RADIO AURORA EXPLORER
NANO-SATELLITE RADAR: DATA INVERSION

A project to characterize the resolution of the new NanoSat Radar.

Performed by Worcester Polytechnic Institute

In co-ordination with SRI International

Project Report Prepared By

Prawal Shrestha   Ashish Maharjan   Hunain Kapadia

28th April 2009

Advisors:

Dr. Hasan Bahcivan
SRI International

Provost John A. Orr
Worcester Polytechnic Institute

ABSTRACT

SRI International, in coordination with the University of Michigan, is leading the Radio Aurora eXplorer (RAX) mission. This mission aims to measure the intensity of plasma irregularities in the auroral ionosphere. Our Major Qualifying Project was concerned with constructing a realistic simulation of the bi-static satellite-radar system and generate synthetic radar data. Moreover, it was also required to develop de-convolution algorithms to invert the synthetic radar data so as to deduce the intensity and magnetic field alignment of the irregularities.

Table of Contents

Table of Figures

Executive Summary

The Radio Aurora eXplorer (RAX) mission is the first satellite mission funded by the National Science Foundation (NSF). This mission has Dr. Hasan Bahcivan of SRI International as the principal investigator and Dr. James Cutler of University of Michigan as the co-principal investigator. The satellite is scheduled to be launched in December 2009 from Kodiak, Alaska; launch is supported by the Department of Defense, Space Test Program.

The primary objective of the RAX mission is to characterize the turbulence in the form of plasma irregularities of the ionosphere in the auroral regions. Plasma turbulence degrades communication and navigation signals and studying it would help improve current communication and navigation technologies. In addition to this, the components of the turbulence parallel to the magnetic field lines dissipate immense energy due to the joule heating phenomenon that takes place. The amount of energy dissipated by the amount of turbulence is related to the alignment of these irregularities.

Our Major Qualifying Project (MQP) is a feasibility check for the RAX mission. The goal of our project is to construct a realistic simulation of the ground-to-space bi-static satellite-radar system and determine how well we can resolve the ionospheric irregularities in altitude and their alignment with the geo-magnetic field lines. Another key focus of the MQP is to develop robust and accurate de-convolution algorithms to invert synthetic radar and satellite data obtained from the simulation in order to determine how well the intensity information of the irregularities can be deduced and also refine the angular resolution of the bi-static satellite-radar system.

The Poker Flat Incoherent Scatter Radar (PFISR) directs an electromagnetic beam towards the ionosphere. The ionospheric irregularities which lie on the path of this beam scatter the incident electromagnetic energy in the shape of hollow cones. As the RAX satellite moves in its orbit, it passes through the conical scattering regions and takes measurements. The measurements are tagged with altitudes (h1 & h2) and aspect angles of the scattering targets. The aspect angle tells us how well the irregularities are aligned with the geo-magnetic field lines.

The PFISR was built at SRI and is now located in Poker Flat, Alaska. It has been used by scientist for various space weather related experiments and can be basically thought of as the function generator in our experiment. The PFISR beam is about 1°. The narrow beam width is extremely important as it allows for higher resolution measurements.

We generated an ambiguity function using the synthetic radar data. It tells us that our altitude resolution is around 5 km while our angular resolution is around 1° to 2°.

Our whole experiment is characterized by the following equation:

$$S_{i,j} = \sum_{h,\theta} \omega_{i,j}\ (h,\theta).\, I(h,\theta)$$

Where I(h, θ) represents the irregularity intensity function, which is assumed to be modeled as a Gaussian and $\omega_{i,j}\ (h,\theta)$ is the Point spread function. $\omega_{i,j}\ (h,\theta)$ quantizes the electromagnetic beam transmitted by the PFISR and determines which part of the beam is illuminating which part of the ionosphere. Convolving these two functions gives us $S_{i,j}$**,** the data sample gathered by the satellite. Here, i represents the index of the pulses that are sent from the radar and j represents the time difference (or index) between the sent and received signals.

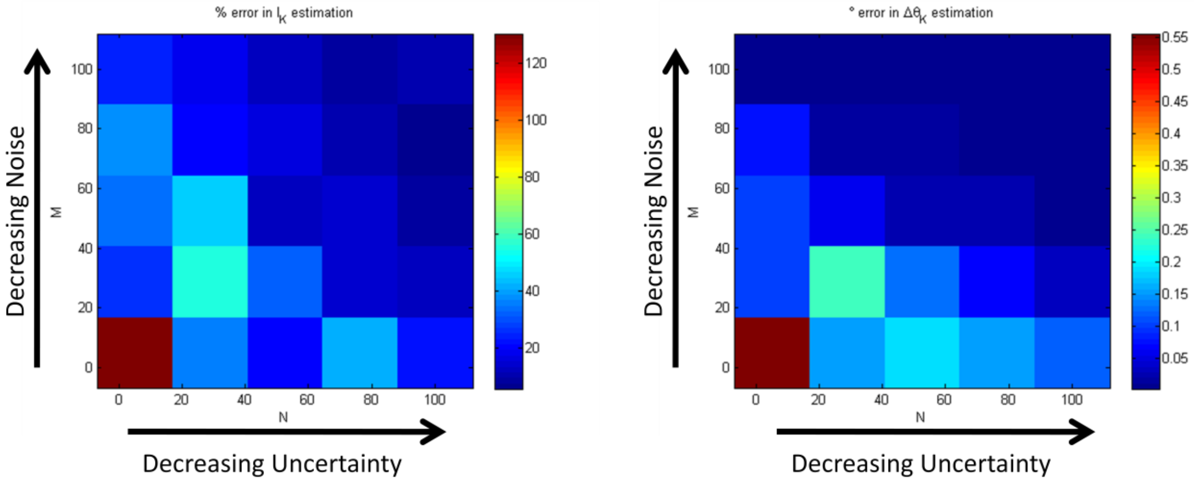
We then add noise and uncertainty to $S_{i,j}$, to compensate for the noise and uncertainty that will be present in the actual samples. The uncertainty is due to the random nature of the scattering process and noise is the receiver additive noise.

Although $S_{i,j}$, has noise and uncertainty, we know precisely the $\omega_{i,j}\ (h,\theta)$ function. Starting from the two functions, we will to determine the parameters of I(h, θ) as accurately as possible by deconvolution. The accuracy of the estimated irregularity parameter was tested by re-synthesizing the $S_{i,j}$ and comparing it to the actual one.

For the deconvolution, three different algorithms were designed and tested. The first one was the exhaustive method which went through all the possible values of the parameters and finds the ones that best fit the actual model. The second was the steepest descent, which involved making an educated guess and looking around it and taking the path with least error to the ideal value. Although accurate, both these processes were computationally intensive and required too much time.

The third method that we used was Newton’s method with least square error minimization, which was both accurate and fast. It involved making an educated guess of the parameter values using exhaustive method against lower resolution data, finding the correction values and then subtracting the correction values from the first guess to find the next guess. The process of finding the correction and deducting it from the guess was repeated until the guess converges to a final value. Due to the presence of error and uncertainty we had certain margin of error in our results.

The following graphs highlights the error after deconvolution has been performed using Newton's method for a range of noise and uncertainty values.



Plot of error in the deconvolution results

The graph shows that for low noise and uncertainty values our estimates of I_k and θ_k are highly accurate. Even for higher noise and uncertainty values, our estimates are fairly accurate, which verifies the robustness and effectiveness of our algorithm. In general, we achieved a 10% error margin in irregularity intensity and 0.1° error margin in aspect angle. This high resolution in aspect angle has never been achieved before in any rocket or satellite based measurement system of magnetic field alignment of ionospheric irregularities. This will be the first time that it's being done using a bi-static satellite-radar system. Conventionally it's been done using mono-static radar which did not allow for high resolution in aspect angle (10° maximum).

The RAX data inversion for mapping ionospheric irregularities has been successfully completed. A realistic simulation of the nano-satellite radar system has been constructive and ambiguity functions for changing satellite positions and PFISR beam direction have been designed where we achieved up to 5 km of altitude resolution and 1-2° of angular resolution. The de-convolution code to compile multiple satellite readings to highly resolve the irregularities in altitude and aspect angle has been tested and verified. It is accurate for up to 0.1° angular accuracy and 4 km of altitude resolution with 20% error margin in irregularity intensity. The de-convolution code could be further built on for more robustness and speed using the same principles and will be further worked on by the staff at SRI International.

1 Introduction

Our Major Qualifying Project is a complement to SRI International's Radio Aurora eXplorer (RAX) mission. It will verify the feasibility of the RAX mission by creating a realistic simulation of the experiment. In this section we will introduce both projects (WPI and RAX) and discuss their objectives.

1.1 Radio Aurora Mission

SRI International's Radio Aurora Mission will implement an innovative idea of using a satellite-radar combination to study the plasma irregularities in the ionospheric irregularities. Understanding the plasma irregularities will help improve technologies in fields of radio communication and navigation.

1.1.1 Science objectives

The primary objective of the RAX mission is to characterize the ionospheric turbulence in the auroral regions. It aims to determine the altitude distribution of high-latitude irregularities as a function of the convection electric field (E_C) and also determine the alignment of these irregularities with the Earth's magnetic field lines. Moreover, the mission will try to quantize the amplitudes of wave parallel electric fields ($E_{||}$) and the electromagnetic energy dissipated by the $E_{||}$ in the form of joule heat dissipation.

1.1.2 A schematic of the experiment

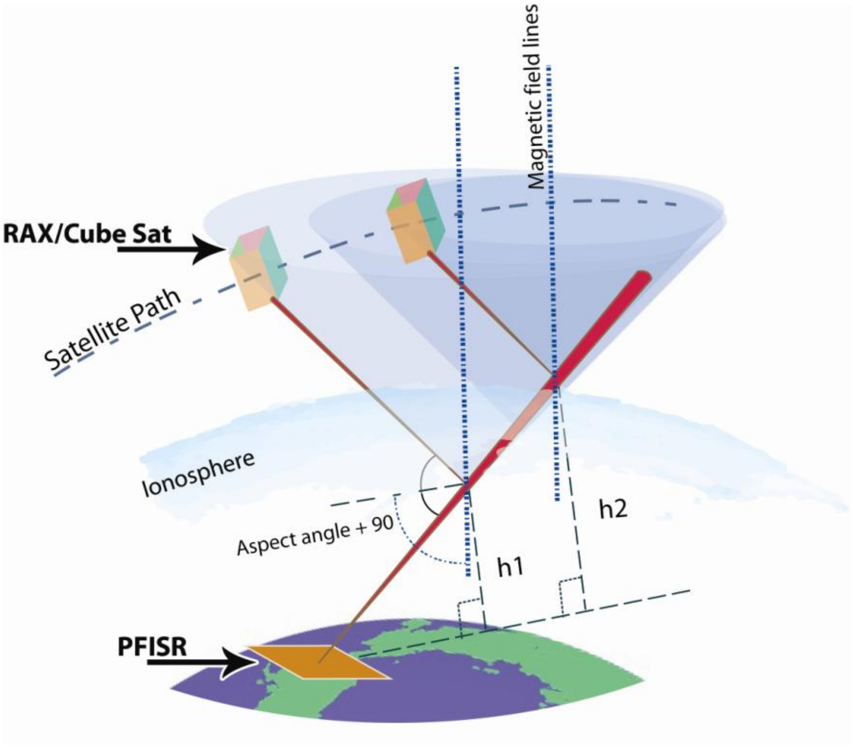


Figure 1.1.1 A schematic of the experiment

As it can be seen in Figure 1.1-1, the PFISR is used to transmit an electromagnetic beam towards the ionosphere. Most of that energy passes through the ionosphere while a very small fraction of it is scattered by the irregularities present in the ionosphere. The scattering occurs in a hollow conical shape. As the satellite orbits the earth, it passes through these conical regions, taking measurements of the amount of energy scattered at different altitudes, h, and aspect angle, θ. The aspect angle is a measure of how well the irregularities in the ionosphere are aligned with the magnetic field of the earth. Quantitatively, it is 90° minus the angle between the magnetic field lines and the bisector of the angle between the incident beam and refracted beam. Taking these measurements allow us to determine the intensities and aspect angle width of the ionospheric turbulence in the auroral regions.

1.1.3 The role of the PFISR

The Poker Flat Incoherent Scatter Radar (PFISR) is used to illuminate the ionospheric irregularities. It is used to direct a known amount of energy in a known direction in the ionosphere. The energy dissipated is reflected off the irregularities in the ionosphere in the direction of the satellite. The PFISR has the capability of generating a very narrow beam of energy which can easily be electronically steered in azimuth, elevation and intensity, without involving any mechanical parts. Sending a large amount of radar power in a narrow beam is of paramount importance for our project since it affects the sensitivity and resolution of the whole system.

1.1.4 The role of the RAX

Set to be launched in December 2009, the Radio Aurora eXplorer (RAX) is used to receive the scattered radiation, referred to as “radio aurora”, from the ionosphere and perform on board radar signal processing in space. The principle payload is a radar receiver which will measure the amplitude and phase of the radio aurora along the satellite track. The signals will be sampled at 1 MHz and stored into a 1.20 GB radar buffer memory where they will be post-processed by the payload processor. The satellite will orbit the earth on a circular orbit at the height of 650 km from the ground with inclination 72° at a speed of approximately 8000 m/s.

1.2 Objectives

The objective of the Radio Aurora Explorer (RAX) satellite mission is to map ionospheric irregularities as a function of altitude and aspect angle with high resolution. The goal of our Major Qualifying Project is to determine how well these irregularities can be resolved in altitude and aspect angle and to characterize their alignment with the geo-magnetic field lines.

The project involves constructing a realistic simulation of the ground-to-space bi-static system and in the process determining the PFISR and RAX radiation pattern, developing robust geometry to accurately locate irregularities and determining scattering zones of these irregularities. The synthetic radar data that is measured by the satellite will then be de-convolved to map irregularity intensities and characterize the altitudinal and angular resolution of the bi-static PFISR-RAX system.

2 Background

2.1 The Auroral Electrojet

Arroral electrojet refers to the current that flows in the D and E regions of the auroral ionosphere.

2.2 Ionospheric irregularities

Ionospheric irregularities are plasma irregularities that lie between the altitudes of 80 km and 500 km in the thermosphere. The auroal electrojet irregularities are densest near the 100 km altitude. The motion of the ionized particles, in the electrojet cause turbulence in the E region. The turbulence interferes with communication and navigation signals, as such, studying and understanding the irregularities will help improve the respective technologies. Moreover, due to the irregularities (especially those parallel to the magnetic field lines), immense energy is dissipated in form of joule heating. As such the irregularities have been the subject of interest for space physicists.

2.3 Radio aurora

The aurora is the colorful display of lights that can be observed in the Northern and Southern Polar Regions. Solar winds carrying charged atomic particles flow towards Earth, and the charged particles collide with the atomic particles of the magnetosphere of Earth's ozone. This generates visible electromagnetic radiation which forms the natural display of lights visible in the Polar Regions. The radio aurora is the scattered radiation from ionospheric irregularities illuminated by a radar.

2.4 PFISR

The Poker Flat Incoherent Scatter Radar is located in Poker Flat, Alaska and is used by scientists to study the upper atmosphere and the ionosphere. This radar was built in a joint collaboration between the National Science Foundation and SRI and is now operated by SRI. The radar is built using a phased-array antenna system which can be remotely controlled by scientists and the radar beam can be electronically steered instantaneously to sense multiple regions of the ionosphere.

The radar consists of 128 panels. Each panel consists of 32 antenna element units placed in an 8 x 4 rectangular array. The whole radar consists of 4,096 antennas as shown in Figure 2.4.1. The antennas are of the crossed dipole type with two dipoles placed at 90 degrees to each other and excited with currents which are 90 degrees out of phase (circular polarization) and can be operated at a frequency of 449MHz. Since scientists can select which antenna's to turn on and which to turn off, the radar is capable of generating different antenna patterns. PFISR is placed so that it faces near the direction of the magnetic north.

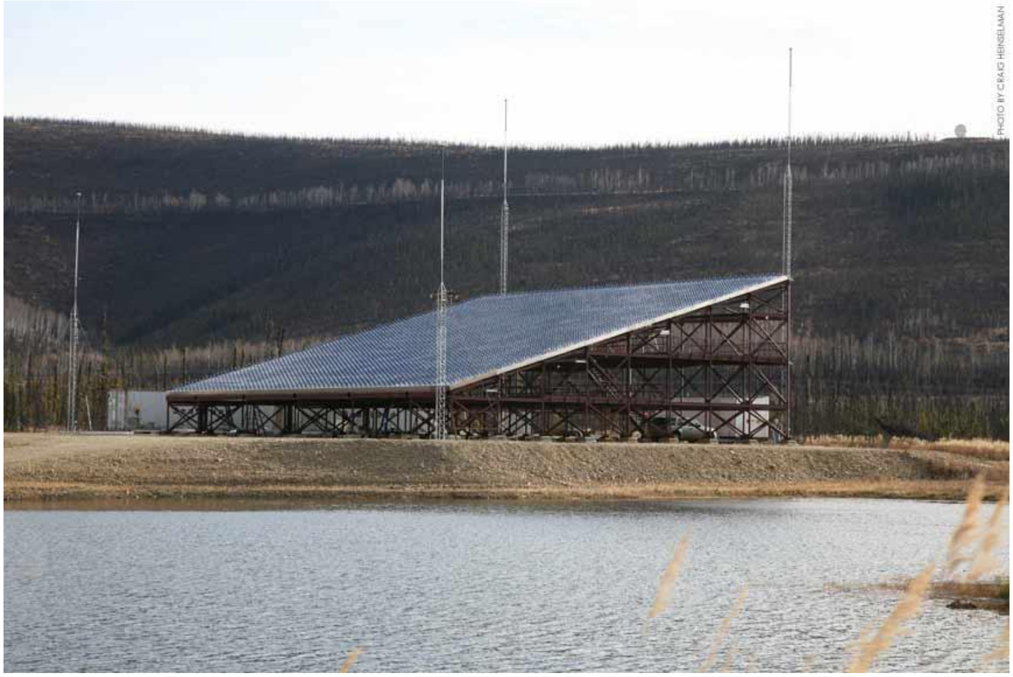


PHOTO BY CRAIG HEINSELMAN

Figure 2.4.1 Poker Flat Incoherent Scatter Radar (PFISR)

By changing the phase of the current in the different antenna elements the radar beam can be electronically steered.

2.5 Coordinate System

The geometry involved in this project uses Cartesian and Spherical coordinate systems. Also, in order to represent the locations of radar, irregularities, and satellites we use Geocentric and Geodetic system.

2.5.1 Cartesian Coordinate System

Cartesian coordinate system contains three axes X, Y, and Z that divides space into eight sub-divisions called octants. The system uses three values to uniquely identify a point in space: x-coordinate, distance from the X axis; y-coordinate, distance from the Y axis; z-coordinate, distance from the Z axis. These values can be either positive or negative. The intersection point of the three axes is the origin where all three coordinates x, y and z are zero. Figure 2.5.1 shows the axes, eight octants and how a point in space is represented in x, y and z coordinate.

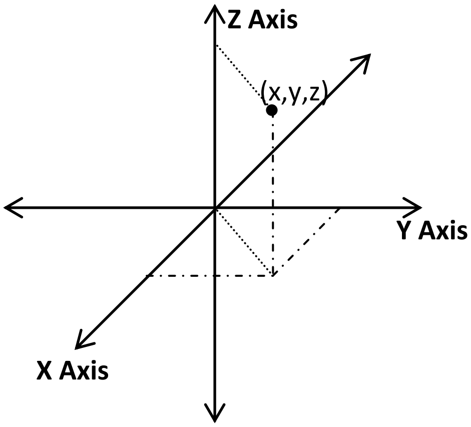


Figure 2.5.1 Cartesian coordinate system

2.5.2 Spherical Coordinate System

Spherical coordinate system, like the Cartesian system helps to represent a point in space. However, the coordinates used are different: radius (r), the distance of the point from the origin; elevation (θ), the positive angle from the z axis to the point; azimuth (φ), the angle from the positive x-axis to the orthogonal projection of the point in the x-y plane. θ ranges from 0° to 180° while φ ranges from 0° to 360°. Figure 2.5.2 shows how a point in space is represented in spherical system.

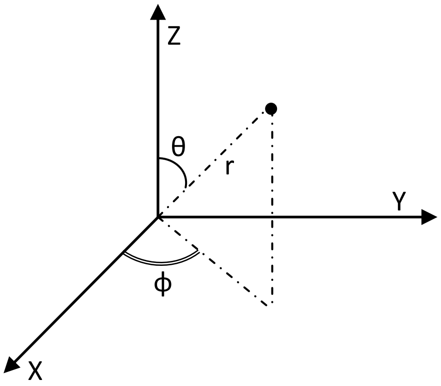


Figure 2.5.2 Spherical coordinate system

2.5.3 Geocentric System

Geocentric system is similar to spherical coordinate system. It uses radial distance, longitude (azimuth in spherical system) and longitude (elevation in spherical system) to represent a point. The only difference is that the geocentric latitude (elevation) is the positive angle from the equator (x-y plane) to the radial distance. In this system, the origin represents the center of the earth. The Z axis is parallel to the earth's rotational axis and points to the geographic North and the X axis lies on the equator of earth and passes through the prime meridian. Also the Earth is considered to be a perfect sphere.

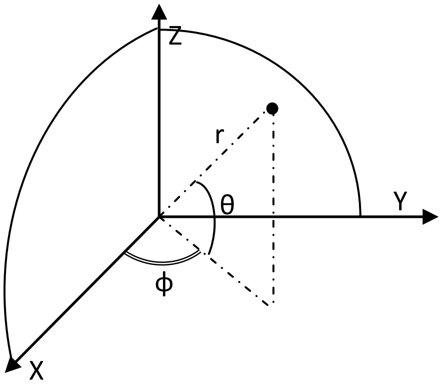


Figure 2.5.3 Geocentric system

2.5.4 Geodetic System

Geodetic system is another coordinate system which can be used to represent a location. It approximates the surface of the Earth as an ellipsoid and uses altitude (h), latitude (θ) and longitude (φ) to describe a point. Geodetic altitude is the length of the perpendicular projection from the point to the surface of the ellipsoid. Geodetic latitude is different from geocentric latitude; it is the angle between the equatorial plane and the projection perpendicular to the surface of the ellipsoid. Figure 2.5.4 shows the representation of both altitude and longitude. Geodetic longitude is the same as geocentric longitude.

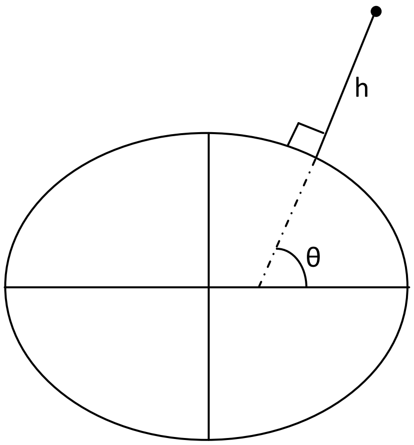


Figure 2.5.4 Altitude and elevation of geodetic system

2.6 Scattering Geometry

The incident electromagnetic beam generated by the PFISR, which is pointed at certain elevation (ε) and rotates about certain azimuth (α), is narrow with a beam width around three degrees and has a spherical beam front. The irregularities falling within the path of the incident beam scatter the signal in the form of a hollow cone shape. The RAX passes through each cone and takes readings. This geometry is shown in **Error! Reference source not found.**. Each reading of the satellite is tagged with certain volume of incident beam, aspect angle, altitude and geodetic location of ionospheric irregularity. Determining these parameters involves complex geometry which is discussed in this section. The geometry involved can be divided into the following steps:

- Determining the geocentric location of the irregularity pointed by bore sight(center of the beam)
- Computing the aspect angle at the geocentric location of the irregularity
- Dividing the volume of incident beam into pieces which is uniform in spherical domain
- Finding location, altitude, aspect angle and j index of each volume piece

We have prepared software in FORTRAN that performs the above steps to provide us with the necessary information. Detailed descriptions of the steps are provided as follows.

