

© 2014 Kirk Timothy Landin

2-D RADIO IMAGING OF EQUATORIAL IONOSPHERIC PLASMA DEPLETIONS
WITH THE C/NOFS SATELLITE: ALGORITHMS AND RESULTS

BY

KIRK TIMOTHY LANDIN

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2014

Urbana, Illinois

Adviser:

Professor Farzad Kamalabadi

Abstract

Local depletions in the equatorial ionospheric electron density, known as spread-F events, have been an active area of scientific research in the last decades. These are of special interest to the research community because of their potentially deleterious impact on trans-ionospheric communication links and navigation systems. The impetus for focused modeling and experimentation is the ultimate goal of predicting these events and subsequent mitigation of their negative impacts.

A promising technique for direct imaging of these irregularities is based on a combination of satellite-borne and ground instruments, together with tomographic inversion techniques, to obtain multi-dimensional images of the ionospheric electron density. However, the inverse problem has some noted challenges due to inherent constraints on the acquisition geometry.

One such system is based on coherent VHF and UHF transmission of radio signals by the *Coherent Electromagnetic Radio Tomography* (CERTO) beacons on board the *Communications/Navigation Outage Forecasting System* (C/NOFS) satellite, measured by a ground array of receivers located in a near-horizontal line around -12 degrees latitude. The resulting signals yield a set of electron density projections, known as the *Total Electron Content* (TEC), which can subsequently be used to produce two-dimensional images of ionospheric irregularities.

The inherently exotic sampling geometry yields an under-determined, limited-angle tomography problem making it difficult to reconstruct accurate vertical variations in electron density. Moreover, the non-uniform sampling dictated by the acquisition geometry demands rigorous quantification of the information content in the reconstructed images. In the present work, we develop algorithms optimized for reconstructing ionospheric images containing such irregularities and apply them to experimental data to illustrate the capability and limitations of this approach for experimental investigation of this ionospheric phenomenon.

To my parents Don and Karen, for their endless love and support.

Acknowledgments

The raw imaging data was provided by the US Naval Research Laboratory (NRL), generated from an experiment lead by P. Bernhardt with data administration by M. Wilkens.

Research funding is provided in part by Sandia National Laboratories. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. (SAND2014-2913 P)



Table of Contents

Chapter 1	Introduction	1
1.1	Main Contributions of the Thesis	1
1.2	Thesis Outline	1
Chapter 2	The Ionosphere	3
Chapter 3	Tomographic Imaging of the Ionosphere	6
3.1	Introduction to Tomography	6
3.2	Introduction to Tomography of the Ionosphere	8
3.3	Satellites, CERTO Beacons, and Data Collection	9
3.4	Differential Phase Technique for Measuring TEC	13
3.5	Characterization of the Forward Problem	13
Chapter 4	Linear Inversion	16
4.1	Least-Squares and Minimum-Norm Solutions	16
4.2	Singular Value Decomposition	21
4.3	Matrix Conditioning	24
4.4	Analysis of the Singular Value Decay	25
4.5	SVD Filtering	26
4.6	Tikhonov Regularization	27
4.7	Tikhonov Regularization with Modified Norms	28
4.8	Statistical Interpretation as MAP Estimation	29
4.9	Total-Variation (TV) Regularization	30
4.10	Edge-Preserving Regularization	32
4.11	Non-Convex Regularization with Iterative Relaxation	33
4.12	Iterative Algorithms	35
4.13	Parameter Choice Methods	37
Chapter 5	Image Reconstruction with C/NOFS-CERTO Data	40
5.1	Satellite Tracking	41
5.2	Data-Stream Analysis	41
5.3	Grid Design	43
5.4	Construction of the Inversion Algorithm	48

Chapter 6	Results	63
6.1	Discussion of Output Images	63
6.2	Dependence on the Reference Image	80
Chapter 7	Conclusion	82
7.1	Dealing with Missing Data	82
7.2	New Questions of Future Interest	82
7.3	Concluding Remarks	85
References		86

Chapter 1

Introduction

1.1 Main Contributions of the Thesis

This thesis examines tomographic imaging of the ionosphere with radio waves, and specifically a dataset collected by the *Naval Research Laboratory* (NRL) from a satellite-to-ground imaging experiment that has been running for approximately half a decade. In this thesis, we use edge-preserving regularization techniques to create a novel algorithm for generating images from this dataset. In doing so, we present some of the first images generated from this data using a closed-form regularization technique. We then provide a pragmatic analysis of the tomographic inversion process and resultant images, which will help scientists working with this data in the future to better understand its characteristics and tendencies. This analysis also gives insight into how to mitigate the effects of some undesirable properties inherent in the data.

1.2 Thesis Outline

The thesis is organized as follows: Chapter 2 presents a general background on the ionosphere itself and motivates the utility of imaging the ionosphere. We then discuss the basic design of the NRL's experiment and the data acquisition systems in Chapter 3. Next, we change directions and introduce many of the ingredients necessary to build a successful image reconstruction algorithm: Chapter 4 introduces the basics of linear inverse problems, regularization techniques, and edge-preserving regularization. Chapter 5 then discusses how these regularization techniques can be applied to inverting real-world data, presents other necessary algorithmic ingredients necessary to perform the image reconstruction, and then describes the full image reconstruction algorithm. Chapter 6 presents the results obtained by using this algorithm on data from the NRL experiment along with analysis of how the

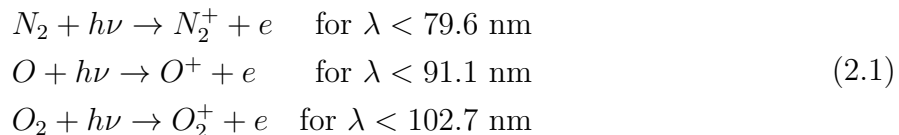
choice of parameters and idiosyncrasies of the algorithm affect the reconstructed images. Finally, Chapter 7 poses a couple of new unanswered questions and gives the concluding remarks.

Chapter 2

The Ionosphere

The *ionosphere* is a layer of electrically conducting plasma that circles the Earth in its upper atmosphere, and it was postulated to exist as early as 1839 by the scientist Carl Friedrich Gauss [1]. In 1901, Guglielmo Marconi had unknowingly leveraged certain reflective properties of the ionosphere when he performed his groundbreaking experiments in transatlantic radio communication. Instead of communications being limited to distances within a line of sight, the ionosphere reflected Marconi's radio waves back towards the earth, which allowed them to travel well beyond the horizon and make their way to the other side of the Atlantic Ocean [2].

The ionosphere exists from 50 km to well past 1000 km in altitude and is formed because the Sun's radiation ionizes some of the common molecular species in the Earth's atmosphere [3].



Due to various physical processes, the precise amount of ionization varies with altitude. At high altitudes, the atmosphere is very thin, providing only a small number of molecules for the Sun's radiation to ionize. Because of this, the level of ionization at high altitudes is very low. At moderate altitudes, the atmosphere is much more dense, providing a much larger source of molecules to ionize, which results in a high level of ionization. At low altitudes, the atmosphere is even more dense, but much of the Sun's radiation has already been filtered out by the upper layers so the level of ionization in the lower ionosphere is much less due to the limited supply of solar radiation [1]. The basic shape of the vertical ionization profile is shown in Figure 2.1.

In 1931, Sydney Chapman put forth the first widely used theoretical framework in which to model the ionosphere. Chapman created a simple model that captured much of the vertical density variations present in the ionosphere, and this model is known as the Chapman

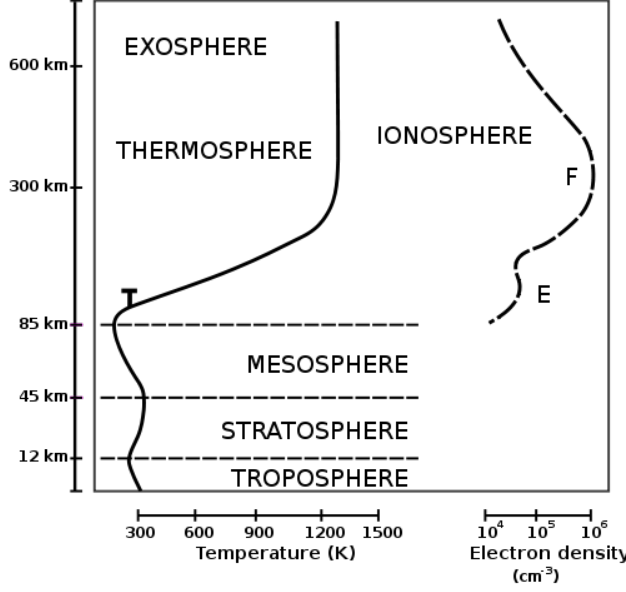


Figure 2.1: Ionosphere Layers, Temperature, and Plasma Density [4]

production function, which is given by

$$q(z) = q_m \exp \left[1 - \frac{z - z_m}{H} - e^{\frac{z_m - z}{H}} \right] \quad (2.2)$$

In this formulation, $q(z)$ is the rate of electron production at z km in altitude. q_m is the maximum production rate, z_m is the altitude at which the production peaks, and H is the scale height, defined as $H = \frac{kT}{mg}$. In this, k is Boltzmann's constant, T is the temperature in Kelvin, m is the molecular weight of the compound, and g is the acceleration due to gravity [3]. The Chapman model is too simple to capture many of the intricate ionosphere happenings that scientists know about today, so it has become somewhat antiquated. Despite its age, the Chapman profile encompasses the basic structure of the ionosphere, so it still forms the core of our understanding. It also drives much of our intuition about the ionosphere's coarse-grained vertical structure.

Analogous to terrestrial weather in the troposphere, the Sun's unequal radiation in the ionosphere creates all sorts of instabilities, and forms the phenomenon known as *space weather* [5]. In the 1930s Berkner and Wells detected an anomaly in the equatorial ionosphere while observing it with a backscatter radar. When taking these radar measurements, they noticed that at certain times the F region, the layer of the ionosphere extending from

approximately 150 km to well past 500 km, would scatter back radio signals that were unexpectedly spread out in range and/or frequency, which did not line up with the pre-conceived model of the ionosphere. Because of the spread out radar measurements returned by the F-layer, these events were named *equatorial spread-F* [5].

As first illustrated by Marconi’s transatlantic radio communication experiment, the ionosphere allows for terrestrial radio links to use *sky waves*, radio waves that are reflected back to earth by the ionosphere in their course of travel [2]. In addition to sky waves, modern radio communications employ satellites, which orbit either within or beyond the ionosphere. Because of this, any radio communication to satellites must travel at least partway through the ionosphere and is subject to any and all of the ionosphere’s deleterious effects. These effects include, but are not limited to, signal attenuation and non-uniform phase and group delay [2]. Many crucial systems such as the *Global Positioning System* (GPS) use communication satellites, so it is advantageous to understand and mitigate these effects as much as possible to ensure that these communication systems perform reliably. In addition to communication systems, many of Earth’s man-made satellites, including the *International Space Station* (ISS), have an orbit located within the ionosphere, so space weather may effect the well-being of the satellites’ electrical systems.

The main ionospheric event that we focus on in this thesis is equatorial spread-F. Our current understanding is that spread-F events are “plasma bubbles” that form and float through the ionosphere [5]. Scientists have linked spread-F bubbles with many radio communication outages [6], and because of this, the knowledge of their exact structure and genesis is of great importance; therefore, these are the subject of very active scientific research.

Chapter 3

Tomographic Imaging of the Ionosphere

3.1 Introduction to Tomography

Tomography is the reconstruction of an unknown quantity, or image, from a number of measurements that are all limited in some way. A form of tomography of great practical use is *projection tomography*, in which an image is reconstructed from any number of line-integral measurements, or projections [7]. Given a figure and a set of line-integrals, it is logical to index each line-integral with two parameters: the angle θ in which the integral intersects the horizontal axis, and the radial coordinate r of the line-integral's closet point to the origin. The set of all line integrals for $r \in (-\infty, \infty)$, $\theta \in [0, \pi)$ forms the *Radon transform* of an image.

Definition 3.1.1. (*Radon Transform*)[7]: *The Radon-transform of a two-dimensional image $f(x, y)$ is the set of projections $s_\theta(r) \forall \theta \in [0, \pi)$ such that:*

$$s_\theta(r) = \int_{-\infty}^{\infty} f(r \cos \theta - t \sin \theta, r \sin \theta + t \cos \theta) dt \quad (3.1)$$

Figure 3.1 depicts the particular line-integral used to compute the projection for a given angle and offset.

To invert the Radon transform we will need employ the use of one more theorem.

Theorem 3.1.1. (*Projection-Slice Theorem*)[7]: *Given a function $f(x, y)$, its Fourier transform, $F(\omega_x, \omega_y)$, the projection $s_\theta(r)$ of $f(x, y)$ at angle θ , and its Fourier transform $S_\theta(\omega_r)$, then*

$$S_\theta(\omega_r) = F(\omega_r \cos \theta, \omega_r \sin \theta) \quad (3.2)$$

which is simply the value of $F(\omega_x, \omega_y)$ at radial coordinate $r, \theta = (\omega_r, \theta)$.

The proof of the projection-slice theorem is very straightforward and can be found in [7]. Equation (3.2) allows us to define $F(u, v)$ completely in terms of $S_\theta(\omega)$ for $\theta \in [0, \pi), \omega \in$

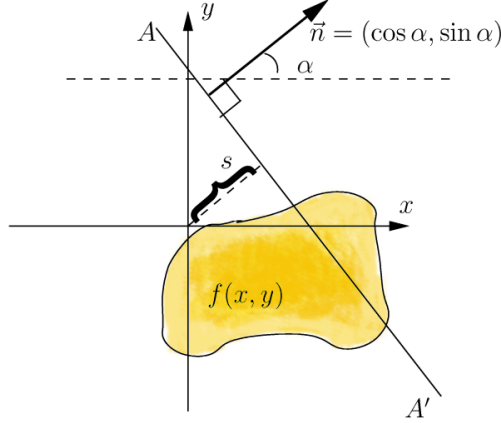


Figure 3.1: Radon Transform Geometry [8]

$(-\infty, \infty)$. Because of this, inverting the Radon transform consists of simply inverting the Fourier transform, $F(\omega_x, \omega_y)$, of $f(x, y)$. Since $F(\omega_x, \omega_y)$ is given in terms of $S_\theta(\omega)$, which is indexed by polar coordinates, the appropriate Jacobian scaling needs to be incorporated, which adds an extra $|\omega|$ term to the integral. Putting it all together yields the inverse operation, known as *filtered back-projection* [7]

Definition 3.1.2. (*Filtered Back-Projection*)[7]: Given a function $f(x, y)$ and its Radon transform $s_\theta(r)$, the inverse of the Radon transform is computed by

$$f(x, y) = \int_0^\pi \int_{-\infty}^\infty S_\theta(\omega) e^{j\omega(x \cos \theta + y \sin \theta)} |\omega| d\omega d\theta \quad (3.3)$$

where $S_\theta(\omega_r)$ is the Fourier transform of the Radon transform, $s_\theta(r)$.

The Radon transform provides the theoretical back-end for projection tomography; however, filtered back-projection fails when the Radon transform data is incomplete, when there exists some (θ, r) for which $s_\theta(r)$ is unavailable. This situation can arise in *limited-angle tomography*, in which the observed θ value is limited to some set $\Theta \subset [0, \pi)$. This situation also arises when Radon transform data is available at only a finite number of points, which is the case with practical data-collection systems. In each of these cases, the under-sampled Radon transform is unable to be inverted, which means that we need to use other methods to reconstruct the image.

3.2 Introduction to Tomography of the Ionosphere

In this thesis, we are primarily interested in imaging the structure of spread-F bubbles, which were first introduced in Chapter 2. These spread-F bubbles exist as variations in plasma density, and a good way to measure the plasma density at any given point is to measure the number of free electrons, or the electron density. Some of the ionosphere’s electrical properties can be leveraged to obtain these electron density measurements through tomographic projection. Since the ionosphere is an electrically dispersive medium, radio waves will propagate at different frequency-dependent velocities [2]. Moreover, these propagation velocities are also dependent on the local electron density.

To put these properties into practice, imagine a two-dimensional region of plasma whose density varies according to the function $f(x, y)$. A simple way to measure electron density is to launch an electromagnetic wave of a known frequency at one side of the region, measure the distance it travels through the region, and measure the time it takes to reach its destination. Since the amount of time needed for the wave to propagate is dependent on the total number of free electrons that the wave encounters; this number, the *total electron content* (TEC) can be computed by performing a simple calculation with the wave’s travel time. The obtained TEC measurement is a line-integral of the plasma’s density function $f(x, y)$ along the wave’s path of travel. If the wave travels in a nearly straight line, the TEC line-integral can be viewed as a straight-line projection of the density function $f(x, y)$ just as in the Radon transform. We can then launch a number of waves at a variety of angles θ and radial offsets r , and use the resulting TEC measurements to create a sampled version of the Radon transform of $f(x, y)$. $f(x, y)$ can then be recovered from the measurements by approximating the inverse to the Radon transform.

This case is somewhat ideal since we have complete control over the exact r and θ coordinates of each transmitted wave. With atmospheric imaging, we do not have this type of fine control over the sampling geometry. Instead, the available r and θ values are heavily constrained by the data collection apparatus. Because of this, the ideal approach of performing plasma tomography via approximating the Radon transform is a far cry from what can actually be done in real life. The exact reasons why are discussed in the next section.

3.3 Satellites, CERTO Beacons, and Data Collection

Due to the inconvenient location of the ionosphere, there is no good way to get a full set of TEC measurements; therefore we are limited to the set of measurements that are able to be obtained in a practical manner. Most practical data collection techniques employ the use of man-made satellites, and they come in two main flavors. The first is a satellite-to-satellite imaging system. These systems employ satellites with two different types of orbits; usually one has a *low earth orbit* (LEO) and the other has a *medium earth orbit* (MEO). One of the satellites is equipped with a radio transmitter, and the other with a radio receiver. Due to the unique nature of each satellite's orbit, the pair of satellites will often get into a configuration in which a region of the ionosphere lies in between them. When this happens, both satellites will activate their radio instruments and generate TEC measurements. The constant change in the satellites' absolute position with respect to the ionosphere and relative position with respect to each other provides a variety of different paths for the radio waves to travel.

A second style of imaging is satellite to ground imaging. In this setup, a radio transmitter is mounted on one or more satellites (both LEO and MEO satellites have been used), and multiple receivers are placed at strategic locations on the ground. When the satellite comes in view of the receivers, they begin to record TEC measurements, and continue to do so until the satellite sets over the other horizon. The constantly changing position of the satellite and the existence of multiple receivers create a variety of different paths for the radio waves to travel. With the CERTO beacons, the receivers use these measurements to compute one TEC integral per second. Since the satellite is never stationary, each TEC integral can be understood as the TEC in a very narrow region surrounding the wave's path of travel between the satellite and the receiver. A graphical depiction of the line integrals is given in Figure 3.2. As shown here, the coverage of these line integrals is very irregular, and this is a far cry from the systematic coverage of the Radon transform. In addition, we see that the middle of the grid is sampled more often than the outsides, and contains ray paths coming from many different angles. In contrast, the edges of the grid only get sampled once. This discrepancy in sampling causes an information non-uniformity in the grid, which will have implications later on in the image reconstruction process.

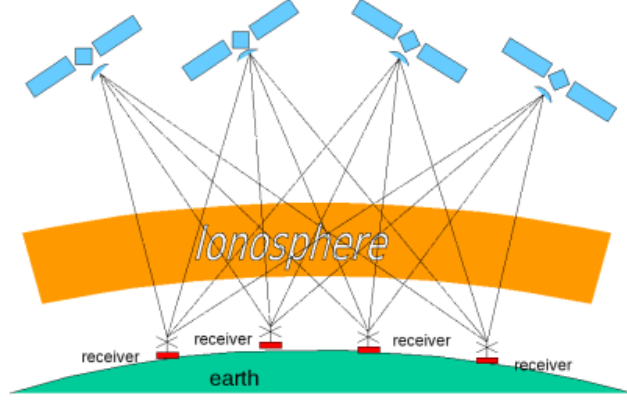


Figure 3.2: Two-Dimensional Space to Ground Imaging [9]

3.3.1 C/NOFS and CERTO

To perform space-to-ground imaging, we employ the *Communications/Navigation Outage Forecasting System* (C/NOFS) satellite. C/NOFS was launched on April 16, 2008, and its main purpose is to perform scientific research pertaining to the ionosphere. To achieve this it uses a wide array of instruments. C/NOFS has an orbit with a 13 degree inclination which means that its orbit intersects the equatorial plane at a 13 degree angle [6]. Due to its low inclination angle, C/NOFS has a nearly *equatorial orbit* [10]. A satellite's orbit type has an enormous impact on the types of space-to-ground imaging it is able to perform. Since C/NOFS' orbit is nearly equatorial, it traverses a wide variety of longitudes in a short period of time and using it as a radio source for imaging will provide good imaging resolution in 2-D planes of constant latitude. Furthermore, C/NOFS orbits in the direction of Earth's rotation, which means that for a ground-based observer it will rise in the west sky and set in the east. On the other hand, C/NOFS has very little latitude variation so it would provide very poor resolution for creating a 2-D image in a plane of constant longitude. Another style of satellite orbit is the *polar orbit*, in which a satellite's path passes directly over both poles on each trip around the Earth. This is an extremely popular style of orbit and a large number of currently deployed satellites travel in an orbit that is nearly polar [10]. Contrary to equatorial orbit satellites, a polar orbit satellite will have excellent imaging resolution in 2-D planes of constant longitude but have poor resolution in 2-D planes of constant latitude.

C/NOFS' equatorial orbit is important because the plasma depletions that make up spread-F bubbles follow the earth's magnetic field lines; therefore, most of their structural variations are in the longitudinal direction [11]. Because of this, C/NOFS will do a much

Table 3.1: CERTO Transmission Frequencies

Frequency	Band
150.012 MHz	VHF
400.032 MHz	UHF
1066.752 MHz	L-band

Table 3.2: Receivers: Geographic Coordinates

Receiver	Latitude (Deg)	Longitude (Deg)
Huancayo	-12.040	-75.320
Ayacucho	-13.160	-74.200
Cusco	-13.530	-71.970
Puerto Maldonado	-12.580	-69.180

better job in capturing a spread-F bubble’s structure than a satellite with a polar orbit.

To be useful for imaging, C/NOFS is equipped with a *coherent electromagnetic radio tomography* (CERTO) beacon instrument which provides the radio signal necessary to perform tomography. The CERTO beacon transmits three radio signals simultaneously at the frequencies listed in Table 3.1, all with a right-handed circular polarization [12]. These three signals are all transmitted as pure sinusoids, without any sort of modulation, which simplifies the job of generating TEC measurements at the receiver. These specific frequencies were chosen for a variety of reasons, and can be referenced in [12].

On the receiving end, four receivers are deployed in southern Peru, in a line of near constant latitude, located near the cities of Ayacucho, Huancayo, Puerto Maldonado, and Cusco [12]. The receivers’ geographic locations are given in Table 3.2, and are displayed as a map in Figure 3.3.

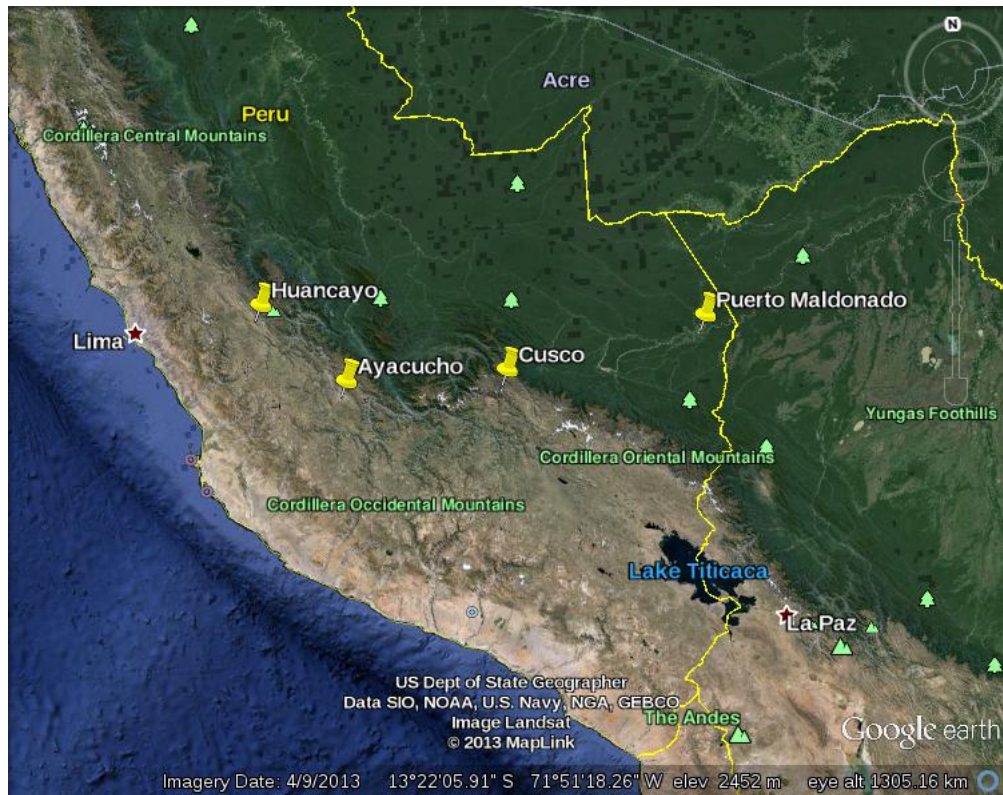


Figure 3.3: The Four Different Receivers (image generated by Google Earth)

3.4 Differential Phase Technique for Measuring TEC

Due to the dispersion of the ionosphere, each of these three frequencies will travel at different velocities, given by [12]

$$P_a = \left(S - \int \frac{\epsilon N}{f_a^2} ds \right) \frac{f_a}{c} + \eta_a \quad (3.4)$$

where S is the distance between the satellite and the receiver, $\epsilon = 40.3$ in mks units, f_a is the frequency of the transmitted signal, c is the speed of light, $3 \times 10^8 \frac{\text{m}}{\text{s}}$, ds is the increment along the path, and η_a is the system phase bias, which is unavoidable in a practical system. Given two waves with frequencies f_a and f_b , the phase delay difference, or *differential phase* between the signals is [12]

$$P_{ab} = P_a - P_b \frac{f_a}{f_b} = \frac{(f_a - f_b)(f_a + f_b)\epsilon}{cf_a f_b^2} \int N ds + \eta_{ab} \quad (3.5)$$

where $\eta_{ab} \triangleq \eta_a - \eta_b \frac{f_a}{f_b}$ is the differential phase bias.