2.6.1 Computing irregularity's geocentric coordinates highlighted by bore sight

We are trying to find the geocentric coordinates of the ionospheric irregularities illuminated by the center of the radar beam; the beam pointing at certain elevation (ε) and rotate about certain azimuth (α). Figure 2.6.1 shows different vectors and their geocentric longitudes that are used for the analysis of the location of the irregularities.

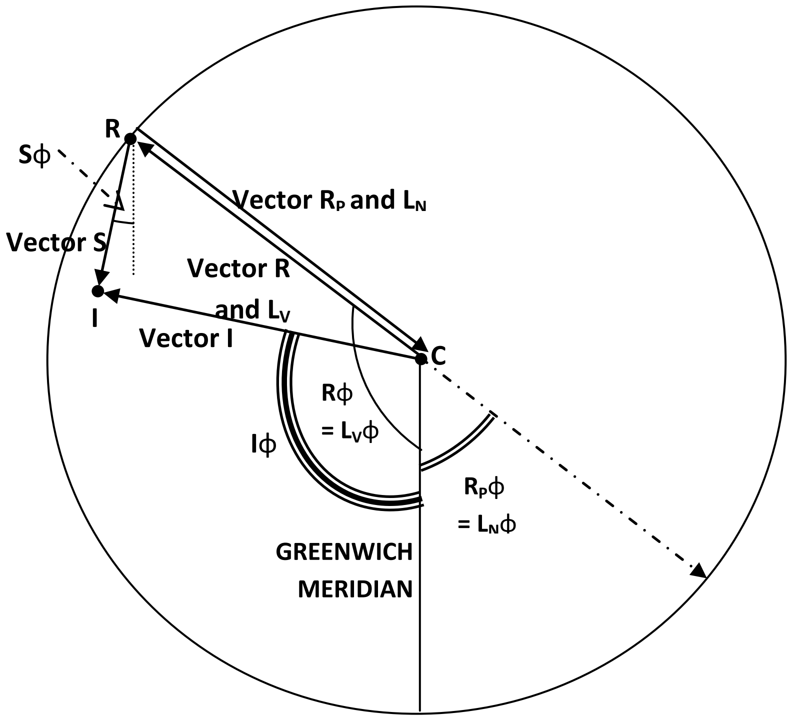


Figure 2.6.1 The longitudes of the different vectors used

In figure above,

C = Center of the Earth

R = Location of the Radar

I = Location of the irregularity in ionosphere

Vector R = Position vector of the radar

Vector R_P = Vector with longitude opposite to Vector R and is perpendicular to Vector R

Vector L_N = Vector with longitude opposite to Vector R and is parallel to the Earth's surface

Vector L_V = Vector with longitude along Vector R and is perpendicular to the Earth's surface

Vector S = Vector that points from the radar, R, to the ionospheric irregularity, I

Vector I = Position vector of the ionospheric irregularity, I

$R\varphi$ = Geocentric longitude of Vector R

$R_P\varphi$ = Geocentric longitude of Vector R_P

$L_N\varphi$ = Geocentric longitude of Vector L_N

$L_V\varphi$ = Geocentric longitude of Vector L_V

$S\varphi$ = Geocentric longitude of Vector S

$I\varphi$ = Geocentric longitude of Vector I

$R\varphi$ = $L_V\varphi$ = -147.47° (Geocentric)

$R_P\varphi$ = $L_N\varphi$ = $R\varphi + 180°$ = 32.53° (Geocentric)

Figure 2.6.2 below shows the vectors in a different angle to highlight their geocentric latitudes used for the analysis of the location of the irregularities.

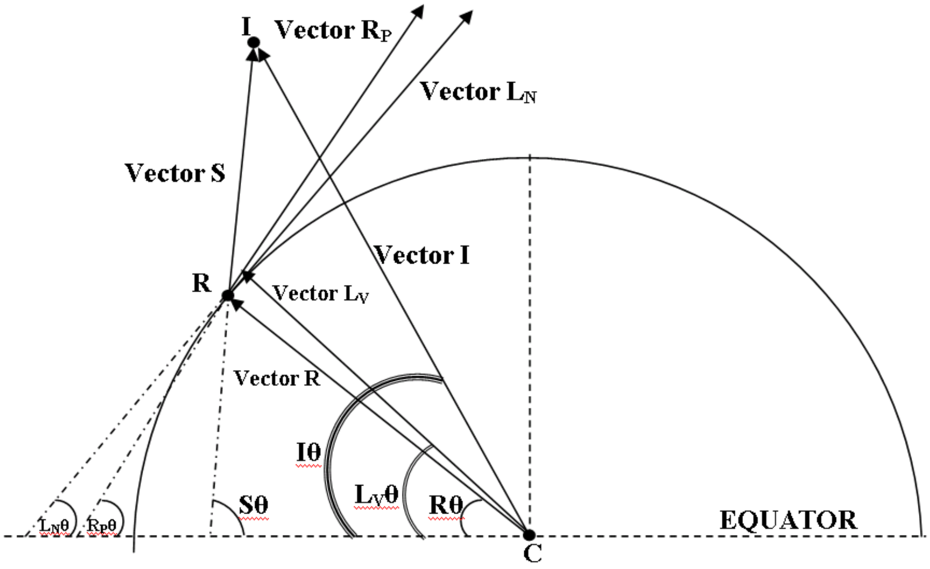


Figure 2.6.2 The latitudes of the different vectors used

In figure above,

$R\theta$ = Geocentric latitude of Vector R

$R_P\theta$ = Geocentric latitude of Vector R_P

$L_N\theta$ = Geocentric latitude of Vector L_N

$L_V\theta$ = Geocentric latitude of Vector L_V

$S\theta$ = Geocentric latitude of Vector S

$I\theta$ = Geocentric latitude of Vector I

Let Δ be the difference in geodetic and geocentric latitude at R,

Δ = 65.15° - 64.9828° = 0.1672°

$R\theta$ = 64.9828° (Geocentric)

$R_P\theta$ = $90° - R\theta$ = 25.0172° (Geocentric)

$L_V\theta$ = $R\theta + \Delta$ = 65.15° (Geocentric)

$L_N\theta$ = $90° - L_V\theta = 90° - 65.15° = 24.85°$ (Geocentric)

Among the vectors, L_V, can be considered the local vertical, L_N, the local North. Let us convert L_V and L_N from Spherical to Cartesian coordinates.

L_V = (L_VX, L_VY, L_VZ)

= $([\text{Cos}(L_V\theta) * \text{Cos}(L_V\varphi)], [\text{Cos}(L_V\theta) * \text{Sin}(L_V\varphi)], [\text{Sin}(L_V\theta)])$

L_N = (L_NX, L_NY, L_NZ)

= $([\text{Cos}(L_N\theta) * \text{Cos}(L_N\varphi)], [\text{Cos}(L_N\theta) * \text{Sin}(L_N\varphi)], [\text{Sin}(L_N\theta)])$

Let the local East be represented by Vector L_E. Now,

Vector L_E = Vector L_V X (–Vector L_N)

Performing the cross product,

L_E = $([L_VY * (-L_NZ) – L_VZ * (-L_NY)], [-(L_VX * (-L_NZ) – L_VZ * (-L_NX))],$
$[L_VX * (-L_NY) – L_VY * (-L_NX)])$

The vectors L_V, L_N and L_E are mutually perpendicular to each other represented by figure 2.6.3 below,

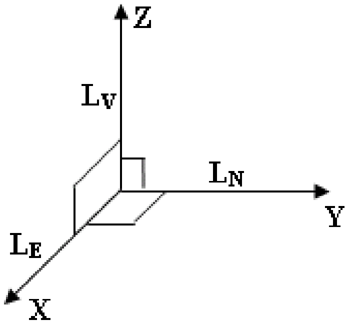


Figure 2.6.3 Local East, North and Vertical at radar position are mutually perpendicular

Unit vectors of L_V, L_N and L_E are thus,

UL_V = $L_V / |L_V|$

UL_N = $L_N / |L_N|$

UL_E = $L_E / |L_E|$

Thus we have developed a coordinate system centered at the position of the radar with respect to the geocentric system. We can manipulate any vector in this system and relate it to the geocentric system by manipulating the unit vectors UL_V, UL_N and UL_E.

Vector S is essential to compute vector I, thus let us go ahead and find vector S. Vector S can be deduced by using the unit vectors UL_V, UL_N and UL_E. If the radar is pointed at certain elevation (ε) and rotate about certain azimuth (α) and the distance between the radar and irregularity is 'd' then,

$SLN \;=\; d * \mathrm{Cos}\,(\varepsilon) * \mathrm{Cos}\,(\alpha)$

$SLE \;=\; d * \mathrm{Cos}\,(\varepsilon) * \mathrm{Sin}\,(\alpha)$

$SLV \;=\; d * \mathrm{Sin}\,(\varepsilon)$

Then, Vector S in geocentric system is given by,

$S_X \;=\; SL_N * UL_NX + SL_V * UL_VX + SL_E * UL_EX$

$S_Y \;=\; SL_N * UL_NY + SL_V * UL_VY + SL_E * UL_EY$

$S_Z \;=\; SL_N * UL_NZ + SL_V * UL_VZ + SL_E * UL_EZ$

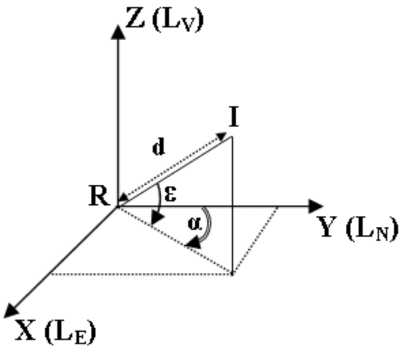


Figure 2.6.4 Radar elevation, azimuth and distance of irregularity location

Finally we can focus on finding the components of the position vector of ionospheric irregularities, vector I.

To find the components of Vector I, we convert Vector S and Vector R to their Cartesian form and then add them up.

From our previous assumption, 'd' is the distance between the radar and irregularity. Let 'r' be the distance between center of Earth and radar then,

$Rx \;=\; r * \mathrm{Cos}\,(R\theta) * \mathrm{Cos}\,(R\varphi)$

$Ry \;=\; r * \mathrm{Cos}\,(R\theta) * \mathrm{Sin}\,(R\varphi)$

$Rz \;=\; r * \mathrm{Sin}\,(R\theta)$

Finally,

$Ix \;=\; Sx + Rx$

$Iy \;=\; Sy + Ry$

$Iz \;=\; Sz + Rz$

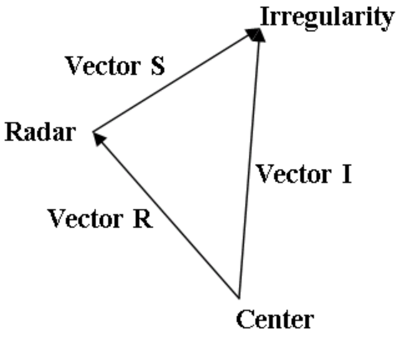


Figure 2.6.5 Components of vector I

To convert the Cartesian coordinate back to Geocentric coordinate,

$Ir \;=\; \mathrm{sqrt}\,(Ix\hat{}2 + Iy\hat{}2 + Iz\hat{}2)$

$I\theta \;=\; \mathrm{arcTan}\,(Iz / (\mathrm{sqrt}\,(Ix\hat{}2 + Iy\hat{}2)))$

$I\varphi \;=\; \mathrm{arcTan}\,(Iy / Ix)$

Thus we have the geometry to calculate geocentric coordinate of the ionospheric irregularity pointed by the center of the beam.

2.6.2 Computing aspect angle at the geocentric location of irregularity

We would like to compute the aspect angle of the ionospheric irregularities in order to observe their alignment with Earth's magnetic lines of forces. The following section demonstrates how we can find the aspect angle of the ionospheric irregularities.

Figure 2.6.6 below shows different vectors that are used for the analysis of the aspect angle of the ionospheric irregularities.

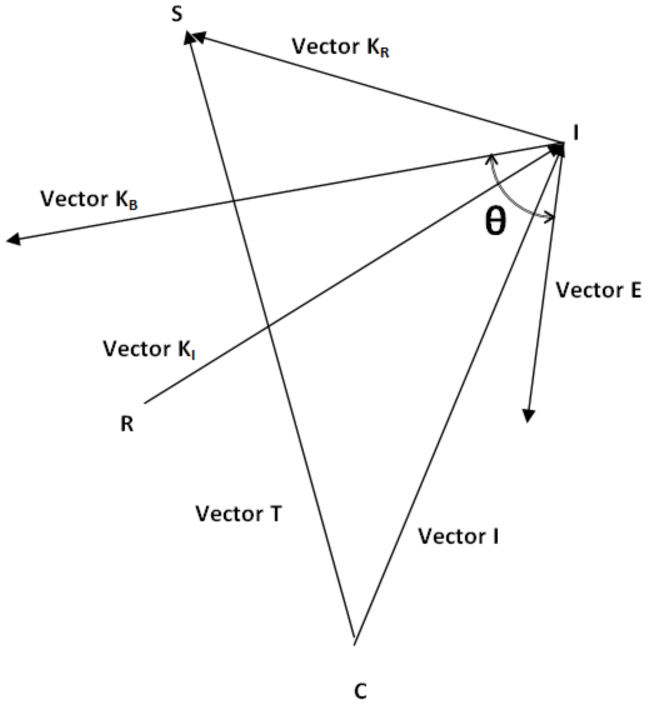


Figure 2.6.6 Aspect angle at the geocentric location of irregularity

In figure above,

C = Center of the Earth

R = Location of the Radar

I = Location of the irregularity in ionosphere

S = Location of the Satellite

Vector T = Position vector of the satellite

Vector I = Position vector of the ionospheric irregularity

Vector E = Vector of the magnetic field at location I

Vector K_I = Vector that points from the radar, R, to the ionospheric irregularity, I

Vector K_R = Vector that points from the ionospheric irregularity, I, to the satellite, S

Vector K_R = Vector that bisects Vector K_R and Vector K_I

θ = Aspect Angle + 90°

In figure 2.6.6, the vectors that need to be determined are vector K_R and K_B. The rest of the vectors are known from the previous section for computing the irregularity's geocentric coordinate.

First we compute Vector K_R

Vector K_R is given by the following relation,

Vector K_R = Vector T – Vector I

Converting Vector T and Vector I to Cartesian,

T_X = Tr * Cos(Tθ) * Cos(Tφ)

T_Y = Tr * Cos(Tθ) * Sin(Tφ)

T_Z = Tr * Sin(Tθ)

I_X = Ir * Cos(Iθ) * Cos(Iφ)

I_Y = Ir * Cos(Iθ) * Sin(Iφ)

I_Z = Ir * Sin(Iθ)

Now,

Vector K_R = $(T_X - I_X, T_Y - I_Y, T_Z - I_Z)$

Next, Let us compute Vector K_B

Vector K_B is the bisector of vector K_I and Vector K_R. Vector K_I points from location R to location I. Let us consider a new vector V_O whose direction is opposite to that of vector K_I thus we will have vectors as shown in figure 2.6.7. Unit Vector of K_B will be the resultant of adding unit vector of K_R and unit vector of K_O.

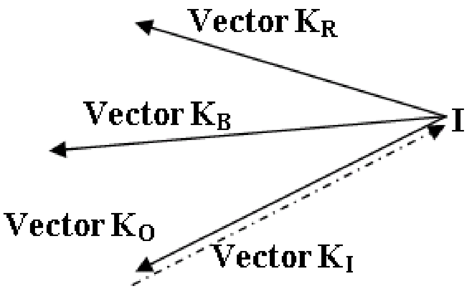


Figure 2.6.7 Computation of vector KB

As vector K_O is the opposite of vector K_I we have,

K_OX = – K_IX

K_OY = – K_IY

K_OZ = – K_IZ

And converting vector K_R to Cartesian,

$K_{RX} \quad = \quad K_Rr * Cos(K_R\theta) * Cos(K_R\varphi)$

$K_{RY} \quad = \quad K_Rr * Cos(K_R\theta) * Sin(K_R\varphi)$

$K_{RZ} \quad = \quad K_Rr * Sin(K_R\theta)$

Now let us find the unit vectors of vector K_R and vector $K_{O,}$

$U_{KR} \quad = \quad K_R / |K_R|$

$U_{KO} \quad = \quad O_O / |K_O|$

Thus,

$U_{KB} \quad = \quad U_{KR} + U_{KO}$

Now to compute the θ,

The θ is the angle between vector E and vector K_B. This angle is the same as the angle between the unit vector of K_B and the unit vector of E.

Unit vector of vector E is given by,

$U_E \quad = \quad E / |E|$

To compute the angle between U_{KB} and U_E we use the concept of dot products. The dot product of vector U_{KB} and U_E is given by,

$U_{KB} . U_E \quad = \quad |U_{KB}| * |U_E| * Cos(\theta)$

Thus,

$\theta \quad = \quad arcCos(U_{KB} . U_E / (|U_{KB}| * |U_E|))$

Where,

$U_{KB} . U_E \quad = \quad U_{KB}{}^T * U_E$

Finally,

Aspect angle = θ – 90

2.6.3 Dividing the volume of incident beam into pieces which are uniform in spherical domain

As mentioned in the beginning, the incident beam is narrow and has a spherical wave front. Also the beam is pointed at certain elevation (ε) and rotates about certain azimuth (α). At certain altitude (h) we can take the wave front of the beam and divide its volume into pieces uniform in the spherical domain. These volumes when multiplied by the power factor of the

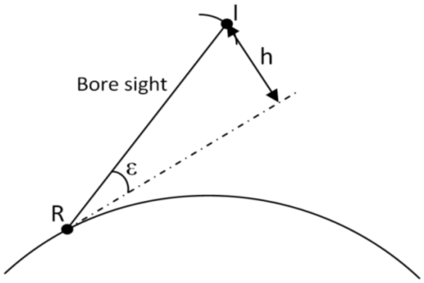


Figure 2.6.8 Radar beam

radar depict how much of the electromagnetic energy propagates from the radar to the irregularity.

Figure 2.6.9 below shows how the total volume is being divided. The total volume, which is within 3° of elevation about the bore sight and 360° of azimuth around the bore sight, is divided by breaking the elevation in steps of dθ which is 0.1°, breaking the azimuth in steps of dφ which is 10° and taking a depth of dr which is 0.1 km.

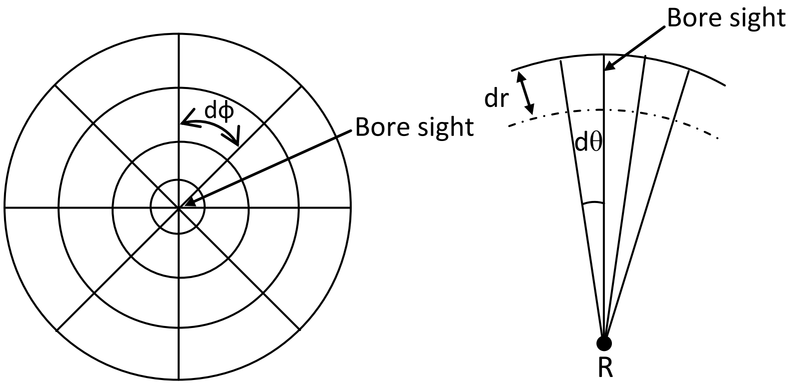


Figure 2.6.9 Division of beam volume

The volume of each piece that is given by the following expression:

$$dv = r^2 * \sin(\theta) * \ dr * d\theta * \ d\varphi$$

Where r is the distance from the radar to the volume piece and θ is the elevation angle from the bore sight to the volume piece. Thus in this way the incident beam is divided into volumes of pieces which is uniform in the spherical domain.

2.6.4 Finding location, altitude, aspect angle and j index of each volume piece

Each volume piece is characterized by a vector pointing from the radar to the piece. This vector can be deduced by rotating the bore sight about certain elevation (θ) and azimuth (φ). θ and φ are characteristic of each piece and is determined while dividing the incident beam into pieces. After we find the vector, we can determine the new location, altitude, aspect angle and j index of each volume piece.

To find the vector pointing from the radar to the volume piece, we start by adding θ to the elevation of the bore sight. In figure 2.6.10, vector S' represents the result of adding the elevation.

Thus,

$S'\theta = S\theta + \theta$

$S'\varphi = \varphi$

$S'r = Sr$

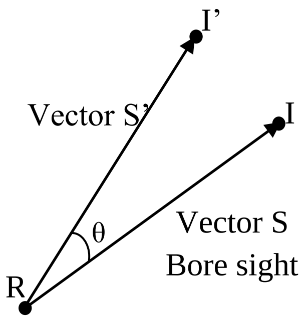


Figure 2.6.10 Addition of elevation to bore sight

Next we setup the local Cartesian coordinate system. Figure 2.6.11 shows the different vectors used to setup the local Cartesian coordinate system.

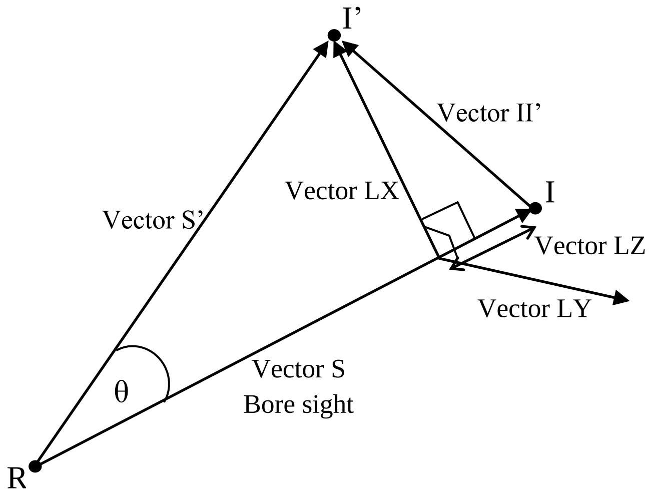


Figure 2.6.11 Local Cartesian coordinate system setup for each beam volume division

In the figure 2.17 above,

R = Location of the Radar

I = Irregularity pointed by bore sight

I' = Irregularity pointed by adding elevation, θ, to the bore sight

Vector S = Bore sight

Vector S’ = Vector formed by adding elevation, θ, to the bore sight

Vector II’ = Vector pointing from I to I’

Vector LX = Local X axis

Vector LX = Local Z axis

Vector LY = Local Y axis

As we can see in figure 2.6.11, vector II’ is given by following relation,

Vector II’ = Vector S’ – Vector S

The local Z axis is along vector S so they have the same azimuth and elevation. Only the radial distance, ‘r’, is different. Vector LZ is given by,

$LZ\theta = S\theta$

$LZ\varphi = S\varphi$

$LZr = Sr - Sr * \cos(\theta)$

Vector LX is the resultant of vector II’ and vector LZ. Thus,

Vector LX = Vector LZ + Vector II’

Vector LY is mutually perpendicular to vector LX and vector LZ. Thus vector LY is the result of the cross product between vector LX and vector LZ.