Given a set of differential phase measurements, the software on the receivers inverts Equation (3.5) to estimate the TEC along the radio signal's path of travel. Due to phase-wrapping, there is an inherent 2π phase ambiguity in the measurements, which maps to an ambiguity in the reported TEC. Using all three of the transmitted frequencies, the TEC ambiguity Ψ is [12]

$$\Psi = 8.3165 \times 10^{16} \text{m}^{-2} \quad (3.6)$$

Because of this, it is advantageous to use all three frequencies in the TEC calculation because the TEC ambiguity due to the measurement of three frequencies is nearly 50 times larger than it would be if only two different frequencies were used [12]. Due to the presence of this TEC ambiguity, the receivers cannot calculate the absolute TEC, so they calculate the TEC relative to an unknown additive offset, which is called the *relative total electron content* (RTEC). The absolute TEC values will need to be estimated from the RTEC measurements during the image reconstruction process.

3.5 Characterization of the Forward Problem

The set of TEC measurements generated with the satellite only correspond to relatively few samples of an entire Radon transform, and the available samples give very non-uniform coverage of the Radon transform space. Because of this, it is better to characterize the

forward model not as a Radon transform, but as a more general class of problem, in which the Radon transform belongs, called a *Fredholm integral equation of the first kind*. The general form of a Fredholm integral equation is

$$g(n, t) = \iint_{\Omega} h(n, t; x, y) f(x, y) dx dy \quad (3.7)$$

In this equation, $f(x, y)$ is the original quantity; $h(n, t; x, y)$, commonly known as the *observation kernel*, describes the transformation applied by the observation geometry; and Ω is the observation kernel's region of support. $g(n, t)$ is the measured quantity, and is parameterized by the receiver number n and the time of observation t . Writing the entire integral equation as an operator $h(\cdot)$ yields the compact notation

$$g(n, t) = h(f(x, y)) \quad (3.8)$$

Furthermore, the Fredholm integral operator $h(\cdot)$ is linear since it only performs scalar multiplication and integration on its input. Because of this linearity, inverting $h(\cdot)$ to get $f(x, y)$ is a *linear inverse problem*, a class of problems that have been, and still are, heavily researched.

The continuous formulation of the Fredholm equation as an integral works well in analytical derivations; however, the integration must be discretized when working with real data on a computer. To perform discretization, a basis $\{\phi_i(x, y)\}$ is chosen for the input space and the initial quantity $f(x, y)$ is approximated as a finite linear combination of these basis functions

$$f(x, y) \approx \sum_i \mathbf{f}_i \phi_i(x, y) \quad (3.9)$$

where vector $\mathbf{f}$ contains the approximate coordinates of $f(x, y)$ with respect to the basis. A widely used family of basis functions is the set of “unit boxes” that make up a regularly spaced grid in the input space. The grid of “unit boxes” is a very intuitive way to discretize the ionosphere, and it is what will be used for the image reconstruction in later sections.

Any real-world observation $g(n, t)$ is already discrete so it can also be converted into a vector $\mathbf{g} \in \mathbb{R}^m$. $h(\cdot)$ can then be represented by a matrix $\mathbf{H} \in \mathbb{R}^{m \times n}$, in which element $\mathbf{H}_{ij}$ is the degree in which coordinate j in vector $\mathbf{f}$ influences measurement i in vector $\mathbf{g}$. Therefore, the complete observation relationship becomes a matrix-vector multiplication $\mathbf{g} = \mathbf{H}\mathbf{f}$, and recovering the original quantity $\mathbf{f}$ is accomplished by inverting this linear system. Ideally, the inversion would only involve computing the matrix inverse $\mathbf{H}^{-1}$; however, life is never that

simple. Due to the limited-angle and irregularly-sampled nature of the problem, matrix $\mathbf{H}$ is far from invertible, so other techniques need to be used to recover $\mathbf{f}$ from the observation $\mathbf{g}$. The next two chapters will present some of these linear inversion techniques and how they can be used to perform successful image reconstruction.

Chapter 4 introduces a set of linear inversion concepts which apply to a wide variety of problems, and then Chapter 5 describes how to apply these general concepts to create an image reconstruction algorithm to deal specifically with the C/NOFS-CERTO data.

Chapter 4

Linear Inversion

As described in Chapter 3, satellite to ground imaging is well approximated by a linear operation of the form $\mathbf{g} = \mathbf{H}\mathbf{f}$. However, the inverse $\mathbf{H}^{-1}$ does not exist, so inverting $\mathbf{H}$ will need to be done in a more strategic fashion that exploits $\mathbf{H}$'s linear structure. This chapter will discuss some of the main strategies for inverting problematic linear operations such as $\mathbf{H}$. We begin by introducing two classic cases of non-invertible matrices, then present the *singular value decomposition* (SVD), an important method to analyze the internal structure of linear operations, and finally introduce more advanced linear inversion techniques, all of which rely on the general intuition provided by the SVD.

4.1 Least-Squares and Minimum-Norm Solutions

Given that $\mathcal{X}$ and $\mathcal{Y}$ are n and m dimensional Euclidean vector spaces of complex numbers, (denoted as $\mathbb{C}^n$ and $\mathbb{C}^m$ respectively), any linear transformation, $f(\cdot) : \mathcal{X} \mapsto \mathcal{Y}$, can be represented as an $m \times n$ matrix $\mathbf{A}$ of complex numbers. The space of these matrices is denoted as $\mathbb{C}^{m \times n}$.

Another important, and closely related, vector space is the n dimensional space of real numbers, denoted $\mathbb{R}^n$. Many of these preliminaries are presented in terms of the complex vector spaces. However, they all hold for the real spaces as well, since any vector $\mathbf{x} \in \mathbb{R}^n$ is a vector in $\mathbb{C}^n$ with all imaginary parts set to zero. The only difference in notation that comes up when working with $\mathbb{C}^n$ vs. $\mathbb{R}^n$ is when dealing with *adjoint operators*. For $\mathbf{A} \in \mathbb{C}^{m \times n}$, the corresponding adjoint operation is the Hermitian transpose $\mathbf{A}^H$ which is the complex conjugate of $\mathbf{A}^T$. For $\mathbf{A} \in \mathbb{R}^{m \times n}$, complex conjugates do not matter so $\mathbf{A}^H = \mathbf{A}^T$. For working with purely real vector spaces, simply take any $\mathbf{A}^H$ defined in this section and replace it with an $\mathbf{A}^T$.

The number of linearly independent rows in a matrix $\mathbf{A}$ is the same as the number of linearly independent columns, and this number is called the *rank* of matrix $\mathbf{A}$ [13]. If

$\mathbf{A} \in \mathbb{C}^{n \times n}$ and $\text{Rank}(\mathbf{A}) = n$, then $\mathbf{A}$ is full-rank, and its linear mapping on a vector, denoted by the equation $\mathbf{b} = \mathbf{A}\mathbf{x}$, is both one-to-one (injective) and onto (surjective). Because of this, the matrix $\mathbf{A}$ is invertible; the inverse matrix $\mathbf{A}^{-1}$ exists. For the equation $\mathbf{A}\mathbf{x} = \mathbf{b}$, given a vector $\mathbf{b}$, $\mathbf{x}$ can be computed as $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$. For a non-square $\mathbf{A} \in \mathbb{C}^{m \times n}$, the largest its rank can be is $\min(m, n)$, since there cannot be more linearly independent row/columns than the total number present in the entire matrix. Next, we define two fundamental subspaces of a matrix.

Definition 4.1.1. (*Range Space*): The range space of a matrix $\mathbf{A}$, denoted as $R(\mathbf{A})$ is the set of vectors

$$R(\mathbf{A}) = \{\mathbf{A}\mathbf{z}, \forall \mathbf{z} \in \mathbb{C}^n\}$$

The range space is simply the span of all column vectors in $\mathbf{A}$, and is the space of all possible vectors in $\mathbb{C}^m$ that $\mathbf{A}$ can map to given any possible input vector $\mathbf{z} \in \mathbb{C}^n$.

If $m > n$ and $\text{Rank}(\mathbf{A}) = n$, $R(\mathbf{A})$ is an n -dimensional subspace of $\mathbb{R}^m$, so the mapping is no longer onto (or surjective). Because of this, $\exists \mathbf{y} \in \mathbb{C}^m : \forall \mathbf{x} \in \mathbb{C}^n, \mathbf{y} \neq \mathbf{A}\mathbf{x}$, which means that $\mathbf{A}$ is no longer invertible. In this case, $\mathbf{A}$ is *over-determined*.

Next, consider the case in which $\mathbf{A} \in \mathbb{C}^{m \times n}$ where $m < n$ and $\text{Rank}(\mathbf{A}) = m$. Here $R(\mathbf{A})$ is the full $\mathbb{C}^m$ space, but the transformation $\mathbf{b} = \mathbf{A}\mathbf{x}$ needs to throw out some information in $\mathbf{x}$ to allow $\mathbf{b}$ to fit inside the lower dimensional range space. Specifically, transformation $\mathbf{b} = \mathbf{A}\mathbf{x}$ throws out all information in $\mathbf{x}$ living in an $n - m$ dimensional subspace of $\mathbb{C}^n$ called the *Null Space* of $\mathbf{A}$.

Definition 4.1.2. (*Null Space*): The null-space of a matrix $\mathbf{A}$, denoted as $N(\mathbf{A})$ is the set of vectors

$$N(\mathbf{A}) = \{\mathbf{x} : \mathbf{A}\mathbf{x} = \mathbf{0}\}$$

Conceptually the null-space of a matrix $\mathbf{A}$ is the set of all vectors that the matrix cannot “see”, that is, all vectors that disappear after passing through $\mathbf{A}$.

In the case when $m < n$, $N(\mathbf{A})$ is non-trivial, that is $N(\mathbf{A}) \neq \mathbf{0}$ and $\forall \mathbf{y} \in \mathbb{C}^m, \exists$ infinitely many $\mathbf{x} : \mathbf{y} = \mathbf{A}\mathbf{x}$. Because of this, the mapping is no longer one-to-one (or injective), so $\mathbf{A}$ is not invertible and is called *under-determined*.

We can extend these concepts to the case where $\text{Rank}(\mathbf{A}) < m$ and $\text{Rank}(\mathbf{A}) < n$. Here, the system is neither injective nor surjective, and is simultaneously over-determined and under-determined.

Many inverse problems that arise in real life may be either over-determined or under-determined, so it is useful to develop notions of an “inverse” operation even though an exact

inverse does not exist. The next section looks at two classic ways to do this.

4.1.1 Least Squares Solutions

Suppose $\mathbf{A} \in \mathbb{C}^{m \times n}$ is an over-determined system of rank $n \leq k < m$. The range space of $\mathbf{A}$ is a k -dimensional subspace of $\mathbb{R}^m$ [13]. Given any vector $\mathbf{b} \in \mathbb{C}^m$, it is very unlikely that $\mathbf{b} \in R(\mathbf{A})$, so a reasonable definition of $\mathbf{x} : \mathbf{Ax} = \mathbf{b}$, would be the vector $\hat{\mathbf{x}}$ such that $\mathbf{A}\hat{\mathbf{x}}$ is closer to $\mathbf{b}$ than any other such $\{\tilde{\mathbf{b}} : \tilde{\mathbf{b}} \in R(\mathbf{A})\}$. A good definition of this closeness is the squared Euclidean distance, or $\|\mathbf{b} - \mathbf{Ax}\|_2^2$. We can now use this to construct a useful notion of “inverse”, called the *least-squares* solution $\mathbf{x}_{LS}$ of a system $\mathbf{Ax} = \mathbf{b}$.

Definition 4.1.3. (*Least-Squares Solution*)[13]: Given $\mathbf{b} \in \mathbb{C}^m$, $\mathbf{x}_{LS}$ is a least-squares solution to $\mathbf{Ax} = \mathbf{b}$ if:

$$\|\mathbf{b} - \mathbf{Ax}_{LS}\|_2^2 \leq \|\mathbf{b} - \mathbf{Ax}\|_2^2 \quad \forall \mathbf{x} \in \mathbb{C}^n \quad (4.1)$$

To compute the least-squares solution, we need to use a few of other theorems and definitions.

Definition 4.1.4. (*Projector*)[13]: A mapping $\mathbf{P}_S : \mathbb{C}^n \mapsto \mathcal{S}$ is a projector associated with subspace $\mathcal{S}$ if it satisfies

$$\mathbf{P}_S(\mathbf{x}) - \mathbf{x} \in \mathcal{S}^\perp, \quad \forall \mathbf{x} \in \mathbb{C}^n \quad (4.2)$$

Theorem 4.1.1. (*Nearest Point Theorem*)[13]: Suppose that a projector $\mathbf{P}_S$ exists for a subspace $\mathcal{S} \in \mathbb{C}^n$. Then for any $\mathbf{x} \in \mathbb{C}^n$, $\mathbf{P}_S(\mathbf{x})$ is the unique vector in $\mathcal{S}$ that is closest to $\mathbf{x}$

$$\|\mathbf{x} - \mathbf{P}_S(\mathbf{x})\|_2 \leq \|\mathbf{x} - \mathbf{s}\|_2 \quad \forall \mathbf{s} \in \mathcal{S}, \mathbf{x} \in \mathbb{C}^n \quad (4.3)$$

with equality iff $\mathbf{s} = \mathbf{P}_S(\mathbf{x})$.

The proof for this theorem can be found in [13].

Due to the nearest point theorem, the least-squares solution $\mathbf{x}_{LS}$ of $\mathbf{Ax} = \mathbf{b}$ satisfies

$$\mathbf{Ax}_{LS} = \mathbf{P}_{R(\mathbf{A})}(\mathbf{b}) \quad (4.4)$$

So the solution to the least squares problem will involve projecting $\mathbf{b}$ onto $R(\mathbf{A})$ and then finding the coordinates of $\mathbf{P}_{R(\mathbf{A})}(\mathbf{b})$ with respect to $\mathbf{A}$. To do that, we will invoke one more, very powerful theorem.

Theorem 4.1.2. (*Fundamental Theorem of Linear Algebra*)[13]: If $\mathbf{A} \in \mathbb{C}^{m \times n}$ matrix, then

$$R^\perp(\mathbf{A}) = N(\mathbf{A}^H), \quad N^\perp(\mathbf{A}^H) = R(\mathbf{A}) \quad (4.5)$$

and

$$R^\perp(\mathbf{A}^H) = N(\mathbf{A}), \quad N^\perp(\mathbf{A}) = R(\mathbf{A}^H) \quad (4.6)$$

These theorems give three important facts about the least-squares solution to $\mathbf{A}\mathbf{x} = \mathbf{b}$. The nearest point theorem says that $\mathbf{A}\mathbf{x}_{\text{LS}}$ is the orthogonal projection of $\mathbf{b}$ onto $R(\mathbf{A})$. The definition of projection states that $(\mathbf{A}\mathbf{x}_{\text{LS}} - \mathbf{b}) \perp R(\mathbf{A})$. Finally, the fundamental theorem of linear algebra says that $\mathbf{b} \perp R(\mathbf{A}) \Rightarrow \mathbf{b} \in N(\mathbf{A}^H)$. Noting these three facts, $\mathbf{x}_{\text{LS}}$ must satisfy the following equation:

$$\mathbf{A}^H(\mathbf{A}\mathbf{x}_{\text{LS}} - \mathbf{b}) = 0 \quad (4.7)$$

Distributing the $\mathbf{A}^H$ operation yields the *normal equations*

$$\mathbf{A}^H \mathbf{A} \mathbf{x}_{\text{LS}} = \mathbf{A}^H \mathbf{b} \quad (4.8)$$

If $\mathbf{A}$ is full column-rank, then $\mathbf{A}^H \mathbf{A}$ is guaranteed to be invertible [13], so solving for $\mathbf{x}_{\text{LS}}$ gives

$$\mathbf{x}_{\text{LS}} = (\mathbf{A}^H \mathbf{A})^{-1} \mathbf{A}^H \mathbf{b} \quad (4.9)$$

The operation on the right-hand side of Equation (4.9) can then be written as a single linear operator $\mathbf{A}^{\text{LS}}$ such that $\mathbf{x}_{\text{LS}} = \mathbf{A}^{\text{LS}} \mathbf{b}$, $\forall \mathbf{b} \in \mathbb{C}^m$.

4.1.2 Minimum Norm Solutions

Suppose $\mathbf{A} \in \mathbb{C}^{m \times n}$ is an under-determined system of rank $m \leq k < n$. Then given a vector $\mathbf{b} \in R(\mathbf{A})$, the system $\mathbf{A}\mathbf{x} = \mathbf{b}$ has an infinite number of possible solutions, $\hat{\mathbf{x}}$. More specifically, each solution $\hat{\mathbf{x}}$ that satisfies $\mathbf{A}\mathbf{x} = \mathbf{b}$ can be written in the form $\hat{\mathbf{x}} = \mathbf{x}_p + \mathbf{s}$, where $\mathbf{s} \in N(\mathbf{A})$ and $\mathbf{x}_p$, the *particular solution*, satisfies $\mathbf{A}\mathbf{x}_p = \mathbf{b}$ [13]. The set that contains all possible solutions $\mathbf{x}_p + \mathbf{s}$ is a translated subspace, or *linear variety*[13].

It is beneficial to determine which single solution in the entire linear variety would be particularly useful to single out as the specific “inverse”. We may know prior information about how the solution should behave, so one possible approach is to find the solution that is closest to a particular guess $\mathbf{x}_0$. Just as with the least-squares solution, we quantify the notion of closeness with the squared Euclidean distance between the two vectors, $\|\mathbf{x} - \mathbf{x}_0\|_2^2$.

$\mathbf{x}_0$ could be any possible value in $\mathbb{C}^n$; however, it is easy enough to define the solution for $\mathbf{x}_0 = \mathbf{0}$ and then simply translate the system of equations to accommodate any other possible $\mathbf{x}_0$. For the choice of $\mathbf{x}_0 = \mathbf{0}$, the squared Euclidean distance, $\|\hat{\mathbf{x}} - \mathbf{x}_0\|_2^2$, becomes $\|\mathbf{x}\|_2^2$, the squared Euclidean norm of $\hat{\mathbf{x}}$. Consequently, this solution is called the *minimum-norm* solution $\mathbf{x}_{\text{MN}}$ of the system $\mathbf{A}\mathbf{x} = \mathbf{b}$.

One extra theorem is needed to construct the minimum-norm solution.

Theorem 4.1.3. (*Dual Projection Theorem*)[13]: Let $\mathcal{T} = \{\mathbf{x}_p\} + \mathcal{S}$ be a translated subspace of $\mathbb{C}^n$. Then the element of $\mathcal{T}$ with minimum norm, i.e.

$$\mathbf{t}_{\text{MN}} = \arg \min_{\mathbf{t} \in \mathcal{T}} \|\mathbf{t}\|_2^2$$

exists, is unique, and it is the only point that satisfies $\mathbf{t}_{\text{MN}} \in \mathcal{T} \cap S^\perp$. Furthermore $\mathbf{t}_{\text{MN}} = \mathbf{P}_{\mathcal{S}^\perp} \mathbf{t}$, the orthogonal projection of $\mathbf{t}$ onto $\mathcal{S}$, for any $\mathbf{t} \in \mathcal{T}$.

The proof can be found in [13].

In the Minimum-Norm problem, $\mathcal{T}$ is a shifted version of $N(\mathbf{A})$, so $\mathbf{x}_{\text{MN}}$ must satisfy

$$\mathbf{x}_{\text{MN}} \perp N(\mathbf{A}) \tag{4.10}$$

Applying the fundamental theorem of linear algebra (Theorem 4.1.2),

$$\mathbf{x}_{\text{MN}} \perp N(\mathbf{A}) \Rightarrow \mathbf{x}_{\text{MN}} \in R(\mathbf{A}^H) \tag{4.11}$$

$$\Rightarrow \exists \mathbf{z} \in \mathbb{C}^m, \text{ st } \mathbf{x} = \mathbf{A}^H \mathbf{z} \tag{4.12}$$

$$\tag{4.13}$$

Substituting $\mathbf{x} = \mathbf{A}^H \mathbf{z}$ into $\mathbf{A}\mathbf{x} = \mathbf{b}$ yields

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{4.14}$$

$$\mathbf{A}\mathbf{A}^H \mathbf{z} = \mathbf{b} \tag{4.15}$$

$$\mathbf{z} = (\mathbf{A}\mathbf{A}^H)^{-1} \mathbf{b} \tag{4.16}$$

$$\tag{4.17}$$

and if $\mathbf{A}$ is full row-rank, $\mathbf{A}\mathbf{A}^H$ is guaranteed to be invertible [13]. Finally, since $\mathbf{x}_{\text{MN}} = \mathbf{A}^H \mathbf{z}$,

the minimum-norm solution is found by

$$\mathbf{x}_{\text{MN}} = \mathbf{A}^H(\mathbf{A}\mathbf{A}^H)^{-1}\mathbf{b} \quad (4.18)$$

Similar to the least-squares solution, the operation in the right-hand side of Equation (4.18) can be written as a single linear operator $\mathbf{A}^{\text{MN}}$ where $\mathbf{x}_{\text{MN}} = \mathbf{A}^{\text{MN}}\mathbf{b}$.

Finally, if $\mathbf{A}$ is neither full row-rank nor full column-rank, the system $\mathbf{A}\mathbf{x} = \mathbf{b}$ is both over and under-determined. In this case the standard least-squares solution is no longer unique. Instead, it is another linear variety, in which there exists a unique vector of minimum-norm [13]. In this case, the theory can be extended to find the *minimum-norm least-squares solution* $\mathbf{x}_{\text{MNLS}}$ of $\mathbf{A}\mathbf{x} = \mathbf{b}$. Its derivation will not be presented here, but can be seen in [13].

4.1.3 The Pseudo-Inverse Operator

In addition to $\mathbf{A}^{\text{LS}}$ and $\mathbf{A}^{\text{MN}}$, there exists another very powerful inverse operator of $\mathbf{A}$ called the *pseudo-inverse*, notated as $\mathbf{A}^+$. Given a system $\mathbf{A}\mathbf{x} = \mathbf{b}$, application of the pseudo-inverse $\mathbf{A}^+\mathbf{b}$ will generate either $\mathbf{x}_{\text{LS}}$, $\mathbf{x}_{\text{MN}}$, or $\mathbf{x}_{\text{MNLS}}$ depending on how $\mathbf{A}$ is rank-deficient. If $\mathbf{A}$ is full-rank, then $\mathbf{A}^+\mathbf{b}$ equals $\mathbf{x}$ itself. This shows that $\mathbf{A}^+$ embodies the entire set of inverse operations, $\mathbf{A}^{-1}$, $\mathbf{A}^{\text{LS}}$, $\mathbf{A}^{\text{MN}}$, and $\mathbf{A}^{\text{MNLS}}$ so it is a very powerful operator. In addition, the process used to construct $\mathbf{A}^+$, known as *singular value decomposition* (SVD), gives us deep insight into the inner structure of $\mathbf{A}$, which will be very useful in generating good inversion methods.

4.2 Singular Value Decomposition

The *singular value decomposition* (SVD) of a matrix is a very important matrix factorization and it has played a prominent role in advancing many disciplines in applied mathematics. Singular value decomposition is an extension of the spectral decomposition of a Hermitian symmetric matrix that can be applied to any matrix, no matter what its shape or structure. In this section, we start by reviewing the classic spectral decomposition, and then explain how it is built upon to create the SVD.

For any Hermitian symmetric matrix $\mathbf{A}$, all of its eigenspaces are mutually orthogonal. Given that each eigenvalue has multiplicity of at most one, the eigenvectors $\mathbf{v}_i$ of $\mathbf{A}$ form an orthogonal basis for $R(\mathbf{A})$ and $R(\mathbf{A}^H)$, that is $\forall i, j$ st $i \neq j$, $\langle \mathbf{v}_i, \mathbf{v}_j \rangle = 0$ and $R(\mathbf{A}) =$

$R(\mathbf{A}^H) = \text{Span}(\{\mathbf{v}_i\})$. In addition, each of these eigenvectors can be normalized such that $\forall i, \|\mathbf{v}_i\|_2^2 = \langle \mathbf{v}_i, \mathbf{v}_i \rangle = 1$. If there exists an eigenspace with multiplicity greater than one, a method such as Gram-Schmidt orthogonalization can be used to create an orthonormal basis for this multi-dimensional eigenspace. Therefore, given a Hermitian-symmetric matrix $\mathbf{A}$, there always exists an orthonormal basis for $R(\mathbf{A})$ made up exclusively by eigenvectors of $\mathbf{A}$. This orthogonality of eigenspaces paves the way for the *Spectral theorem*.

Theorem 4.2.1. (*Spectral Theorem*)[13]: *If $\mathbf{A} \in \mathbb{C}^{n \times n}$ is Hermitian, then there exists a unitary matrix $\mathbf{V} \in \mathbb{C}^{n \times n}$, and a real diagonal matrix $\mathbf{\Lambda} \in \mathbb{R}^{n \times n}$ such that*

$$\mathbf{\Lambda} = \mathbf{V}^H \mathbf{A} \mathbf{V} \quad (4.19)$$

or equivalently

$$\mathbf{A} = \mathbf{V} \mathbf{\Lambda} \mathbf{V}^H = \sum_{k=1}^n \lambda_k \mathbf{v}_k \mathbf{v}_k^H \quad (4.20)$$

For clarity, a *unitary matrix* $\mathbf{V} \in \mathbb{C}^{n \times n}$ is a matrix in which $\mathbf{V}^H \mathbf{V} = \mathbf{I}_n$, and the columns of $\mathbf{V}$ form an orthonormal set in $\mathbb{C}^n$ [13]. The unitary matrix used in the spectral theorem can be constructed from normalized versions of $\mathbf{A}$'s eigenvectors. In this case, for each column $\mathbf{v}_i$, of $\mathbf{V}$

$$\begin{aligned} \mathbf{v}_i^H \mathbf{A} \mathbf{v}_i &= \lambda_i \mathbf{v}_i^H \mathbf{v}_i \\ &= \lambda_i \end{aligned} \quad (4.21)$$

since $\mathbf{v}_i^H \mathbf{v}_i = \|\mathbf{v}_i\|_2^2 = 1$. Due to this and the fact that all $\mathbf{v}_i$ are orthogonal, it follows that $\mathbf{V}^H \mathbf{A} \mathbf{V}$, must be a diagonal matrix $\mathbf{\Lambda}$. In addition, Equation (4.21) says that the i th diagonal entry of $\mathbf{\Lambda}$ must be the eigenvalue λ_i corresponding to eigenvector $\mathbf{v}_i \in \mathbf{V}$.

Using the factorization $\mathbf{A} = \mathbf{V} \mathbf{\Lambda} \mathbf{V}^H$, let us walk through what happens when Hermitian-symmetric matrix $\mathbf{A}$ is left-multiplied to a given vector $\mathbf{x}$. First, we multiply $\mathbf{d} = \mathbf{V}^H \mathbf{x}$. Since $\mathbf{V}^H = \mathbf{V}^{-1}$, $\mathbf{d}$ will contain the coordinates of $\mathbf{x}$ with respect to the $\{\mathbf{v}_i\}$ basis. Next $\mathbf{f} = \mathbf{\Lambda} \mathbf{d}$, and since $\mathbf{\Lambda}$ is diagonal, this operation just scales coordinate $\mathbf{d}_i$ up by a factor of λ_i . Lastly, $\mathbf{y} = \mathbf{V} \mathbf{f}$ simply maps the scaled coordinates in $\mathbf{f}$ back to their original vectors in $\{\mathbf{v}_i\}$. Putting this all together, what the operation $\mathbf{A} \mathbf{x}$ does is take each $\mathbf{v}_i$ component of $\mathbf{x}$ and scale it up by its corresponding eigenvalue λ_i .

A good example of this diagonalization in action is with the *finite impulse response* (FIR) filtering of a finite-length signal $\mathbf{s} \in \mathbb{C}^n$. FIR filtering is performed by convolving the input signal $\mathbf{s}$ with kernel $\mathbf{h}$. Since convolution is a linear operation over a Euclidean space,

there exists a matrix $\mathbf{H}$ that embodies the operation. From basic signal processing, we know that convolution can also be performed by transforming $\mathbf{s}$ into the frequency domain, multiplying the transformed version by the frequency response of $\mathbf{h}$, and then transforming it back. The transformation into the frequency domain, the *discrete Fourier transform* (DFT), is an orthogonal transformation, so it can be computed with the unitary DFT matrix $\mathbf{V}^H$. Multiplication by the frequency response of $\mathbf{h}$ is simply a diagonal scaling, so it can be performed by multiplying with diagonal matrix $\mathbf{\Lambda}$. The inverse DFT can be accomplished by multiplying by the inverse of $\mathbf{V}^H$, which is $\mathbf{V}$ since it is a unitary matrix. By doing this, we have decomposed the filtering operation H into three steps $\mathbf{V}\mathbf{\Lambda}\mathbf{V}^H$, and this decomposition is precisely the eigendecomposition of $\mathbf{H}$. Regardless of whether the FIR filtering is performed in one step or decomposed into three steps, the same diagonal scaling will happen to each of the frequency components in $\mathbf{s}$.

The SVD takes this intuition one step further. Just as the Spectral theorem did for Hermitian symmetric matrices, the SVD generates a diagonal representation for any matrix, no matter what its geometry. For a rank r matrix $\mathbf{A} \in \mathbb{C}^{m \times n}$, there exist matrices $\mathbf{U} \in \mathbb{C}^{m \times r}$ and $\mathbf{V} \in \mathbb{C}^{n \times r}$, such that $\mathbf{U}^H \mathbf{U} = \mathbf{V}^H \mathbf{V} = \mathbf{I}_r$, and diagonal matrix $\mathbf{\Sigma} \in \mathbb{C}^{r \times r}$ such that [13]

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^H = \sum_{k=1}^r \mathbf{u}_k \sigma_k \mathbf{v}_k^H \quad (4.22)$$

Furthermore, the diagonal entries σ_k in $\mathbf{\Sigma}$ are non-negative and satisfy the ordering [13]

$$\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0 \quad (4.23)$$

The columns, $\mathbf{u}_i$, of $\mathbf{U}$ are called the *left singular vectors*; the columns, $\mathbf{v}_i$, of $\mathbf{V}$ are called the *right singular vectors*; and the σ_i values are called the *singular values*. The main difference between the SVD and the eigendecomposition is that the SVD diagonalizes the matrix $\mathbf{A}$ with respect to two separate orthonormal bases $\{\mathbf{u}_i\}$ and $\{\mathbf{v}_i\}$, while the eigendecomposition diagonalizes $\mathbf{A}$ with respect to only a single orthonormal basis $\{\mathbf{v}_i\}$.

We use the SVD to analyze matrix multiplication in the same way as the eigendecomposition. $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^H$, so the first two steps $\mathbf{f} = \mathbf{\Sigma}\mathbf{V}^H \mathbf{x}$ are exactly the same as working with the eigendecomposition; we take the coordinates of $\mathbf{x}$ with respect to the $\{\mathbf{v}_i\}$ basis, and then scale each coordinate by its corresponding singular value σ_i . However, the last step, $\mathbf{y} = \mathbf{U}\mathbf{f}$, differs. Instead of mapping the scaled coordinates back to their original basis $\{\mathbf{v}_i\}$, they are mapped to a new basis $\{\mathbf{u}_i\}$ given by matrix $\mathbf{U}$. Therefore, the operation $\mathbf{A}\mathbf{x}$ is

accomplished by taking the coordinates of $\mathbf{x}$ with respect to the $\{\mathbf{v}_i\}$ basis, scaling these coordinates by their corresponding $\sigma_i \in \mathbf{\Sigma}$, and mapping these coordinates into the $\{\mathbf{u}_i\}$ basis.

This is the power of the SVD, that any linear transformation can be decomposed into a simple diagonal scaling as long as the correct orthonormal bases $\{\mathbf{u}_i\}$ and $\{\mathbf{v}_i\}$ are chosen. Each of the matrices $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{\Sigma}$, can be analyzed to give the precise behavior of any linear transformation and how it affects each individual component of its input. We take a look at some of this analysis in the next section.

4.3 Matrix Conditioning

The first way in which the SVD can be used to analyze inverse problems is by providing the matrix *condition number*. The condition number of a matrix $\mathbf{A}$ is defined as [13]

$$\text{Cond}(\mathbf{A}) = \frac{\sigma_1}{\sigma_n} \quad (4.24)$$

where σ_1 is the largest and σ_n is the smallest singular value of $\mathbf{A}$. Using the understanding from Section 4.2 in which the multiplication of a matrix is decomposed into separate parts, each dealing with a pair of singular vectors, σ_1 gives the largest “Amplification” of any singular vector pair, and σ_n gives the smallest amplification. The condition number σ_1/σ_n gives the discrepancy in amplification that occurs when multiplying by matrix $\mathbf{A}$. For the operation $\mathbf{Ax} = \mathbf{b}$, as the condition number of $\mathbf{A}$ increases, the disparity in resolution between singular vector pairs increases, and the output vector $\mathbf{b}$ becomes dominated by only a few singular vectors, the ones with the largest amplification factors. Furthermore, as the resolution of the smaller singular vectors becomes less and less, they get harder and harder to distinguish from the noise. When the conditioning gets even worse, these singular vectors essentially join the null space of $\mathbf{A}$, even though none of their singular values are exactly zero. This illustrates an important difference between theory and practice with linear systems. In theory, a linear system is either singular or non-singular with no gray area in between. In practice, matrices exist in a full spectrum of near and not-so-near singularity, and a matrix will become more and more “singular” as its condition number becomes larger and larger.

This notion of matrix conditioning is very important when inverting $\mathbf{A}$. For a singular value $\sigma_i \in \mathbf{A}$, the corresponding singular value in $\mathbf{A}^{-1}$ is $1/\sigma_i$, which means that $\mathbf{A}^{-1}$ will perform the inverse scaling on any given component. If $\mathbf{A}$ amplifies a component, $\mathbf{A}^{-1}$ will

suppress it, and if $\mathbf{A}$ suppresses a component, $\mathbf{A}^{-1}$ will amplify it. This becomes problematic when $\mathbf{A}$ has a large condition number since there exist components of $\mathbf{x}$ that $\mathbf{A}$ may nearly annihilate. If this is the case, $\mathbf{A}^{-1}$ will perform the inverse of near-annihilation and amplify these components of $\mathbf{b}$ by an obscene amount. In doing this, $\mathbf{A}^{-1}$ will also amplify all of the noise and other errors that live in these observation components, which in turn will dominate the recovered vector $\mathbf{x}$. If the conditioning of $\mathbf{A}$ is bad enough, directly computing $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ will produce nothing but noise, giving an answer that is completely useless.

This can be likened to the scenario in which we are at a pier and wish to estimate the length of a ship that is 500 yards away. We can estimate the ship's length by first holding up a ruler to note how long the ship appears from our point of view, and then scale that measurement out the 500 yards to get an estimate of the ship's true length. Common sense tells us that the best place to stand is with our line of sight perpendicular to the ship's length, which will give us the best resolution. However we could still estimate the ship's length if we observe it at 45 degrees from perpendicular. In this case we would also need to take into account how the viewing angle affects the perceived length and do the correct inverse operation after obtaining our measurement with the ruler. We could do the same thing if we were looking from a 60 degree angle too, but the inverse scaling would be more severe. As our viewing angle gets further and further from perpendicular, the inverse transformation becomes more and more severe, so any little error that we make with the ruler will have a larger and larger effect on our estimate of the ship's length. At an extreme angle such as 85 degrees from perpendicular, any little error in the measurement with our ruler could result in the estimated ship length deviating by hundreds of yards. This intuition encompasses exactly what happens with matrix conditioning. As the condition number of matrix $\mathbf{A}$ increases, any little error in the measurement $\mathbf{b}$ will effect the estimate of $\mathbf{x}$ by a greater and greater amount.

4.4 Analysis of the Singular Value Decay

In addition to the ratio between the first and last singular values of a matrix, the shape in which the singular values decay also plays an important role in how a matrix inversion should be treated. Poorly conditioned problems can be separated into two main types based on their pattern of singular value decay: *rank-deficient* problems and *discrete ill-posed* problems [14]. In a rank-deficient problem, the singular values decay gradually up until a certain point and then drop off sharply to very small values. Most of the useful information is contained

in the components that happen before the drop-off, and many inversion methods simply extract this information to create a new matrix of lower rank without all of the tiny singular values present in the original [14]. Because of this, we consider the number of singular values occurring before the drop-off to be the practical “rank” of the matrix.

On the other hand, discrete ill-posed problems do not have a defined drop-off in their singular values. Instead, the singular values decay fairly steadily toward zero [14]. Since there is no defined cutoff between which components contain useful information and which components contain noise, inversion algorithms for discrete ill-posed problems need to gradually decrease the influence of the data components as the singular values of A decrease, and in doing so strike a good balance between fitting the data and fitting our expectations of what the solution should be [14]. Discrete ill-posed problems can easily be created by discretizing a continuous ill-posed problem such as a Fredholm integral equation [14]. Since satellite-to-ground imaging geometry presented in Section 3.5 is a Fredholm integral equation, we can expect its matrix version to be a discrete ill-posed problem.

4.5 SVD Filtering

In Section 4.3, it was mentioned that for matrices with a large condition number, the small singular values can blow up in the matrix inversion, causing the recovered signal to be dominated by undesirable noise. One way to mitigate this problem is to modify the singular values of the inverse matrix to suppress this undesired noise amplification. This process is called *SVD filtering* and is a form of *regularization*, a process in which we take a poorly conditioned problem and modify it to stabilize its inversion [13]. For a poorly conditioned system $\mathbf{Ax} = \mathbf{b}$, the singular values of $\mathbf{A}^{-1}$ can be modified to create the *regularized inverse*, denoted as $\mathbf{A}^\#$. Note that changing the singular values of $\mathbf{A}^{-1}$ will obviously change the transformation itself. However, if the singular values are modified with care, $\mathbf{A}^\#\mathbf{b}$ will yield a better estimate of $\mathbf{x}$ than $\mathbf{A}^{-1}\mathbf{b}$, since $\mathbf{A}^\#$ has greatly reduced the effect of noise in $\mathbf{b}$ on the estimate of $\mathbf{x}$. Note that in the case of zero-noise, $\mathbf{A}^{-1}\mathbf{b} = \mathbf{x}$ but $\mathbf{A}^\#\mathbf{b} \neq \mathbf{x}$, due to the fact that $\mathbf{A}^\#$ is not the actual inverse of $\mathbf{A}$. Because of this non-equality, the expression $\mathbf{A}^\#\mathbf{b}$ is a *biased estimator* of $\mathbf{x}$. This illustrates a relationship in statistics called the *bias-variance trade-off*, which is the trade-off between over-fitting and under-fitting the data.

Two simple schemes for performing SVD filtering are known as *threshold* and *threshold to zero*. In the threshold method, we select a regularization parameter λ , and any singular value $1/\sigma_i$ in $\mathbf{A}^{-1}$ that is larger than λ is clipped to equal λ exactly [13]. Threshold to

zero, on the other hand, will take any singular value $1/\sigma_i$ that is larger than λ and set it equal to zero, completely removing that component from the inversion. Due to this, threshold to zero is also called *truncated singular value decomposition* (TSVD), and is a very popular method for approaching inverse problems. Since this completely removes the offending components from the inversion, the $\mathbf{A}^\#$ created by the threshold-to-zero method is a lower-rank approximation to $\mathbf{A}^{-1}$, and it is an optimal approximation to $\mathbf{A}^{-1}$ as guaranteed by the *Eckart-Young theorem* [13].

4.6 Tikhonov Regularization

A third method of SVD filtering modifies the singular values by a smooth function instead of using hard cutoffs. Let σ_i be the i th singular value of $\mathbf{A}$ and μ_i be the i th singular value of $\mathbf{A}^\#$. Then $\mu_i = g_\lambda(\sigma_i)$ [13], where

$$g_\lambda(\sigma) = \frac{\sigma}{\sigma^2 + \lambda} \quad (4.25)$$

If σ_i is much greater than $\sqrt{\lambda}$, the mapping in Equation (4.25) starts to look like $1/\sigma_i$, which is the corresponding singular value of $\mathbf{A}^{-1}$. In this case, the singular value is barely modified. If σ_i is much less than $\sqrt{\lambda}$, the mapping begins to look like σ/λ , in which $\mu_i \rightarrow 0$ as $\sigma_i \rightarrow 0$. In this case, the singular value is almost truncated, similar to what would happen in the threshold-to-zero method.

Given the smooth modification of the singular values given by Equation (4.25), this regularization problem can be given a *variational characterization*, that is, the regularized solution $\hat{\mathbf{x}}$ can be expressed as the optimum of some cost function. In this case, the cost function is [13]

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 + \lambda \|\mathbf{x}\|^2 \quad (4.26)$$

where λ is the same as in Equation (4.25). This method is called *Tikhonov regularization*, and is a classic way to approach inverting poorly conditioned inverse problems.

Intuitively, Tikhonov regularization can be thought as a “tug of war” between the residual $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2$ and the regularization term $\lambda \|\mathbf{x}\|^2$. The regularization term pulls evenly on all components of $\mathbf{x}$. However, components in $\mathbf{x}$ corresponding to larger singular values of $\mathbf{A}$ have a larger influence on the residual term, so the residual pulls much harder on them. Due to the harder pull of the residual term on these components, they will adhere very well to

the data. However, the other terms, which contain mostly noise, will get gobbled up by the regularization term and will be barely present in the regularized solution.

The variational characterization given in Equation (4.26) is great because it provides a much more efficient method for computing the regularized solution $\hat{\mathbf{x}}$ than performing a complete SVD on the matrix $\mathbf{A}$. Since both the residual and the regularization term are quadratic, the $\hat{\mathbf{x}}$ in Equation (4.26) can be computed by minimizing the augmented system [13]

$$\hat{\mathbf{x}} = \arg \min_x \left\| \begin{pmatrix} \mathbf{A} \\ \sqrt{\lambda} \mathbf{I} \end{pmatrix} \mathbf{x} - \begin{pmatrix} \mathbf{b} \\ \mathbf{0} \end{pmatrix} \right\|^2 \quad (4.27)$$

for which there exist many fast methods for computing exact and approximate minimizers. Because of this, a big advantage of Tikhonov regularization is that it can be computed quickly.

4.7 Tikhonov Regularization with Modified Norms

Tikhonov regularization can be modified by changing the norm used in its regularization term $\lambda \|\mathbf{x}\|^2$. For a matrix $\mathbf{W}$, such that $\mathbf{W}$ is symmetric and positive-definite, we can define a weighted norm $\|\mathbf{x}\|_{\mathbf{W}}$ as [13]

$$\|\mathbf{x}\|_{\mathbf{W}} = \sqrt{\mathbf{x}^T \mathbf{W} \mathbf{x}} = \|\mathbf{W}^{1/2} \mathbf{x}\|_2 \quad (4.28)$$

where $\mathbf{W}^{1/2}$ is the square root of $\mathbf{W}$, that is $(\mathbf{W}^{1/2})^T \mathbf{W}^{1/2} = \mathbf{W}$. To perform regularization, $\mathbf{W}$ can be tailored to penalize a specific characteristic of the solution $\mathbf{x}$ that is deemed undesirable.

Sometimes, we desire a weighting matrix $\mathbf{L}$ which is not symmetric positive-definite. In that case, $\mathbf{L}$ will not create a valid norm such as that presented in Equation (4.28), but we can use $\mathbf{L}$ to create a *seminorm* by [14]

$$\text{Seminorm}(\mathbf{x}) = \|\mathbf{L}\mathbf{x}\| \quad (4.29)$$

This may not be a true norm because $\|\mathbf{L}\mathbf{x}\|$ will equal zero whenever $\mathbf{x}$ is in the null-space of $\mathbf{L}$, and this violates the positive-definiteness property of a norm, thus garnering the name seminorm [14].

In imaging applications, a very popular choice of $\mathbf{L}$ is a discrete approximation to the

spatial derivative. In many cases, the image of interest contains mainly smooth textures, in which the spatial derivatives will have a relatively low magnitude. Choosing $\mathbf{L}$ to be the spatial derivative will favor these smooth, realistic images over rough images that may never arise in the real world. This notion of regularizing with respect to spatial derivatives will play a very large role in the construction of a good algorithm to invert the C/NOFS-CERTO image data.

4.8 Statistical Interpretation as MAP Estimation

So far, Tikhonov regularization has been presented as a variational problem, one in which the optimal solution $\hat{\mathbf{x}}$ is the optimal argument to some cost function. However, Tikhonov regularization can also be cast in a statistical framework (where it may be referred to as *ridge regression*[14]). Here, we cast Tikhonov regularization as *maximum a posteriori (MAP) estimation* of one random variable from another

Given two random variable $\mathbf{X}$ and $\mathbf{Y}$, the MAP estimate of $\mathbf{X}$ given a realization $\mathbf{y}$ of $\mathbf{Y}$ is

$$\hat{\mathbf{x}} = \arg \max_{\mathbf{x}} p(\mathbf{x}|\mathbf{y}) \quad (4.30)$$

where $p(\mathbf{x}|\mathbf{y})$ is the probability of $\mathbf{x}$ given $\mathbf{y}$, and this can be constructed from the known forward probability distribution $p(\mathbf{y}|\mathbf{x})$ via Bayes' rule

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{y})} \quad (4.31)$$

Given a realization $\mathbf{y}$ of random variable $\mathbf{Y}$, the denominator of Equation (4.31) does not depend on $\mathbf{x}$; therefore, Equation (4.30) can be rewritten as

$$\hat{\mathbf{x}} = \arg \max_{\mathbf{x}} p(\mathbf{y}|\mathbf{x})p(\mathbf{x}) \quad (4.32)$$

Every statistical estimation problem has a corresponding variational form in which the solution is the optimum of some deterministic function. It turns out that Tikhonov regularization (with or without seminorms) is just the variational form of MAP estimation when the prior on $\mathbf{X}$ is Gaussian, and $\mathbf{Y} = \mathbf{H}\mathbf{X} + \mathbf{W}$ where $\mathbf{W} \sim \mathcal{N}(0, \Sigma_{\mathbf{W}})$ and $\mathbf{H}$ is a deterministic $m \times n$ matrix.

The Gaussian prior on $\mathbf{X}$ is

$$p(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n |\Sigma_{\mathbf{X}}|}} e^{-\frac{1}{2} \|\mathbf{x} - \mu_{\mathbf{X}}\|_{\Sigma_{\mathbf{X}}^{-1}}^2} \quad (4.33)$$

and the conditional distribution of $\mathbf{Y}$ given $\mathbf{X}$ is

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^m |\Sigma_{\mathbf{W}}|}} e^{-\frac{1}{2} \|\mathbf{y} - \mathbf{H}\mathbf{x}\|_{\Sigma_{\mathbf{W}}^{-1}}^2} \quad (4.34)$$

Since $\ln(\cdot)$ is monotonically increasing in its argument, the MAP estimator in Equation (4.32) is maximized if we maximize its log

$$\begin{aligned} \hat{\mathbf{x}} &= \arg \max_{\mathbf{x}} p(\mathbf{y}|\mathbf{x}) p(\mathbf{x}) \\ &= \arg \max_{\mathbf{x}} [\ln p(\mathbf{y}|\mathbf{x}) + \ln p(\mathbf{x})] \\ &= \arg \min_{\mathbf{x}} [-\ln p(\mathbf{y}|\mathbf{x}) - \ln p(\mathbf{x})] \\ &= \arg \min_{\mathbf{x}} \|\mathbf{y} - \mathbf{H}\mathbf{x}\|_{\Sigma_{\mathbf{W}}^{-1}}^2 + \|\mathbf{x} - \mu_{\mathbf{X}}\|_{\Sigma_{\mathbf{X}}^{-1}}^2 \end{aligned} \quad (4.35)$$

where the last step is accomplished by removing any terms that do not include $\mathbf{x}$ since they do not affect the solution.

There are many tools available for analyzing inverse problems and estimators that rely on a statistical framework, so using any of these tools requires us to use the MAP estimator definition of regularization. For non-Tikhonov regularization schemes, the statistical interpretation can also be derived, but the form may not be as pretty.

4.9 Total-Variation (TV) Regularization

The general form of Tikhonov regularization, where $\hat{\mathbf{x}} = \arg \min_{\mathbf{x}} (\text{residual} + \text{regularization term})$, opens the door to a very general framework of regularization strategies that each use an additive penalty term to bias the estimate in a desired way. The general form of these regularization strategies is

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x}} J_0(\mathbf{b} - \mathbf{A}\mathbf{x}) + \sum_{k=1}^n \lambda_n J_n(L_k(\mathbf{x} - \mathbf{z}_k)) \quad (4.36)$$

where $J_0(\cdot), J_1(\cdot), \dots, J_n(\cdot) : \mathbb{R}^m \mapsto \mathbb{R}^+$ are weighting functions, and $L_1, L_2, \dots, L_n : \mathbb{R}^n \mapsto \mathbb{R}^m$ are linear transformations. All of these methods minimize a cost function combining the residual term $J_0(\mathbf{b} - \mathbf{A}\mathbf{x})$ with any number of regularization terms $J_k(L_k(\mathbf{x} - \mathbf{z}_k))$ to get the desired solution.

A popular method is to let $L = \mathbf{I}_n$, $\mathbf{z} = \mathbf{0}$, and $J_1(\cdot) = \|\cdot\|_1$ which creates *total variation* (TV), or one-norm regularization. In one of its forms, total-variation regularization solves the equation [15]

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x}} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda \|\mathbf{L}\mathbf{x}\|_1 \quad (4.37)$$

Total-variation regularization has become popular in part for the following reasons: First, it penalizes large values in the regularization term $\mathbf{L}\mathbf{x}$ less heavily than the quadratic penalty used by Tikhonov regularization. As mentioned before, many imaging applications let $\mathbf{L}$ be the spatial derivative operator, and in doing so they favor smooth images over rough ones. However, there are cases in which the desired image is piecewise-smooth, smooth in local regions punctuated by fairly sharp discontinuities. TV regularization penalizes these sharp discontinuities less harshly than Tikhonov regularization, so it will allow the reconstructed image to form more edges [15]. This will be discussed in more detail in Section 4.10. Another reason that TV regularization has become popular is because the 1-norm has been shown to be a good approximation of the 0-norm, or sparsity norm [16]. *Sparse recovery*, also referred to as *compressive sensing* in recent literature, deals with inverse problems in which the desired solution is sparse, that is, the solution has a low 0-norm. TV regularization has proven useful in some of these applications.

Since TV regularization has at least one term that is not quadratic, the solution to the TV regularization problem is no longer a least-squares problem, so it must be solved by other methods. The optimization in Equation (4.37) is still convex; however, it is not differentiable at the point where $\lambda \|\mathbf{L}\mathbf{x}\|_1 = 0$. This may create trouble for certain optimization algorithms, so one method is to approximate the 1-norm by a smooth function where [15]

$$|w| \approx \sqrt{w^2 + \beta} \quad (4.38)$$

for a small value of β . This approximation can then be solved using a variety of convex optimization solvers.

A second method for computing the TV regularized solution switches Equation (4.37) for

the modified equation [15]

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x}} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_1 + \lambda \|\mathbf{L}\mathbf{x}\|_1 \quad (4.39)$$

Since Equation (4.39) uses the 1-norm exclusively, it can be rewritten as minimizing the augmented system

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x}} \left\| \begin{pmatrix} \mathbf{A} \\ \lambda \mathbf{L} \end{pmatrix} \mathbf{x} - \begin{pmatrix} \mathbf{b} \\ \mathbf{0} \end{pmatrix} \right\|_1 \quad (4.40)$$

which turns into the form $\hat{\mathbf{x}} = \arg \min_{\mathbf{x}} \|\mathbf{p} - \mathbf{G}\mathbf{x}\|_1$. A computationally efficient method for computing the minimizer to Equation (4.40) is called *iteratively re-weighted least-squares* (IRLS), and computes $\hat{\mathbf{x}}$ by solving a sequence of standard least-squares problems [15]. The main benefit of this method is that it does not need to employ any sort of general nonlinear optimization technique since all steps in computing the solution involve solving standard least-squares problems, which have a closed-form solution. This process can be sped up even more by approximating the solutions to each of the least-squares problems using an iterative solver. In my experience, this method is a very good way to solve TV regularization problems.

4.10 Edge-Preserving Regularization

Any reconstruction scheme that regularizes with respect to spatial derivatives favors smooth images over rough ones, so it will suppress the formation of sharp gradients in the output image. As mentioned in Section 4.9, there are times in which the desired image output behaves like a piecewise constant function. That is, the image is relatively smooth in most regions, but these regions of smoothness are punctuated by occasional sharp boundaries.