Vector LY = Vector LZ X Vector LX

Thus we have successfully determined the local X, local Y, and local Z. Next we have to rotate the local X, vector LX, φ degrees about the local Z, vector LZ, as shown in figure 2.6.12. Let us name the new resulting vector, vector RT.

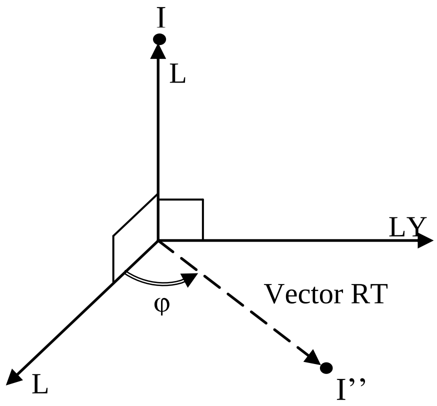


Figure 2.6.12 Rotation of local X about the local Z

The local x, y and z coordinates of RT is given by,

$RT_{LX} = |\text{Vector LX}| * \text{Cos}(\varphi)$

$RT_{LY} = |\text{Vector LX}| * \text{Sin}(\varphi)$

$RT_{LZ} = 0$

Let ULX, ULY and ULZ be the unit vectors of the local X, Y and Z axis.

$ULX = LX / |LX|$

$ULY = LY / |LY|$

$ULZ = LZ / |LZ|$

Now, the geocentric Cartesian coordinates of RT is given by,

$RT_{X} = RT_{LX} * ULX_{X} + RT_{LY} * ULY_{X} + RT_{LZ} * ULZ_{X}$

$RT_{Y} = RT_{LX} * ULX_{Y} + RT_{LY} * ULY_{Y} + RT_{LZ} * ULZ_{Y}$

$RT_{Z} = RT_{LX} * ULX_{Z} + RT_{LY} * ULY_{Z} + RT_{LZ} * ULZ_{Z}$

Next we find the vector II'', which is vector pointing from the end of the bore sight to the end of vector RT, and is given by,

Vector II' = Vector RT – Vector LZ

Now the vector pointing from the radar to the volume piece is given by,

Vector newS = Vector S + Vector II'

The position vector of the new Irregularity location is given by,

Vector newI = Vector R + vector news

Thus, we have the position vector of the new volume piece.

The vector pointing from the irregularity to the satellite is given by,

Vector I''T = Vector T – Vector newI

Hence we have successfully determined the location of the volume piece. As we know the position vector of volume piece, the vector pointing from radar to the volume piece and the vector pointing from the volume piece to the satellite, we can determine the aspect angle as discussed in the previous section.

Solving for the altitude of the volume piece requires some mathematical processing. We know the vector pointing from the radar to the irregularity. The Cartesian coordinates of the vector can be represented by the following equation:

$\mathrm{newS}_X = \mathrm{SL}_N * \mathrm{UL}_N X + \mathrm{SL}_V * \mathrm{UL}_V X + \mathrm{SL}_E * \mathrm{UL}_E X$

$\mathrm{newS}_Y = \mathrm{SL}_N * \mathrm{UL}_N Y + \mathrm{SL}_V * \mathrm{UL}_V Y + \mathrm{SL}_E * \mathrm{UL}_E Y$

$\mathrm{newS}_Z = \mathrm{SL}_N * \mathrm{UL}_N Z + \mathrm{SL}_V * \mathrm{UL}_V Z + \mathrm{SL}_E * \mathrm{UL}_E Z$

We have three equations above and three unknowns SLN, SLV and SLE. The local vertical SLV gives us the altitude of the irregularity.

Solving for the altitude, we get the following equation:

$$Altitude = \frac{(ULEX * ULNX - ULEX * ULNZ)(ULEY * SX - ULEX * SY) - (ULEY * ULNX - ULEX * ULNY)(ULEZ * SX - ULEX * SZ)}{(ULEY * ULVX - ULEX * ULVY) * (ULEZ * ULNX - ULEX * ULNZ) - (ULEY * ULNX - ULEX * ULNY)(ULEZ * ULVX - ULEX * ULVZ)}$$

The j index is the number formed by rounding off the time taken for wave to propagate from the radar to the satellite after getting scattered. As we know the vector going from radar to volume piece and volume piece to satellite, we can find the total distance travelled. This distance when divided by the speed of light gives us the total time of travel. And by rounding off the time taken we get the j index.

3 Methodology

3.1 Physics Assumptions

The irregularity intensities are assumed to be uniform in horizontal direction. Hence, intensity depends only on altitude and aspect angle. The aspect angle dependence is assumed to be Gaussian in the intensity function I(h,θ).

3.2 Forward Model

The forward model is formulated as

$$S_{i,j} = \sum_{h,\theta} \omega_{i,j}\,(h,\theta).\,I(h,\theta) \tag{3.1}$$

where I(h,θ) is the irregularity intensity function as a function of irregularity altitude, h, and aspect angle, θ. $\omega_{i,j}$ is the point spread function which blurs the intensity function and $S_{i,j}$ is the data that is received by the satellite. **i** is the index of the pulse sent from the PFISR and j is the time delay (index) between the transmission and reception of the beam.

3.2.1 Irregularity intensity function, I(h,θ)

I(h,θ) is the irregulari

ty intensity at different altitudes and aspect angles. The altitude is the vertical distance between the irregularity location and the transmitting radar plane and the aspect angle is the angle between the earth's magnetic field line and bisector of radar incident and reflected beam.

In a simpler model it is modeled as a simple Gaussian function given by the equation 3.2.

$$I(h,\theta) = I_0 e^{-\left(\frac{h-h_0}{\Delta h_I}\right)^2} e^{-\left(\frac{\theta}{\Delta\theta_I}\right)^2} \tag{3.2}$$

where h is the altitude, θ is the aspect angle, h_0 is the center of Gaussian height, and Δh and Δθ are altitude and aspect angle widths.

The equation is later modified as a sum of seven separate simple Gaussian functions as shown in equation 3.3 to obtain higher altitude resolution. The height range of the ionosphere that we are more concerned about is from 90km to 120 km. Hence each of these individual Gaussian functions would be centered at separation of 5km from 90km to 120km which is denoted by the value of h_0 in the equation.

$$I(h,\theta) = \sum_{k=1}^{7} I_k e^{-\left(\frac{h-h_{0k}}{\Delta h_k}\right)^2} e^{-\left(\frac{\theta}{\Delta\theta_k}\right)^2} \tag{3.3}$$

Since all functions have height width of 5km, Δh_k equals 5. The expanded version of the equation would be:

$$I(h,\theta) = I_1 e^{-\left(\frac{h-90}{5}\right)^2} e^{-\left(\frac{\theta}{\Delta\theta_1}\right)^2} + I_2 e^{-\left(\frac{h-95}{5}\right)^2} e^{-\left(\frac{\theta}{\Delta\theta_2}\right)^2} + I_3 e^{-\left(\frac{h-100}{5}\right)^2} e^{-\left(\frac{\theta}{\Delta\theta_3}\right)^2} + I_4 e^{-\left(\frac{h-105}{5}\right)^2} e^{-\left(\frac{\theta}{\Delta\theta_4}\right)^2} + I_5 e^{-\left(\frac{h-110}{5}\right)^2} e^{-\left(\frac{\theta}{\Delta\theta_5}\right)^2} + I_6 e^{-\left(\frac{h-115}{5}\right)^2} e^{-\left(\frac{\theta}{\Delta\theta_6}\right)^2} + I_7 e^{-\left(\frac{h-120}{5}\right)^2} e^{-\left(\frac{\theta}{\Delta\theta_7}\right)^2} \quad (3.4)$$

3.2.2 Point spread function, $\omega_{i,j}$

$\omega_{i,j}$ is the point spread function in the forward model of the experiment which blurs the intensity function. It quantizes the electromagnetic wave energy being transmitted from the PFISR and is constructed for each satellite position, PFISR pointing angle, the magnetic field geometry, PFISR transmitter antenna pattern, and RAX receiving antenna pattern.

In the initial phase of the project it was modeled as a simple Gaussian, equation 3.5, and sinc function, equation 3.6. ω_0 is a scalar constant.

$$\omega(h,\theta) = \omega_0 e^{-\left(\frac{h}{\Delta h_\omega}\right)^2} e^{-\left(\frac{\theta}{\Delta\theta_\omega}\right)^2} \quad (3.5)$$

$$\omega(h,\theta) = \left(\frac{\sin\left(2\pi * \frac{h}{\Delta h_\omega}\right)}{2\pi * \frac{h}{\Delta h_\omega}}\right)^2 \cdot \left(\frac{\sin\left(2\pi * \frac{\theta}{\Delta\theta_\omega}\right)}{2\pi * \frac{\theta}{\Delta\theta_\omega}}\right)^2 \quad (3.6)$$

The equation was later modified to accommodate the volume of the radar beam being transmitted to the ionosphere. A very narrow radar beam with a beam angle width of 3 degrees is projected to the ionosphere that forms a conic shape. The volume of the beam is divided into smaller quantities of δv as a function of change in elevation angle (θ), radar azimuth angle (φ) and distance (r) from the central bore sight as discussed in section 2.7.3, where

$$\delta v = (r^2 \sin\theta)\, \delta r\, \delta\theta\, \delta\phi \quad (3.7)$$

The purpose of dividing radar beam into smaller quantities of δv is to relate the power intensity of each of those δv with the data received by the satellite. For any given satellite position represented in the forward model as **i**, $\omega_{i,j}$ refers to the sum of all δv from which the signals reach satellite at the same time.

Different satellite positions are chosen such that their **i** values can be correlated with the radar pulse number transmitted from the PFISR. The pulses from radar are transferred at every 10ms. Hence the distance between two consecutive satellite positions would be the distance travelled by the satellite in 10ms.

For any given satellite position, the satellite samples at the frequency of 1MHz thus forming sample window of 1μs, which is referred as **j** in the forward model. As said before, each ω_j is the summation of energies reflected from all δv that falls into that particular sample window.

The total number of j depends on the time delay between transmitted and received signal. Time delay is given by the following equation.

$$\text{Time Delay} = \frac{\text{distance from radar to irregularity} + \text{distance from irregularity to satellite}}{\text{speed of light}} \tag{3.8}$$

Then **j** index is given by the floor value of the following equation.

$$\mathbf{j} = \left(\frac{\text{Time Delay}}{\text{RAX sampling time period}}\right)_{\text{floor value}} \tag{3.9}$$

Now, ω function can be written as:

$$\omega_{i,j}(h, \theta) = \sum_{k=1}^{N} (r_k{}^2 \sin(\theta_k)\ \delta r\ \delta\theta\ \delta\phi \tag{3.10}$$

$$\omega_{i,j}(h, \theta) = \sum_{k=1}^{N} \delta v_k \quad \delta r\ \delta\theta\ \delta\phi \tag{3.11}$$

where k refers to total number of δv for each **i** and **j** pair.

$\omega_{i,j}$ is further modified to take into account varying power of beam at different δv.

$$\omega_{i,j}(h, \theta) = \sum_{k=1}^{N} \delta v_k * (P_{in})_k\ \ \delta r\ \delta\theta\ \delta\phi \tag{3.12}$$

P_{in} was initially modeled as a Gaussian function $e^{-\left(\frac{\theta_k}{\Delta\theta}\right)^2}$ and later replaced by the actual PFISR power gain pattern in section 4.1.

3.2.3 $S_{i,j}$

S(h,θ) is the data measured by the satellite which is basically measurement of energy being reflected from particular set of δv.

In a simpler model, S(h,θ) was simply computed as a convolution result of I(h,θ) and Gaussian and sinc ω(h,θ).

$$S(h,\theta) = \sum_{h,\theta} \omega_0 e^{-\left(\frac{h}{\Delta h_\omega}\right)^2} e^{-\left(\frac{\theta}{\Delta\theta_\omega}\right)^2} * I_0 e^{-\left(\frac{h-h_0}{\Delta h_I}\right)^2} e^{-\left(\frac{\theta}{\Delta\theta_I}\right)^2} \tag{3.13}$$

$$S(h,\theta) = \sum_{h,\theta} \left(\frac{\sin\left(2\pi * \frac{h}{\Delta h_\omega}\right)}{2\pi * \frac{h}{\Delta h_\omega}} \right)^2 . \left(\frac{\sin\left(2\pi * \frac{\theta}{\Delta\theta_\omega}\right)}{2\pi * \frac{\theta}{\Delta\theta_\omega}} \right)^2 * I_0 e^{-\left(\frac{h-h_0}{\Delta h_I}\right)^2} e^{-\left(\frac{\theta}{\Delta\theta_I}\right)^2} \tag{3.14}$$

For each changes in ω, S function was modified as well. When volume of radar beam was considered, S function was modified into:

$$S_{i,j}(h,\theta) = \sum_{h,\theta} \left(\sum_{k=1}^{N} \delta v_k \quad \delta r\, \delta\theta\, \delta\phi \right) . I_0 e^{-\left(\frac{h-h_0}{\Delta h_I}\right)^2} e^{-\left(\frac{\theta}{\Delta\theta_I}\right)^2} \tag{3.15}$$

With the accommodation of power gain factor, the $S_{i,j}$ equation was modified into:

$$S_{i,j}(h,\theta) = \sum_{h,\theta} \left(\sum_{k=1}^{N} \delta v_k\, (P_{in})_k \quad \delta r\, \delta\theta\, \delta\phi \right) . I_0 e^{-\left(\frac{h-h_0}{\Delta h_I}\right)^2} e^{-\left(\frac{\theta}{\Delta\theta_I}\right)^2} \tag{3.16}$$

$S_{i,j}$ was modified for the last time to consider the I(h,θ) which divided ionospheric intensity model into seven divisions based on the altitude.

$$S_{i,j}(h,\theta) = \sum_{h,\theta} \left(\sum_{k=1}^{N} \delta v_k\, (P_{in})_k \quad \delta r\, \delta\theta\, \delta\phi \right) . \left(\sum_{k=1}^{7} I_k e^{-\left(\frac{h-h_{0k}}{\Delta h_k}\right)^2} e^{-\left(\frac{\theta}{\Delta\theta_k}\right)^2} \right) \tag{3.17}$$

3.2.4 Uncertainty and noise

The uncertainty is due to the fact that the scattering from ionospheric irregularities is a random process. The measured $S_{i,j}$ is not exactly a function of a particular I model but contains statistical uncertainty. To accommodate such situation uncertainty factor was computed using the equation 3.18.

$$\text{Uncertainty factor} = 1 + \frac{\text{randn}}{\sqrt{\text{N}}} \tag{3.18}$$

where N is the number of radar pulses involved in the measurements of $S_{i,j}$. The higher the number of pulses involved in the $S_{i,j}$ computation, the lower is the uncertainty. $S_{i,j}$ with uncertainty in consideration is then given by multiplying the actual $S_{i,j}$ with the uncertainty factor.

$$\left(S_{i,j}\right)_{\text{uncertainty}} = \left(S_{i,j}\right)_{\text{actual}} * \text{ Uncertainty factor} \tag{3.19}$$

For this experiment, the noise in the measured ${S_{i,j}}_{\text{actual}}$ was modeled by equation 3.20 and added to the actual $S_{i,j}$ modeled by equation 3.21.

$$\text{noise} = \left(\frac{{S_{i,j}}_{\text{max}}}{\text{M}}\right) * \text{randn} \tag{3.20}$$

$${S_{i,j}}_{\text{noise}} = {S_{i,j}}_{\text{actual}} + \text{ noise} \tag{3.21}$$

When both uncertainty and noise are simultaneously considered the $S_{i,j}$ is given by the equation 3.22.

$$S_{i,j} = \left({S_{i,j}}_{\text{actual}} + \left(\frac{{S_{i,j}}_{\text{max}}}{\text{M}}\right) * \text{randn}\right) * \left(1 + \frac{\text{randn}}{\sqrt{\text{N}}}\right) \tag{3.22}$$

4 Results

4.1 Radiation pattern simulation

The radiation pattern of the PFISR was modeled using MATLAB. Initially, the radiation pattern of a single dipole was plotted to understand basic electric fields and plotting mechanisms. Thereafter, two dipoles were placed together and the electric field pattern plotted to get the radiation pattern of a single crossed-dipole antenna. Once verified, the radiation patterns of all other 4,096 antennas were aggregated and plotted together to model the radiation pattern of the PFISR.

4.1.1 Single dipole patterns

Dipole antennas consist of a single linear metallic wire with a feed point at the center. They are commonly used in high-frequency (HF) and ultrahigh-frequency (UHF) communications, TV and FM broadcasting.

Figure 4.1.1 Single dipole

If a single dipole is aligned along the z-axis as follows,

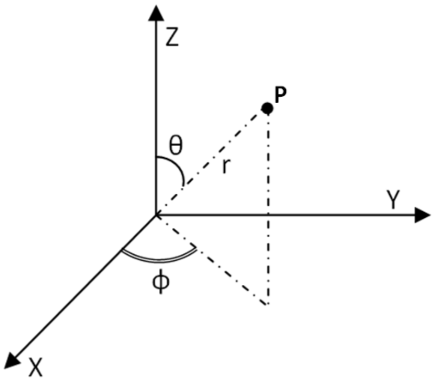


Figure 4.1.2 Single dipole aligned along z-axis

Its electric field is given by the following equation:

$$E(\theta) = I * dl * \sin(\theta) * \left(jZ \,/\, (4 * \pi * B)\right) * \left(\left(\frac{B^2}{r}\right) - \left(\frac{jB}{r^2}\right) - \left(\frac{1}{r^3}\right)\right) e^{j\,(\omega t - Br)} \qquad (4.1)$$

Where dl is the length of the dipole (infinitesimal), I is the current, r is the distance, t is the time and Z is the impedance of free space which is assumed to be 377Ω.

Since our dipoles are orientated in the x and y planes we changed this equation to meet our conditions. This was done by specifying a point P in the far field zone and taking its unit vector. The cross product of the unit vector of P and the unit vector of the axis (X or Y) would give us the angle between P and the axis, which we could substitute in (1) to calculate the electric field at P due to the dipole on the axis. Fields for many infinitesimal dipoles were calculated by varying *r* and integrated together to calculate the electric field of a dipole of length 33.41cm.

Using this procedure, the radiation pattern for the dipole parallel to the y-axis was modeled. The power of the dipole parallel to the y-axis was plotted as a function of theta (where theta = 0° is considered to be parallel to the z-axis).

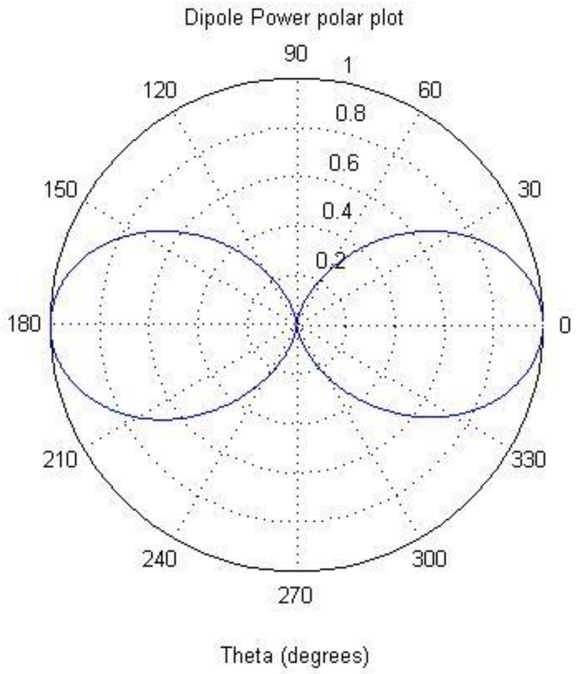


Figure 4.1.3 Radiation pattern of a dipole parallel to y-axis

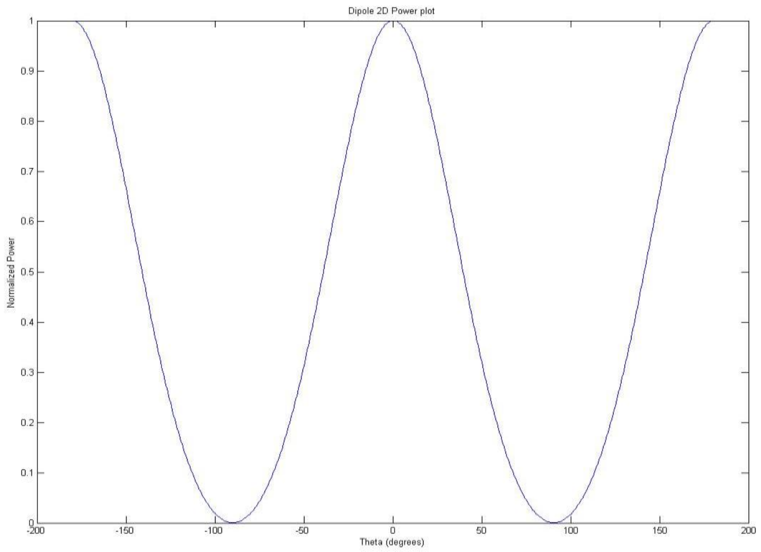


Figure 4.1.4 2D Radiation pattern of a dipole parallel to y-axis

It can be seen that when the dipole is aligned along the y-axis, it transmits minimum amount of energy when theta = 90° or theta =270° (-90°) i.e. at its ends.

The picture becomes clearer when the radiation pattern is plotted in 3D where the color represents the power intensity when the dipole is aligned in the y-axis.

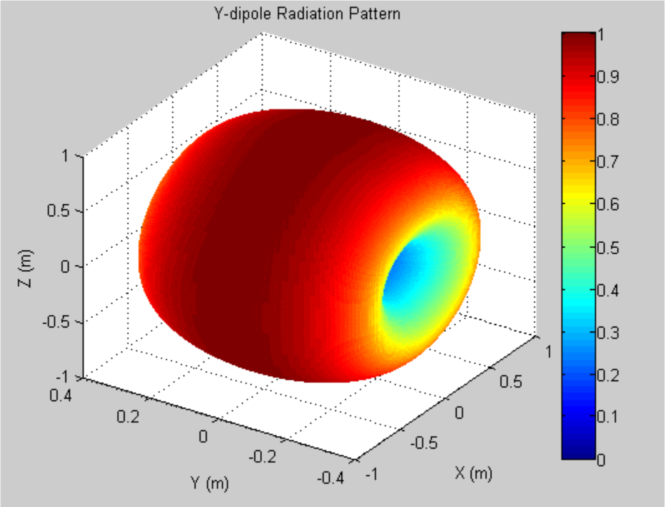


Figure 4.1.5 3D Radiation pattern of a dipole parallel to y-axis

4.1.2 Single crossed-dipole patterns

A crossed dipole antenna consists of two dipoles placed perpendicular to each other. When fed 90 degrees out of phase it results in circular polarization. This configuration was modeled by calculating the electric field of the dipole parallel to the y-axis and adding it to the electric field of the dipole parallel to the x-axis. The current in the y dipole was 90 degrees out of phase of the current fed to the x dipole. The following represents how a crossed-dipole antenna looks like.