Consider reconstructing an image such as this using a Tikhonov regularization method that penalizes the spatial gradient of the reconstructed image. Since the penalty term $\|\mathbf{L}\mathbf{x}\|^2$ is quadratic in the magnitude of the derivative, any regions with a high gradient magnitude will be heavily penalized, much more so than those with low gradient magnitudes. Because of this disparity, Tikhonov regularization will inhibit the formation of edges, smearing a sharp boundary out over a large area. Plainly stated, if there are any sharp boundaries in the original image, Tikhonov regularization will destroy them. To counteract this problem, we need to choose a regularization scheme that penalizes large gradients less harshly than Tikhonov regularization. Section 4.9 mentioned that TV regularization does a good job at

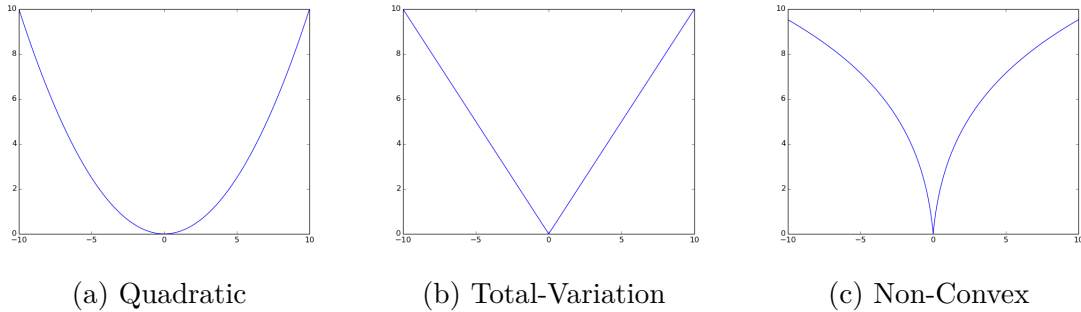


Figure 4.1: Cost Function Varieties

preserving edges, and it does better than Tikhonov regularization because its penalty term only scales linearly with the magnitude of the gradient instead of quadratically. Because of this, we can expect TV regularization to allow some sharp edges to form in the reconstructed image.

What if TV regularization still is not good enough at edge-preservation, and we desire something that penalizes large gradients even less? This problem becomes more difficult because the l_1 norm used by TV regularization is the smallest l -norm that is still convex. This means that any l -norm that reduces the penalty disparity even more between large and small gradient magnitudes will be non-convex, and this makes the optimization calculations much more difficult. Figure 4.1 depicts the cost functions used by Tikhonov regularization, TV regularization, and a non-convex regularization method. Note how the scaling of the penalty term becomes less severe as the level of edge-preserving utility increases.

Despite the woes that arise when optimizing non-convex functions, there has been research done on this exact problem of non-convex regularization, and one such method to approach it is presented next.

4.11 Non-Convex Regularization with Iterative Relaxation

In Section 4.10, we noted how useful it would be to perform regularization with non-convex norms. Instead of using pure l -norms, we show how to perform regularization with a family of non-convex weighting functions that provide the same utility as a non-convex l -norm.

[17] presents an algorithm that is able to perform regularization with non-convex weighting functions as long as the functions satisfy certain regularity conditions. The functions that

we choose to use are a family of $\rho(\cdot)$ functions defined as [17]

$$\rho_{q,T}(t) = \frac{1}{1 + \left(\frac{t}{T}\right)^q} \quad (4.41)$$

where q and T are parameters that control the shape of Equation (4.41). These $\rho(\cdot)$ functions can either be convex or non-convex depending on the choice of parameters. For $0 < q \leq 0.5$, Equation (4.41) is convex, and for $q > 0.5$ it is non-convex. The parameter T behaves much like a horizontal scaling parameter, a smaller T will compress the function horizontally, and a larger T will stretch it horizontally.

These $\rho(\cdot)$ functions are used to define the regularization weighting function $\sigma(t)$ where

$$\rho(t) = \frac{\partial \sigma(t)}{\partial t} \quad (4.42)$$

For $q > 0.5$, $\rho(x)$ is a decreasing function in x ; therefore, its integral $\sigma(x)$ will grow less and less rapidly as x gets larger. An example of a possible $\sigma(x)$ is given in the “Non-Convex” pane of Figure 4.1. $\sigma(x)$ is then used to perform regularization, and it will provide more edge preserving utility than TV regularization because it penalizes large gradient magnitudes even less than TV does.

Taking the general form of regularization given in Equation (4.36) and tweaking it to fit these new $\sigma(\cdot)$ functions yields a cost function $J(\cdot)$ where

$$J(\mathbf{f}) = Q(\mathbf{f}) + \lambda \sum_{m=0}^{M-1} \sigma[V_m(\mathbf{f})] \quad (4.43)$$

For the optimization algorithm to work, the pieces of Equation (4.43) need to satisfy the following regularity conditions: $Q, V_m : \mathbb{R}^n \mapsto [0, \infty)$, for $m \in \{0, 1, \dots, M\}$ and each of them are convex functionals that are continuously differentiable. $\sigma(\cdot) : [0, \infty) \mapsto [0, \infty)$ is a continuously twice-differentiable, strictly concave function where $\sigma(0) = 0$, $\sigma'(0) = 1$, and $0 < \sigma'(t) \leq 1$ [17].

Given that these conditions are met, the global minimum of $J(\cdot)$ can be found by iteratively minimizing a sequence of local cost functions $J_0(\cdot)$ whose minima gradually approach the global minimum of $J(\cdot)$.

The local $J_0(\cdot)$ functions are given by[17]

$$J_0(\mathbf{f}, \mathbf{e}) = Q(\mathbf{f}) + \lambda \sum_{m=0}^{M-1} \mathbf{e}_m V_m(\mathbf{f}) \quad (4.44)$$

where $\mathbf{e}$ is a vector of weighting values. Note that when $\mathbf{e}$ is held constant, Equation (4.44) is convex in $\mathbf{f}$.

So far, computing the optimum of $J(\cdot)$ involves sequentially minimizing the $J_0(\cdot)$ functions given by Equation (4.44), where each $J_0(\cdot)$ function contains a different weighting vector $\mathbf{e}$. Furthermore, the minimum $\hat{\mathbf{f}}_k$ of the current function $J_{0,k}(\cdot)$ is used to compute the weighting vector $\mathbf{e}$ for the next function $J_{0,k+1}(\cdot)$.

To compute the next iteration's $\mathbf{e}$ vector, we define the set of scalar functions $E : \mathbb{R}^n \mapsto (0, 1]^M$ where [17]

$$E_m(\mathbf{f}) = \rho[V_m(\mathbf{f})], \forall m \in \{0, 1, \dots, M-1\} \quad (4.45)$$

These $E_m(\mathbf{f})$ functions are then applied to the current $J_0(\cdot)$ minimizer, $\hat{\mathbf{f}}_k$, to generate the weighting vector $\mathbf{e}$ for the next iteration. Intuitively, it makes sense to think of the regularization term in cost function $J_{0,k+1}(\cdot)$ as a convex approximation of the true regularization term, $\lambda \sum_{m=0}^{M-1} \sigma[V_m(\mathbf{f})]$, taken at point $\hat{\mathbf{f}}_k$.

The complete algorithm proceeds as follows: After initializing $\mathbf{f}^{(0)}$ to be the initial guess, the algorithm iterates between

$$\begin{aligned} \mathbf{e}_m^{(k)} &\leftarrow E_m(\mathbf{f}^{(k)}) \quad \forall m \\ \mathbf{f}^{(k+1)} &\leftarrow \arg \min_{\mathbf{f}} J_0(\mathbf{f}, \mathbf{e}^{(k)}) \end{aligned} \quad (4.46)$$

until a suitable stopping point $\mathbf{f}^*$ is found.

[17] then proves that this algorithm is globally convergent given that all of the regularity conditions are satisfied.

4.12 Iterative Algorithms

In the past sections, all of the regularization methods presented have one thing in common. They each augment the residual term with a set of regularization terms to generate a cost function, which can be minimized to produce the estimate using some sort of closed-form or iterative optimization technique. A different class of methods called *iterative reconstruction*

algorithms skip forming the cost function and proceed directly to iteratively computing the estimate using just the residual.

In the first iterations, these algorithms will resolve coarse structures in the image, and they will continue to resolve finer and finer structures as the iterations progress. To regularize with these algorithms they are simply stopped before convergence. Ideally, we stop the iteration after all of the essential coarse-grained structures have been formed, but before the algorithm fits the image to all of the fine structures due to noise and inconsistencies in the data.

One of the most popular iterative algorithms in imaging is the *algebraic reconstruction technique* (ART), which is based on a classic method called *Kaczmarz's method*. Kaczmarz's Method and ART are called *row-action methods* because they compute updates to the current guess $\mathbf{x}_k$ only using one row of the model matrix at a time. To solve the system $\mathbf{Ax} = \mathbf{b}$, Kaczmarz's method uses a fixed-point iteration, and its update equation is [14]

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \frac{\mathbf{b}_i - \mathbf{a}_i^T \mathbf{x}^{(k)}}{\|\mathbf{a}_i\|_2^2} \mathbf{a}_i \quad (4.47)$$

Kaczmarz's method cycles through each row $\mathbf{a}_i$ of matrix $\mathbf{A}$ and projects $\mathbf{x}^{(k)}$ onto the hyperplane in $\mathbb{R}^n$ which satisfies $\mathbf{a}_i^T \mathbf{x} = \mathbf{b}_i$. It can be shown that the guess will then work its way towards the intersection of all the hyperplanes, which is the solution to $\mathbf{Ax} = \mathbf{b}$. If $\mathbf{Ax} = \mathbf{b}$ is under-determined, then the algorithm will converge to the solution of $\mathbf{Ax} = \mathbf{b}$ that is closest to the initial guess, $\mathbf{x}^{(0)}$ [15].

Another row-action technique, the *multiplicative algebraic reconstruction technique* (MART), operates in a very similar way to Kaczmarz's method, but it uses multiplicative updates instead of additive updates. The general form of the MART algorithm iteration is [18]

$$\mathbf{x}_j^{(k+1)} = \mathbf{x}_j^{(k)} \left(\frac{\mathbf{b}_i}{\mathbf{a}_i^T \mathbf{x}^{(k)}} \right)^{\gamma_j \delta_i \mathbf{A}_{ij}} \quad (4.48)$$

For MART, the update formula in Equation (4.48) is defined individually for each element $\mathbf{x}_j$ of vector $\mathbf{x}$. Recall that in the ART algorithm, each step calculates the additive residual $\mathbf{b}_i - \mathbf{a}_i^T \mathbf{x}^{(k)}$ corresponding to a matrix row $\mathbf{a}_i$ and then adds it to the current guess $\mathbf{x}^{(k)}$ to create the new guess $\mathbf{x}^{(k+1)}$. In MART, the multiplicative residual, $\mathbf{r}_{ij} = \frac{\mathbf{b}_i}{\mathbf{a}_i^T \mathbf{x}^{(k)}}$, corresponding to matrix row $\mathbf{a}_i$ is calculated, exponentially scaled by the j -th element in matrix row $\mathbf{a}_i$, scaled by a damping factor $\gamma_j \delta_i$, and then multiplied into the corresponding entry of $\mathbf{x}^{(k)}$.

This update is given by

$$\mathbf{x}_j^{(k+1)} = \mathbf{x}_j^{(k)} \mathbf{r}_{ij}^{\gamma_j \delta_i \mathbf{A}_{ij}} \quad (4.49)$$

Unlike ART, the convergence properties of MART are a little more subtle and use information theoretic measures such as *Shannon entropy* and *Kullback-Leibler (KL) divergence*. More information on the convergence of MART can be found in [18].

Recently, a researcher at the *Naval Research Laboratory* (NRL) used the MART algorithm to reconstruct two images from data collected with the C/NOFS-CERTO system, and the results look good [19]. This goes to show that the MART algorithm is something worth considering when working with this and other similar datasets.

A big advantage of iterative algorithms is their simplicity. There are very few parameters to tune, so they provide a great vehicle for rapidly characterizing some of the behavior of a particular inverse problem. This lack of tunable parameters can also be a disadvantage since we have less control over the exact form of regularization that is used. ART reconstruction has had great success in tomography applications, but it may converge very slowly when used for inverse problems that do not come from tomography [14].

4.13 Parameter Choice Methods

Each of these regularization procedures relies on a parameter λ which is used to shift the balance of power between the residual term and the regularization term. In statistical parlance, the precise choice of λ determines where the given estimate will lie with respect to the bias-variance trade-off. A lower λ value will pull the solution closer to the exact prescription of the data; however, in doing so it may over-fit the data with all of its noise and defects. Because of this, a lower λ value will decrease the bias, but increase the variance of the estimate. For higher λ , the algorithm will shift more weight onto the regularization term and care less about strictly adhering to the data. Setting λ too high may over-smooth the data and remove meaningful information from the image. Because of this, a higher λ reduces the variance of the estimate, but at the expense of adding more bias.

Constructing the best solution to an inverse problem involves choosing a λ value that optimizes the balance between data-fit and regularization. Methods for finding optimal λ values are still a very active and interesting area of research in the engineering and statistics communities. This section describes two popular parameter choice methods.

The first of these is *L-curve analysis*. When performing an inversion, if the regularization

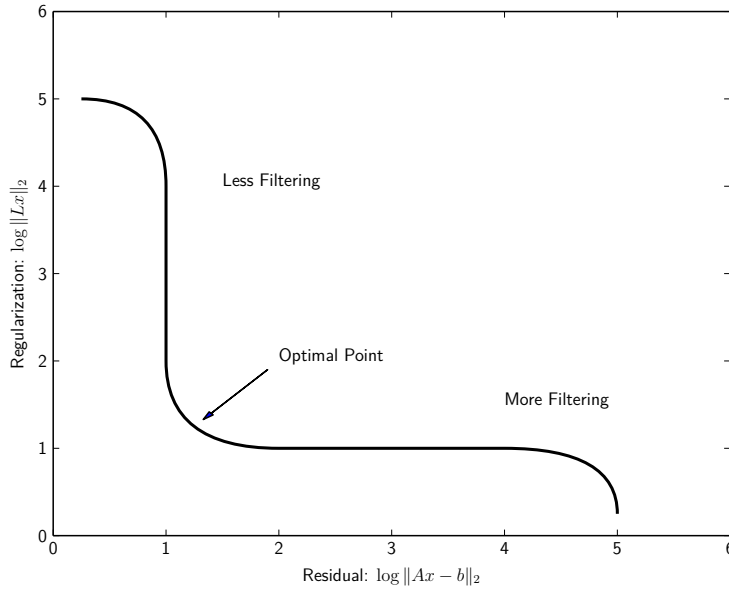


Figure 4.2: Example L-Curve

term $\|\mathbf{L}\mathbf{x}\|_2$ is very large, it is an indicator that the solution may be over-fitting the data. On the other hand, if the residual magnitude $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2$ is very large, it is an indicator that the solution may be over-smoothing the data. Given the true solution and a set of measurements, we can compute multiple solutions with different levels of regularization and trace out a curve comparing the regularization term and the residual magnitude for all tested values of λ . For the family of discrete ill-posed problems, of which the C/NOFS-CERTO imaging is a member, the plot of $\log \|\mathbf{L}\mathbf{x}\|_2$ vs. $\log \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2$ will often have an L-shape and is referred to as the *L-curve* [14]. An example of a possible L-curve plot is shown in Figure 4.2. Both conventional wisdom and empirical studies say that the optimal λ parameter corresponds to the point in directly in the elbow of the “L”, since it provides the greatest balance between regularization error and the residual error [14]. The biggest drawback of the L-curve method is that it requires use of the ground-truth solution, which is not very helpful since it is unavailable in real imaging problems. The way to get around this is to perform L-curve analysis and parameter tuning on simulated data. If the simulated data accurately models the actual system, then the optimal L-curve parameter taken from the simulation may be a very good choice for reconstruction with the actual data. For the particular case of Tikhonov regularization, other properties of the L-curve have been proven, and some of these can be found in [14].

In contrast to the L-curve criterion, the next method for parameter selection does not need access to the ground-truth solution. Instead, it uses the notion of predictive risk. The motivating intuition for this is that a well-chosen regularization parameter should be able to predict any piece of missing data. Consider the case in which a single data-point $\mathbf{b}_i$ is intentionally left out of the observation vector $\mathbf{b}$, the system is inverted with regularization parameter λ to generate solution $\mathbf{x}_\lambda$, and then $\mathbf{x}_\lambda$ is fed forward through the model to generate a guess $\mathbf{b}_{\lambda,i}$ of the missing data-point $\mathbf{b}_i$.

Using the notion of predictive risk, the goal is to minimize the average error between each of the actual data values $\mathbf{b}_i$ and their guesses $\mathbf{b}_{\lambda,i}$. Under the assumption that the choice of i is uniformly distributed on $[0, M - 1]$, we desire to choose the regularization parameter that minimizes the mean-square predictive risk given by

$$\hat{\lambda} = \arg \min_{\lambda} \mathbb{E}_i[(\mathbf{b}_i - \mathbf{b}_{\lambda,i})^2] \quad (4.50)$$

The *cross-validation* method directly estimates the optimal λ value for m -dimensional vector $\mathbf{b}$ via the equation [15].

$$\hat{\lambda} = \arg \min_{\lambda} \frac{1}{m} \sum_{k=1}^m ([\mathbf{A}\mathbf{x}_\lambda]_k - \mathbf{b}_k)^2 \quad (4.51)$$

A big drawback of using the cross-validation procedure presented in Equation (4.51) is that for every value of λ , the method must compute an individual solution $\forall m \in [0, M - 1]$, and this can take a long time if the data vector is large.

A more computationally tractable version is called *generalized cross-validation* (GCV). In GCV, only one, not M computations need to be performed for each choice of λ . Here, the optimal λ value is chosen to minimize the generalized cross-validation function given by [14].

$$\hat{\lambda} = \arg \min_{\lambda} \frac{\|\mathbf{A}\mathbf{x}_\lambda - \mathbf{b}\|_2^2}{\text{Tr}(\mathbf{I}_m - \mathbf{A}\mathbf{A}_\lambda^\#)^2} \quad (4.52)$$

A drawback of GCV is that it assumes the regularized inverse $\mathbf{A}_\lambda^\#$ exists in closed form. This is true for Tikhonov regularization, but not for most other regularization methods. Other forms of GCV have been proposed to deal with some of these other cases, and a brief introduction to some of these can be found in [14].

Chapter 5

Image Reconstruction with C/NOFS-CERTO Data

Chapter 4 presents a set of techniques that are useful in dealing with general linear inversion problems, many of which will play a part in inverting the C/NOFS-CERTO data. This next chapter will present some more techniques that are specifically needed for this particular inverse problem, and it will then combine them with ideas presented in Chapter 4 to create a complete algorithm capable of generating images from the C/NOFS-CERTO data-set.

Another topic presented in this chapter is how to deal with the actual C/NOFS-CERTO data. As opposed to simulated data, real-world data is very far from ideal: the limited-angle nature of this data is even more severe than initially hoped, with less than ten percent angular coverage of any given voxel. There is also a lot of missing data due to technical problems encountered in the data gathering. This data is also subject to noise that is not white nor Gaussian. All of these facts can make working with real data a fairly challenging task. An additional problem encountered when working with this data is that the process of inversion is far from a controlled experiment; we are unable to load a test case into the ionosphere, image it, and tweak variables to quantify each of their effects. Instead, we can only take measurements of what is already happening in the ionosphere, which will involve many unknown variables and variable interactions. In addition, the inversion process will also introduce many more variables that may interact with some of the physical variables already present. To state it again, this is not a controlled experiment, and because of that, we need to analyze the generated images with a discerning eye. Some structures that appear in reconstructed images may be due to physical features in the ionosphere, and others may be completely artificial, added in by the regularization process. In this case, there are no good quantitative indicators as to which camp a structure is a part of. To deal with all of this, we have to rely on human judgment and intuition to separate the real structures from the fake. To be able to make this judgment, we need to develop a good intuitive understanding of the data, and its structure and limitations. It is also important to develop a deep understanding of the inversion methods used. Every inversion method will introduce a unique set of artifacts into the image reconstruction, so it is important to understand these

artifacts and be able to identify them when they appear.

Despite the obvious difficulties, analysis of datasets such as this are very useful and have provided good knowledge into the workings of the ionosphere. Data sets such as this may also exhibit new structures or phenomena in the ionosphere that other researchers may have not considered.

5.1 Satellite Tracking

The first necessary ingredient for interpreting the data is the ability to track the C/NOFS satellite through its orbit and reconstruct its position at any given date or time. The *North American Aerospace Defense Command* (NORAD) assigns satellites a unique identifier called a *two-line element* (TLE) [20]. These identifiers contain information about the satellite's orbit at a specific time, from which the orbit can be extrapolated to calculate the satellite's position at a future date and time. Due to imperfect orbit modelling and other physical happenings, these TLE identifiers will be updated periodically, possibly multiple times a day, to fix any errors that creep in [20]. Two other things that are needed to track the C/NOFS satellite are an Earth gravity model and a satellite orbit model. For the Earth gravity model I use WGS84 and for the satellite orbit model I use SGP4. Information about both of these can easily be found on the Web. For performing the orbit calculations, I use a computer library called PyEphem, which at the time of writing is freely available on the Python Package Index (PyPI). Calculating a satellite orbit is a fairly common procedure, so in addition to PyEphem, there are many other computer libraries that will give similar functionality.

5.2 Data-Stream Analysis

Each satellite pass recorded by the receivers generates a data stream covering approximately 11 minutes worth of observations. This data stream is then used by the reconstruction algorithm to reconstruct a single image.

Due to inherent 2π phase ambiguities that are present in any phase-comparison algorithm [12], the imaging data can only be reported as a relative quantity. Because of this, each receiver records the current measured TEC relative to all other measurements it has taken during the observation. It then reports all of its measurements as *relative TEC* (RTEC),

relative to an unknown additive offset. To compute the absolute TEC we need to estimate what the unknown offset should be and then add each of these reported RTEC measurements to that estimate.

The measurements are reported in custom units, called TEC units (TECU), which are equal to $10^{16} \frac{\text{el}}{\text{m}^2}$. To intuitively motivate these units, the actual density is in $\frac{\text{el}}{\text{m}^3}$, but the line integral integrates over one of the m^{-3} dimensions, leaving the units to be in terms of $\frac{\text{el}}{\text{m}^2}$.

The C/NOFS-CERTO data set contains observations scattered over a four-year span, from 2009 to late 2012, and each observation contains anywhere from one to four receivers. For practical reasons, this thesis only looks at observations that contain three or four receivers. The data set contains many three-receiver observations which are spread over the entire four year window, and there are many fewer observations with all four receivers active, with almost all of them occurring in 2012.

Figure 5.1 shows what a typical data-stream looks like. The first thing to notice is that only one receiver reports data from every point in time. The pragmatics of building and administering a distributed data-collection system on the other side of the world are extremely difficult, so holes in the data streams are to be expected. The extreme example of missing data is shown by the receiver at Ayacucho, which contains only one short burst of noisy data near the beginning of the window. Most of the four-receiver measurements contain a stream from Ayacucho that looks like this, so we might guess that the receiver has had some sort of internal failure and will most likely require maintenance to get running at full capacity again. Because of this receiver problem, many of the four-receiver observations gathered in 2012 are effectively three-receiver datasets and this needs to be kept in mind when analyzing the reconstructed images.

Many of the reported data-streams from a single receiver have a “Cup” shape in which higher readings appear at the edges of the window, and the RTEC gradually decreases to the center of the window. Many streams report a value near zero in the center, and this is not a reading of 0 TEC. Instead, it is where the observed TEC is equal to the TEC offset for that receiver, that is, the reported RTEC is zero.

Typical measurement orders of magnitude are as follows: the electron density in the grid usually achieves its maximum value somewhere near $10^{12} \frac{\text{el}}{\text{m}^3}$, and the TEC measurements will combine these densities to create measurements on the order of $10^{16} \frac{\text{el}}{\text{m}^2}$.