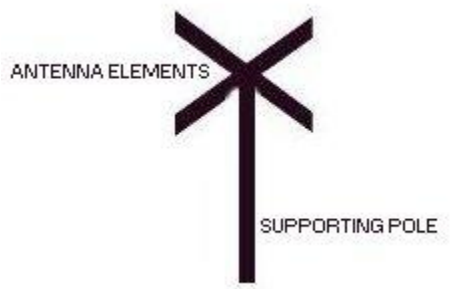


Figure 4.1.6 Crossed dipole

Adding the two electric fields and squaring them gives us the power of the transmitted field which is shown below as a function of theta (where theta= 0° is parallel to the z-axis).

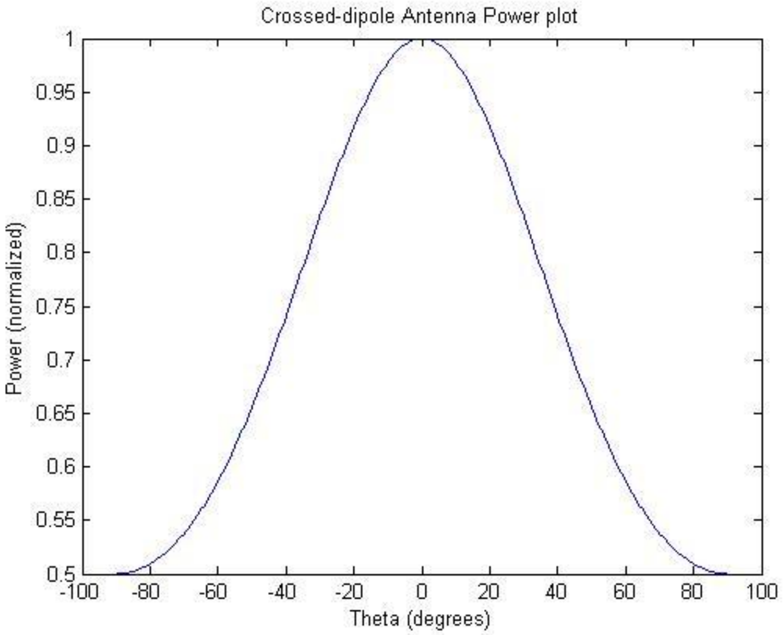


Figure 4.1.7 2D power plot of a crossed dipole

When plotted in 3D, it is observed that the radiation pattern has a spherical shape where the power intensity is highest at the peak. The colors in this figure represent the intensity of the power.

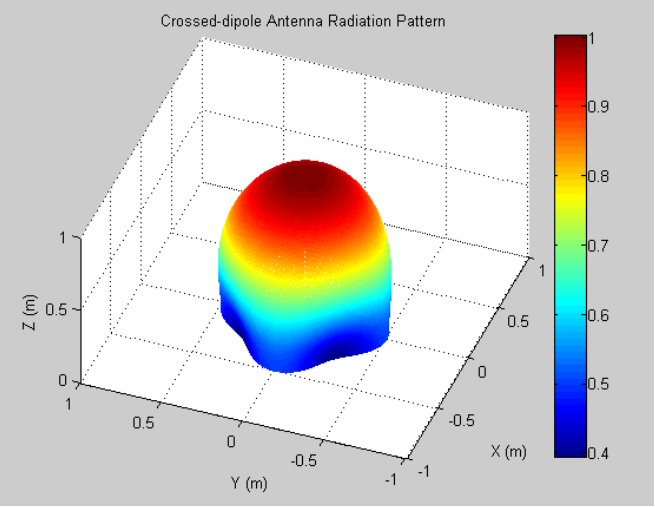


Figure 4.1.8 3D power plot of a crossed dipole

After obtaining the power pattern for one crossed dipole antenna, we proceeded to obtain the power pattern for multiple dipole antennas.

4.1.3 Panel of crossed-dipoles pattern (32 Antennas)

A panel of the PFISR consists of 32 crossed-dipole antennas arranged in an 8 x 4 rectangular plane. To determine the radiation pattern of this configuration, the electric field of each crossed dipole was calculated and all of them were summed up to find the total pattern generated. In MATLAB this was modeled by changing the r in Eq 4.1 for each dipole. By keeping the point P fixed and changing r, the electric field caused at P by dipoles placed at different r's from P can be calculated and summed together to give the total electric field at P.

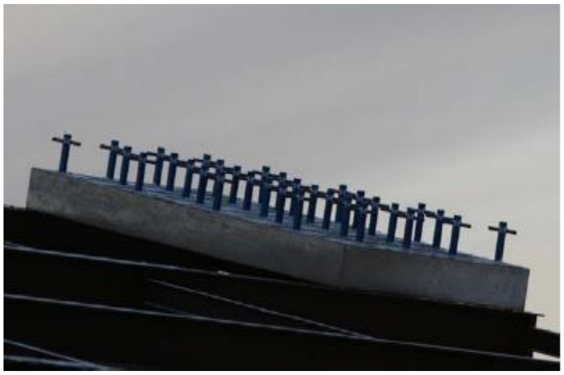


Figure 4.1.9 A panel of crossed dipoles (32 antennas)

The length of the dipoles was set to be 33.4 cm and each antenna was separated from the other antenna by a distance of 44.76 cm.

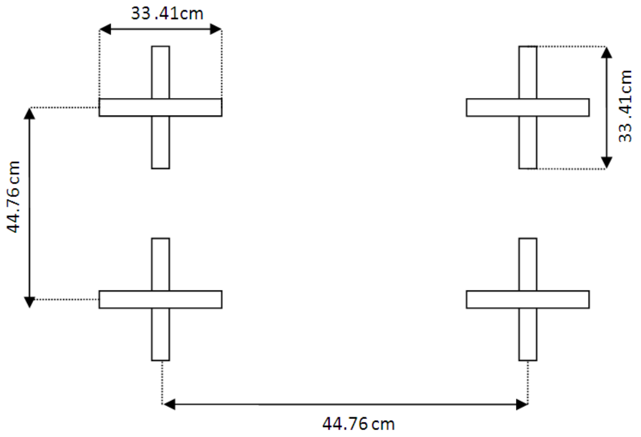


Figure 4.1.10 Antenna Geometry

In a panel there are 4 of these crossed dipoles in the across and 8 in the down. The power was calculated and plotted against θ which gave the following result.

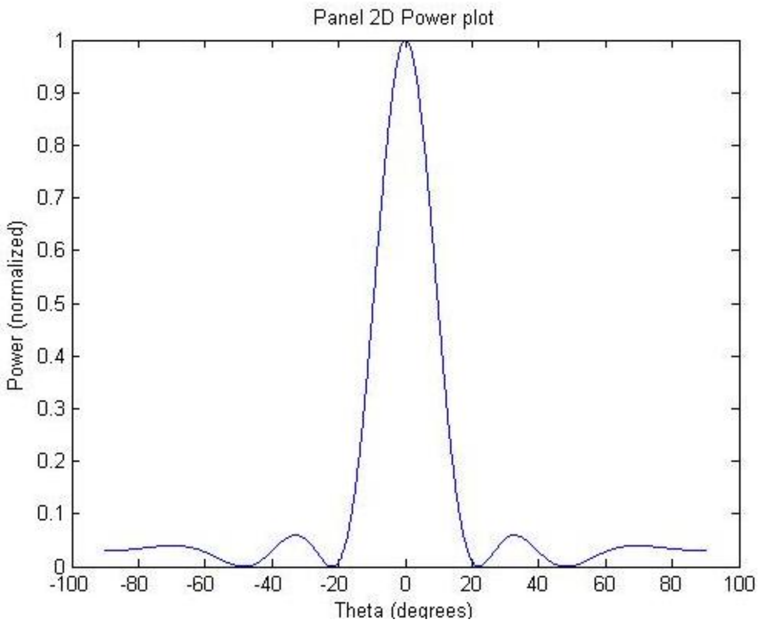


Figure 4.1.11 2D power plot of a panel (32 antennas)

It can be seen that the beam width has decreased to about 40 degrees and that side lobes are also present. The max power of the beam is when theta = 0°. In order to get a better sense of the side lobes, the power was plotted in dB scale against theta.

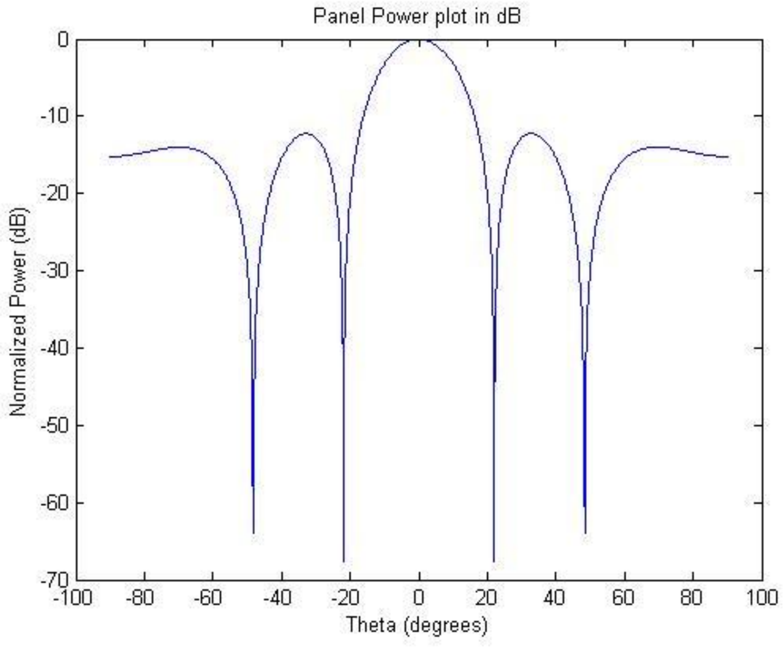


Figure 4.1.12 dB power plot of a panel

The power was also plotted as a function of theta and phi. In the plot below, the color represents the intensity of the power and the x and y axis represent theta and phi respectively.

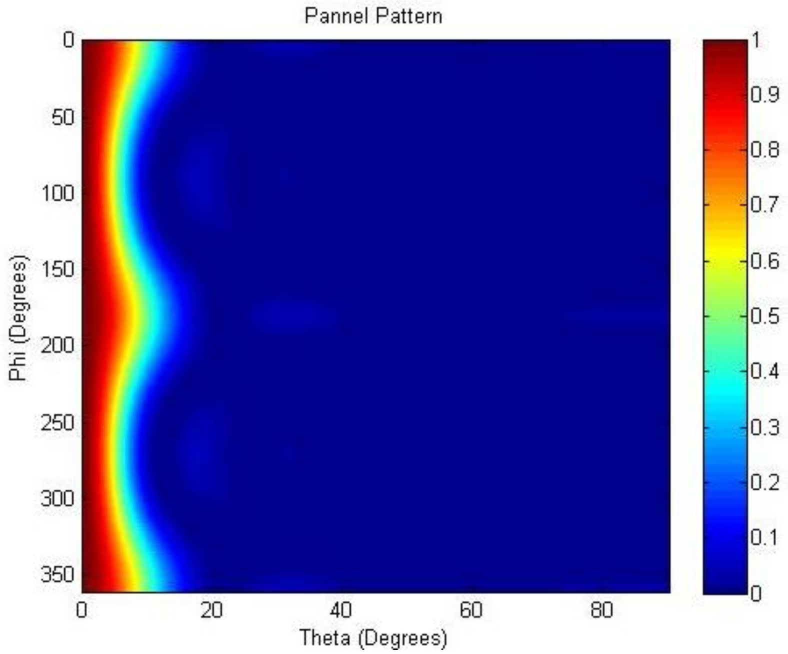


Figure 4.1.13 Power of a panel as a function of θ and ϕ

To get a better sense of the shape of the radiation pattern, it was plotted in 3D.

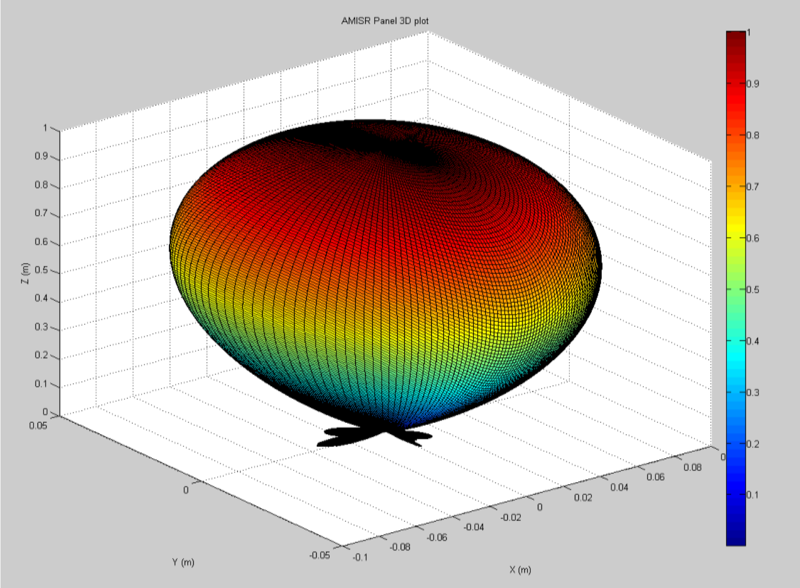


Figure 4.1.14 3D power plot of a panel

It can be observed that the shape remains spherical but the intensity at the top of the sphere becomes narrower. This trend continues when we plot the patterns for the whole PFISR.

4.1.4 Whole PFISR antenna pattern

The whole PFISR consists of 128 panels. Since each panel consists of 32 antennas, the whole PFISR consists of 4,096 antennas placed in a 64 x 64 rectangular configuration. The electric field for the whole PFISR was calculated in the same manner at it was calculated for a panel, but for a theta sweep of 0-5 degrees.

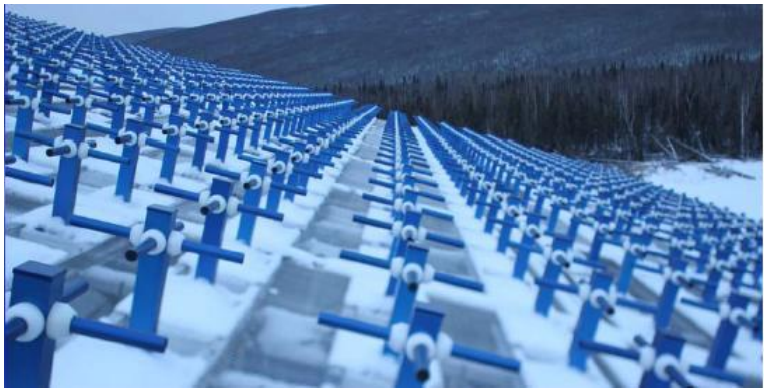


Figure 4.1.15 Whole PFISR (128 panels)

The power calculated was plotted as a function of theta.

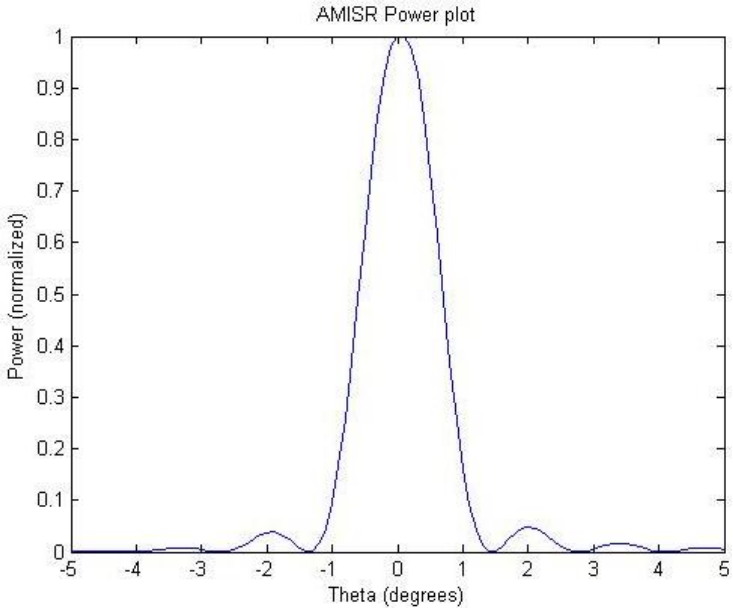


Figure 4.1.16 PFISR power plot

It can be seen that the beam width (null-to-null) decreases to about 3 degrees (1.5 degrees on each side) and the side lobes become more subdued which is more evident when the power is plotted in dB scale as follows in fig

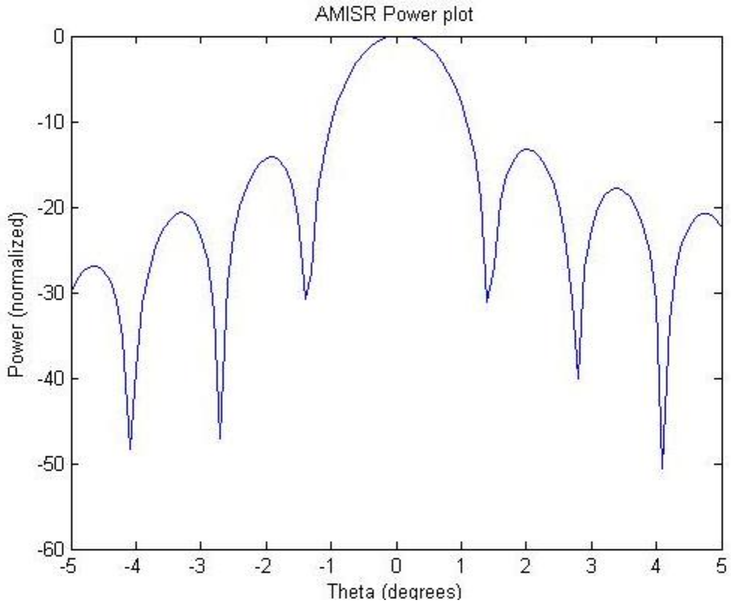


Figure 4.1.17 dB PFISR power plot

To better analyze the beam, the radar pattern is plotted as function of theta and phi as follows where the colors represent the intensity of the power.

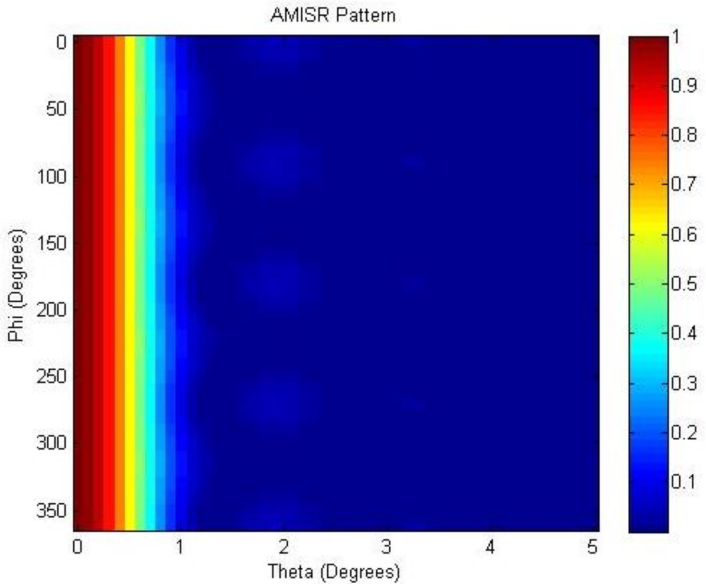


Figure 4.1.18 PFISR power plot as a function of θ and ϕ

4.1.5 Observations and conclusions

It can be observed that as the number of antennas increase the beam width becomes smaller and smaller. This enables us to achieve a higher resolution in our measurements of the I function. It can also be seen that with increasing antennas the side lobes also become less intense. A narrow beam width with small side lobes, are important characteristics of the PFISR, which makes it an integral part of this project. The power calculated for the whole PFISR was used to model the w function so that the intensity of the power falling at each point can be known to make further calculations.

4.2 Satellite antenna pattern

Once the radar beam is reflected from the irregularities in the ionosphere towards the RAX, antennas strategically placed on the RAX pick up the received signals. In order to determine the behavior and gain of the antennas, we simulated their gain pattern. Knowing the precise gain of the antennas would allow us to calculate our results with greater accuracy.

4.2.1 The satellite

The RAX is shaped like rectangular box with cross's along its sides. It measures 10cm x 10cm x 30cm and has three levels. On its four sides, it has crosses.

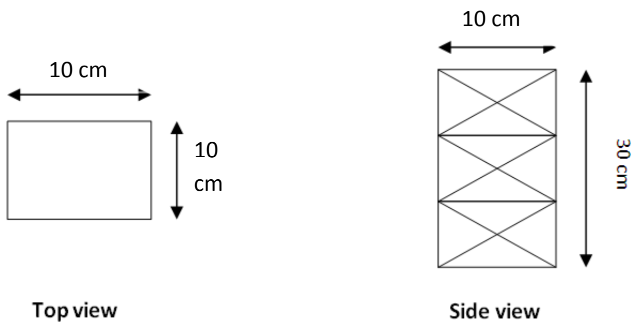


Figure 4.2.1 RAX Cubesat geometry

These borders of the RAX and the crosses on the four sides constitute the ground plane of the antenna's that will be placed on the bottom of the satellite.

The antenna placement on the RAX was simulated using Numerical Electromagnetic code (NEC). NEC was used because of the complex ground plane that the antennas would have on the RAX. NEC made it easier to model the ground plane by specifying the exact wire elements and their geometry.

4.2.2 Antenna placement

The antenna's on the satellite consists of 4 monopoles placed perpendicular to each other so that they look like one crossed-dipole antenna when looking from top. They have 4 different feed lines and each of them is fed a current which is 90 degrees out of phase from the other. The figure below gives the top view and side view of the antennas on the satellite.

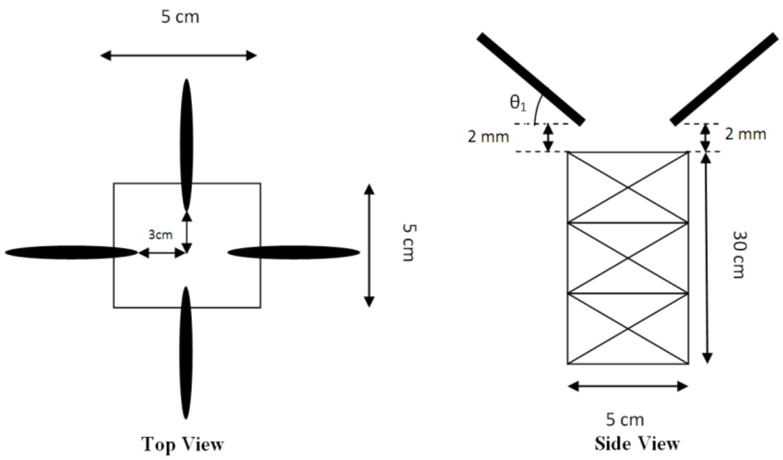


Figure 4.2.2 Antenna placement in RAX

Due to design constraints, the feed point of the antennas remains fixed. The only two parameters that are flexible are the tilt angle (θ_1) and the length of the monopoles. The simulations were run with varying tilt angle and antenna length. The aim was to come up with the maximum gain in the θ = 0° (parallel to the z-axis) direction.