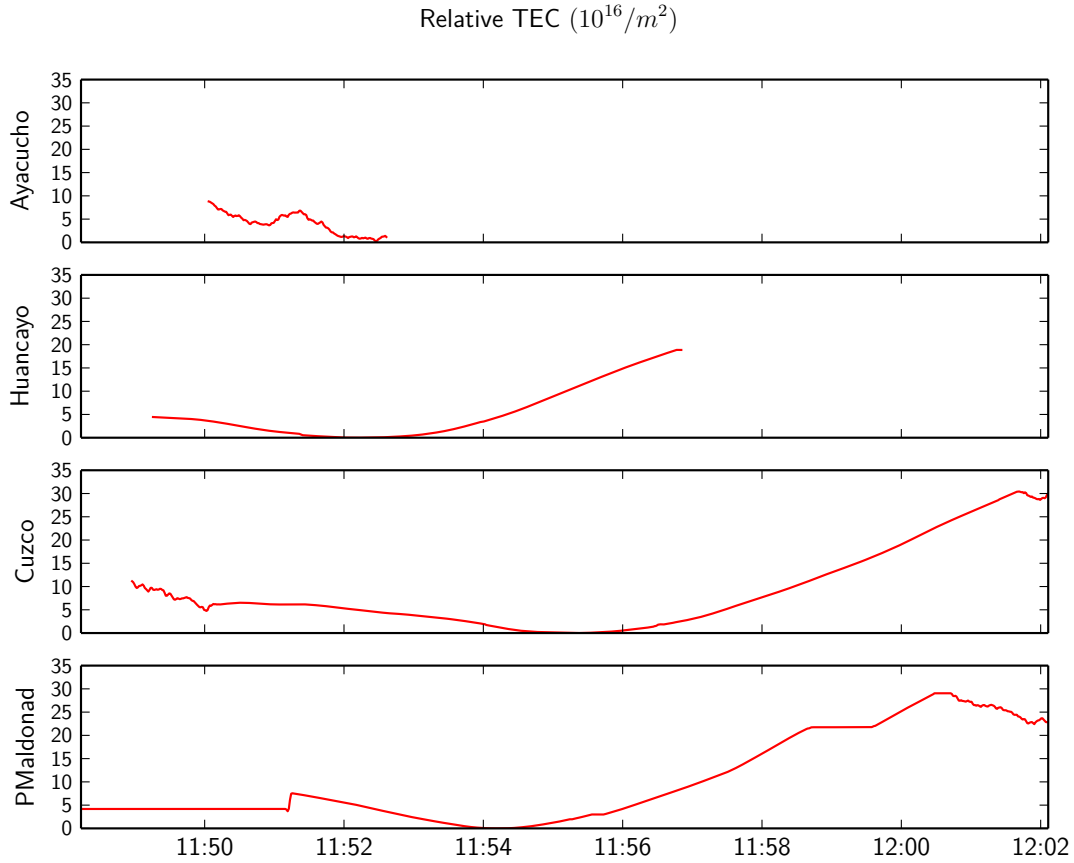


Figure 5.1: Sample Data Stream (11 Minute Window on 6/30/2012)

5.3 Grid Design

Section 3.5 discusses the fact that discretization of the ionosphere’s electron density is necessary to perform image reconstruction, and that this is done by approximating the continuously varying electron density with a finite set of basis functions. The set of basis functions that we choose to use is a grid of polar “squares”. These polar squares are regions whose bottom and top are given by two lines of constant altitude, and whose sides are given by two lines of constant longitude. They are squares in the longitude vs. altitude coordinate space, but after being transformed back to the standard Euclidean they are no longer completely square, but close enough for practical purposes. Also note that the polar squares will no longer be uniformly sized after their transformation back to Euclidean space, since a square at higher altitude will be larger than one that is at lower altitude. One of these squares

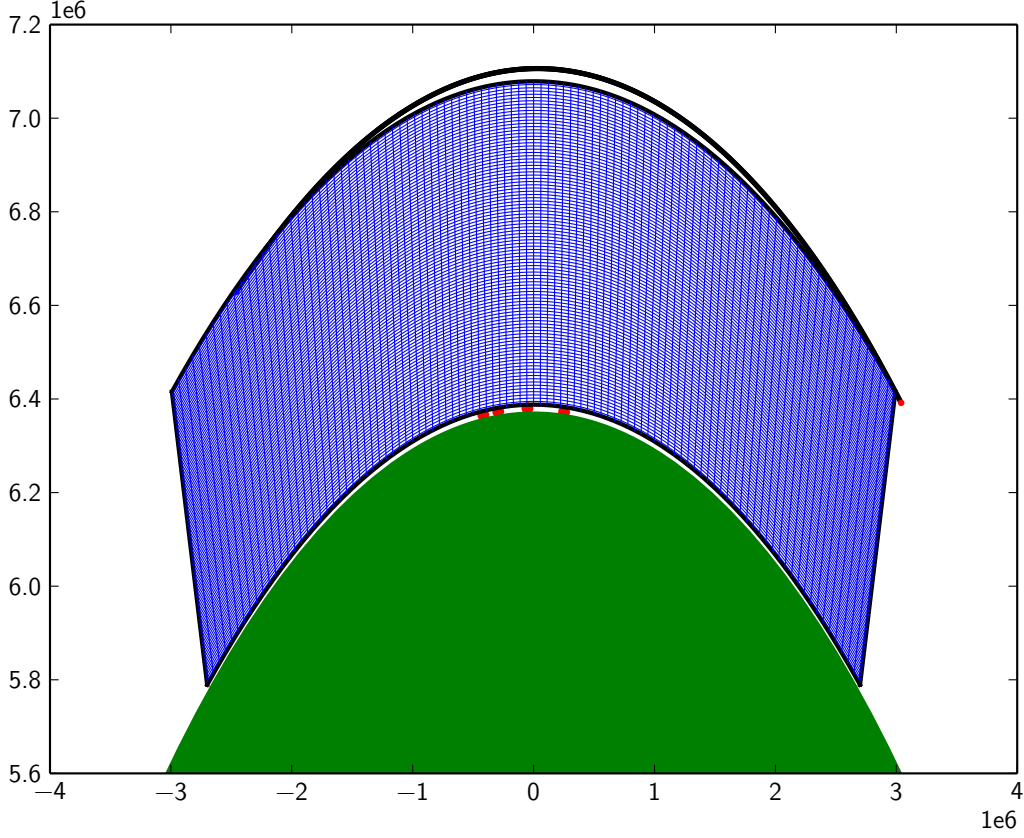


Figure 5.2: Full Grid Before Pruning

is known as a *voxel*, which is simply an n -dimensional generalization of a pixel. This grid geometry is shown in Figure 5.2

With this grid set up, we are now able to specify the exact geometry of the forward problem. Given the $m \times n$ voxel grid G , each voxel in the grid is assumed to have uniform electron density throughout. Every voxel in grid G is given a lexicographic ordering, and in doing so the grid G can be written as an $m \times n$ dimensional vector $\mathbf{g}$, in which the element $\mathbf{g}_i$ contains the electron density contained in voxel i of the ordering. As discussed in Chapter 3, the entire observation operation is linear, so it can be represented by a single model matrix $\mathbf{H}$, which maps from the space of electron density vectors to the space of observation vectors. Let the observation vector $\mathbf{b}$ contain the observations from all receivers at all available times, lexicographically ordered just as with the voxels in the grid. With this formalism, the entire

observation equation reduces to a matrix multiplication.

$$\mathbf{b} = \mathbf{H}\mathbf{g} \quad (5.1)$$

To build a grid to use the real data, the first thing that is needed is to fix a coordinate system. The natural choice for this setup is Earth-Centered, Earth-Fixed (ECEF) coordinates, which are Cartesian (x, y, z) coordinates that are fixed in their position relative to the earth. In ECEF coordinates, the origin is given by the center of the earth, the x axis passes through the equator at 0° longitude, the y axis passes through the equator at $+90^\circ$ longitude, and the z axis passes through the North Pole [21].

Another coordinate system commonly used for satellites is geodetic coordinates, which are indexed by latitude, longitude, and altitude. Geodetic coordinates approximate the earth as a giant ellipsoid, and contain the latitude, longitude, and altitude of the satellite with respect to this ellipsoidal approximation. Converting geodetic coordinates to ECEF requires the use of a very simple formula which can be found in [21].

Next, we present how to use this grid to programmatically generate a forward model to be used for the inversion. The program needs two things to work, a parameterized description of the imaging grid and the location of the satellite and corresponding receiver for every data point present in the observation. The program will then iterate through each data point in the observation, and use the corresponding receiver/satellite position pair to generate the appropriate row of the observation matrix $\mathbf{H}$. It does this as follows: First, we generate a straight line connecting the receiver and satellite of the form parameterized by value t .

$$\mathbf{x}(t) = \mathbf{x}_{\text{recv}} + t(\mathbf{x}_{\text{sat}} - \mathbf{x}_{\text{recv}}) \quad (5.2)$$

Next, all intercepts of the ray path with the elevation grid lines are computed. For this calculation, the earth is approximated by a sphere, which is a reasonable approximation since the grid only lives over a small portion of the earth. Elevation intercepts can be found by simply equating the Euclidean norm of the parameterized line position $\mathbf{x}(t)$ with the elevation of each grid-line e_i .

$$\|\mathbf{x}(t)\| = e_i \quad (5.3)$$

Next, the intercepts of the ray path with each of the longitude grid-lines are computed. To do this, planes are constructed that radiate out from the center of the earth along each of the longitudinal grid lines. Each of these planes can be parameterized by a normal vector $\mathbf{n}_i$, and line $\mathbf{x}(t)$ intersects the longitude plane whenever its coordinates are orthogonal to

the normal vector $\mathbf{n}_i$.

$$\mathbf{x}^T(t)\mathbf{n}_i = 0 \quad (5.4)$$

For a detailed explanation of how these calculations can be implemented, refer to [3].

After the elevation and longitudinal intercepts are calculated, the algorithm needs to decide whether this specific observation is valid or not. In most tomography applications, regions outside the imaging grid have no effect on the measurement. However, that is not the case with the ionosphere since ionization does not just stop in a region because we have decided not to include it in the imaging grid. Because of this, regions outside the grid can affect the TEC measurements if we are not careful. We do not need to worry about electron density past the top of the grid since the top is bounded by the satellite’s trajectory, which means the radio signals will spend little to no time radiating through space above the grid. We also do not need to worry about density past the bottom of the grid since the grid extends slightly past the bottom of the ionosphere. On the other hand, we do need to worry about the sides of the grid. If the ray path enters or exits the grid from one of the sides, and not the top or bottom, the extra density on either of those sides will affect the TEC measurements and possibly corrupt the reconstruction. An easy way to fix this is to simply remove the data point generated by any ray path that enters or exits the grid through a side, and not the top or bottom. This can be accomplished in code by simply checking the first and last grid intercepts of any ray path.

After an observation is deemed valid, its grid intercepts are sorted and used to compute which voxels the ray passes through. This is done by computing the midpoint between each pair of adjacent grid intercepts and then using the parametric model of the grid to determine which voxel it belongs to. The final step is to calculate the distance that the ray path spends inside each voxel, which is easily done with Equation (5.2). This distance is then entered into matrix element $\mathbf{H}_{ij}$, where i is the observation index and j is the voxel index. The operation is repeated for each satellite/receiver position pair available in the data.

5.3.1 Other Grid Variants

Due to the irregular sampling geometry generated by the imaging model, a square-shaped grid will always contain voxels in its bottom corners that are never sampled. To fix this problem, I have tried using two separate modifications to a standard square grid.

The first modification is to create an “Extended Grid”, that is, to augment a standard square grid with two extra grids on each side, which have larger voxels than the main center

grid. These side grids allow more data points to be used, since many fewer ray paths will enter or exit through the sides of this larger grid and need to be discarded. The voxels in these side grids are then included as part of the inversion process, but are discarded afterwards and not included in the output image. In doing so, the side grids allow the reconstruction to focus on the same square area as the original grid, but use more data points to create a better solution. The reason behind choosing large voxel sizes in the side grids is to keep state dimension as low as possible while still enjoying the benefits of more usable data.

In practice, this method has some serious problems. The sharp change in the voxel size between the grids triggered some very bad artifacts in the image near the boundaries. Initially, I thought that this might be due to some faulty calculations in the regularization functions since computing derivatives on non-uniform grids is more difficult than with a uniform grid. However, I then tried inverting with Kaczmarz’s method (described in Section 4.12) which does not use any explicit regularization scheme. Even with this method, the reconstructed images still contained the boundary artifacts; therefore, there must be something else responsible for creating the artifacts, and I currently do not know what that is. Due to these problems, I abandoned the side grid method, and switched to the method described next. I think that there may still be some utility in using side grids, but first I would need to track down the source of these artifacts.

The second grid modification that I tried was simpler and worked much better. This version starts out with a standard square grid and removes any dead voxel from the model before the inversion process begins. Other than producing a non-square output image, there were no added artifacts introduced by this procedure. Since this method works well, it is what I use to create all of the images appearing in later sections.

Table 5.1: Grid Parameters Used for C/NOFS-CERTO Data Analysis

Parameter	Value
Earth Radius	6378×10^3 m
Center Latitude:	-13.16 degrees
Center Longitude:	-71.5 degrees
Longitude Width:	50 degrees
# Horizontal Voxels:	100
# Vertical Voxels:	100

5.3.2 Physical Grid Parameters

Table 5.1 lists the grid parameters used when inverting the C/NOFS-CERTO Data. A longitudinal width of 50 degrees means that each voxel covers $1/2$ degree of longitude, which is approximately 58 km at a 350 km elevation. This resolution will provide enough information to distinguish neighboring spread-F plumes from one another. However, it will not provide enough information to resolve intra-plume structures very well.

Another important detail to note is that we are interested in imaging in a 2-D plane despite the fact that the physical acquisition geometry is inherently 3-D. To fix this discrepancy, we use 3-D voxels in the acquisition geometry. These voxels are split up in the longitude and altitude directions as prescribed by Table 5.1 and shown in Figure 5.2; however, they extend across all possible latitudes. In essence, this just ignores the fact that differences in latitude even exist. This is fine to do for a couple of reasons: First, the entire observation apparatus is very localized in latitude; all receivers are within 1.5 degrees latitude of each other, and the C/NOFS satellite orbit has a very low inclination to the Equator. The other reason that this approximation is valid is that, as mentioned in Section 3.3.1, the structure of spread-F bubbles tends not to vary much over different latitudes due to the effects of Earth’s magnetic field. Because of this, we can expect the electron density to be fairly constant over the small range of latitudes that are observed. In addition, the location of this imaging grid is very close to the Earth’s magnetic equator [5].

5.4 Construction of the Inversion Algorithm

5.4.1 SVD Analysis of the Imaging Geometry

After generating the forward model, image reconstruction now becomes a matrix inversion, and one of the first analysis tools presented in Chapter 4 is the SVD. This particular SVD analysis was performed on the imaging geometry taken from a four-receiver observation in 2012, and most other datasets have very similar looking SVD analyses. Figure 5.3 shows the normalized decay of singular values for this imaging geometry using a 100×100 voxel grid. The singular values decay very smoothly throughout the entire plot, and decay very fast at the beginning, faster than an exponential curve for approximately the first 500 components. Section 4.4 says that the satellite to ground imaging geometry is a discrete ill-posed problem since it arises from the discretization of a Fredholm integral equation, which is ill-posed itself.

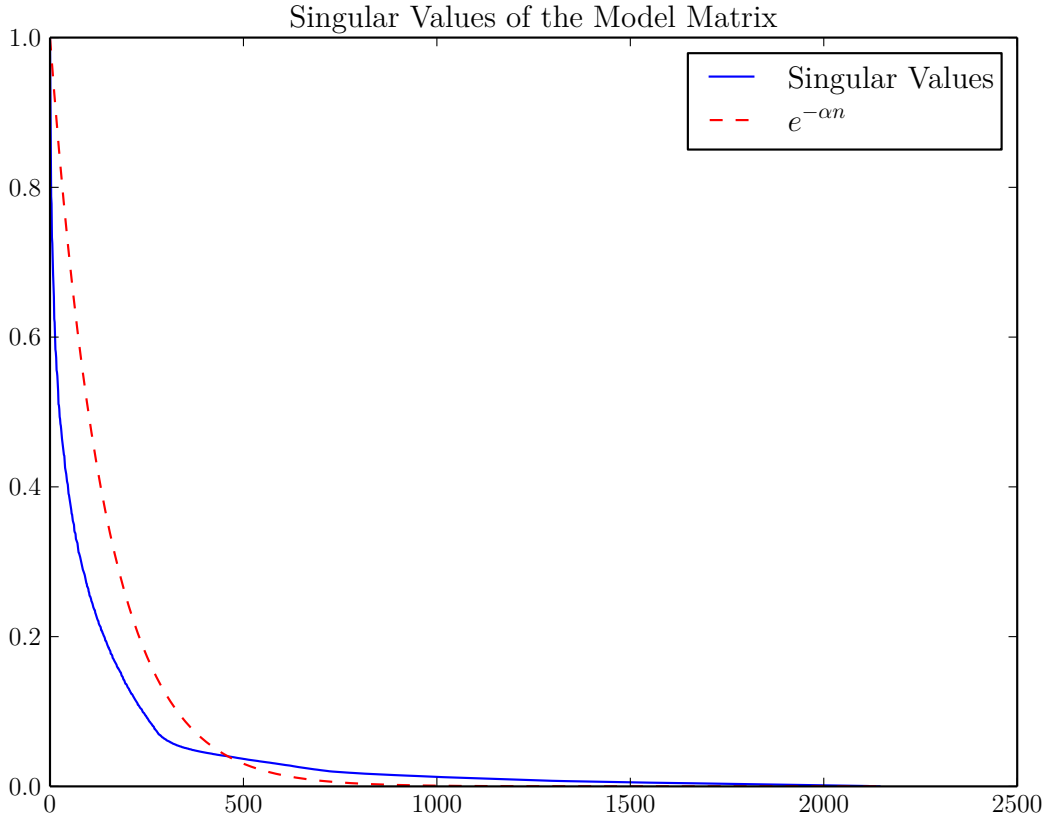


Figure 5.3: Singular Values for a 100×100 Grid with Four Active Receivers

Section 4.4 also says that the singular values of a discrete ill-posed problem will smoothly decay, and that is exactly what happens in Figure 5.3

The condition number of the model matrix is on the order of 10^{14} , and from a practical standpoint this says that direct inversion of the model matrix cannot be trusted at all. A general rule of thumb in direct matrix inversion is that every order of magnitude increase in the condition number results in one fewer significant digit of accuracy in the reconstructed solution. Given double-precision floating-point computations and noise-free data, directly inverting this model would give a solution with only two significant digits of accuracy. This is certainly not good. Even worse, the measurement noise will eat into those two remaining significant digits of accuracy resulting in a reconstruction that cannot be trusted. Due to this, direct inversion of the system is not at all practical.

5.4.2 Estimating the Reference Image

Due to the limited-angle nature of this tomography problem, the data contains very little information about the vertical textures in the ionosphere. Because of this, the reconstruction algorithm needs a lot of extra help to reconstruct these textures. To give it this extra help, we provide it with an external reference image which the algorithm will look to for guidance. In the grand scheme of things, this reference image can be viewed as another regularization parameter, albeit one that is multi-dimensional and has a very direct effect on the structure of the output image. Because of this, the reference image needs to be chosen carefully since it will play a large role in defining what the vertical profiles look like in the reconstruction. This section presents a couple of techniques which can be used to generate the reference image.

Chapter 2 presented a parameterized model for the ionosphere called the Chapman profile, given by Equation (2.2). Given estimates of the Chapman profile parameters, Equation (2.2) can be used to generate a basic reference image. When using the Chapman profile to generate a reference image, I did not develop any sophisticated methods for estimating the model parameters. Instead, I set the peak altitude to 350 km, which is a safe value, and set the other parameters so that the generated image looked reasonable. To estimate the electron density in the profile peak, I used a simple trial and error scheme in which I ran test images with different peak densities through the forward model and chose the one that generated measurements that best lined up with the actual data streams.

Constructing a reference image with a lot of guesswork, as done here, is a good first step, but it is far from ideal. A much better way to generate the reference image is to build it with data from other sources to create an image that better approximates the actual vertical profile. For external sources, there are two good candidates which this data could come from. The *Jicamarca Radio Observatory* (JRO) is located near Lima, Peru, and is just a little ways north of the four satellite receivers. Jicamarca uses many different instruments to observe the ionosphere, and much of the data that they collect is freely available online. One of these instruments is a specialized radar called an *ionosonde*. An ionosonde transmits a series of radio bursts at a range of different frequencies up into the ionosphere. Due to the range of plasma densities present in the ionosphere, radio waves of different frequencies will be reflected back by plasma at different altitudes, and these reflections are then used to generate a very accurate vertical profile of the ionosphere’s electron density [5].

Another good source of information is in-situ or “in-place” measurements taken by instruments on the satellite. An in-situ measurement will take physical samples of the ionosphere

to generate very accurate measurements of the electron density along the satellite’s path of travel. The C/NOFS satellite has multiple in-situ instruments on board [6], and this data can be used to even further augment the image reconstruction.

5.4.3 Estimating Receiver Measurement Offsets

Section 5.2 mentions that the receivers report their measurements in Relative TEC, which means that the image reconstruction algorithm must estimate the TEC offsets for each receiver. This section provides a brief discussion as to how that estimation can be accomplished.

One way to estimate the receiver offsets is to assume that all receivers have the exact same TEC offset. Given that most of the signals that the receivers see travel through the same general regions of electron density, it is somewhat reasonable to assume that each receiver is going to have offset values that are not too disparate from each other. Experimentally, I found that adding a constant offset to the data-streams barely changes the shape of any structures in the reconstructed image. Instead, it almost exclusively modifies the scale at which the electron density is reconstructed. That is, adding a positive offset to the data-stream will cause the image to have the same structure, but the reported electron densities will be slightly higher than if the offset was left out.

It turns out that this constant offset approximation does not work well in practice because it does not take into account the fact that all data streams may not be available at all times. Consider a receiver that only reports data for the first third of the observation window and a second receiver that only reports data for the last third. The radio signals that generate these observations will travel through completely different parts of the ionosphere, so the offsets for these two receivers may have very little in common. A much better way to go about estimating the receiver offsets is to leverage what we know about the forward model. Under the assumption that the true electron density follows what is prescribed by the generated reference image, the reference image can be used to estimate the TEC offset for each receiver. To do this, the reference image is fed forward through the imaging model to generate an artificial data-stream that is measured in absolute TEC units. Given a receiver, the minimum value of this artificial data-stream is then as the estimate of its actual TEC offset. Because of its superior ability to deal with partial data streams, this estimation method is used when reconstructing actual data-streams in Chapter 6.

5.4.4 Conjugate Gradient Method

Exact solutions to linear systems, computed using Gaussian elimination, etc., are useful in many situations; however, they come with a $\mathcal{O}(n^3)$ computational complexity, where n is the number of unknowns. Because of this, calculating an exact solution becomes less and less tractable as the size of the vectors increases. One way to significantly speed up the computation of these matrix inversions is to just approximate the solution. This works really well for an iterative scheme such as the iterative relaxation algorithm presented in Section 4.11, because each step of the iterative relaxation algorithm needs to compute a minimizer that is just “good enough” to keep the algorithm moving in the right direction. Because of this, significant computational savings can be gained by replacing each exact matrix inversion with an approximation. This section and the next present a few different iterative algorithms to perform approximate matrix inversion.

The *conjugate gradient* (CG) method is an iterative method to solve linear systems. It is theoretically an exact method, since for an n dimensional system, running CG for n steps will yield an exact solution. However, due to the effects of finite-precision arithmetic, the output of n steps of the CG algorithm will not be exactly equal to the true solution. Because of this, CG tends not to be used as an exact matrix solver but is very useful at generating approximate solutions, which can be done by stopping its iterations before convergence.

Given a matrix $\mathbf{A}$ that is symmetric and positive definite, the solution $\mathbf{x}^*$ to the system $\mathbf{Ax} = \mathbf{b}$ can be written as the minimizer of the quadratic form

$$\mathbf{x}^* = \arg \min_{\mathbf{x}} \frac{1}{2} \mathbf{x}^T \mathbf{Ax} - \mathbf{b}^T \mathbf{x} \quad (5.5)$$

Instead of obtaining $\mathbf{x}^*$ by solving the system $\mathbf{Ax} = \mathbf{b}$ directly, it can be computed by finding the optimizer to Equation (5.5). Substantial computational savings can arise if this minimizer $\mathbf{x}^*$ is just approximated.

A classic method for computing minima is known as *steepest descent*. Given the current guess $\mathbf{x}_k$ for function $f(\mathbf{x})$, steepest descent computes the next guess $\mathbf{x}_{k+1}$ by performing a *line-search*, or 1-D minimization, in the direction of the negative gradient $-\nabla f(\mathbf{x}_k)$ since that is the direction in which the value of $f(\mathbf{x})$ decreases the fastest. The optimal line-search parameter is then set as the step size α_k . This gives the iteration

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k) \quad (5.6)$$

For the quadratic form $f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{Ax} - \mathbf{b}^T \mathbf{x}$, $\nabla f(\mathbf{x}) = \mathbf{Ax} - \mathbf{b}$, so the gradient at point $\mathbf{x}_k$

is equal to the residual $\mathbf{r}_k = \mathbf{A}\mathbf{x}_k - \mathbf{b}$. Since $\mathbf{A}$ is positive-definite, the exact minimizer α_k of the line-search can be found by setting the derivative along the search direction to zero.

$$\frac{\partial f(\mathbf{x} - \alpha \nabla f(\mathbf{x}))}{\partial \alpha} = 0 \quad (5.7)$$

The derivative achieves this when the new residual/gradient, $\nabla f(\mathbf{x}_{k+1})$, is orthogonal to the current search direction $\mathbf{r}_k$, that is $\nabla^T f(\mathbf{x}_{k+1})\mathbf{r}_k = 0$. Plugging this orthogonality into the update equations gives the optimal choice of parameter α_k to be [22]

$$\alpha_k = \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{r}_k^T \mathbf{A} \mathbf{r}_k} \quad (5.8)$$

Despite being intuitively satisfying, the steepest descent algorithm runs into many convergence problems because choosing the negative gradient as the next search direction is far from optimal. Consider the case in which there exists a long narrow valley at the bottom of the $f(\mathbf{x})$ surface, which will happen if matrix $\mathbf{A}$ is poorly conditioned, and suppose the current guess $\mathbf{x}_k$ sits close to the bottom of the valley. The optimal search direction choice would be to travel along the bottom of the valley and land the next guess right on the minimizer $\mathbf{x}^*$. Instead, the residual points directly to the nearest dip in the valley, which could be very different from the optimal direction. In addition, since $\mathbf{x}_k$ already lives close to the bottom of the valley, $\mathbf{x}_{k+1}$ will not be very far away from $\mathbf{x}_k$ even if the minimizer $\mathbf{x}^*$ is still a long ways off. This sequence of estimates will just slosh around in the bottom of the valley making their way to the optimizer $\mathbf{x}^*$ very slowly. At an intuitive level, consider snowboarding on a half-pipe. The global minimum of the mountain is at the very bottom, next to the chair-lift. Because of this, the optimal “search direction” is to travel straight down the center of the half-pipe to the bottom. Most of the time, a snowboarder is not at the center of the half-pipe, but instead up on the edges. When on the edges, the gradient does not point down towards the bottom of the mountain, but instead back towards the center of the half-pipe, which is certainly not in the optimal search direction.

One of the main reasons that the steepest descent can fail to converge is that it keeps searching in the same directions over and over. It would be great if it could search in a direction, find the optimal value in that direction, and then never search in that direction again. This may sound too good to be true, but it is very possible since $f(\mathbf{x})$ is quadratic. Consider the case in which $\mathbf{A} = \mathbf{I}_n$, the n -dimensional identity matrix. In two dimensions, the contour plot of $f(\mathbf{x})$ is a set of circular rings surrounding the optimal point $\mathbf{x}^*$. Given

any starting point $\mathbf{x}_0$, let the first search direction be along the first coordinate axis. After this step, guess $\mathbf{x}_1$ contains the first coordinate of $\mathbf{x}^*$. Next, find guess $\mathbf{x}_2$ by searching in the direction of the second coordinate axis. $\mathbf{x}_2$ will pick up the second-coordinate of $\mathbf{x}^*$, and thus $\mathbf{x}_2 = \mathbf{x}^*$. The reason that this case works is that the surface $f(\mathbf{x})$ is completely symmetric and the search directions are orthogonal. However, $f(\mathbf{x})$ rarely has such a nice contour plot, and instead the circular contour rings are all stretched out. Because of this, instead of choosing the search directions to be orthogonal, as was done in this example, they should be chosen to be $\mathbf{A}$ -orthogonal [22]; that is, given search directions $\mathbf{p}_i$,

$$\mathbf{p}_i^T \mathbf{A} \mathbf{p}_j = 0 \quad \forall i \neq j \quad (5.9)$$

The basic intuition behind this choice of directions is that $\mathbf{A}$ orthogonal search directions will become orthogonal if the contour plot of $f(\mathbf{x})$ is “un-stretched” to make it completely circular again.