The initial configuration which was received showed the antennas to be of length 14cm and tilted at an angle of 36°.

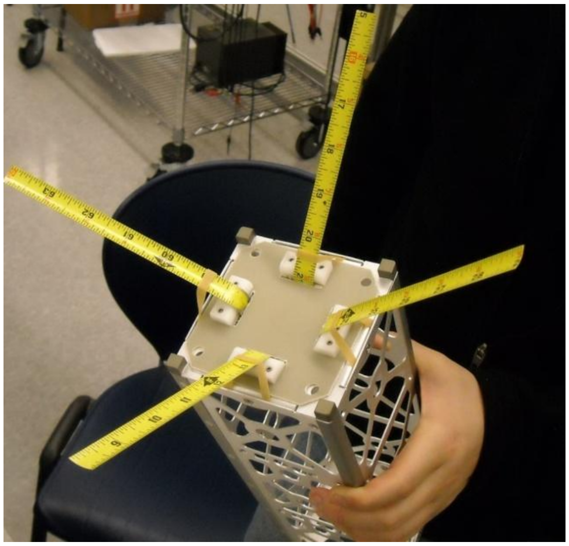


Figure 4.2.3 Initial configuration of antenna in RAX

The results for this configuration were as follows:

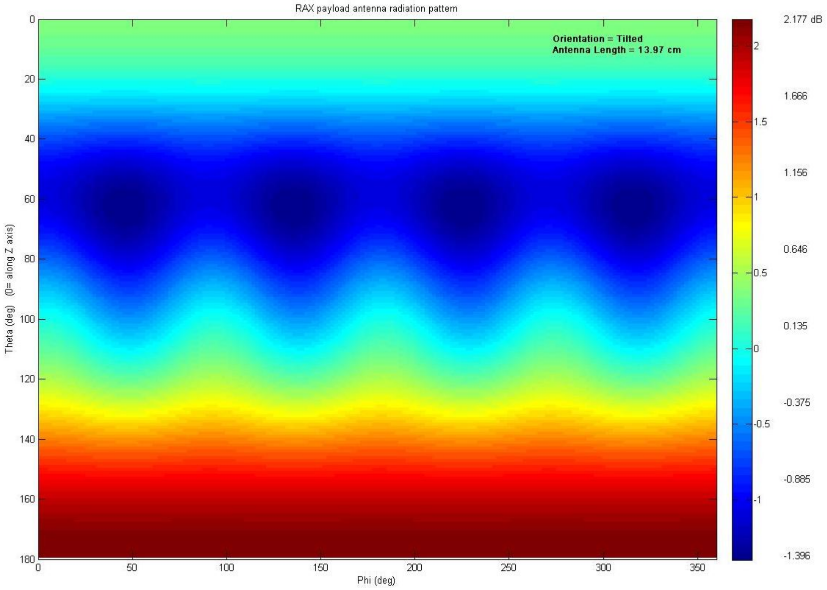


Figure 4.2.4 Radiation pattern of initial configuration of antenna in RAX

It can be seen that in the θ = 0° direction, the gain of the antennas is low. In order to get a higher gain, a range of simulations were carried out for different antenna lengths and tilt angles. The highest gain was achieved for an antenna length of 30 cm and a tilt angle of 0°. The following was the result.

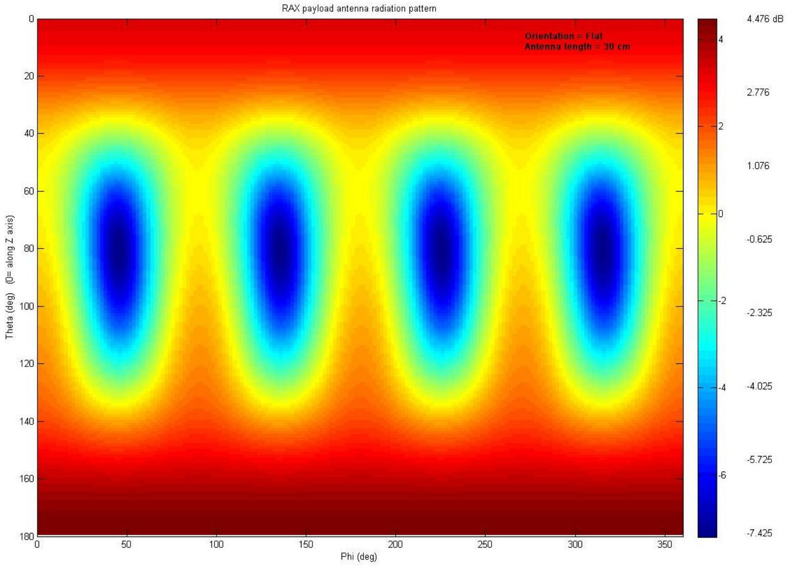


Figure 4.2.5 Radiation pattern of a proposed configuration of antenna in RAX

4.3 Zones of scattering

The zones of scattering represent the aspect angles at different geodetic satellite locations for a particular elevation and azimuth of the radar beam. We plotted the aspect angles at different geodetic longitudes and latitudes which fall within three degrees; the plot is referred to as *Loci Plot of Perpendicularity*. The significance of this plot is that it provides us the scattering zone where the experiments will be conducted. The scattering zone is defined as |θ| < 3°. We plotted the scattering zones within 55° to 70° geodetic latitude and -165° to -135° longitude. The radar beam was set to an elevation of 50° and azimuth of 0°.

4.3.1 100 km altitude

Figure 4.3.1 below represents the zone of scattering at an altitude of 100 km.

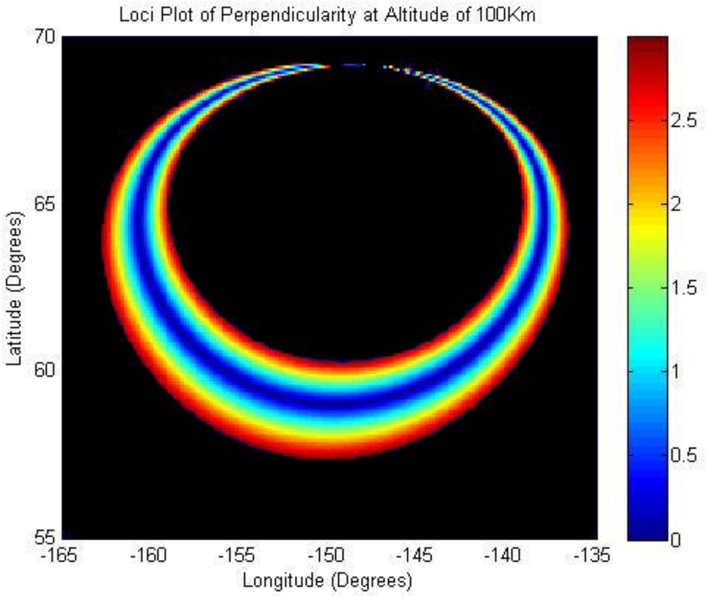


Figure 4.3.1 Loci plot of perpendicularity at altitude of 100km

As you can see in the figure above the ring represents the area where the aspect angle is within 3 degrees. The middle of the ring, which is deep blue in color, represents near zero degree of aspect angle. This tells us that the ionospheric irregularities at that location are closely aligned (within 3°) with the geomagnetic field and are desirable locations (high SNR) to take measurements. Scattering is more intense within the ring. Also narrow the ring width, the more field aligned are the irregularities. The black regions within and around the colored ring have aspect angle greater than three degrees.

4.3.2 300 km altitude

Figure 4.3.2 below represents the zones of scattering at an altitude of 300 km. Similar to the scattering zone at 100 km, this one also has a ring nature with the central region of the ring representing near zero degree of aspect angle.

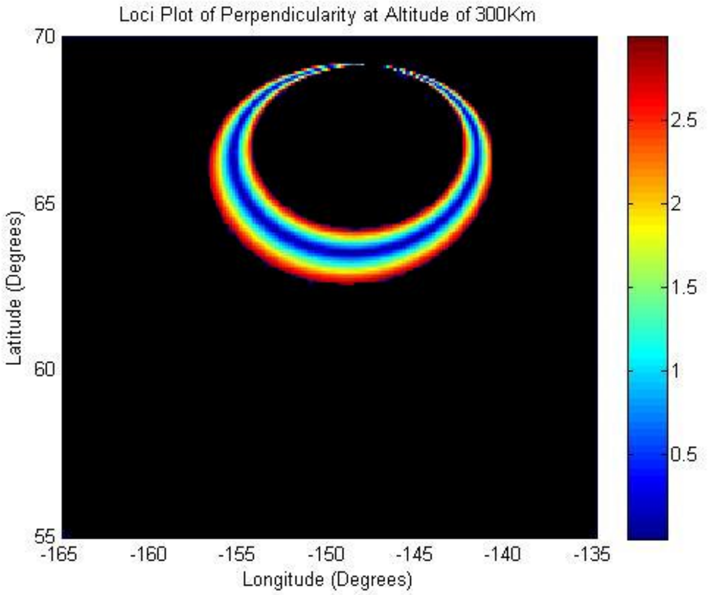


Figure 4.3.2 Loci plot of perpendicularity at altitude of 300km

4.4 Altitude and aspect angle ambiguity function

The altitude and aspect angle ambiguity function is a plot of the value of ω as a function of altitude and aspect angle for a particular satellite location and particular radar beam elevation and azimuth. The significance of this plot is that it tells us about the altitude and aspect angle resolution for a particular ωi,j(h, θ). In other words it tells us how far our reading deviates from the bore sight in terms of altitude and aspect angle for each $\omega_{i,j}(h, \theta)$. The ambiguity function is analogous to the window of view of a telescope. It specifies the h and θ regions associated with sample $S_{i,\ j}$. By changing the satellite location and radar direction we can move around the sky taking observations at different points, the readings falling within the altitude and aspect angle resolution specified by the ambiguity function.

The data we obtained from software for each $\omega_{i,j}(h, \theta)$ was discrete and varied in non-uniform steps. In order to plot the ambiguity function, we implemented a smoothing process to obtain uniformly spaced data points. For this, we took a certain range of altitude and aspect angle, which we divided with respect to a fixed step size, and summed the effect of the discrete values of $\omega_{i,j}(h, \theta)$ at those sets of altitude and aspect angle. Following equation was used to calculate the smoothed data points:

$$\omega(h,\theta) = \sum_{k=1}^{N} \omega_k * C_o * e^{\left(-\left(\frac{h-h_k}{\Delta h}\right)^2 - \left(\frac{\theta-\theta_k}{\Delta\theta}\right)^2\right)} \tag{4.2}$$

In the above equation, N is the number of discrete data points in each $\omega_{i,j}(h, \theta)$, ω_K is the discrete value of $\omega_{i,j}(h, \theta)$, C_O is an arbitrary constant, h and θ are the new height and aspect angle within the new range while h_K and θ_K are the discrete height and theta value. Δh and $\Delta\theta$ are also constants. It should be noted that this smoothing process is only for plotting purposes and the values obtained were not used in any other calculations.

4.4.1 Role of antenna pattern

The antenna pattern of the PFISR provides us information about the distribution of the transmitting power of the antenna which in turn determines the shape of $\omega_{i,j}(h, \theta)$. For the PFISR, transmitting power is high along the center. As such while synthesizing $\omega_{i,j}(h, \theta)$, we considered the volume that falls within three degrees of elevation from the center of the beam. This ensures that most of the volume is considered during $\omega_{i,j}(h, \theta)$ synthesis.

4.4.2 Change of ambiguity function due to satellite motion at a fixed altitude

We plotted the ambiguity function at eleven different satellite locations keeping the altitude fixed. This was done for following five altitudes of irregularity: 90 km, 95 km, 100 km, 105 km, 110 km, 115 km and 120 km. The ambiguity functions can be found in appendix A. Figure 4.4.1 shows the ambiguity function for altitude 90 km.

For the plots, the radar was set to an elevation of 50° and azimuth of 0°. The pulse width of the transmitted electromagnetic beam is considered to be 1 µsec. Also, following 11 satellites with geodetic locations as follows were used:

S.No.	Latitude	Longitude	Altitude
1.	59.824833	-157.515667	665.950000
2.	59.872217	-157.440733	665.965000
3.	59.919600	-157.365800	665.980000
4.	59.966983	-157.290867	665.995000
5.	60.014367	-157.215933	666.010000
6.	60.061750	-157.141000	666.025000
7.	60.109133	-157.066067	666.040000

8.	60.156517	-156.991133	666.055000
9.	60.203900	-156.916200	666.070000
10.	60.251283	-156.841267	666.085000
11.	60.298667	-156.766333	666.100000

It can be seen from the figures below that the altitude resolution for a fixed altitude is around 5 km and the aspect angle resolution in around 1° to 2°. Moreover, as the satellite position progresses, the ambiguity function is shifted away from the center (aspect angle of 0°).

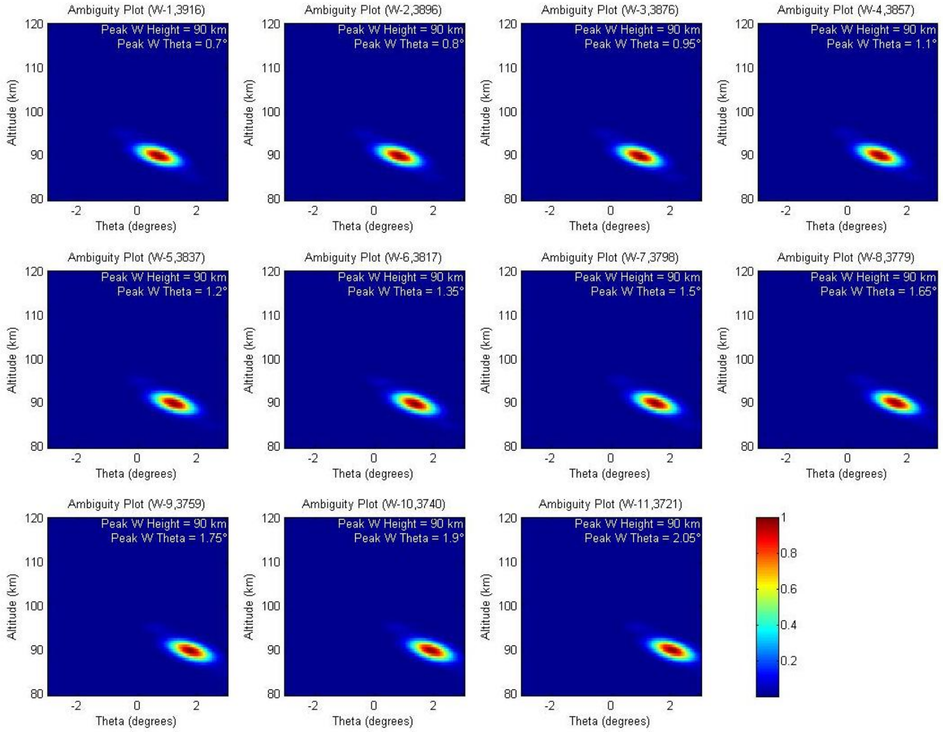


Figure 4.4.1 Ambiguity functions for the 11 satellite locations with irregularity altitude 90 km

4.4.3 Change of ambiguity function for 1 µsec, 10 µsec and 100 µsec pulse widths at 100 km and 300 km

Next for the same satellite locations and same PFISR pointing elevation and azimuth, we plotted the ambiguity function for pulse widths of 1 µsec, 10 µsec and 100 µsec at center of altitudes 100 km and 300 km. The ambiguity functions reside in appendix B. Similar to the previous section, the altitude resolution is around 5 km and the aspect angle resolution around 2°. Also, for the particular eleven satellite positions, the function shifts towards the right.

4.4.4 Change of ambiguity function for scattering angles 90° and 135° in elevation

Keeping the elevation and azimuth of the radar fixed again at 50° and 0° respectively we plotted the ambiguity function for scattering angles 90° and 135° in elevation. The altitude and elevation resolution is different in the two ambiguity functions. Ambiguity function for 90° scattering angle has a higher resolution as it can be seen by comparing figure 4.4.2(a) and 4.4.2(b). The satellite geodetic location used for producing the 90° scattering angle ambiguity function is 63.5° latitude, -148° longitude and 665.9 km altitude. The satellite geodetic location used for producing the 135° scattering angle ambiguity function is 67.5° latitude, -142° longitude and 665.9 km altitude.

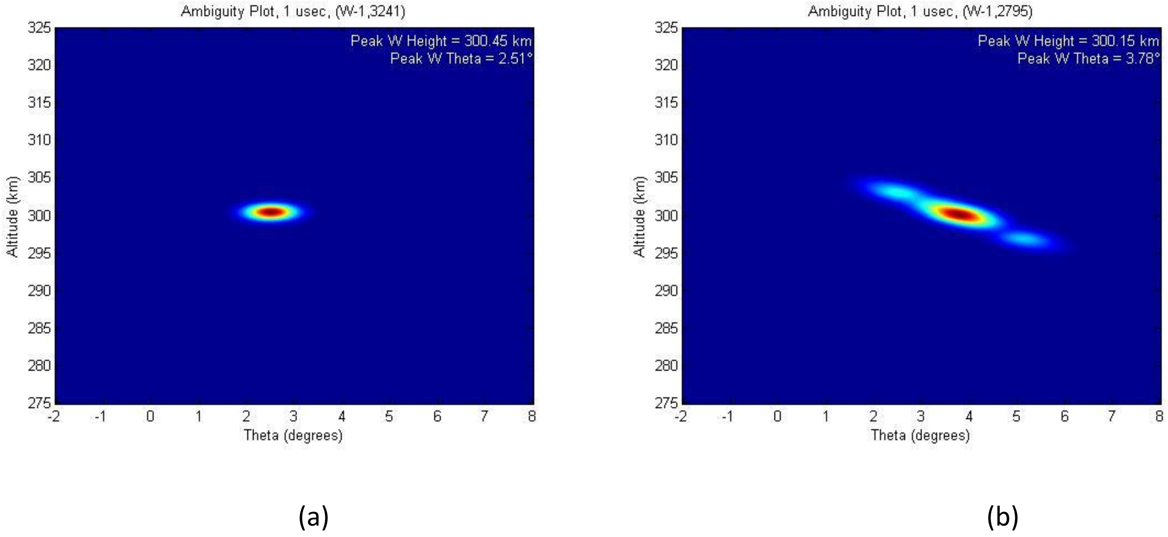


Figure 4.4.2 Ambiguity functions for scattering angles of 90° and 135° in elevation

4.4.5 Change of ambiguity function for scattering angles 0°, 90°, 170° and 270° in azimuth

For an elevation of 50° and azimuth of 0° radar beam direction, we plotted the ambiguity functions at satellite locations 0°, 90°, 170° and 270° azimuth around the radar. These locations are highlighted in the loci plot of perpendicularity, figure 4.4.3. The ambiguity functions are shown in figure 4.4.4. The altitude resolution is around 5 km and the aspect angle resolution around 2°.

The satellite locations used for producing the ambiguity functions are as follows:

Scattering Angle	Satellite Location (Geodetic)		
	Latitude	Longitude	Altitude
0°	58.425	-147.45	665.965
90°	64.98329	-161.95	665.95
170°	69.06	-154.95	665.95
270°	64.9829	-136.95	665.95

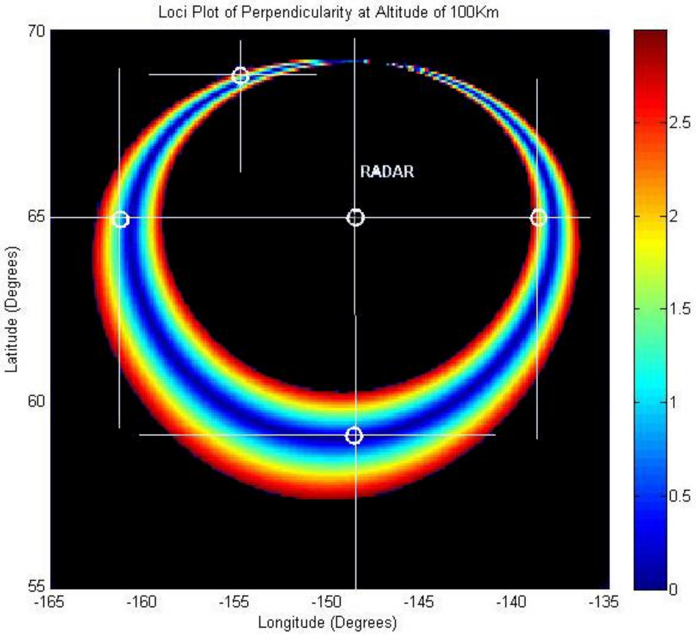
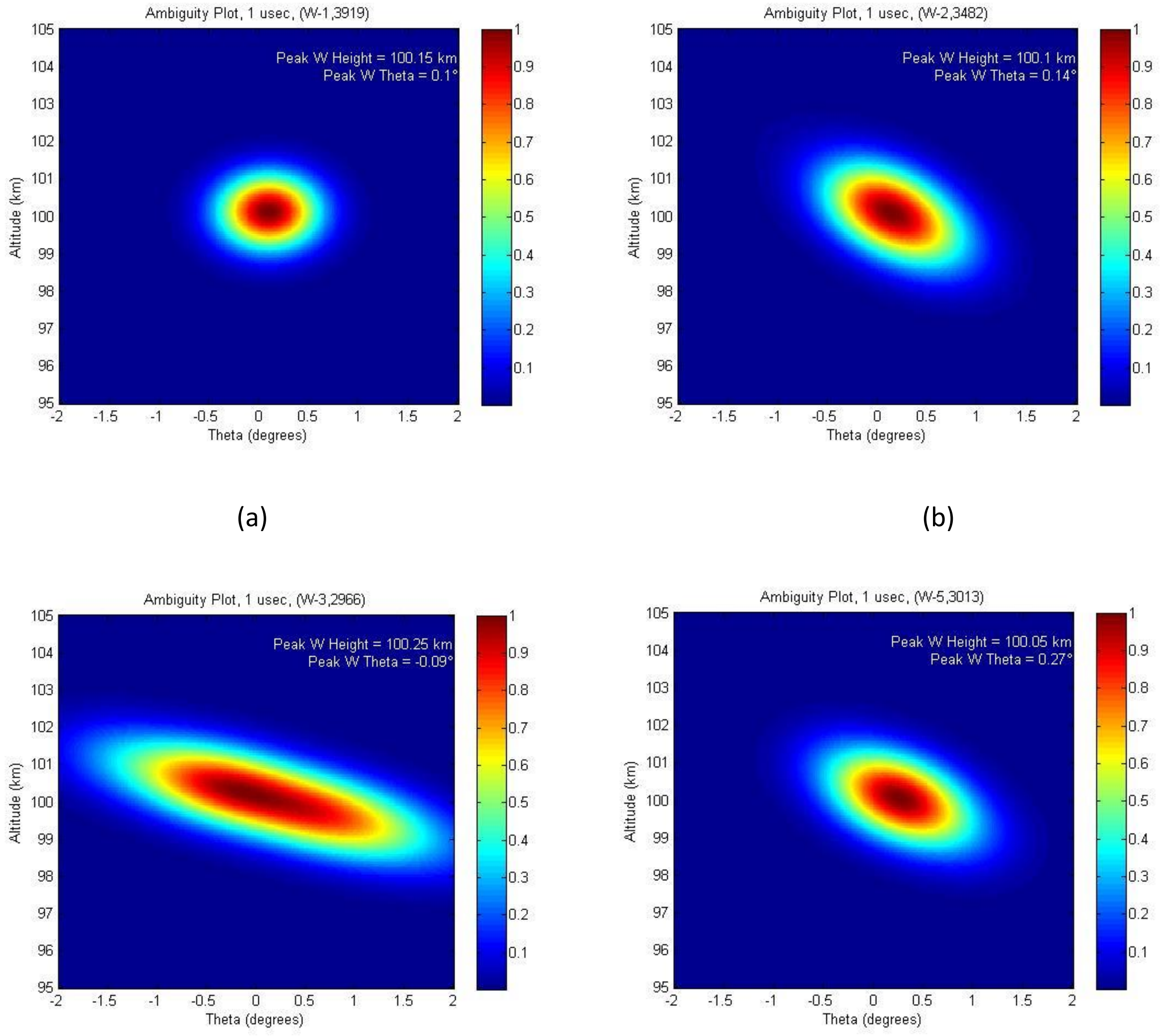


Figure 4.4.3 Loci plot of perpendicularity at altitude of 100km



(a)

(b)

(c)

(d)

Figure 4.4.4 Change of ambiguity function for scattering angles 0°, 90°, 170° and 270° in azimuth

4.5 Inversion of synthetic data for irregularity estimates

In the forward model, I and ω function are known and satellite data is computed as a convolution of I and ω function. In the inversion process, however, satellite data and ω function are known but intensity function, I, is unknown. Inversion is hence a deconvolution algorithm to best estimate the unknown I function. The first approach taken was exhaustive method.

4.5.1 Exhaustive Approach

Exhaustive is a very simple and primitive way of searching for the best fit values of the parameters. Using the simple forward model, satellite data, S_{act}, is computed using certain Δh_I and $\Delta\theta_I$ values for I function and certain Δh_ω and $\Delta\theta_\omega$ for ω function. Then in inversion process, Δh_I and $\Delta\theta_I$ values of I function are assumed to be unknown and hence approximated. The approximation is done by estimating I function, I_{est}, for all possible values of Δh_I and $\Delta\theta_I$, then convolving with ω function to get S_{est}. The parameters that result in S_{est} that best fits with the actual forward model satellite data, S_{act}, are the best estimates of the Δh_I and $\Delta\theta_I$ parameters. It is assumed that the Gaussian behavior of I function is known and also the possible ranges of Δh_I and $\Delta\theta_I$.

The best fit is measured in terms mean square error (MSE) between S_{act} and S_{est} given by the equation 4.3.

$$\mathrm{MSE} = \sqrt{\sum_{i=1}^{M}\sum_{j=1}^{N}\frac{\left(S_{ij\,act} - S_{ij\,est}\right)^2}{M * N}} \qquad (4.3)$$

The S_{est} with the lowest MSE has the least deviation from the S_{act} and hence corresponding Δh_{I_est} and $\Delta\theta_{I\_est}$ values are considered the best estimates of those parameters.

If the resolution of Δh_I and Δh_{I_est}, and $\Delta\theta_I$ and $\Delta\theta_{I\_est}$ match, the estimated parameter values would exactly match the actual values. Otherwise, it would converge to the closest possible combination based on Δh_{I_est} and $\Delta\theta_{I\_est}$ resolution.

Figure 4.5.1 below shows Gaussian ω_{act} function(a), I_{act} function(b), their convolved result S_{act}(c), first I_{est}(d), first S_{est}(e), best match S_{est}(f) and corresponding I_{est}(g)

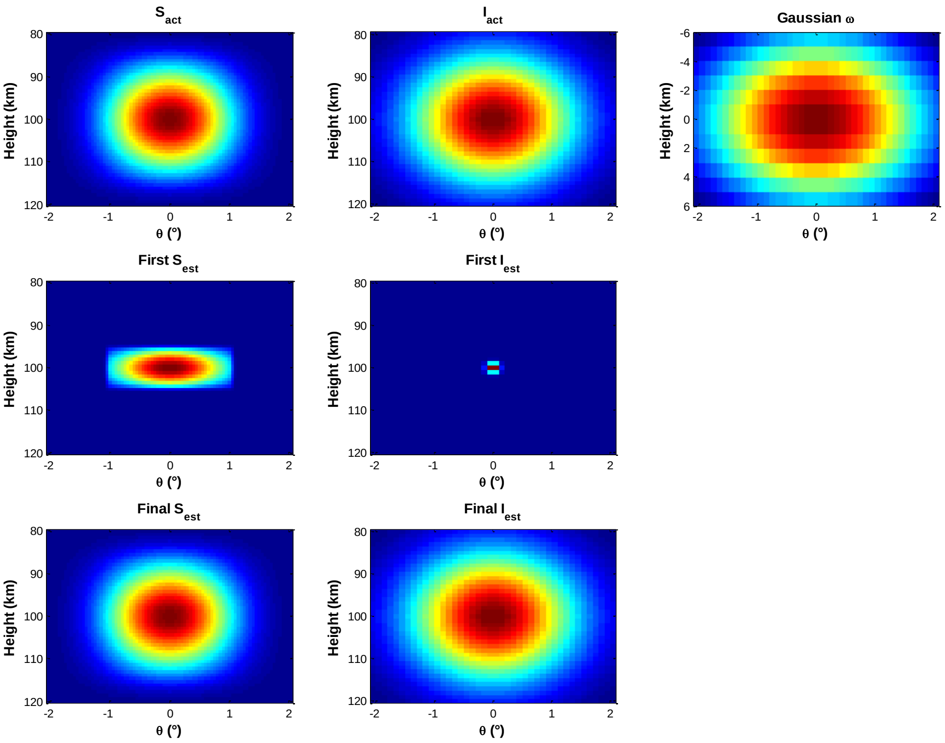


Figure 4.5.1 Exhaustive method of de-convolution for Gaussian ω

	Δh	Δθ
I_{act} Parameters	15	1.5
First I_{est} Parameters	1	0.1
Final I_{est} Parameters	15	1.5

Similar results were obtained when ω_{act} modeled as a sinc function as shown in eqn 3.6.

Figure 4.5.2 below shows sinc ω_{act} function(a), I_{act} function(b), their convolved result S_{act}(c), first I_{est}(d), first S_{est}(e), best match S_{est}(f) and corresponding I_{est}(g)

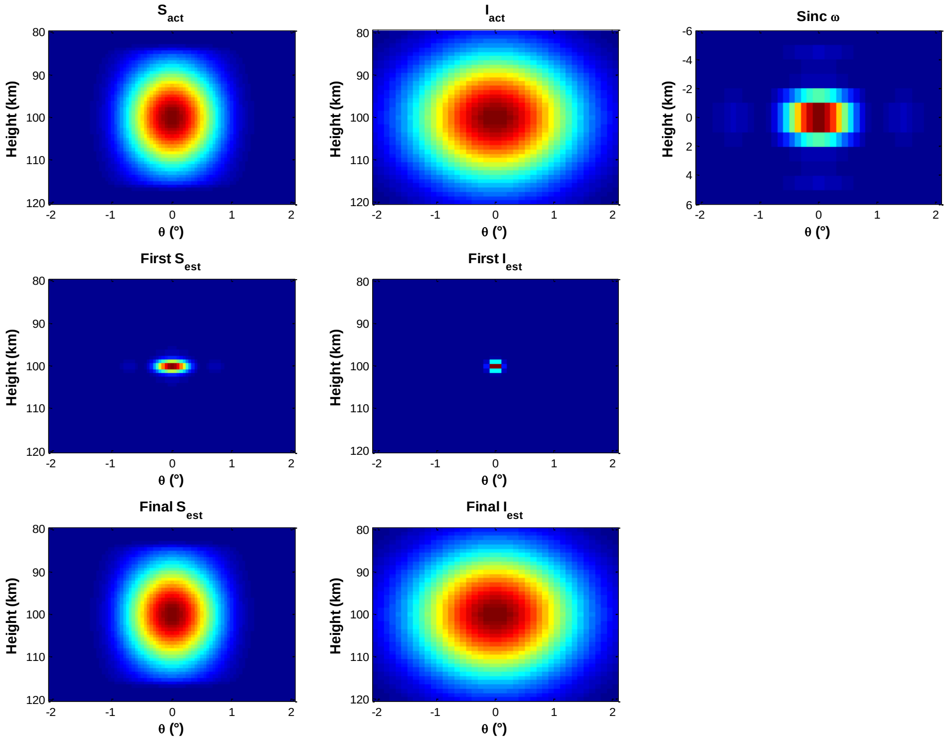


Figure 4.5.2 Exhaustive method of de-convolution for sinc ω

	Δh	Δθ
I_{act} Parameters	15	1.5
First I_{est} Parameters	1	0.1
Final I_{est} Parameters	15	1.5

The exhaustive method of checking each possible combination of parameters would only be effective as long as the number of parameters and the ranges of the parameters are small. The use of this method for higher dimensional model and higher range would be very costly in terms of computational time. With the expectation that the models would have more dimensions and different range, the alternative efficient algorithm was sought for which led us to the method of steepest descent.

4.5.2 Method of steepest decent

The method of steepest descent is much more effective method to converge towards the best estimate of parameters. In this method at first a set of values for the parameters are guessed and their fit with the actual model computed. Then the fitness of the models generated by all the parameter values around the first guess is computed. If any of them has better fitness that would be the next guess of the parameters. The iteration continues until a set of parameter values is reached which produces the best fit of the model. This conclusion can be reached when none of points around the current parameter values produce any better fit to the model.

In the figure 4.5.3 below, for the first pair of estimated parameters, Δh_{est} and $\Delta\theta_{est}$, represented by the point A, I_{est}, S_{est} and MSE between S_{act} and S_{est} are computed. Following this, S_{est} and MSE values are computed for all other points B,C,D,E,F,G,H and I based on parameter values represented by these points. The lowest MSE among them would then be the next estimates of the parameters. If none of those MSE are smaller than S_{est} at A, the parameters represented by A are the best estimates of the parameters.

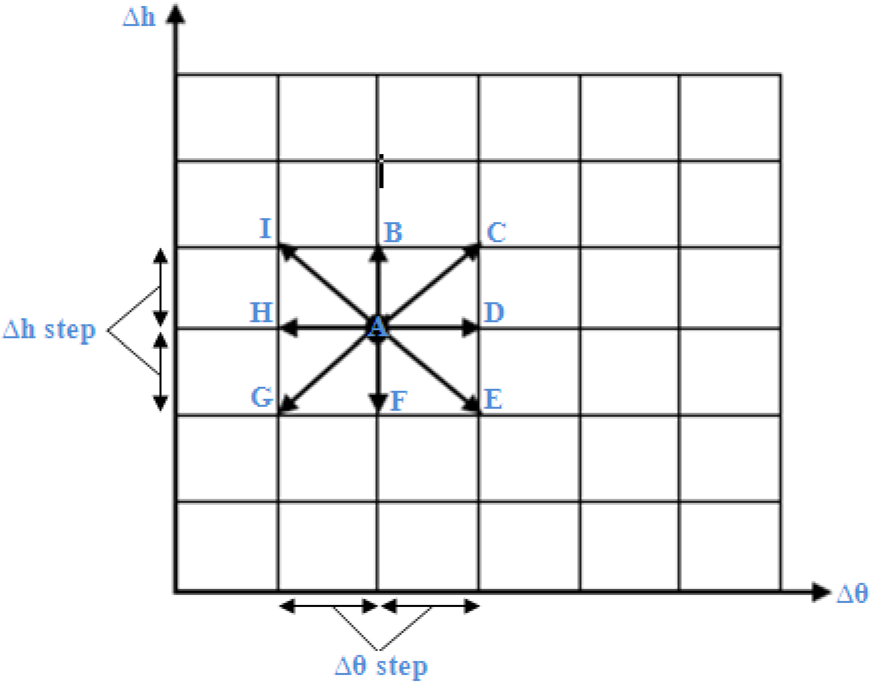


Figure 4.5.3 Steepest Descent Method

Again depending on the resolution of Δh_I and Δh_{I_est}, and $\Delta\theta_I$ and $\Delta\theta_{I\_est}$ match, the estimated parameter values might exactly match the actual values or converge to the closest possible combination based on Δh_{I_est} and $\Delta\theta_{I\_est}$ resolution.

Figure 4.5.4 below shows the Gaussian ω $_{act}$ function(a), I_{act} function(b), their convolved result S_{act}(c), first I_{est}(d), first S_{est}(e), best match S_{est}(f) and corresponding I_{est}(g).

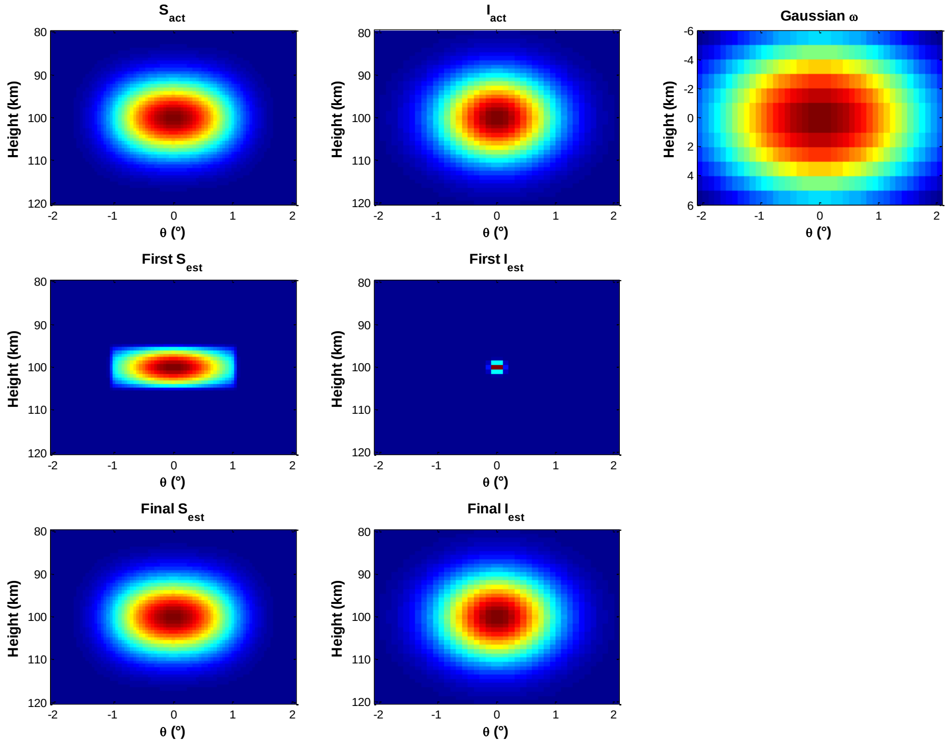


Figure 4.5.4 Steepest descent method of de-convolution for Gaussian ω

	Δh	Δθ
I_{act} Parameters	10	1.0
First I_{est} Parameters	1	0.1
Final I_{est} Parameters	10	1.0

Figure 4.5.5 below shows the Gaussian ω $_{act}$ function(a), I_{act} function(b), their convolved result S_{act}(c), first I_{est}(d), first S_{est}(e), best match S_{est}(f) and corresponding I_{est}(g).

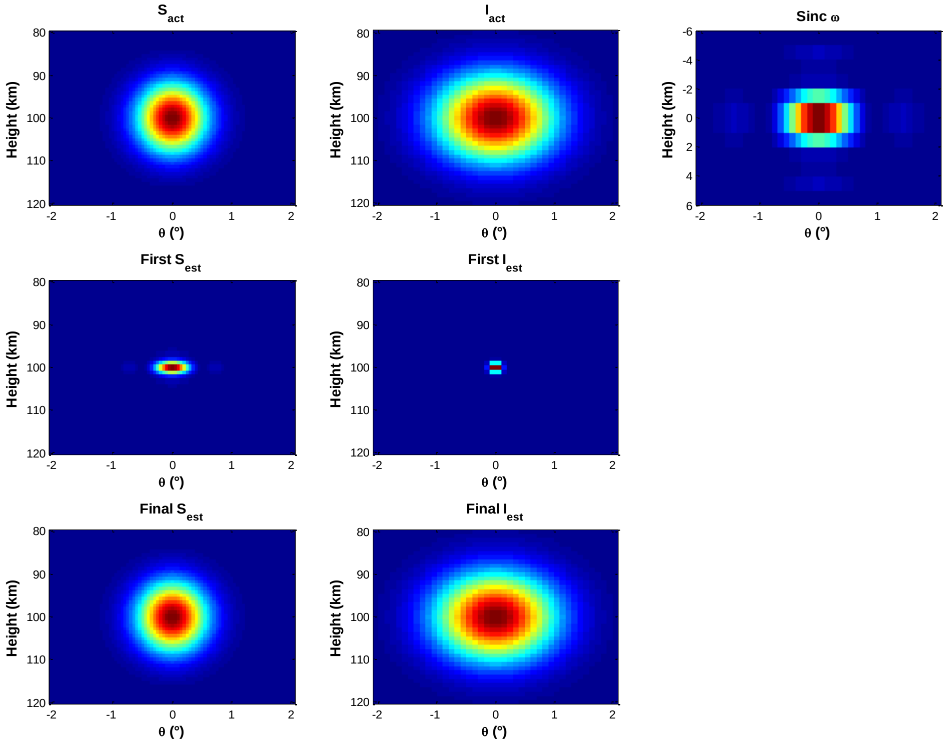


Figure 4.5.5 Steepest descent method of de-convolution for Gaussian ω

	Δh	Δθ
I_{act} Parameters	10	1.0
First I_{est} Parameters	1	0.1
Final I_{est} Parameters	10	1.0

ω model was later modified as shown in eqn 3.16 which takes into account both the radar beam volume divisions, δv, and the power intensity of the those individual δv represented by P_{in}. Power gain factor was initially modeled as a Gaussian $e^{-\left(\frac{\theta}{\Delta\theta}\right)^2}$ and later substituted by the actual PFISR power gain factor computed in section 4.1.

4.5.2.1 Effect of Uncertainty

The effectiveness of the steepest descent algorithm was tested for the consideration of uncertainty case. Uncertainty was added to I_{act} function using the equation 4.4 where uncertainty factor is given by the equation 3.18.

$$I_{uncertain} = I_{act} * \text{Uncertainty factor} \qquad (4.4)$$

Figure 4.5.6 shows the $I_{uncertain}$ and corresponding $S_{uncertain}$ at different values of N, for Gaussian ω_{act}. It also shows the final S_{est} and corresponding final I_{est}. Figure 4.5.7 shows the same thing but for the sinc ω_{act}.

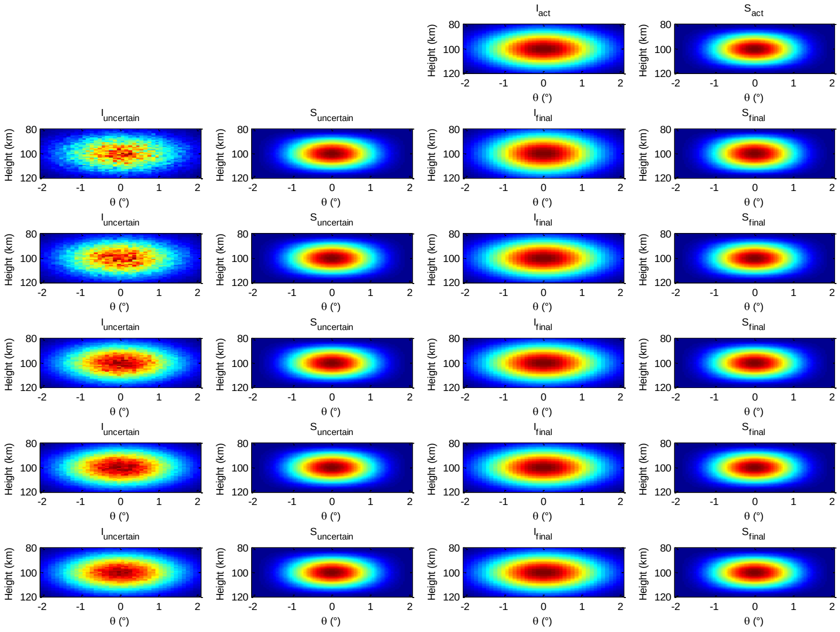


Figure 4.5.6 Effect of uncertainty on steepest descent method of de-convolution with Gaussian ω

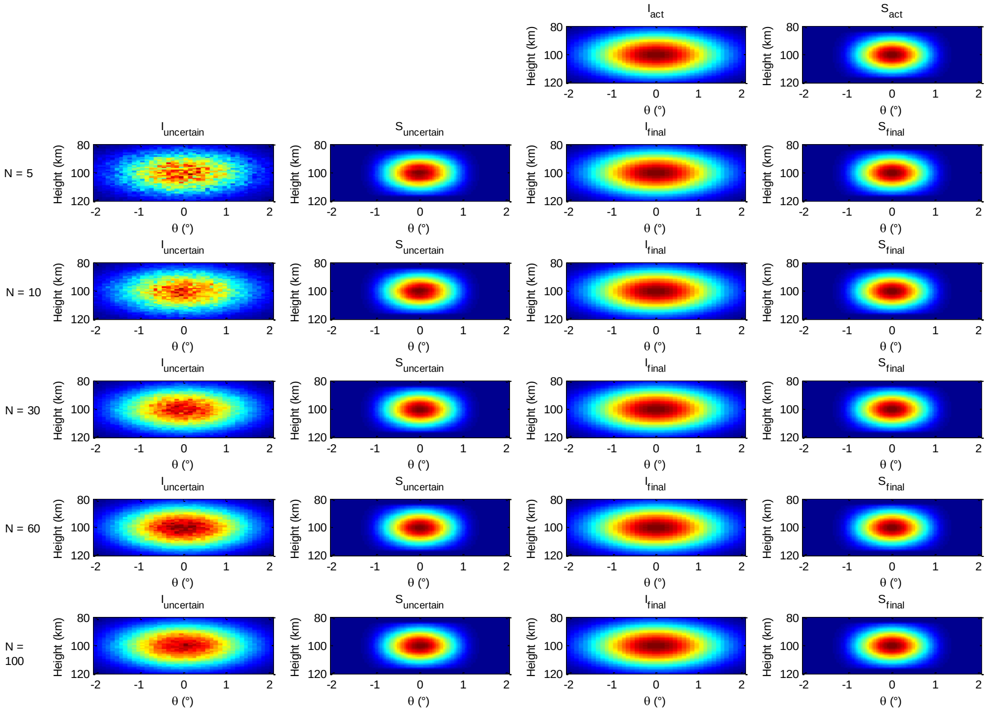


Figure 4.5.7 Effect of uncertainty on steepest descent method of de-convolution with sinc ω

Figure 4.5.7: Effect of Uncertainty on Δh and Δθ approximation (Sinc Kernel)

To get the better statistical approximation, for each value of N, the estimation algorithm is ran 10 times and deviation of the final Δh_{I_est} and $\Delta\theta_{I\_est}$ from Δh_I and $\Delta\theta_I$ are calculated. Figure 4.5.8 below shows the deviation of the estimated parameter values from the actual parameter values when ω_{act} was modeled as Gaussian function. Figure 4.5.9 that follows show the similar graph for sinc ω_{act} and the modified ω_{act} that considered PFISR beam volume and power gain factor.