Choosing search directions in this manner gives rise to the celebrated conjugate gradient (CG) method. The complete proof for CG is somewhat involved so it will not be explained here. [22] provides a very in-depth explanation of the full CG derivation including much of the motivating intuition, and [23] provides a more concise derivation of the method.

The last point to note is that the CG method involves orthogonalizing the current search direction against all previous search directions, which could be computationally intensive if done manually. However, due to some surprising properties of the search directions, this full orthogonalization can be performed by manually orthogonalizing a new search direction against only the one preceding search direction [23], which makes CG iterations very efficient.

After setting the initial residual as the first search direction

$$\mathbf{d}_0 = \mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0 \quad (5.10)$$

the full CG iteration proceeds as follows [22]:

$$\begin{aligned}
\alpha_i &= \frac{\mathbf{r}_i^T \mathbf{r}_i}{\mathbf{d}_i^T \mathbf{A} \mathbf{d}_i} && \text{Compute the optimal step-size} \\
\mathbf{x}_{i+1} &= \mathbf{x}_i + \alpha_i \mathbf{d}_i && \text{Compute new guess} \\
\mathbf{r}_{i+1} &= \mathbf{r}_i - \alpha_i \mathbf{A} \mathbf{d}_i && \text{Compute new residual} \\
\beta_{i+1} &= \frac{\mathbf{r}_{i+1}^T \mathbf{r}_{i+1}}{\mathbf{r}_i^T \mathbf{r}_i} && \text{Orthogonalize to the previous search direction} \\
\mathbf{d}_{i+1} &= \mathbf{r}_{i+1} + \beta_{i+1} \mathbf{d}_i && \text{Compute new search direction}
\end{aligned} \tag{5.11}$$

5.4.5 CGLS/LSQR Methods

In the same way that the conjugate gradient algorithm computes the solution to a linear system iteratively, there are a number of algorithms that can be used to compute the least-squares solution to a linear system in the same iterative fashion. Iterative least-squares algorithms share many of the same benefits as iterative matrix solvers. These algorithms will converge to an exact answer; however, their iterations can be terminated early to give a good approximation of the true least-squares solution. Most times, this approximation is achieved with much less computational effort than an exact solution.

As shown in Section 4.1.1, the least squares solution $\mathbf{x}$ to a system $\mathbf{A}\mathbf{x} = \mathbf{b}$ satisfies the *normal equations*

$$\mathbf{A}^T \mathbf{A} \mathbf{x} = \mathbf{A}^T \mathbf{b} \tag{5.12}$$

As long as $\mathbf{A}$ is nonsingular, $\mathbf{A}^T \mathbf{A}$ is positive definite so the inversion of $\mathbf{A}^T \mathbf{A} \mathbf{x} = \mathbf{A}^T \mathbf{b}$ can be recast as minimizing the quadratic form.

$$\mathbf{x}^* = \arg \min_{\mathbf{x}} \frac{1}{2} \mathbf{x}^T \mathbf{A}^T \mathbf{A} \mathbf{x} - \mathbf{b}^T \mathbf{A} \mathbf{x} \tag{5.13}$$

A sensible-sounding idea is to find the minimum by application of the conjugate gradient method, presented in Section 5.4.4. Direct application of CG creates a couple problems. First, if $\mathbf{A}$ is large $\mathbf{A}^T \mathbf{A}$ may take a very long time to compute, and if $\mathbf{A}$ was initially sparse, $\mathbf{A}^T \mathbf{A}$ may no longer be. Second and more important, the condition number of $\mathbf{A}^T \mathbf{A}$ is the condition number of $\mathbf{A}$ squared, which means that all of the noise-sensitivity in $\mathbf{A}$ is going to be magnified that much more in $\mathbf{A}^T \mathbf{A}$. A slight modification of the CG algorithm gets around having to compute $\mathbf{A}^T \mathbf{A}$ directly, and this modification is the *conjugate gradient*

least-squares (CGLS) algorithm. In CGLS, the extra $\mathbf{A}^T$ is factored out of the calculation for the majority of an iteration and is then put back at the end [15]. This bypasses the expensive matrix-matrix multiplication to form $\mathbf{A}^T \mathbf{A}$ and avoids its associated conditioning problems. The entire CGLS iteration can be found in [15].

Another iterative least-squares algorithm, the *LSQR* algorithm uses many of the same principles as the CGLS algorithm, but in a much different way. Just as with the CGLS algorithm, the LSQR algorithm looks to iteratively build up the least squares solution $\mathbf{x}_{\text{LS}}$ as a combination of basis vectors. However, LSQR uses completely orthogonal vectors instead of $\mathbf{A}$ -orthogonal vectors as CGLS does. Let these basis vectors $\mathbf{v}_i$ be the columns of a matrix $\mathbf{V}$. Then LSQR gives the exact least-squares solution $\mathbf{x}_{\text{LS}}$ by [23]

$$\mathbf{x}_{\text{LS}} = \mathbf{V} \mathbf{y}_{\text{LS}} = \sum_{i=1}^n \mathbf{y}_i \mathbf{v}_i \quad (5.14)$$

where $\mathbf{y}$ is the coordinate vector of $\mathbf{x}_{\text{LS}}$ with respect to the $\{\mathbf{v}_i\}$ basis. Approximate least-squares solutions can then be computed by truncating this sum to only p terms where $p < n$.

In each iteration of LSQR, another basis vector $\mathbf{v}_k$ is added to the set, and the optimal coordinates $\mathbf{y}_k$ are found such that the current approximate $\mathbf{x}_k$ minimizes $\|\mathbf{x} - \mathbf{x}_{\text{LS}}\|_2^2, \forall \mathbf{x} \in \text{Span}(\{\mathbf{v}_k\})$. Thus, after computing the optimal coordinates $\mathbf{y}_k$ for iteration k , the current guess $\mathbf{x}_k$ is given by

$$\mathbf{x}_k = \mathbf{x}_0 + \mathbf{V}_k \mathbf{y}_k \quad (5.15)$$

Again, the desire at each step is to find the $\mathbf{y}_k$ such that

$$\mathbf{y}_k = \arg \min_{\mathbf{y}} \|\mathbf{b} - \mathbf{A}(\mathbf{x}_0 + \mathbf{V}_k \mathbf{y})\|_2^2 \quad (5.16)$$

In LSQR the optimal guess $\mathbf{x}_k$ is formed by a linear combination of $\mathbf{x}_0$ and each of the vectors in $\mathbf{V}_k$, while in CGLS, the optimal guess $\mathbf{x}_k$ was formed by a linear combination of $\mathbf{x}_0$ and each of the past search directions

To compute $\mathbf{V}_k$ for each iteration, an orthogonal decomposition known as *bidiagonalization* is used. In bidiagonalization, a matrix $\mathbf{A}$ is transformed by two matrices with orthonormal columns, $\mathbf{U}$ and $\mathbf{V}$, into an upper bidiagonal matrix $\mathbf{B}$ where

$$\mathbf{U}^T \mathbf{A} \mathbf{V} = \mathbf{B} \quad (5.17)$$

Bidiagonalization is similar to singular value decomposition presented in Section 4.2, but

it is faster to compute. One such method for computing the bidiagonalization is called *Lanczos bidiagonalization* in which the matrices $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{B}$ are all built up in an iterative fashion [23]. LSQR needs to use a modified version of Lanczos bidiagonalization in which the resulting $\mathbf{B}$ is lower bidiagonal instead of upper bidiagonal. With this modification, the partially constructed factorization satisfies the following relationship [23]:

$$\mathbf{A}\mathbf{V}_k = \mathbf{U}_{k+1}\mathbf{B}_k \quad (5.18)$$

where for a given matrix $\mathbf{M}$, $\mathbf{M}_i$ denotes the sub-matrix containing the first i columns of $\mathbf{M}$.

LSQR begins by seeding $\mathbf{u}_1$, the first vector of $\mathbf{U}$ as [23]

$$\mathbf{u}_1 = \frac{\mathbf{b} - \mathbf{A}\mathbf{x}_0}{\beta_1} \quad \text{where } \beta_1 = \|\mathbf{b} - \mathbf{A}\mathbf{x}_0\| \quad (5.19)$$

If we substitute the current guess of $\mathbf{x}_k$ given by Equation (5.15) into the squared-error term $\|\mathbf{A}\mathbf{x}_k - \mathbf{b}\|_2^2$, this error term will simplify to [23]

$$\|\mathbf{A}(\mathbf{x}_0 + \mathbf{V}_k\mathbf{y}) - \mathbf{b}\|_2^2 = \|\mathbf{U}_{k+1}\mathbf{B}_k\mathbf{y} - \beta_1\mathbf{U}_{k+1}\mathbf{e}_1\|_2^2 \quad (5.20)$$

$$= \|\mathbf{U}_{k+1}(\mathbf{B}_k\mathbf{y} - \beta_1\mathbf{e}_1)\|_2^2 \quad (5.21)$$

$$= \|\mathbf{B}_k\mathbf{y} - \beta_1\mathbf{e}_1\|_2^2 \quad (5.22)$$

$$(5.23)$$

where $\mathbf{e}_1$ is the vector $[1, 0, \dots, 0]^T$, and $\mathbf{U}_{k+1}$ can be removed in Equation (5.22) since it has orthonormal columns. Unlike minimizing $\|\mathbf{A}(\mathbf{x}_0 + \mathbf{V}_k\mathbf{y}) - \mathbf{b}\|_2^2$ over $\mathbf{y}$ directly, the minimizer $\hat{\mathbf{y}}$ for $\|\mathbf{B}_k\mathbf{y} - \beta_1\mathbf{e}_1\|_2^2$ is very easy to compute since the expression is comprised of a lower-bidiagonal matrix, $\mathbf{B}_k$, and a vector with only one nonzero entry, $\beta_1\mathbf{e}_1$. With a bit more math, the simplified minimization given in Equation (5.22) can be performed in an iterative fashion, in which $\mathbf{x}_k$ is updated directly from $\mathbf{x}_{k-1}$. More information on LSQR, Lanczos bidiagonalization, and orthogonal factorizations can be found in [23].

To sum it all up, the LSQR algorithm computes a better and better approximation to the exact least-squares solution by iteratively computing the bidiagonalization of $\mathbf{A}$ and using this partial factorization at each step of the process to update the guess $\mathbf{x}_k$ by means of solving the simplified least-squares minimization given in Equation (5.22). The LSQR algorithm provides a more stable solution than the CGLS algorithm when run for many iterations. However, there is no advantage if the algorithm is terminated after a small

number of iterations [14]. So depending on the application, LSQR may be preferable to CGLS.

5.4.6 Constructing the Variational Characterization of the Problem

Many of the ingredients necessary for performing a successful tomographic inversion of the C/NOFS-CERTO data have been presented, so this section looks at how they can be combined to create a suitable cost function, which can then be optimized to generate a reconstructed image. The electron density in the ionosphere is a relatively smooth quantity which may have sharp edges when irregularities are present. Section 4.10 gave a discussion about how we might develop a regularization scheme which will preserve boundaries, and then Section 4.11 presented an iterative relaxation algorithm which can be used to achieve this edge-preserving regularization. This section will apply that exact algorithm to the problem at hand. The use of this specific regularization scheme was originally presented in [24] in which it was applied to imaging with GPS satellites.

As mentioned in previous sections, the C/NOFS-CERTO data provides much more information about horizontal textures than vertical; therefore, the reconstruction process will employ a reference image to help create the vertical profile, and this process is described in Section 5.4.2. The reference profile is generated with the Chapman production function parameterized with a peak electron density of $4 \times 10^{11} \frac{\text{el}}{\text{m}^2}$ at a height of 350 km.

The general form of the cost function to be used is

$$J(\mathbf{f}) = Q(\mathbf{f}) + \lambda \sum_{m=0}^{M-1} \sigma[V_m(\mathbf{f})] \quad (5.24)$$

which is described in detail in Section 4.11. We will use two regularization terms ($M = 2$), the first to regularize horizontal textures and the second to regularize vertical textures.

The first regularization term penalizes the weighted magnitude of the horizontal spatial derivative using the expression

$$J_{\text{hor}}(\mathbf{x}) = \lambda_{\text{hor}} \sum_k \phi_{\text{hor}}([\mathbf{D}_1 \mathbf{x}]_k) \quad (5.25)$$

This regularization term uses a simple first-order finite-difference approximation to the hor-

horizontal first-derivative, where

$$f'[x] \approx -f[x] + f[x + 1] \quad (5.26)$$

The differentiation operation is denoted as multiplication by matrix $\mathbf{D}_1$.

The vertical regularization is slightly more involved since it involves penalizing a weighted difference, $\psi(\mathbf{x} - \mathbf{x}_0)$, of the solution image $\mathbf{x}$ and the reference image $\mathbf{x}_0$. [24] finds that the second-derivative operation is particularly useful to use here, which essentially will penalize the difference in curvature between vertical profile and the reference profile. [24] says that using a first derivative generally forces the reconstructed voxel values to conform too much to the values presented in the reference image, not giving them enough freedom to follow the data. Using the second-derivative in the penalty, the weighted difference can be written as $\hat{\psi}(\mathbf{D}_2(\mathbf{x} - \mathbf{x}_0))$, where multiplication by the matrix $\mathbf{D}_2$ is used to denote evaluation of the second-derivative. To compute the second-derivative, we use a second order finite-difference approximation of the form

$$f''[x] \approx f[x - 1] - 2f[x] + f[x + 1] \quad (5.27)$$

Combining all of these gives a vertical regularization term of the form

$$J_{\text{vert}}(\mathbf{x}) = \lambda_{\text{vert}} \sum_k \phi_{\text{vert}}([\mathbf{D}_2(\mathbf{x} - \mathbf{x}_0)]_k) \quad (5.28)$$

Combining the vertical and horizontal regularization terms with the residual gives the full cost function

$$J(\mathbf{x}) = \|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 + \lambda_{\text{hor}} \sum_i \phi_{\text{hor}}([\mathbf{D}_1\mathbf{x}]_i) + \lambda_{\text{vert}} \sum_i \phi_{\text{vert}}([\mathbf{D}_2(\mathbf{x} - \mathbf{x}_0)]_i) \quad (5.29)$$

As discussed in Section 4.11, both $\phi_{\text{vert}}(\cdot)$ and $\phi_{\text{hor}}(\cdot)$ are non-convex functions that satisfy the regularity conditions, so Equation (5.29) can be optimized using the iterative relaxation algorithm. Equation (5.29) will first get broken down into local cost functions of the form

$$J_{0,k}(\mathbf{x}) = \|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 + \lambda_{\text{hor}} \sum_i e_{\text{hor},i}[\mathbf{D}_1\mathbf{x}]_i^2 + \lambda_{\text{vert}} \sum_i e_{\text{vert},i}[\mathbf{D}_2(\mathbf{x} - \mathbf{x}_0)]_i^2 \quad (5.30)$$

which means that at step k next guess $\mathbf{x}_k$ will satisfy

$$\mathbf{x}_k = \arg \min_{\mathbf{x}} J_{0,k}(\mathbf{x}) \quad (5.31)$$

$$= \arg \min_{\mathbf{x}} \|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 + \lambda_{\text{hor}} \sum_i \mathbf{e}_{\text{hor},i} [\mathbf{D}_1 \mathbf{x}]_i^2 + \lambda_{\text{vert}} \sum_i \mathbf{e}_{\text{vert},i} [\mathbf{D}_2(\mathbf{x} - \mathbf{x}_0)]_i^2 \quad (5.32)$$

$$(5.33)$$

Equation (5.32) is just a sum of expanded inner-products, so it can be rewritten as

$$\mathbf{x}_k = \arg \min_{\mathbf{x}} \|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 + \lambda_{\text{hor}} \|\mathbf{D}_1 \mathbf{x}\|_{\mathbf{W}_{\text{hor}}}^2 + \lambda_{\text{vert}} \|\mathbf{D}_2(\mathbf{x} - \mathbf{x}_0)\|_{\mathbf{W}_{\text{vert}}}^2 \quad (5.34)$$

where $\mathbf{W}_{\text{hor}}$ is a diagonal matrix with $\mathbf{e}_{\text{hor}}$ as the main diagonal, and $\mathbf{W}_{\text{vert}}$ is a diagonal matrix with $\mathbf{e}_{\text{vert}}$ as the main diagonal. These three terms can then be combined into a single term which will be

$$\mathbf{x}_k = \arg \min_{\mathbf{x}} \left\| \begin{pmatrix} \mathbf{H} \\ \lambda_{\text{hor}}^{1/2} \mathbf{W}_{\text{hor}}^{1/2} \mathbf{D}_1 \\ \lambda_{\text{vert}}^{1/2} \mathbf{W}_{\text{vert}}^{1/2} \mathbf{D}_2 \end{pmatrix} \mathbf{x} - \begin{pmatrix} \mathbf{y} \\ \mathbf{0} \\ \lambda_{\text{vert}}^{1/2} \mathbf{W}_{\text{vert}}^{1/2} \mathbf{D}_2 \mathbf{x}_0 \end{pmatrix} \right\|_2^2 \quad (5.35)$$

where $\mathbf{W}_{\text{hor/vert}}^{1/2}$ is the diagonal matrix with its main diagonal equal to $\sqrt{\mathbf{e}_{\text{hor/vert}}}$. Equation (5.35) has a simple $\hat{\mathbf{x}} = \arg \min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$ form, which can be solved directly via QR factorization, or approximated with either the CGLS or LSQR algorithms, which were presented in Section 5.4.5.

5.4.7 Sparse Matrix Formats

The final ingredient needed make the inversion algorithm run is a set of special formats for storing sparse matrices and efficient matrix-vector multiplication algorithms that take advantage of this sparsity.

For a standard $n \times n$ matrix $\mathbf{A}$ the cost of a matrix-vector multiplication $\mathbf{A}\mathbf{x}$ is $\mathcal{O}(n^2)$, and the cost of a matrix-matrix multiplication $\mathbf{A}\mathbf{B}$ is $\mathcal{O}(n^3)$. As the matrix gets larger, repeated computations can start to really slow down an algorithm. If a large matrix $\mathbf{A}$ is sparse, that is, most of its entries are equal to 0, then substantial computational savings can be achieved if all of the zero-entries in $\mathbf{A}$ are ignored. This idea is the core of all sparse matrix formats and operations. There are a wide variety of formats for storing sparse matrices, and each

one has its advantages and disadvantages depending on a matrix's sparsity structure and what operation need to be performed.

The two formats introduced here are *compressed sparse row* (CSR) and *compressed sparse column* (CSC), which are useful in a wide variety of applications. The CSR format stores a sparse matrix $\mathbf{A}$ in three separate arrays. The value array $\mathbf{v}$ stores every nonzero element in $\mathbf{A}$ in a row-major format [25]. That is, it starts with the first row of $\mathbf{A}$, stores all nonzero entries in that row in the order they appear, and then does the same with each subsequent row. The size of $\mathbf{v}$ is precisely the number of nonzero entries in $\mathbf{A}$, abbreviated as NNZ. The column-index array $\mathbf{c}$ is the same length as the value array and each entry in this array contains the column index of its corresponding element in the value array [25]. The row-pointer array $\mathbf{r}$ is a smaller array whose length is equal to the number of rows in $\mathbf{A}$ plus one [25]. The first entry in $\mathbf{r}$ contains the index in $\mathbf{v}$ of the first element in row 1, the second entry in $\mathbf{r}$ contains the index in $\mathbf{v}$ of the first element in row 2, and so on, with the last entry in $\mathbf{r}$ containing administrative information. The combination of these three vectors together $(\mathbf{v}, \mathbf{c}, \mathbf{r})$ gives a full description of matrix $\mathbf{A}$ and contains all information about $\mathbf{A}$'s sparsity structure [25]. From this it is fairly easy to see that a matrix-vector multiplication, $\mathbf{A}\mathbf{x}$, will now only take $\mathcal{O}(\text{NNZ})$ operations to compute, since only the nonzero entries are stored and operated upon. If $\mathbf{A}$ is sparse this can be a significant improvement over the $\mathcal{O}(n^2)$ operations requires for a full dense matrix-vector multiply.

The compressed sparse column (CSC) format is very similar to the CSR format, but it stores all entries in $\mathbf{v}$ in a column-major format. That is, it stores the nonzero entries of the first column, followed by the nonzero entries of the second column, and so on [25]. Instead of storing the column indices of the entries, CSC stores the row indices, and instead of storing row pointers, it stores column pointers. Matrix-vector multiplications also take $\mathcal{O}(\text{NNZ})$ operations with the CSC format [25].

Due to the inherent compression of a matrix in the CSR and CSC formats, changes to a matrix tend to take longer to do. Changes to the sparsity structure and matrix slicing in the non-major direction can take much longer than with classic dense matrix storage due to the fact that they involve searching through index lists and/or moving large chunks of data around. Because of this, sparse matrices may be constructed using other formats and then converted to either CSR or CSC when it is time for the heavy computations to begin.

5.4.8 Putting Everything Together

Every portion of the algorithm has been discussed in detail in the previous sections, so this section will explain how each of these parts interconnect along with some of the extra “glue” that holds them together.

Starting after the forward model matrix and observation vector are generated, the first step is to remove from the model matrix all voxels in the grid that are not sampled during the observation, as discussed in Section 5.3.1. The next step is to use the reference image to estimate each receiver’s TEC offset, as described in Section 5.4.3.

Due to the nature of the observation geometry and the structure of finite-difference derivative approximations, the model matrix $\mathbf{H}$ and both derivative matrices, $\mathbf{D}_1$ and $\mathbf{D}_2$, are all large and sparse. Because of this, these three matrices are stored in compressed sparse row (CSR) matrix format and use sparse matrix-vector multiplications. This significantly improves the speed of the algorithm execution.

Getting to the meat of the algorithm, we set up the variational characterization of the problem as presented in Section 5.4.6. In each step of the optimization the local cost function $J_0(\cdot)$ is constructed and its approximate minimizer is computed using the LSQR algorithm. After this approximate minimizer $\mathbf{x}_k$ is obtained, we clip all negative values of $\mathbf{x}_k$ to zero since negative electron density cannot physically occur. In many cases, a clipping operation such as this can cause very undesirable artifacts. The main reason that it works here is that the guess $\mathbf{x}_k$ is clipped after every single iteration, so the hard clipping becomes more of a gentle nudging of the guess towards the set of positive solutions. If the algorithm waited until the very end to do any clipping, the results would be disastrous.

The optimization is either iterated a finite number of times (solving a fixed number of $J_0(\cdot)$ local optimizations) or it is run until convergence, when the change between the current guess $\mathbf{x}_k$ and the previous guess $\mathbf{x}_{k-1}$ differ from each other by a small enough amount.

Chapter 6

Results

6.1 Discussion of Output Images

6.1.1 Structure of Spread-F

In this chapter we take a look at a few images generated by the reconstruction algorithm laid out in Chapter 5. Before discussing the output images, it is instructive to get familiar with the basic physical structure of spread-F irregularities. This background will allow us to look at the images with a more discerning eye, and help identify structures that may be due to spread-F events.

Spread-F irregularities are postulated to be driven by Rayleigh-Taylor instabilities that happen due to a density inversion in the ionosphere's plasma [11]. This inversion can be thought of conceptually as a glass full of oil and water which is inverted so the water sits above the oil. The oil has a lower density than the water, so it will naturally rise to the top to correct the inversion, and it does this by rising up through the water in a number of distinct plumes.

Spread-F irregularities form in much the same way; low density plasma that is trapped below a higher density layer gets disturbed and rises through the high density plasma in a number of distinct bubbles, each of which can have a finger-like shape [11]. A larger plume may also split into several smaller plumes as it makes its way up through the high density layer [5].

At times of active weather in the ionosphere, many spread-F bubbles may form, and research suggests that these bubbles have a somewhat periodic spacing. Data analysis published in 2010 using a multi-year dataset recorded the distance between bubbles when they exhibited periodic structure and found that 200-300 km inter-bubble spacing was most common, followed by either 100-200 km spacing or 300-400 km spacing [26].

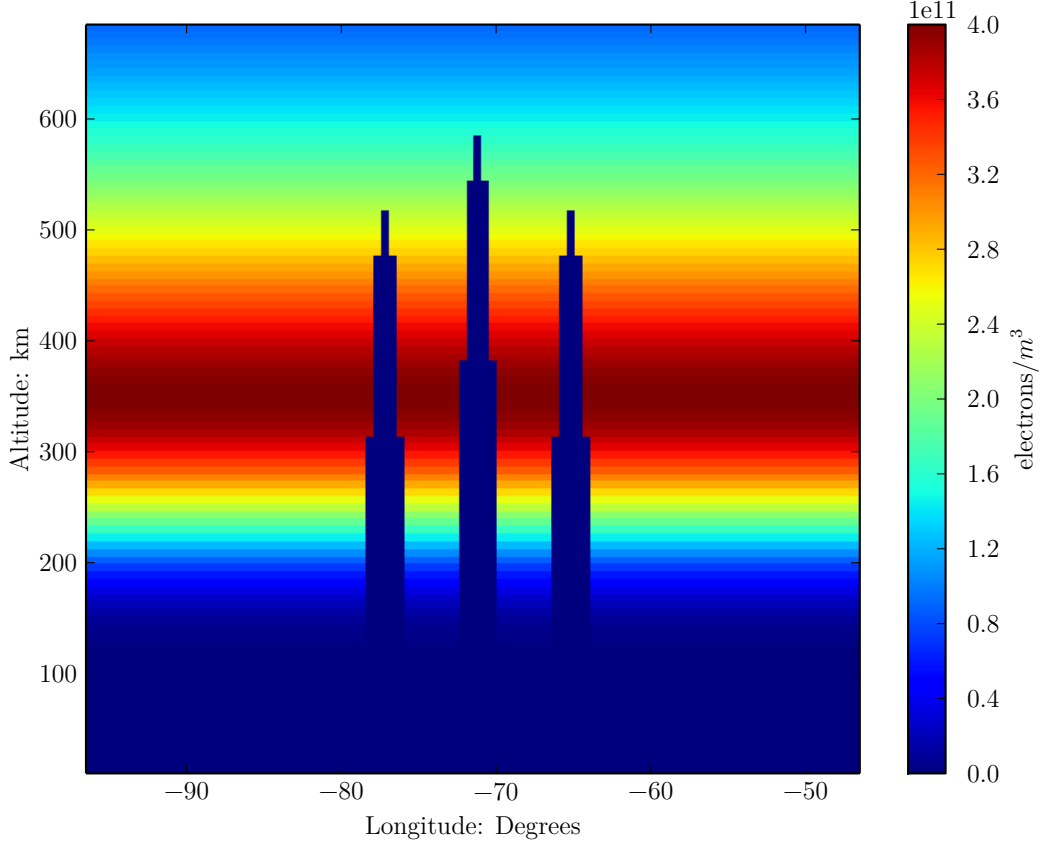


Figure 6.1: Initial Fake Irregularity

6.1.2 Simulated Reconstructions

We first run the reconstruction algorithm on a test image containing an artificially generated ionosphere with an irregularity, shown in Figure 6.1. The test image was created by superimposing a simple approximation of three low-density bubbles on top of a basic Chapman profile with peak electron density at 350 km. The satellite trajectory is taken from an actual observation, and three artificial receivers are spaced evenly across the bottom of the grid. The inversion parameters are given in Table 6.1, and the reconstructed output is given in Figure 6.2.