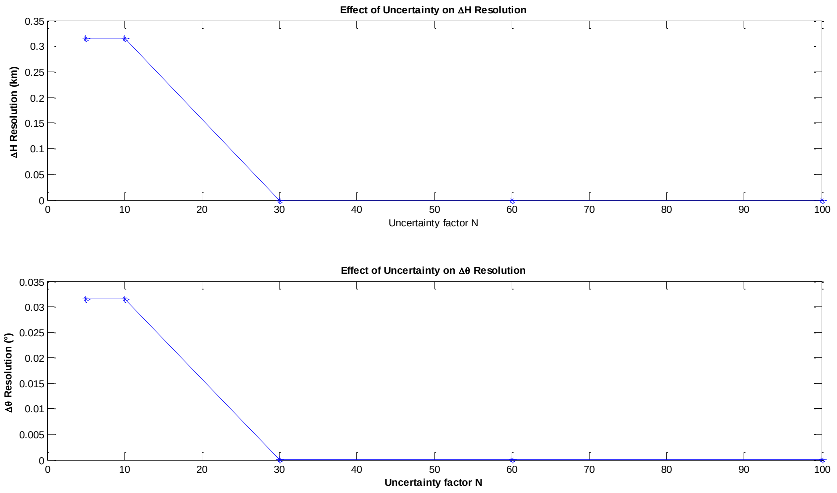


Figure 4.5.8 Δh and Δθ resolution for Gaussian ω

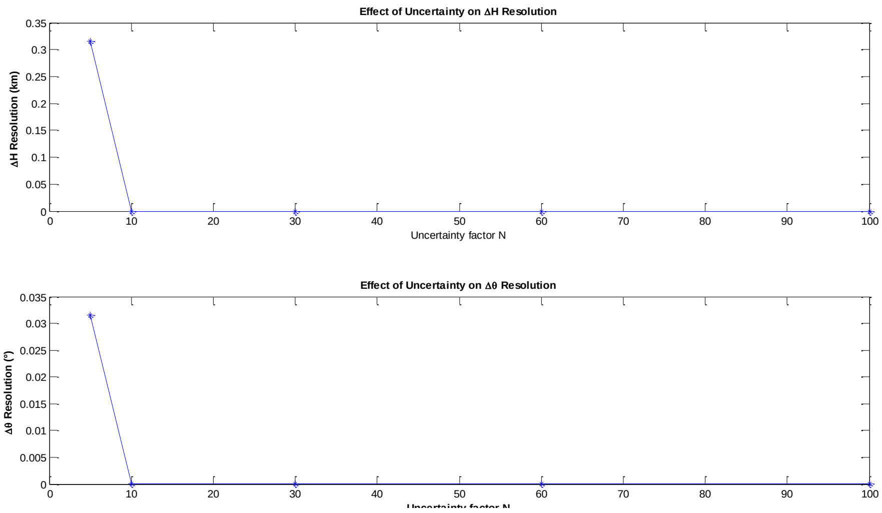


Figure 4.5.9 Δh and Δθ resolution for sinc ω

4.5.2.2 Effect of Noise

The effectiveness of the steepest descent algorithm was also tested for the consideration of noise case. Noise was added to S_{act} function using the equation 4.5 where noise is given by the equation 3.20.

$$S_{\mathrm{noise}} = S_{\mathrm{act}} + \mathrm{noise} \qquad (4.5)$$

Figure 4.5.10 and Figure 4.5.11 show the S_{noise} at different values of M, for the Gaussian ω_{act} and sinc ω_{act} respectively. They also show the final S_{est} and corresponding final I_{est}.

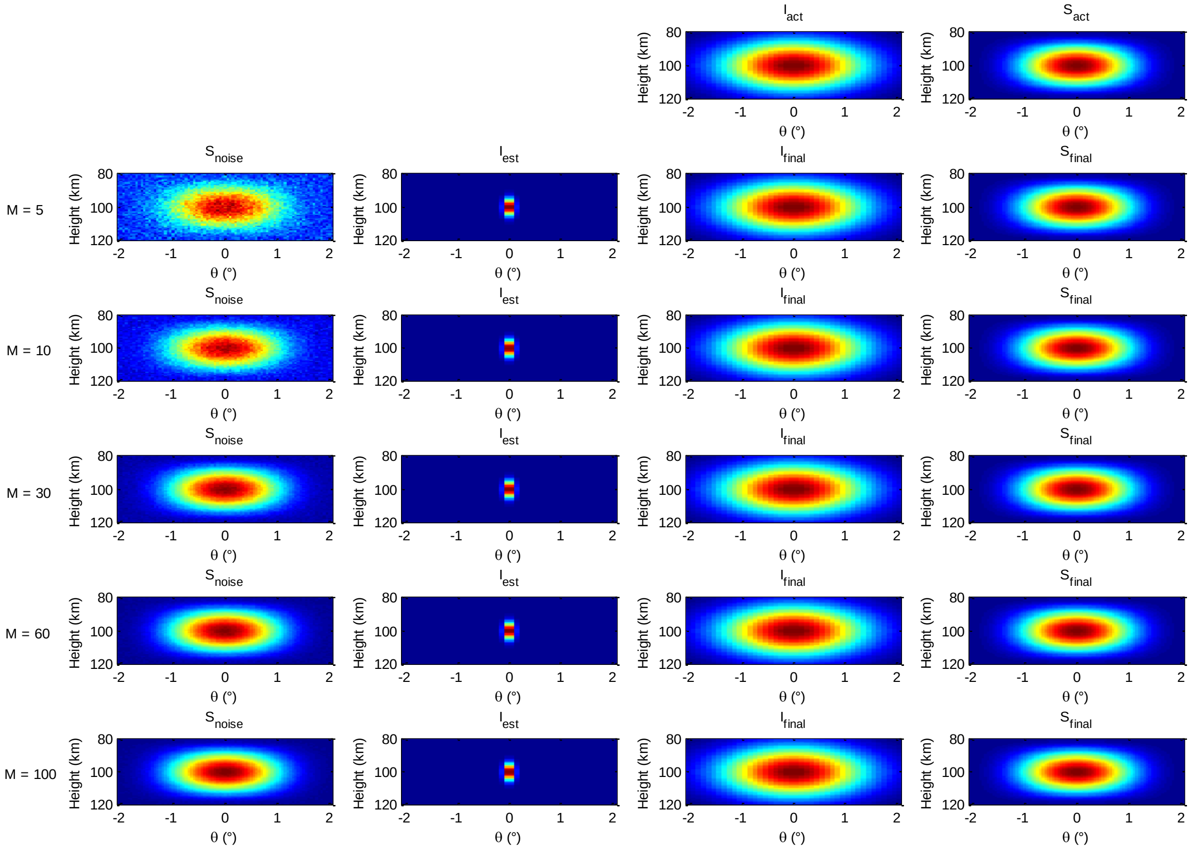


Figure 4.5.10 Effect of noise on steepest descent method of de-convolution with Gaussian ω

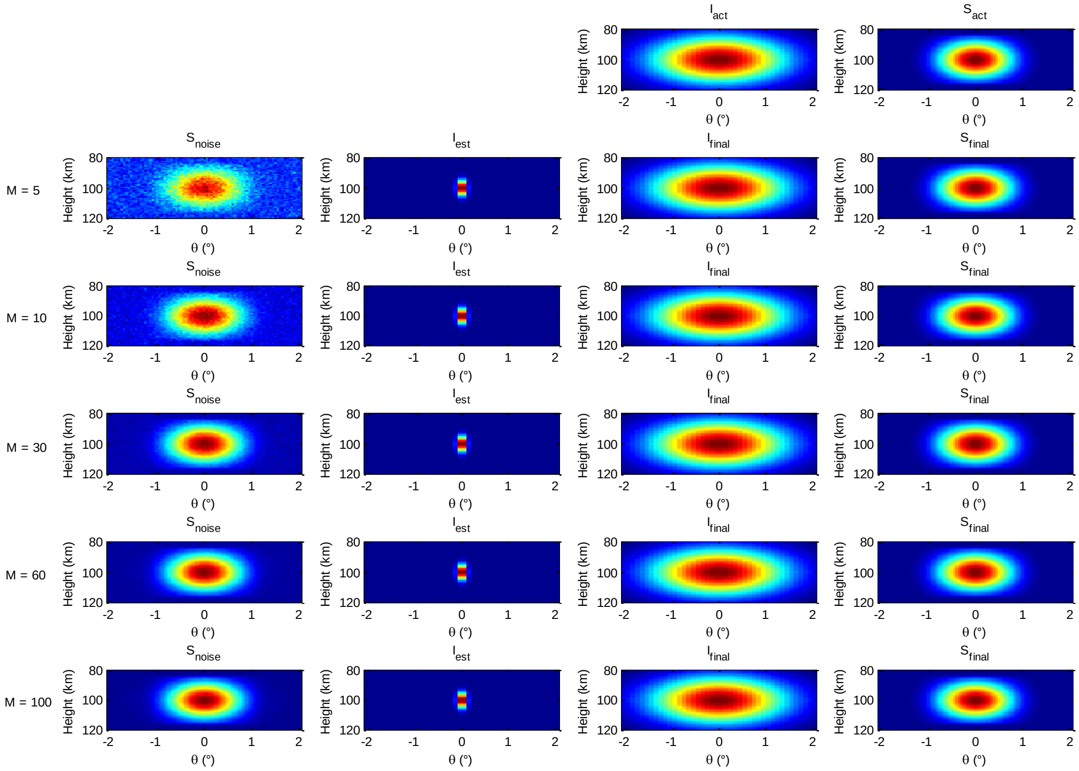


Figure 4.5.11 Effect of noise on steepest descent method of de-convolution with sinc ω

To get the better statistical approximation, for each value of M, the estimation algorithm is ran 10 times and deviation of the final Δh_{I_est} and $\Delta\theta_{I\_est}$ from Δh_I and $\Delta\theta_I$ are measured. Figure 4.5.12 below shows the deviation of the estimated parameters from the actual parameter values for ω_{act}. Figure 4.5.13 shows the deviation for sinc ω_{act} that considered PFISR beam volume and the power gain factor.

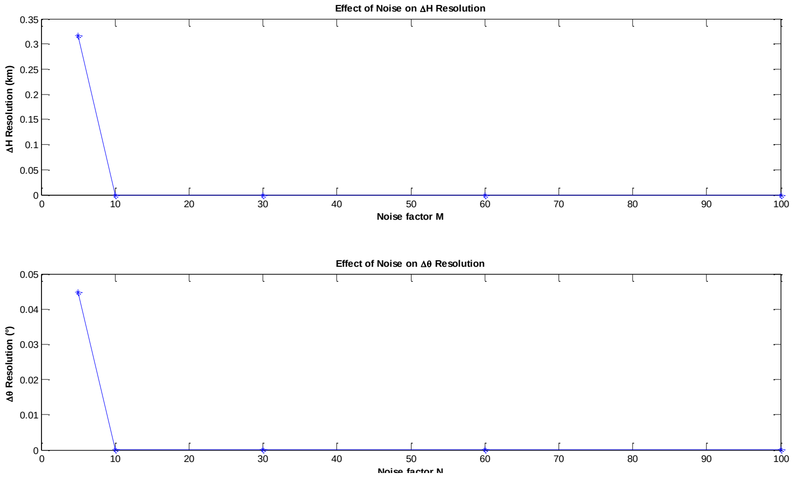


Figure 4.5.12 Δh and Δθ resolution for Gaussian ω

Figure 4.5.12: Δh and Δθ resolution for Gaussian kernel

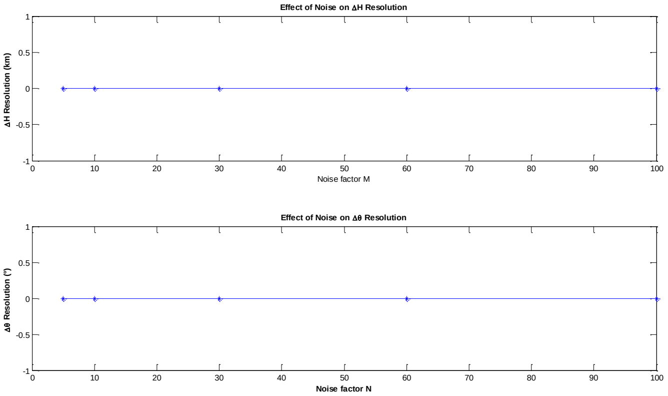


Figure 4.5.13 Δh and Δθ resolution for sinc ω

Figure 4.5.13: Δh and Δθ resolution for Sinc kernel

With the modified ω function that accommodated radar beam volume and transmitting power gain factor, I_{act} function was modified to attain higher altitude resolution as shown in equation 3.3 where Δh was 5 km and I_k and $\Delta\theta_k$ were the parameters of function that were to be approximated. With so many unknown parameters, it would be impossible to calculate MSE in all possible directions and keep track of it. So, given the failure of steepest descent method for multivariable parameters, an alternative and more robust algorithm was sought. The approach of Newton's method was taken.

4.5.3 Newton's method

As with other methods, Newton's method also begins first with guessing values for unknown parameters. However, instead of randomly guessing first values of the parameters, the advantage of experiment geometry and the lower resolution S_{act} data was taken to make more educated guess.

The ambiguity function for the experiment as shown in figure 4.4.1 earlier, shows that for a certain height of bore sight, the ambiguity function expands to about 5km in width centered at the bore sight height. Each of these ambiguity function further refers to certain i and j index of ω and S. Since individual Gaussian functions in the modified I_{act} also have width of 5km, if specific $\omega_{i,j}$ with bore sight height of h km are selected, the corresponding $S_{i,j}$ could be considered sole influence of $I_k e^{-\left(\frac{h-h_{0_k}}{\Delta h}\right)^2} e^{-\left(\frac{\theta}{\Delta\theta_k}\right)^2}$, where h_{0_k} equals the bore sight height of h km.

Based on this theory, for a particular i and j value, $S_{i,j}$ can be simplified to

$$S_{i,j} = \sum_{h,\theta} \omega_{i,j}\,(h,\theta) * I_k e^{-\left(\frac{h-h_{0_k}}{\Delta h}\right)^2} e^{-\left(\frac{\theta}{\Delta\theta_k}\right)^2} \qquad (4.6)$$

Since, $S_{i,j}$ and $\omega_{i,j}$ are known, I_k and $\Delta\theta_k$ can be computed using the exhaustive method as discussed earlier in section 4.5.1. The method is continued for all the 7 known h_{0_k} values and hence compute first estimates of all 7 unknown pairs of I_k and $\Delta\theta_k$.

After getting the first set of parameter estimations, they are used to synthesize S_{est} and compared with S_{act} to compute error matrix, e, which is simply S_{est} subtracted from the S_{act}.

$$e = S_{act} - S_{est} \tag{4.7}$$

Then a Jacobian matrix is calculated which is formed with the elements that are partial derivatives of the error matrix in terms of each of 14 unknown parameters. The partial derivatives are shown in Appendix C.

Hence the Jacobian matrix would be:

$$J_{m,n} = \frac{\partial\left(e_{i,j}\right)_m}{\partial p_n} \tag{4.8}$$

The correction vector, x, is given by the equation X.

$$Jx = e \tag{4.9}$$

Since there are more error measurements than parameters, this system of equations may be over determined. Therefore, the x that minimizes the 2-norm of the residual was found rather than solving for the exact values. (Lowe)

$$x = \min\|Jx - e\|^2 \tag{4.10}$$

Since $\|Jx - e\|^2 = (Jx - e)^T(Jx - e)$, it can be shown that this minimization has the solution as the normal equations,

$$J^T J x = J^T e \tag{4.11}$$

where J^T is the transpose of J. It is assumed that the function is locally linear over the range of typical errors which is true for our experiment.

Once the correction vector x is obtained, the next value for parameters would be:

$$p_{next} = p_{previous} - x \tag{4.12}$$

With this algorithm, the parameters were expected to converge to certain value but some of them were oscillating between two sets of values. Hence to ensure that the estimated parameters converge to certain values, at the cost of slow convergence, p_{next} was changed to:

$$p_{next} = p_{previous} - \frac{x}{2} \tag{4.13}$$

4.5.3.1 *Effect of uncertainty and noise*

For the Newton's approach both uncertainty and noise were added at the same time unlike before when they were introduced and analyzed separately. Equation 3.22 which is repeated below for convenience shows how the noise and uncertainty factor are added.

$$\left(S_{i,j}\right)_{\text{noise +uncertainty}} = \left(S_{i,j} + \frac{S_{i,j_{\text{max}}}}{M}\right)\left(1 + \frac{\text{randn}}{\sqrt{N}}\right) \tag{4.14}$$

Figure 4.5.14 below shows the percentage error of estimated I_k parameters and degree error of estimated $\Delta\theta_k$ parameters for 10 iterations of different values of M and N.

Figure 4.5.15 and 4.5.16 show the percentage error of estimated I_k parameters and degree error of estimated $\Delta\theta_k$ parameters respectively, for different values of M and N. 10 iteration values are converted into single values using the following equations.

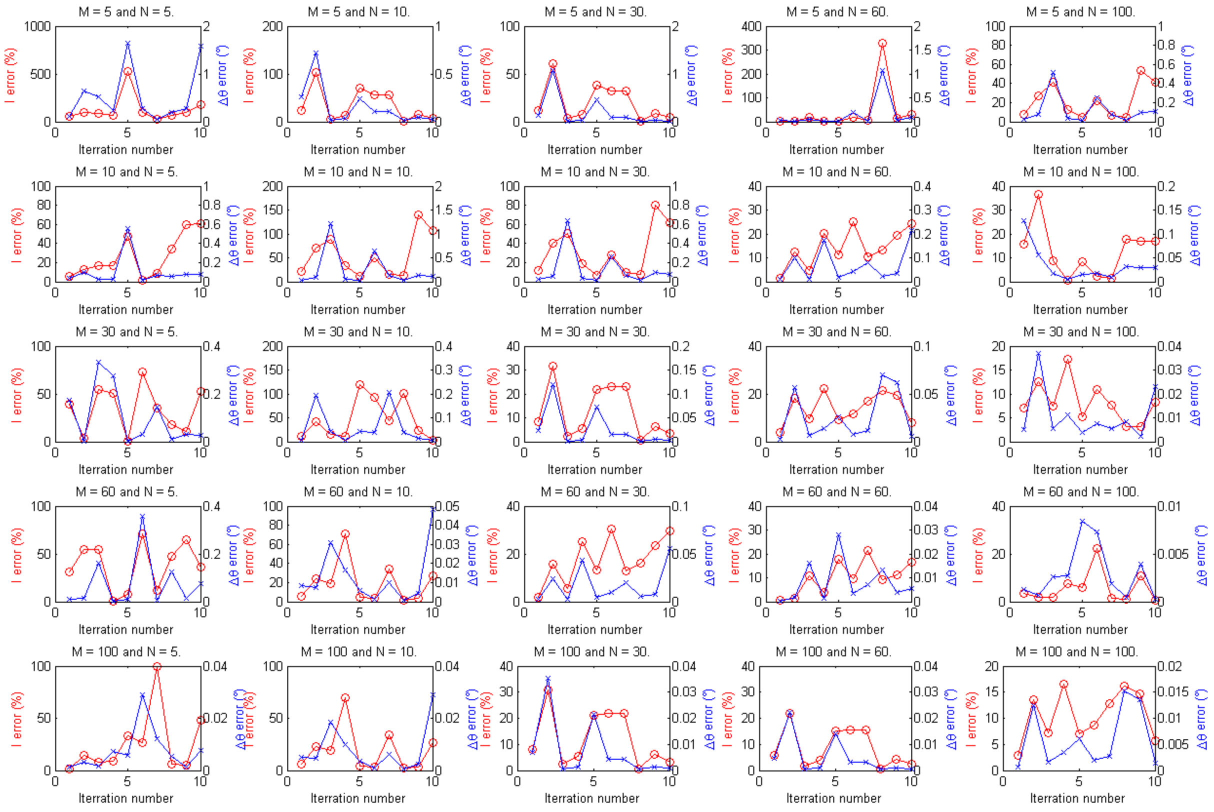


Figure 4.5.14 Error plots of Ik and Δθ for different uncertainty and noise parameters

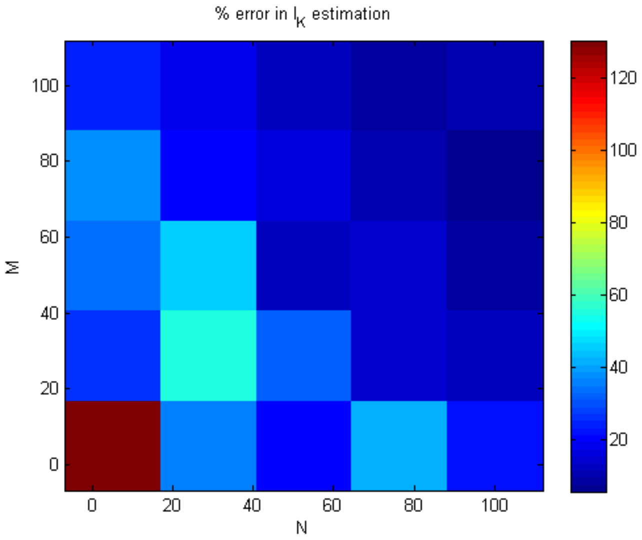


Figure 4.5.15 Percentage error in Ik estimation for different uncertainty and noise parameters

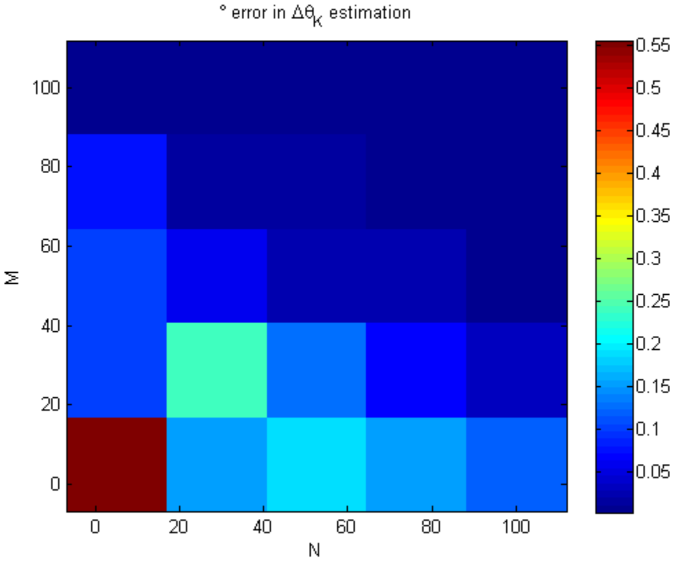


Figure 4.5.16 Degree error in Δθ estimation for different uncertainty and noise parameters

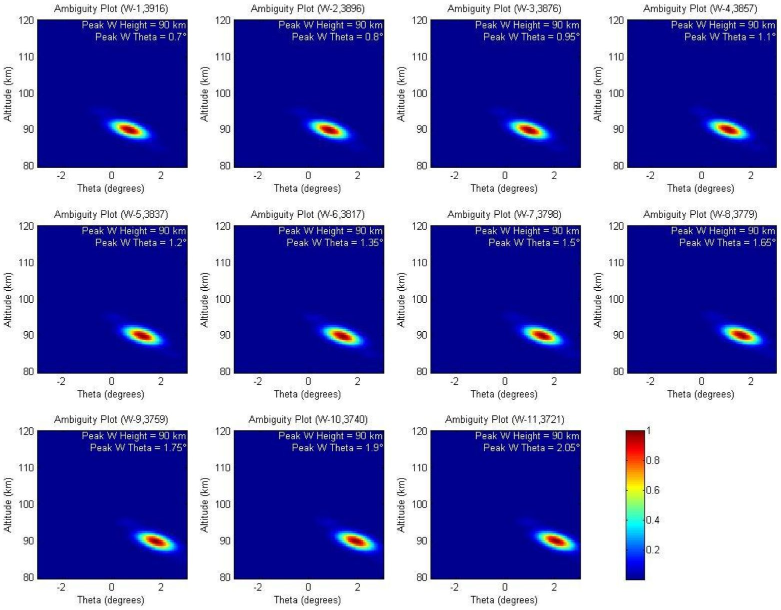
5 Conclusion

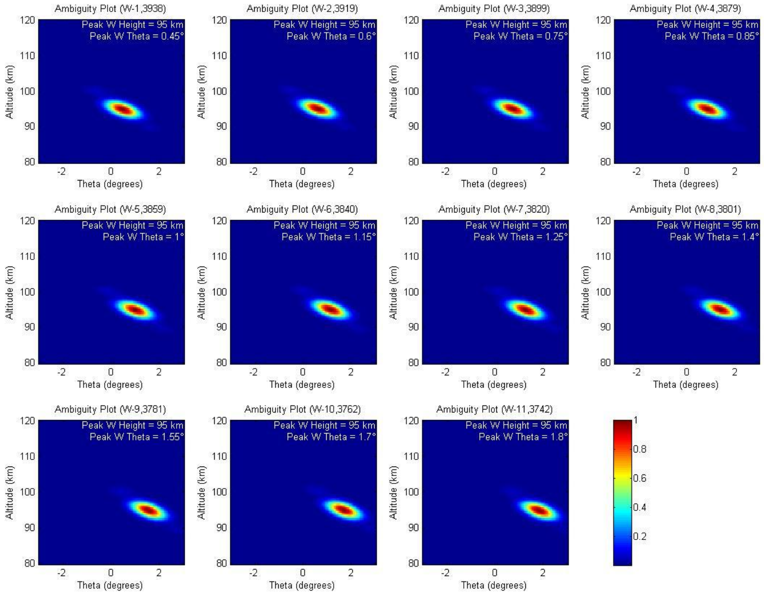
We successfully created a simulation of the satellite radar mission and generated synthetic data. We ensured that the simulation was as realistic as possible by implementing accurate scattering geometry and by using the PFISR radiation pattern to account for the power factor of the radar. Through this data we were able to generate ambiguity functions which showed that we can achieve 5 km altitude resolution and 1° to 2° aspect angle resolution. Three de-convolution algorithms were implemented among which the Newton's method along with method of least square minimization proved to be the most effective. Our results show that we can estimate the irregularity intensities with less that 10% error and also increase our aspect angle resolution up to 0.1°. Moreover, by plotting the scattering zones we also looked at the ideal locations for the satellite to make experiment readings. Overall, the project was a success and we can conclude that the RAX mission will provide high altitude and aspect angle resolution to characterize the ionospheric irregularities.