To visualize the receiver locations in Figure 6.2, look at the bottom boundary of the image and discard all of the white un-sampled voxels to get the small region in the center.

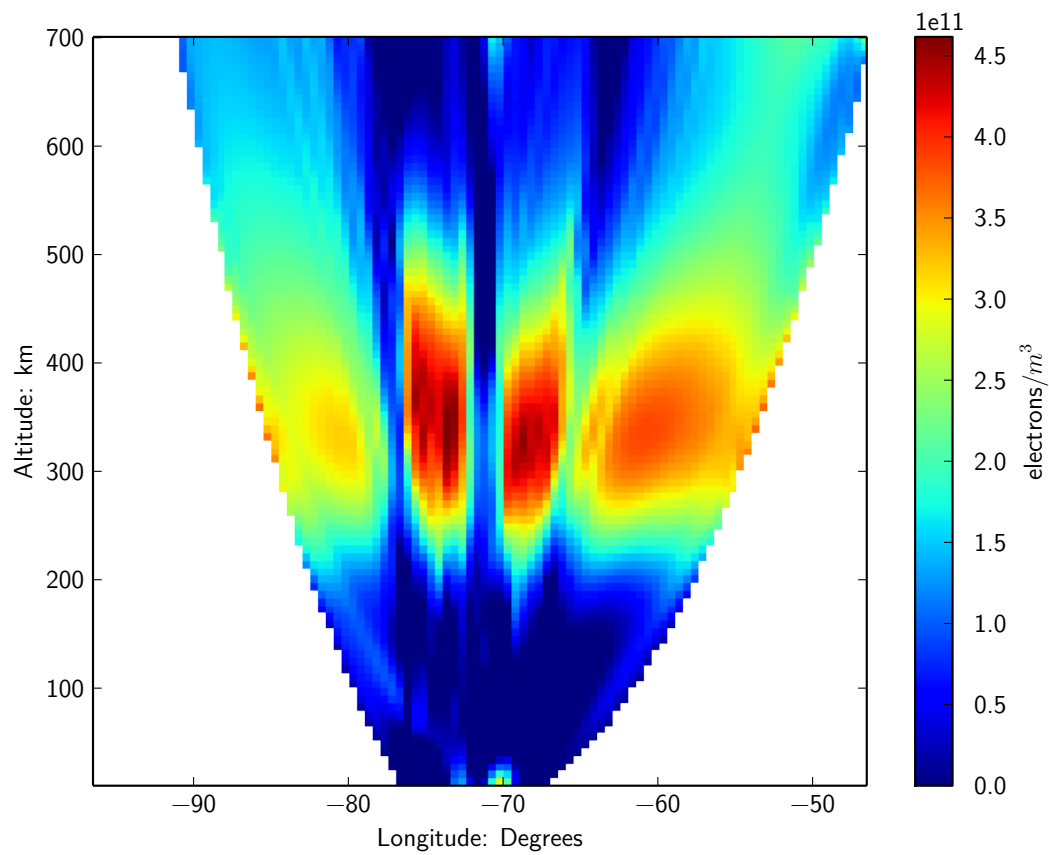


Figure 6.2: Inversion of Fake Irregularity

Table 6.1: Algorithm Parameters Used When Inverting the Test Image

Parameter	Value
# of Iterations	50
λ_{hor}	1
λ_{vert}	100

A receiver sits near each edge of this region, and one sits directly in the middle. Taking note of where these receivers are located provides a good illustration of just how severe the angle-limitation is in the tomographic reconstruction. For the majority of locations on the grid, approximately five to ten percent of the possible observation angles are covered. Despite this severe angle limitation, the algorithm is still able to accurately place the bubbles in the reconstructed image. Also note that the center irregularity is resolved a little better than the two irregularities on the outside. This is most likely due to the fact that the center irregularity is seen by receivers from both the left and right, while the side irregularities are only seen from one side.

As exhibited in Table 6.1, the λ parameter for the vertical regularization is much larger than that of the horizontal. This discrepancy arises because we use a limited-angle acquisition geometry that ties all of the vertical textures to small singular values in the model matrix. Because of this, the vertical textures need a lot of extra help during image reconstruction, which requires us to set the regularization parameter higher. On the other hand, the horizontal textures are tied to the larger singular values, so they need much less help from the regularization term during reconstruction.

Figure 6.2 also exhibits some of artifacts that can arise during image reconstruction. The first artifact to notice is what I call the “Cat-Eye” effect. The background density in Figure 6.1 is spread evenly across the entire width of the image, but in Figure 6.2, the background density values in the center of the image are more extreme than they are at the edges. This causes the higher electron density values to appear “smashed” toward the center of the image. Another notable artifact is that the region directly above an irregularity is always filled in with too much low density plasma, and this gives the appearance that the irregularities extend farther up in the image than they actually do. This specific artifact happens because the imaging geometry provides very little information about vertical textures. The reconstruction algorithm knows that the receivers report low-density measurements when they look towards an irregularity; however the data does not clearly communicate that the region of low density stops abruptly once the bubble ends. Because of this, the reconstruction algorithm reconciles the reported low density measurements with the reference image’s prescription of the ionosphere, in which low density plasma lives at the top and bottom, but not in the middle. Given this information, the reconstruction algorithm concludes that this low density measurement must have mainly come from the top and bottom of the image, even though a great deal of it actually came from the middle. Despite this problem, the plasma bubbles still show up in the ionosphere peak, albeit at a higher electron density than

they actually are. In a way, this can be likened to a young kid using a coloring book who has not yet learned how to color inside the lines. The outlines of the structures are all present, but the fill colors are not quite in the correct place.

Next we examine Figure 6.3, which also reconstructs the simulated image in Figure 6.1, but under the influence of additive Gaussian white noise. This image was also reconstructed with the same regularization parameters as Figure 6.2, which are listed in Table 6.1. At first glance, the overall structure is very similar between both of the images. This is good because it shows that this particular style of noise at moderate levels will not compromise the basic structure of the reconstruction. Other than that, the noisy reconstruction has more noise-induced artifacts apparent in its reconstructed textures: The vertical and horizontal gradients are less defined, and there are streaks of high-density plasma that extend further towards the top and bottom of the image than those in the noise-free version. A very simple way to smooth out some of these noise-induced artifacts is to set the regularization parameters higher. Doing this will partially eliminate these artifacts, but they will not completely disappear.

Due to the application of regularization and its resulting artifacts, these images are biased estimators of the actual quantity being observed. To successfully interpret these images, we need to be aware of these biases and in a sense, “undo” some of their effects in our interpretation of the reconstructions. That is, we should imagine what possible “ground-truth” models could have generated a given image reconstruction and then decide which of these possible models best match the image given what we know about the ionosphere. This imaging problem is far from a controlled experiment, and there are many unknown variables, and interactions. Because of this, the most insight will be gained by a scientist who has carefully studied the data and inversion techniques, knows their tendencies, and can use that intuition to pull information out of the reconstructed images that may have otherwise been overlooked or misinterpreted.

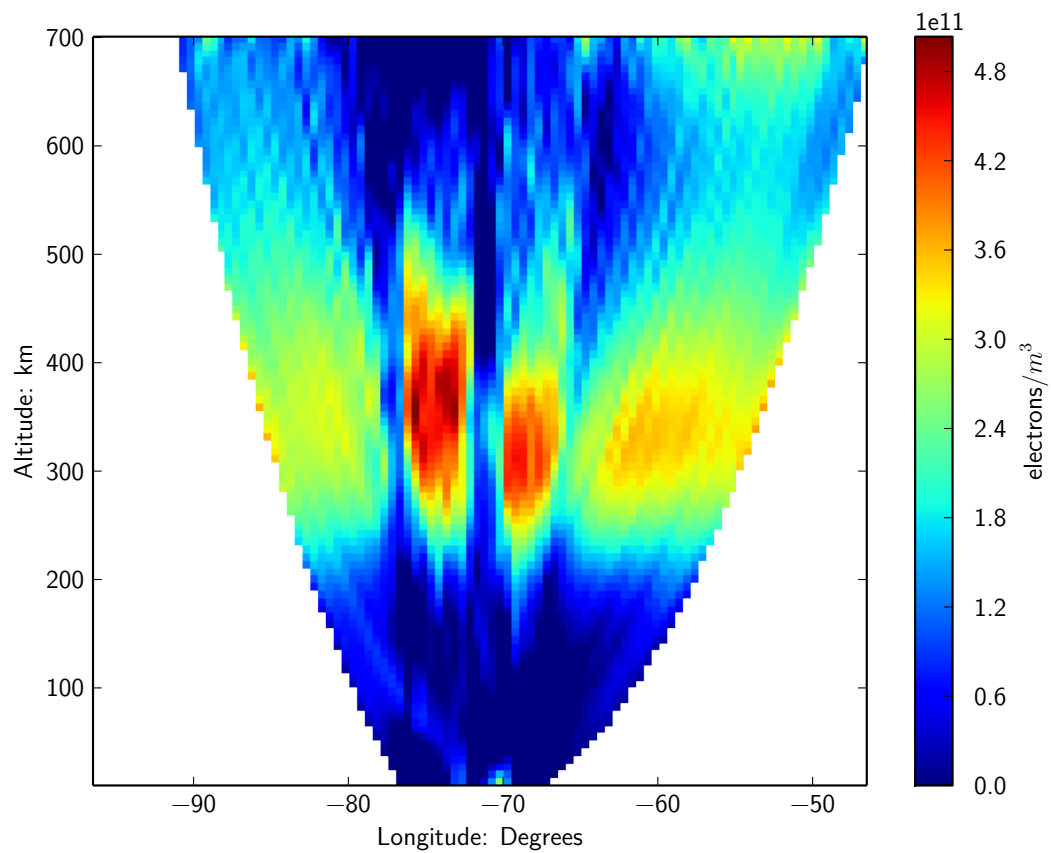


Figure 6.3: Inversion of Fake Irregularity with Heavy Noise

6.1.3 Inversion of the C/NOFS-CERTO Data

In this section, we examine the reconstructions of real data streams that were collected near mid-summer 2012. Section 3.3.1 notes that the C/NOFS satellite rises in the West sky and sets in the East, and this fact is very useful when interpreting the reconstructed images and their corresponding data streams. A useful trick is to visualize the image formation as the viewing of a weather radar. At the beginning of the observation each of the receivers, which are located slightly below the center of the bottom image boundary, look towards the C/NOFS satellite on the left side of the image. As time progresses, C/NOFS moves from left to right just above the top of the image, and the receivers' lines of sight follows it on its travel. In addition to this, the receivers' lines of sight sweep radially across the image, just as if they were attached to hinges at the bottom of the plot. Another point to note is that data points at the very beginning and very end of the observation window may not be used in the image reconstruction since the satellite sits very low in the sky at those times, resulting in ray paths that may exit the grid through the sides and be removed, as described in Section 5.3. Finally, we need to remember that each of the data-streams are reported in TEC relative to an unknown additive offset; therefore, the absolute magnitudes of these measurements have little meaning. Instead, the important features in these data-streams are how they change with respect to themselves over the duration of the data collection. Much of the following image analysis is going to refer to both the data-stream plots and the reconstructed images, so it is important to understand these basic notions of how they relate to one another.

June 30, 2012

The first image is created from measurements taken on June 30, 2012, at 11:48 Universal Time. The reconstruction is given in Figure 6.4, the data-streams are given in Figure 6.5, and the regularization parameters are in Table 6.2.

Table 6.2: Parameters for June 30, 2012

Parameter	Value
# Iterations	50
λ_{hor}	1
λ_{vert}	500

The first structure to note in the reconstruction is that the high density in the peak drops

off significantly to the right of -76 degrees longitude. The data-stream shows that this change in peak density occurs very close to when the receiver at Ayacucho stops reporting data. In addition, at 11:51 the data-stream from Puerto Maldonado departs from its suspiciously artificial looking flat-line that it has been reporting up until this point. We now have two possible culprits, and let us first consider the case of the lost readings from Ayacucho. Losing a data-stream at a given time t_l will cause the forward-model $\mathbf{A}$ to become more under-determined at all times $t \geq t_l$. Due to this, the squared residual norm $\|\mathbf{Ax} - \mathbf{b}\|^2$, which was originally summing over four receivers worth of data, will now only sum over three receivers worth. Since all terms in the sum are nonnegative, this means that the residual norm will shrink as soon as a data-stream cuts out. Since the residual norm suddenly becomes smaller and the regularization parameter λ stays the same, the balance of power between the residual and the regularization term will shift in favor of the regularization term. Stated again, when the data-stream from Ayacucho drops out, the residual becomes weaker in its “tug-of-war” with the regularization term. Because of this, the solution will shift towards higher levels of regularization and possible over-smoothing. Another way to look at this happening is that, for the small portion of the image in which all four data-streams are present, the image has the ability to resolve structures that would otherwise remain invisible when observed by fewer receivers. This may explain why the left side of the image is more structured than the rest.

Even though the entrance and exit of data-streams have a large effect on the left side of the image, notice that the exit of Huancayo’s data-stream at 11:57 is barely noticeable in the image reconstruction. This may suggest that the leap from three to four active data-streams makes a large difference in the “tug-of-war” between the residual and the regularization terms, while the jump from two to three data-streams may not affect the balance of power as much.

Next, consider the case in which Puerto Maldonado’s mysterious flat-line may have caused the artifact. This is an example multiple data-streams reporting very different events, and they may fight with the rest of the data, causing undesired artifacts to form.

Since most of the variation in this image can be explained by fairly overt anomalies in the data, it is fairly safe to say that no spread-F bubbles appear to be present. For the most part, the data presented in Figure 5.1 is well-behaved and plays well with the inversion process. The next two data-sets presented are a little more ornery, making their reconstructions a little more difficult.

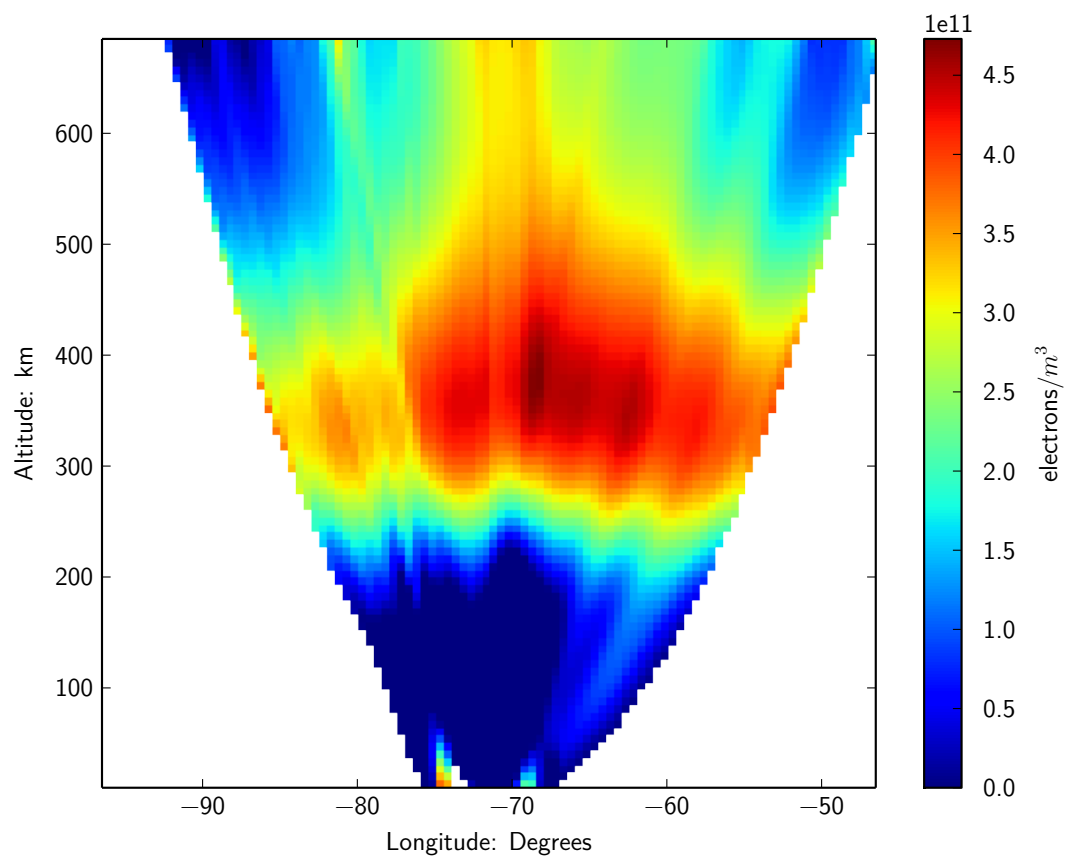


Figure 6.4: Reconstruction: June 30, 2012 11:48 UT

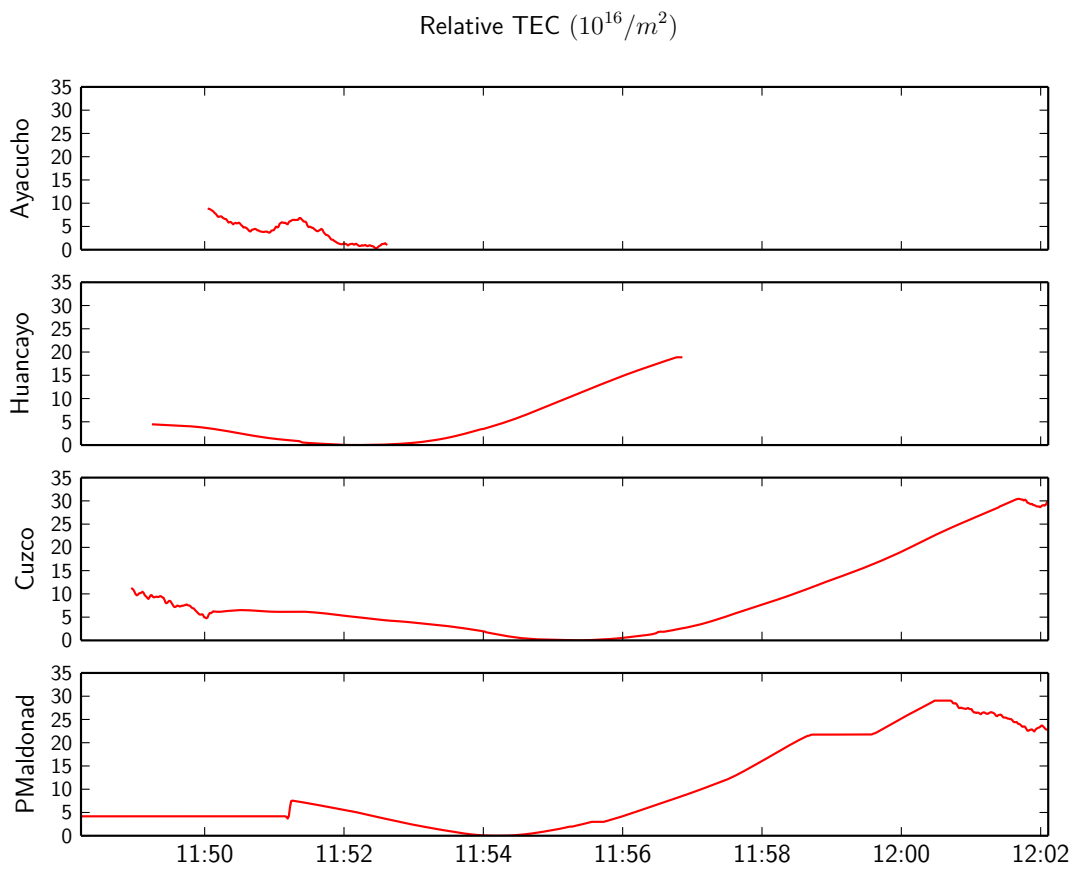


Figure 6.5: Data Stream: June 30, 2012 11:48 UT

June 25, 2012

Table 6.3: Parameters for June 25, 2012

Parameter	Value
# Iterations	50
λ_{hor}	5
λ_{vert}	3000

The next image, given in Figure 6.6, is created from measurements taken on June 25, 2012, at 13:17 Universal Time (UT), with data-stream shown in Figure 6.7, and inverted with the parameters given in Table 6.3.

The most striking feature in the image is the large gap in density right around -75 degrees longitude. Looking at the data-stream in Figure 6.7, the large drop-off in reported density from Huancayo slightly after 13:21 might explain why the giant gap appears in the peak. This also may explain why there is a large area of high density plasma reported up at the top of the image: Huancayo has dropped off and is reporting low measurements, but none of the other data-streams do. In trying to fit the geometry of the reconstructed image to satisfy all four data-streams during this somewhat bizarre portion of the data-reporting, the area of high density was placed way up at the top of the image.

Inside the density gap at -75 degrees latitude, there appear two small patches of higher density. These may hint at the presence of a spread-F bubble, however it is somewhat wishful thinking at this point. To confirm spread-F, we would need more convincing evidence to overturn the more likely alternative that these patches are simply artifacts of the edge-preserving regularization.

The next feature to notice is the fact that the peak density falls off considerably to the right of -65 degrees latitude. At this point, there are only two receivers active, Cuzco and Puerto Maldonado, and they report less and less electron density as they near the end of the observation. This does not make a whole lot of sense physically, but this is what the data says. Also, if there is any useful information in the rightmost portion of the image, it may have been over-smoothed by the regularization, since the regularization term will be much more dominant when only two data-streams are present.

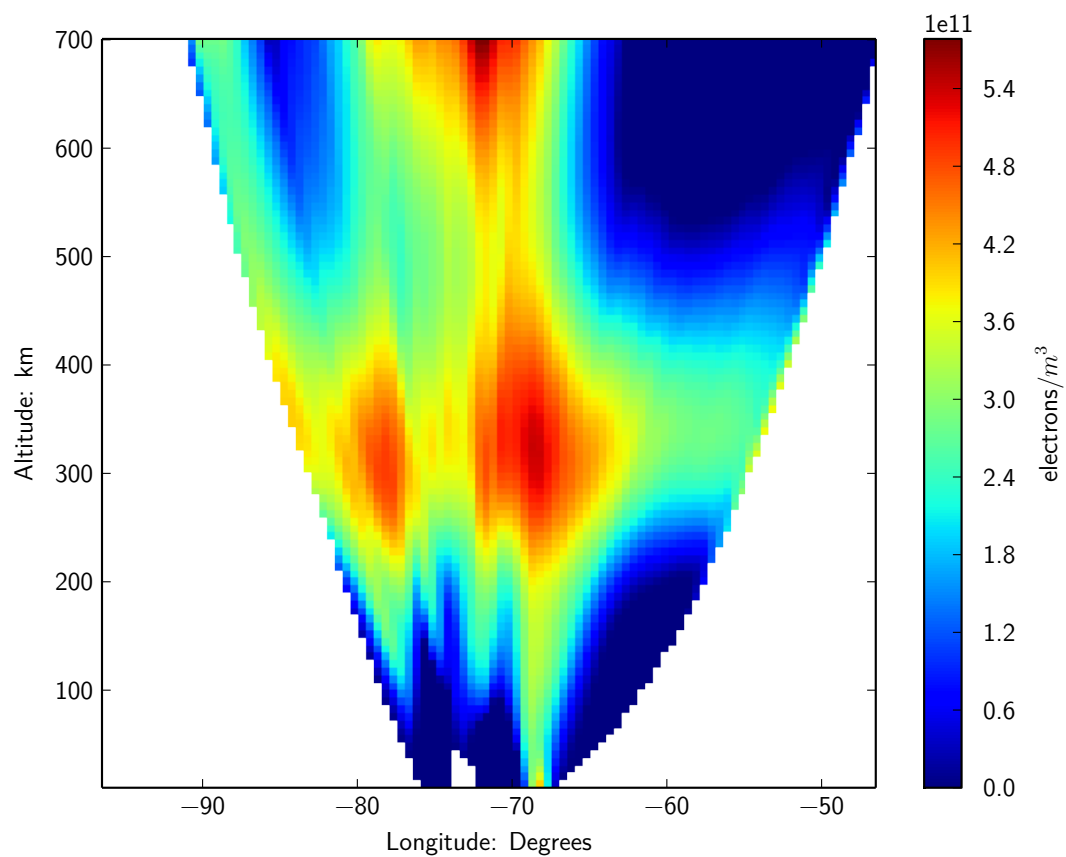


Figure 6.6: Reconstruction: June 25, 2012 13:17 UT

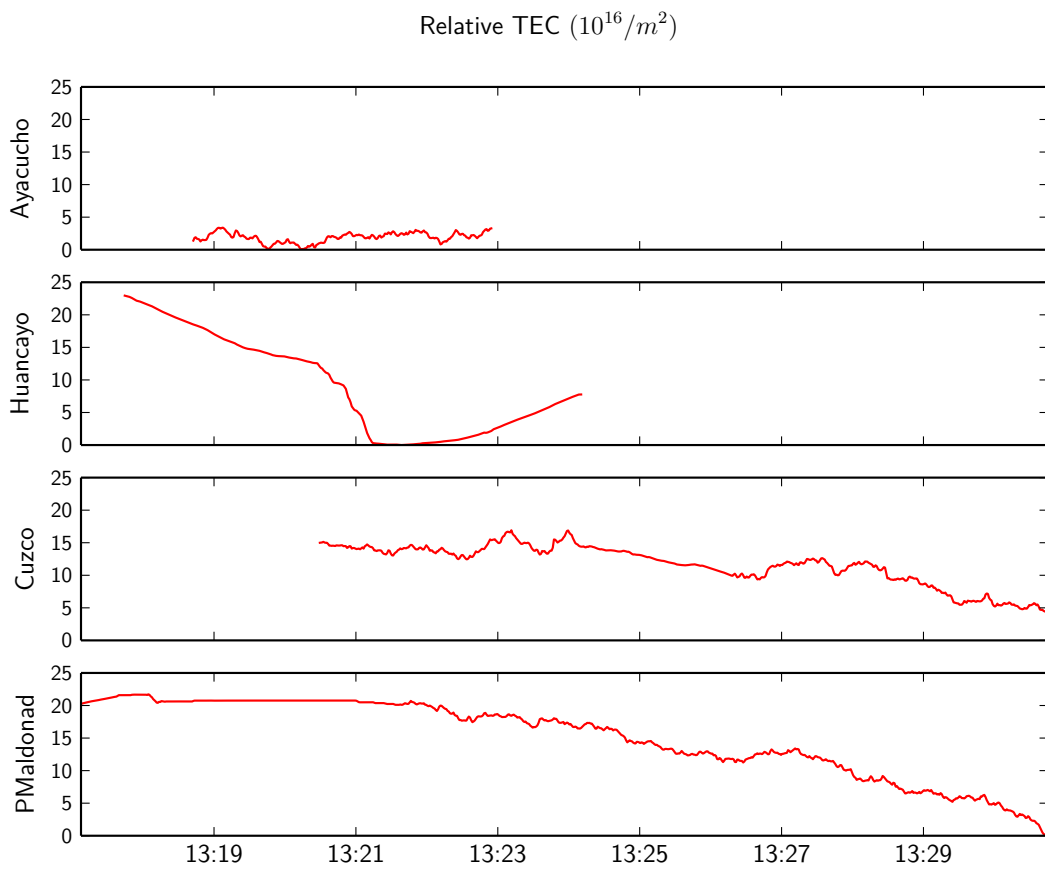


Figure 6.7: Data Stream: June 25, 2012 13:17 UT

July 1, 2012

Table 6.4: Parameters for July 1, 2012

Parameter	Value
# Iterations	50
λ_{hor}	10
λ_{vert}	1500

The final image is created with measurements taken on July 1, 2012 at 17:00 UT, and is shown in Figure 6.8, with inversion parameters listed in Table 6.4, and corresponding data-stream given in Figure 6.9.

Here are some features to notice in the data-stream shown in Figure 6.9: First, Ayacucho generated an observation file that had no data, so its data-stream is empty. Next, notice that Huancayo’s data-stream is very clean and follows the “cup shape”, which is indicative of a quiescent ionosphere. On the other hand, both Cuzco and Puerto Maldonado report somewhat turbulent measurements with both small and large-scale oscillations. Mapping each of these data-streams to the image in Figure 6.8, the reconstruction algorithm tries to reconcile the “clean” measurement from Huancayo with the “dirty” measurements from Cuzco and Puerto Maldonado, and it results in a very ugly looking region of high density plasma appearing at the top of the image where it most definitely does not belong. This artifact is reminiscent of the region of misplaced high density at the top of Figure 6.6, and seems to be created by a somewhat similar problem, in which the reconstruction algorithm attempts to reconcile multiple disparate data-streams. The small region of high density plasma below 100 km altitude at approximately -73 degrees is most likely caused by the same data discrepancy.

Looking to the right of -70 degrees longitude, the data-stream from Huancayo drops out and we are free of its imposing artifacts. The remaining data-streams, Cuzco and Puerto Maldonado, suggest the presence of multiple spread-F bubbles in this region. Looking at the data-stream from Puerto Maldonado in Figure 6.9 for all times past 17:04, there are a lot of tiny oscillations, which are probably caused by observation noise, but there also exist larger scale oscillations with a period of approximately 45 seconds. These larger-scale oscillations indicate that Puerto Maldonado is observing alternating regions of higher and lower TEC. Looking at the spot in Figure 6.8 just to the right of -70 degrees, these oscillations prominently appear in the rendering of the ionosphere peak. Furthermore, the regions of lower density plasma that cut through the peak look very similar to the reconstructions of

simulated spread-F bubbles exhibited in Figure 6.2. These bubbles even cause displaced low-density plasma readings to form at the top of the image, just as discussed in Section 6.1.2. These are all indicators that the structures are indeed caused by spread-F bubbles. The three noticeable bubbles are spaced out approximately 3.5 degrees from each other, which would translate to approximately a 400 km spacing at 350 km altitude. This 400 km spacing lines up with what was discussed in Section 6.1.1.

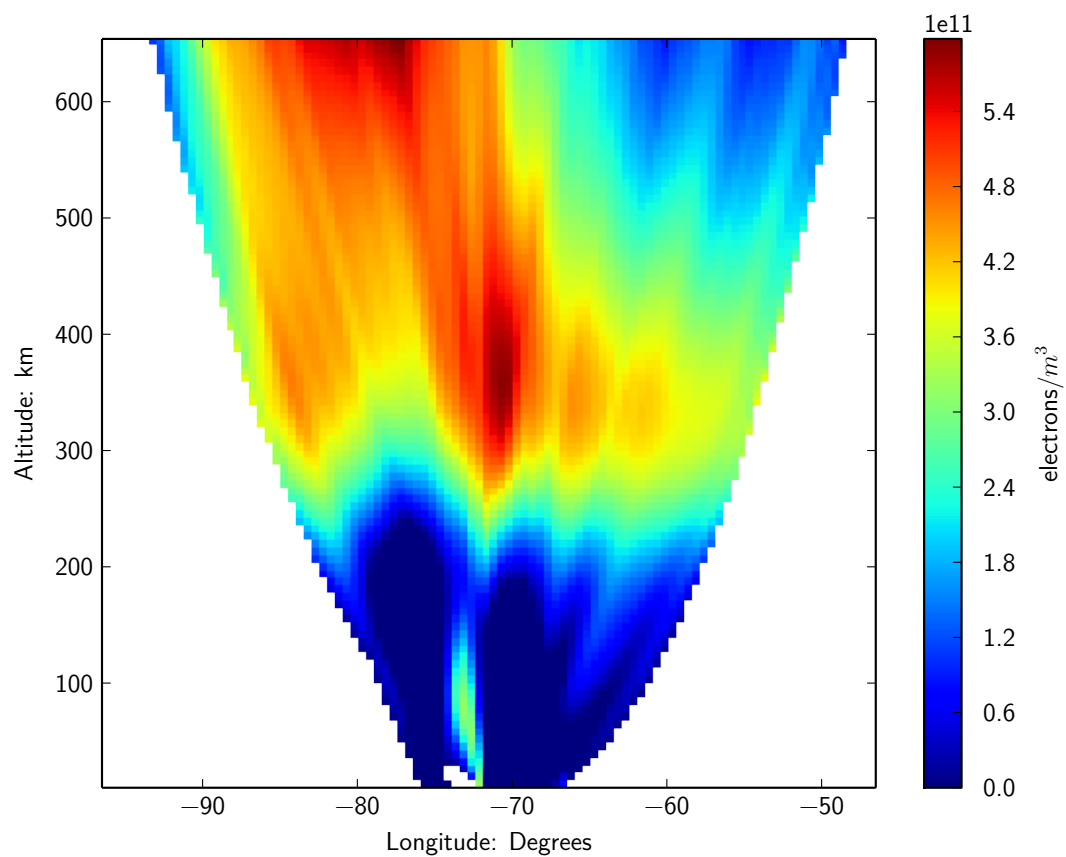


Figure 6.8: Reconstruction: July 1, 2012 17:00 UT

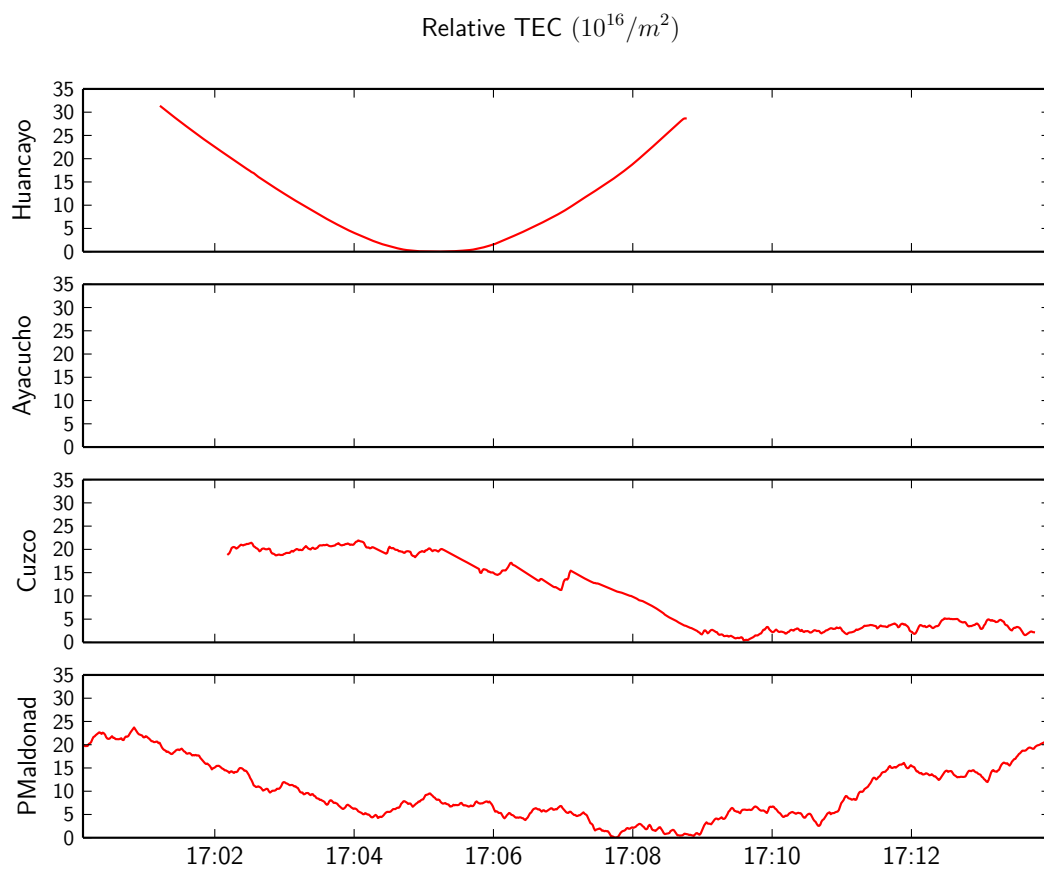


Figure 6.9: Data Stream: July 1, 2012 17:00 UT

6.2 Dependence on the Reference Image

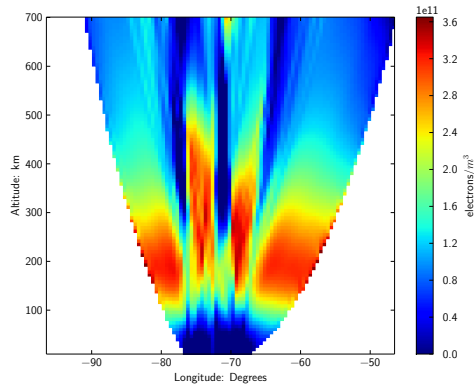
When presented with the results of an inverse problem, a question that every scientist should ask is, “How physically accurate are these results expected to be?” Attempting to answer this question helps us build a mental picture of which parts of the image are trustworthy, which parts should not be taken too literally, and which parts may be improved with further work.

Section 5.4.2 discusses the need to use a reference image to influence the formation of vertical textures in the image. The reference image is needed because the actual data gives very little information about these vertical textures. Section 5.4.2 also mentions that the current form of the algorithm uses the Chapman profile equation (2.2) to generate a simple estimate of the vertical texture. Here we show how changes to the reference images can affect the reconstruction.

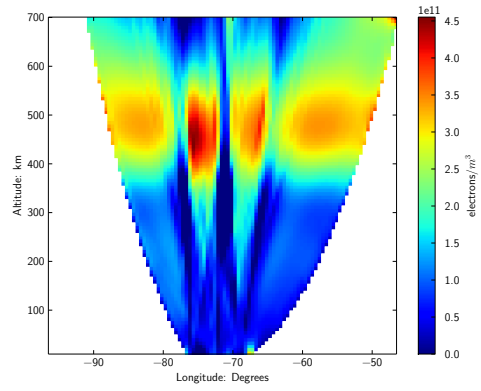
Figure 6.10 displays the test irregularity image (Figure 6.1) reconstructed two different times, in which all parameters are identical except the reference image. In these two reconstructions, only the peak altitude in the Chapman profile is modified. The original image in Figure 6.1 has its background peak is at 350 km, and these two inversions are performed with reference peak altitudes of 200 km and 500 km. In Figure 6.10 both of the reconstructions place the irregularities in the correct place; however, the vertical profile peaks are very different, strictly adhering to what was prescribed by the reference image.

This experiment is repeated in Figure 6.11, but instead the scale height (peak width) of the reference image is changed. The results are very similar to what happened in Figure 6.10. The reconstructed images put all of the irregularities in the correct place, but the background scale heights are very different and closely follow what was provided by the reference image.

Both of these examples show that putting care into choosing the reference image is very important if we desire the vertical profile of the reconstructed images to have any physical meaning. Because of this, it would be very useful to incorporate external data, such as that from the ionosonde at the Jicamarca observatory, in future versions of this inversion algorithm. This extra data will provide more information about the vertical behavior of the ionosphere, and the reconstruction process will use this information to generate images in which the vertical textures are more physically significant.

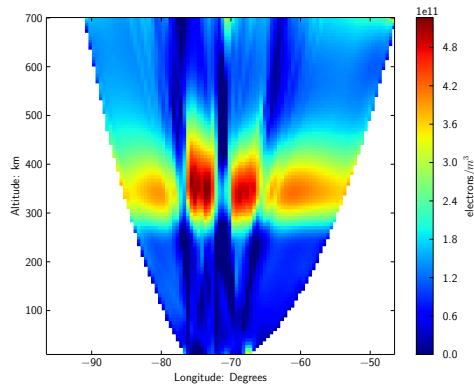


(a) 200 km Peak

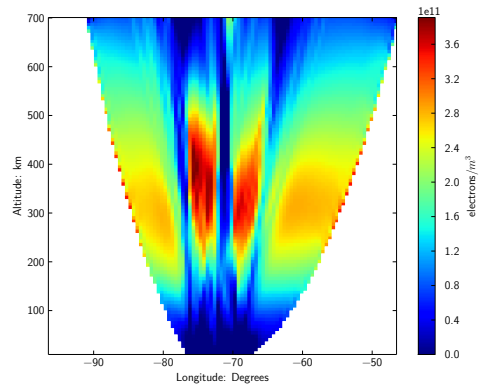


(b) 500 km Peak

Figure 6.10: Various Peak Density Altitudes



(a) 50 km Scale Height



(b) 150 km Scale Height

Figure 6.11: Various Scale Heights

Chapter 7

Conclusion

7.1 Dealing with Missing Data

One of the recurring themes throughout the inversion analysis presented in Section 6.1.3 is that all four reported data-streams did not cover the complete image. Much to the contrary, there were only small portions of each image covered by every data-stream. In addition to this, the abrupt entrance and exit of data-streams during the observation shifted the balance of power between the residual and the regularization term in the reconstruction process, which caused some portions of the image to over-fit the data and other portions to under-fit. These problems are brought on by the fact that our regularization scheme applies the regularization uniformly to the entire image. To fix this problem, we need to extend the regularization scheme to something that can adaptively adjust the amount of regularization to fit the level of available data at any given point in the image. Doing this will eliminate the disparity between regions of data over-fitting and data under-fitting. However, it is no small task since adaptive regularization is still an open problem with a lot of exciting research going on.

7.2 New Questions of Future Interest

The pragmatics of analyzing and generating images from real data sets such as this can help inspire new questions in imaging, whether big or small. This section briefly mentions two questions that this specific problem poses, and some possible approaches to answering them. Both of these questions deal with resolution: First, “What is the best possible resolution given a specific sampling configuration of satellite positions and receivers?” Second, “How can the inversion algorithm be designed to have *Temporal Resolution*, or track changes to the image as a function of time?”

7.2.1 Grid Resolution

Section 3.3 mentions that the inherent physical limitations on the satellites and receivers result in a grid with non-uniform sampling, in which the center of the grid is sampled more often than the outsides. In addition, the receivers stay stationary during the entire data collection, so each region of the grid will be sampled at a different set of angles than all other regions. Because of this, the best possible image resolution varies across the grid. This begs the question, “What is the best achievable resolution given the physical configuration of the receivers and the satellite trajectory?” The center of the grid will certainly have a higher possible resolution than the edges, but to be precise, the question becomes “What are the analytical precision limits on the reconstructed image?”

One approach to this problem is to recast the inversion as maximum a posteriori (MAP) Estimation as described in Section 4.8, and then use different information theoretic tools to place hard bounds on the achievable variance on this estimator, which could be used to create bounds on the possible imaging resolution. After this, the next step is attempting to achieve these analytical resolution bounds with a practical system that can be used with real inversions, possibly designing an adaptive grid generation tool that would tailor the imaging grid to the specific distribution of information given by the acquisition geometry.

7.2.2 Temporal Resolution

Decades ago, panoramic photos could not be taken in one single frame; instead, the photographer would pan the camera from one side to the other to collect the entire image. Suppose the photographer panned the camera from left to right: a mischievous kid would start by standing on the left side, quickly run behind all of the adults while the camera was panning through the middle, and then get in position on the right side just in time to be imaged by the camera again. Because of this, the kid appears in the photo twice, much to his mother’s dismay. The reason that the family photo depicts something physically impossible is that the imaging system incorrectly assumed that the quantity being imaged, in this case the family, remained stationary over the entire image acquisition. However, this assumption was incorrect, so the generated image will contain artifacts, which in the case of the family photo, the little kid appears twice.

Panoramic photos are not usually taken in that fashion these days, so little kids will need to find new ways to cause their parents distress. However this assumption of stationarity still generates artifacts with modern day cameras. Consider the case of photographing a sprinter

at close range. If the shutter speed and the film speed are too slow, the photograph will not capture a clean image of the sprinter, but rather a blurred one, in which the sprinter is smeared across the photo. Again, this happens because of the assumption that the quantity being imaged remains static for the entire image acquisition.

The example of the sprinter and the slow film is analogous to what may happen in the imaging of the ionosphere. With the C/NOFS satellite, it takes approximately ten minutes to collect all of the data to form an image, and the current version of the reconstruction algorithm assumes that the ionosphere’s electron density remains relatively static for the entire duration.

In the case of a quiescent ionosphere, very little does happen, so this static assumption is perfectly valid. However, spread-F bubbles may move around and change shape over this ten minute period. If this motion is significant, the static imaging assumption may be invalid, causing “smearing” artifacts in the reconstructed images. These artifacts are the same as with the photographing of a sprinter; the spread-F bubbles will be smeared across part of the image, which will occlude many of the bubbles’ fine details.

The way to mitigate these smearing artifacts is to employ an imaging model that takes into account the dynamics of the underlying system. Building an imaging model such as this involves first generating an underlying physical model that approximates the system dynamics and then estimating this model’s parameters during the image reconstruction process.

One big cause of spread-F bubble evolution is the presence of horizontal winds in the ionosphere[5]. Because of this, spread-F bubbles may get blown sideways during the data collection process. This sideways motion can cause the smeared plumes to appear wider in the image than they actually are, and it can hide many of the bubble’s horizontal textures. A very simple way to correct the effects of horizontal wind is to assume that each bubble has a constant shape, but moves horizontally across the image at a constant velocity. If this velocity can be estimated beforehand, the only needed algorithmic modification is to “put the receivers on roller-skates”, that is, to pretend that the receivers move horizontally relative to the grid at the estimated speed and direction of the wind. If this is done, the bubbles will appear to stand still despite getting blown by the wind.

If we are unable to provide an a-priori estimate of the wind velocity, then the wind velocity must be estimated during the image inversion. One effective way to do this is to use a recursive estimator such as a Kalman filter during the reconstruction process. [27] gives a basic explanation of how Kalman filtering can be used during image reconstruction. Using a

Kalman filter, we also gain the ability to use more complex models of ionosphere dynamics to provide even better removal of smearing artifacts. However, the computational cost of the Kalman filtering approach can become prohibitive when the dimension of the physical model becomes too large.

7.3 Concluding Remarks

This thesis developed, demonstrated, and analyzed the very first application of edge-preserving regularization techniques to the C/NOFS-CERTO dataset. These regularization techniques combine real data together with scientific models of the ionosphere to create a global inversion cost function which can be minimized to generate a solution that strikes a good balance between fitting the data and fitting the models.

These techniques, which use an explicitly defined cost function, give the scientist a large degree of control over the inversion process since the regularization parameters can be carefully tuned to fit the specific geometry and noise-level of the data. In addition, this thesis provides analysis of an edge-preserving algorithm in action, working on the C/NOFS-CERTO data, and then gives a discussion of how parameter choices and algorithm idiosyncrasies affect the reconstructed images. This analysis can be used as a basic set of guidelines from which scientists can make a more informed analysis of images generated with the C/NOFS-CERTO data. Scientists can do this by understanding which portions of images are generated by physical occurrences, and which portions may just be artifacts of the inversion process.

Due to the fact that tomography of the ionosphere is an inverse problem with many unknown factors, good intuition is necessary to help filter out what in an image is significant and what is not. As research in this style of tomography progresses, scientists will quantify more and more of these unknown factors, and this will allow them to rely less on intuition and more on provable facts, which will only improve the quality of images that they can generate.

This thesis provides the first step of many toward building interesting and useful images from the C/NOFS-CERTO dataset. It also provides additional insight into the world of limited-angle tomography as a whole.

References

- [1] F. Kamalabadi, “Tomographic remote sensing of space plasmas using spectroscopic measurements,” Ph.D. dissertation, Boston University, 2001.
- [2] E. Kudeki, “Applications of radiowave propagation,” 2010, UIUC ECE 458 Lecture Notes.
- [3] P. C. Tzanos, “Tomographic reconstruction of ionospheric electron density from space and ground based total electron content measurements,” M.S. thesis, University of Illinois at Urbana-Champaign, 2002.
- [4] Wikimedia Commons, “Earth’s atmosphere and ionosphere,” 2007, accessed 24-March-2014. [Online]. Available: http://en.wikipedia.org/wiki/File:Atmosphere_with_Ionosphere.svg
- [5] M. C. Kelley, *The Earth’s Ionosphere - Plasma Physics and Electrodynamics*, ser. International Geophysics. Elsevier, 2009, vol. 43.
- [6] O. de La Beaujardière, “C/NOFS: a mission to forecast scintillations,” *Journal of Atmospheric and Solar-Terrestrial Physics*, vol. 66, no. 17, pp. 1573 – 1591, 2004. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1364682604001695>
- [7] R. E. Blahut, *Theory of Remote Image Formation*. Cambridge University Press, 2004.
- [8] Wikimedia Commons, “Radon transform,” 2005. [Online]. Available: http://upload.wikimedia.org/wikipedia/commons/5/5d/Radon_transform.png
- [9] M. Van de Kamp, O. Amm, and K. Kauristie, “3D ionospheric tomography using GEOTRIM GPS measurements,” accessed 24-March-2014. [Online]. Available: <http://www.space.fmi.fi/MIRACLE/Geotrim/index.html>
- [10] Air Command and Staff College Space Research Electives Seminars, *AU-18 Space Primer*. Maxwell Air Force Base, Alabama: Air University Press, 2009.
- [11] M. C. Kelley, J. J. Makela, O. de La Beaujardière, and J. Retterer, “Convective ionospheric storms: A review,” *Reviews of Geophysics*, vol. 49, no. 2, 2011. [Online]. Available: <http://dx.doi.org/10.1029/2010RG000340>

- [12] P. A. Bernhardt and C. L. Siefring, “New satellite-based systems for ionospheric tomography and scintillation region imaging,” *Radio Science*, vol. 41, no. 5, p. RS5S23, 2006. [Online]. Available: <http://doi.wiley.com/10.1029/2005RS003360>
- [13] Y. Bresler, S. Basu, and C. Couvreur, “Hilbert spaces and least squares methods for signal processing,” January 2013, UIUC ECE 513 Lecture Notes.
- [14] P. C. Hansen, *Rank-deficient and Discrete Ill-posed Problems: Numerical Aspects of Linear Inversion*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1998.
- [15] R. Aster, B. Borchers, and C. Thurber, *Parameter Estimation and Inverse Problems (International Geophysics)*. Elsevier Academic Press, Jan. 2005.
- [16] R. Baraniuk, “Compressive sensing [lecture notes],” *IEEE Signal Processing Magazine*, vol. 24, no. 4, pp. 118–121, July 2007.
- [17] A. H. Delaney and Y. Bresler, “Globally convergent edge-preserving regularized reconstruction: an application to limited-angle tomography,” *IEEE Transactions on Image Processing*, vol. 7, no. 2, pp. 204–21, Jan. 1998. [Online]. Available: <http://www.ncbi.nlm.nih.gov/pubmed/18267394>
- [18] C. Byrne, “The multiplicative algebraic reconstruction technique solves the geometric programming problem,” University of Massachusetts Lowell, Tech. Rep., 2007. [Online]. Available: <http://faculty.uml.edu/cbyrne/dgp2.pdf>
- [19] M. Hei, P. Bernhardt, C. Siefring, J. Huba, J. Krall, C. Valladares, R. Heelis, M. Hairston, W. Coley, J. Chau, C. D. L. Jara, and M. Wilkens, “Radio tomographic images of post-midnight equatorial plasma depletions.” presented at USNC-URSI National Radio Science Meeting, Boulder, CO, 8 January 2014.
- [20] T. S. Kelso, “Frequently asked questions: Two-line element set format,” accessed 29-March-2014. [Online]. Available: <http://celestrak.com/columns/v04n03/>
- [21] G. Cai, B. M. Chen, and T. H. Lee, *Unmanned Rotorcraft Systems*. Springer, 2011.
- [22] J. R. Shewchuk, “An introduction to the conjugate gradient method without the agonizing pain,” Pittsburgh, PA, Tech. Rep., 1994.
- [23] G. H. Golub and C. F. Van Loan, *Matrix Computations*. Johns Hopkins University Press, 1996.
- [24] J. K. Lee and F. Kamalabadi, “GPS-based radio tomography with edge-preserving regularization,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 1, pp. 312–324, 2009.

- [25] Y. Saad, *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, 2003.
- [26] J. J. Makela, S. L. Vadas, R. Muryanto, T. Duly, and G. Crowley, “Periodic spacing between consecutive equatorial plasma bubbles,” *Geophysical Research Letters*, vol. 37, no. 14, July 2010. [Online]. Available: <http://doi.wiley.com/10.1029/2010GL043968>
- [27] M. D. Butala, R. A. Frazin, Y. Chen, and F. Kamalabadi, “Tomographic imaging of dynamic objects with the ensemble kalman filter,” *IEEE Transactions on Image Processing*, vol. 18, no. 7, pp. 1573–87, July 2009. [Online]. Available: <http://www.ncbi.nlm.nih.gov/pubmed/19447717>