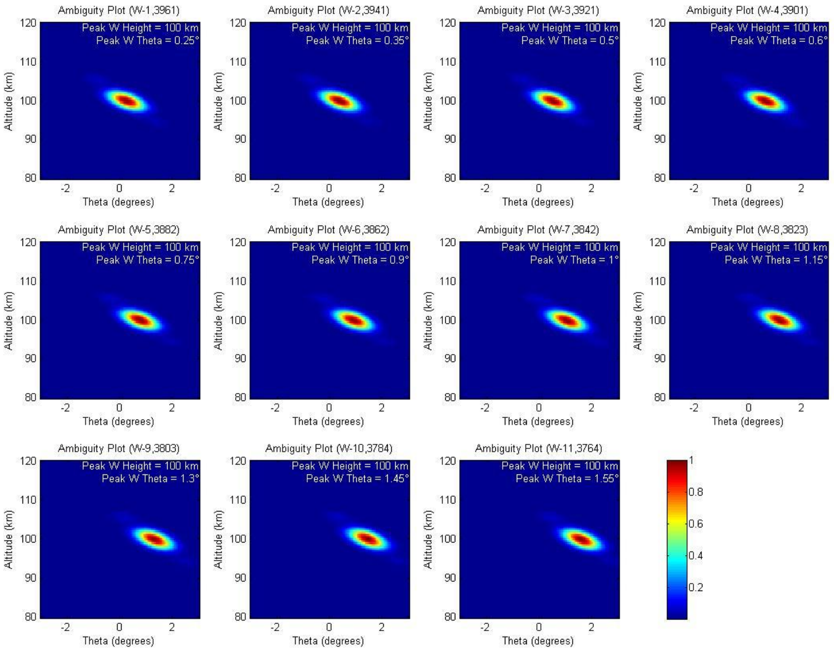
Works Cited

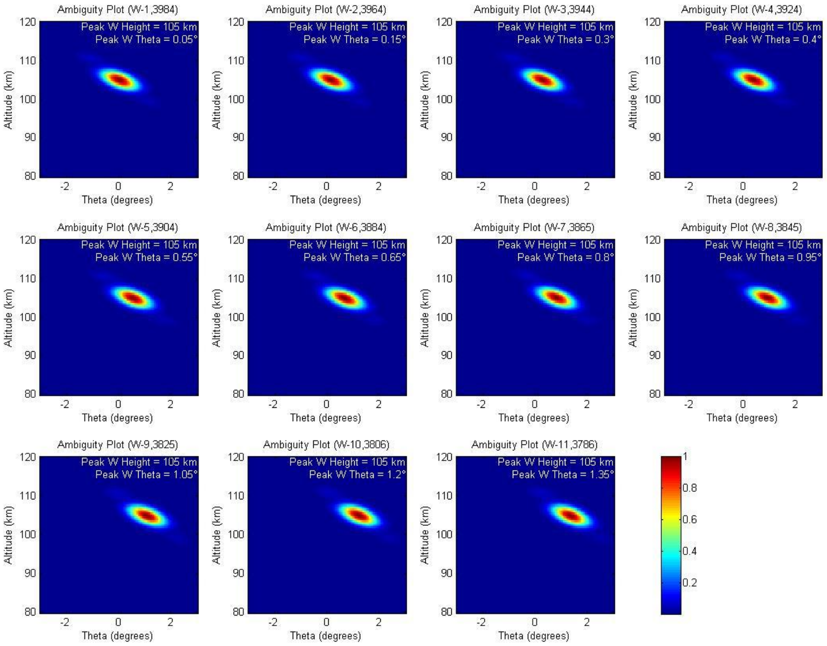
Lowe, David G. http://www.cs.ubc.ca/. 1 March 2002. February 2009 <http://www.cs.ubc.ca/spider/lowe/papers/pami91/pamilatex.html>.

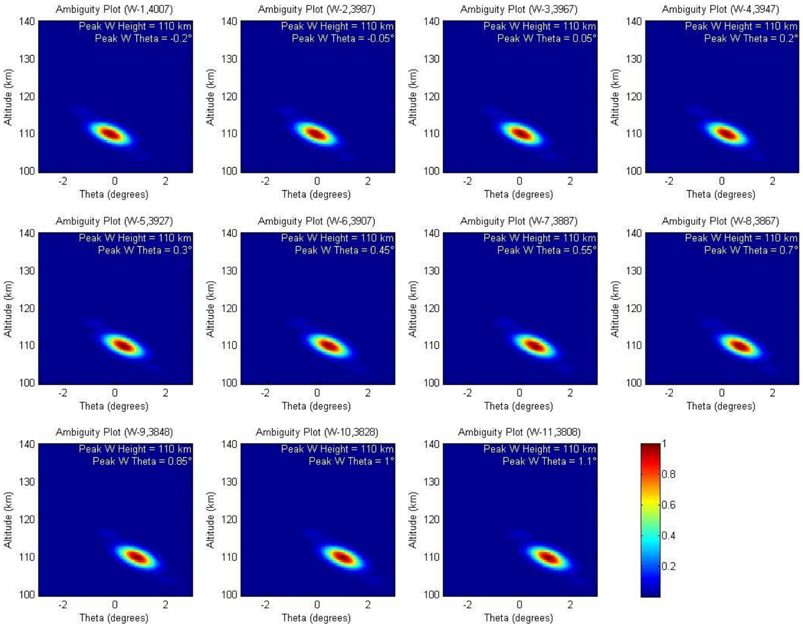
Appendix A: Change of ambiguity function due to satellite motion at five different altitudes

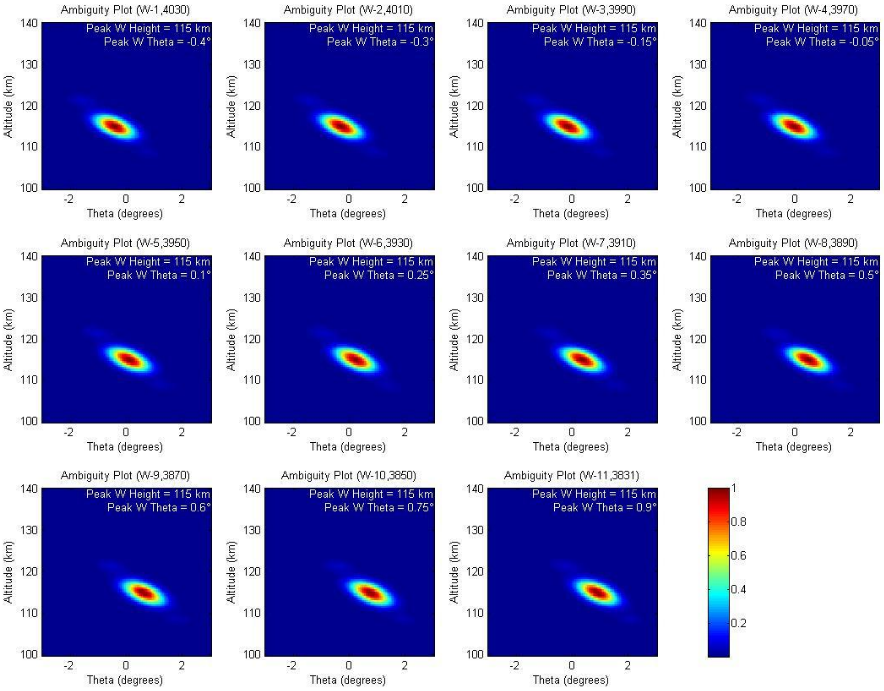


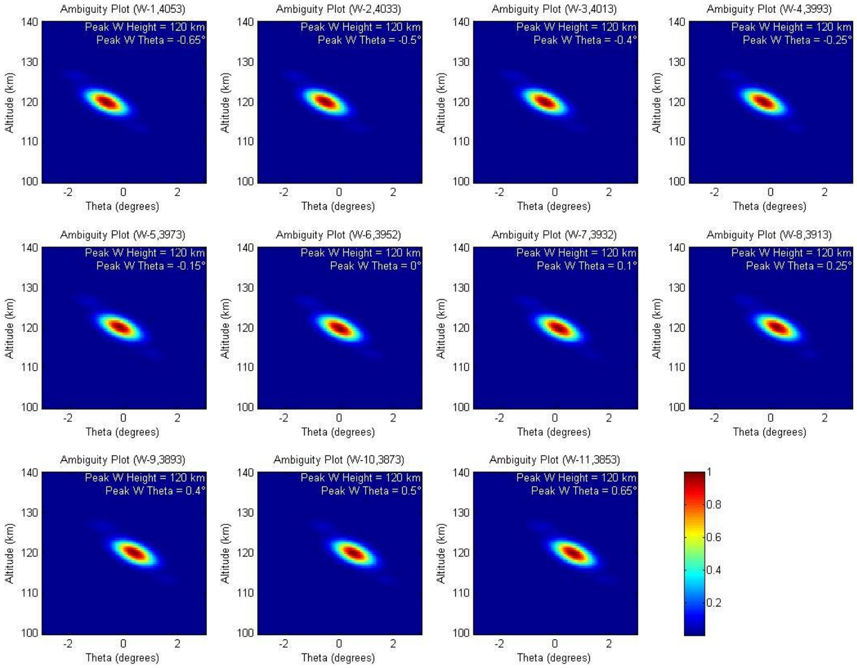



Ambiguity Plot (W-1,3961)
Peak W Height = 100 km
Peak W Theta = 0.25°
Ambiguity Plot (W-2,3941)
Peak W Height = 100 km
Peak W Theta = 0.35°
Ambiguity Plot (W-3,3921)
Peak W Height = 100 km
Peak W Theta = 0.5°
Ambiguity Plot (W-4,3901)
Peak W Height = 100 km
Peak W Theta = 0.6°
Ambiguity Plot (W-5,3882)
Peak W Height = 100 km
Peak W Theta = 0.75°
Ambiguity Plot (W-6,3862)
Peak W Height = 100 km
Peak W Theta = 0.9°
Ambiguity Plot (W-7,3842)
Peak W Height = 100 km
Peak W Theta = 1°
Ambiguity Plot (W-8,3823)
Peak W Height = 100 km
Peak W Theta = 1.15°
Ambiguity Plot (W-9,3803)
Peak W Height = 100 km
Peak W Theta = 1.3°
Ambiguity Plot (W-10,3784)
Peak W Height = 100 km
Peak W Theta = 1.45°
Ambiguity Plot (W-11,3764)
Peak W Height = 100 km
Peak W Theta = 1.55°
Altitude (km)
120
110
100
90
80
-2
0
2
Theta (degrees)
1
0.8
0.6
0.4
0.2


Ambiguity Plot (W-1,3984)
Peak W Height = 105 km
Peak W Theta = 0.05°
Ambiguity Plot (W-2,3964)
Peak W Height = 105 km
Peak W Theta = 0.15°
Ambiguity Plot (W-3,3944)
Peak W Height = 105 km
Peak W Theta = 0.3°
Ambiguity Plot (W-4,3924)
Peak W Height = 105 km
Peak W Theta = 0.4°
Ambiguity Plot (W-5,3904)
Peak W Height = 105 km
Peak W Theta = 0.55°
Ambiguity Plot (W-6,3884)
Peak W Height = 105 km
Peak W Theta = 0.65°
Ambiguity Plot (W-7,3865)
Peak W Height = 105 km
Peak W Theta = 0.8°
Ambiguity Plot (W-8,3845)
Peak W Height = 105 km
Peak W Theta = 0.95°
Ambiguity Plot (W-9,3825)
Peak W Height = 105 km
Peak W Theta = 1.05°
Ambiguity Plot (W-10,3806)
Peak W Height = 105 km
Peak W Theta = 1.2°
Ambiguity Plot (W-11,3786)
Peak W Height = 105 km
Peak W Theta = 1.35°
Altitude (km)
120
110
100
90
80
-2
0
2
Theta (degrees)
1
0.8
0.6
0.4
0.2

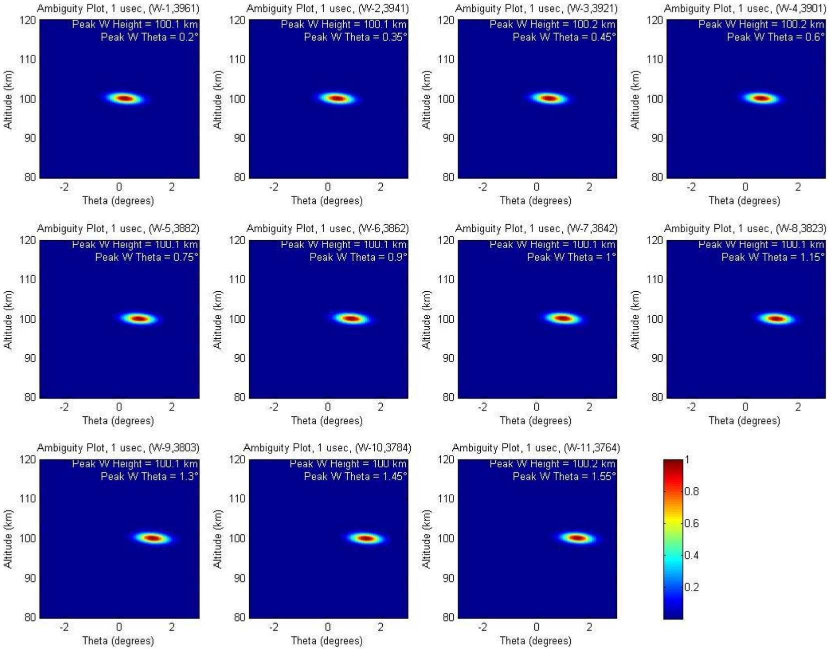

Ambiguity Plot (W-1,4007)
Peak W Height = 110 km
Peak W Theta = -0.2°
Ambiguity Plot (W-2,3987)
Peak W Height = 110 km
Peak W Theta = -0.05°
Ambiguity Plot (W-3,3967)
Peak W Height = 110 km
Peak W Theta = 0.05°
Ambiguity Plot (W-4,3947)
Peak W Height = 110 km
Peak W Theta = 0.2°
Ambiguity Plot (W-5,3927)
Peak W Height = 110 km
Peak W Theta = 0.3°
Ambiguity Plot (W-6,3907)
Peak W Height = 110 km
Peak W Theta = 0.45°
Ambiguity Plot (W-7,3887)
Peak W Height = 110 km
Peak W Theta = 0.55°
Ambiguity Plot (W-8,3867)
Peak W Height = 110 km
Peak W Theta = 0.7°
Ambiguity Plot (W-9,3848)
Peak W Height = 110 km
Peak W Theta = 0.85°
Ambiguity Plot (W-10,3828)
Peak W Height = 110 km
Peak W Theta = 1°
Ambiguity Plot (W-11,3808)
Peak W Height = 110 km
Peak W Theta = 1.1°
Altitude (km)
140
130
120
110
100
Theta (degrees)
-2
0
2
1
0.8
0.6
0.4
0.2

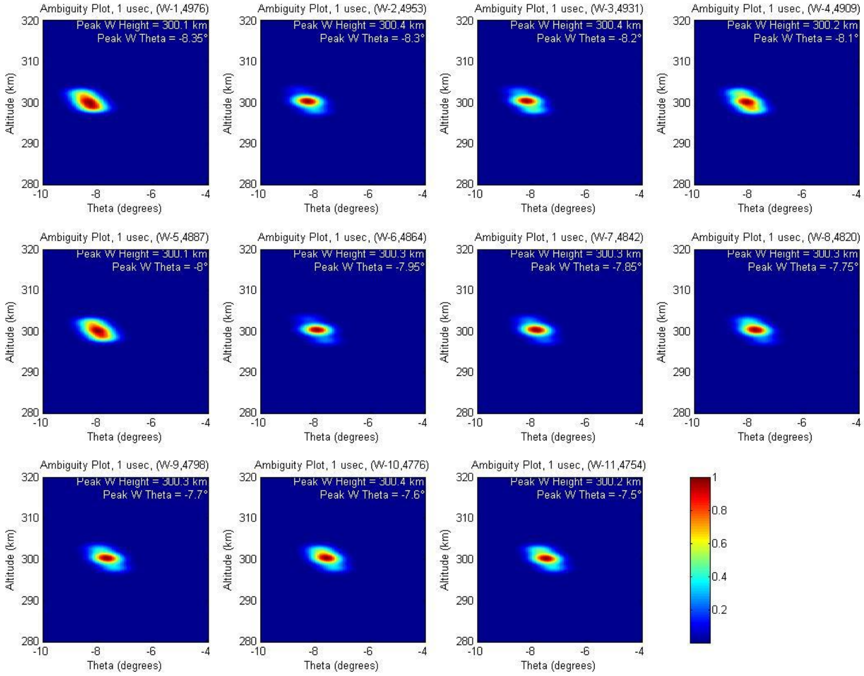

Ambiguity Plot (W-1,4030)
Peak W Height = 115 km
Peak W Theta = -0.4°
Ambiguity Plot (W-2,4010)
Peak W Height = 115 km
Peak W Theta = -0.3°
Ambiguity Plot (W-3,3990)
Peak W Height = 115 km
Peak W Theta = -0.15°
Ambiguity Plot (W-4,3970)
Peak W Height = 115 km
Peak W Theta = -0.05°
Ambiguity Plot (W-5,3950)
Peak W Height = 115 km
Peak W Theta = 0.1°
Ambiguity Plot (W-6,3930)
Peak W Height = 115 km
Peak W Theta = 0.25°
Ambiguity Plot (W-7,3910)
Peak W Height = 115 km
Peak W Theta = 0.35°
Ambiguity Plot (W-8,3890)
Peak W Height = 115 km
Peak W Theta = 0.5°
Ambiguity Plot (W-9,3870)
Peak W Height = 115 km
Peak W Theta = 0.6°
Ambiguity Plot (W-10,3850)
Peak W Height = 115 km
Peak W Theta = 0.75°
Ambiguity Plot (W-11,3831)
Peak W Height = 115 km
Peak W Theta = 0.9°
Altitude (km)
140
130
120
110
100
Theta (degrees)
-2
0
2
1
0.8
0.6
0.4
0.2

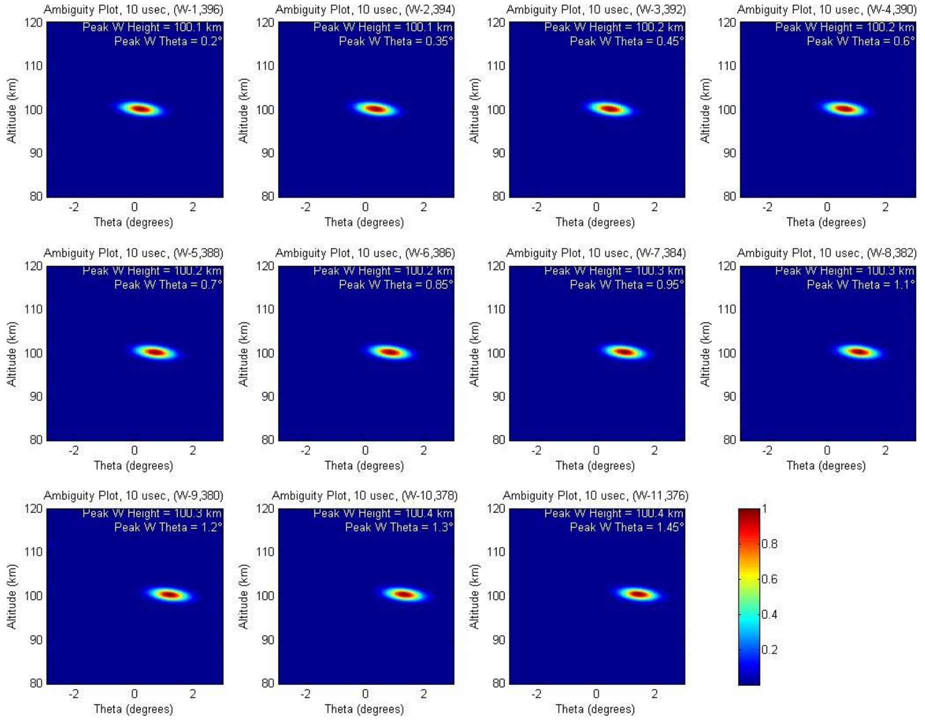


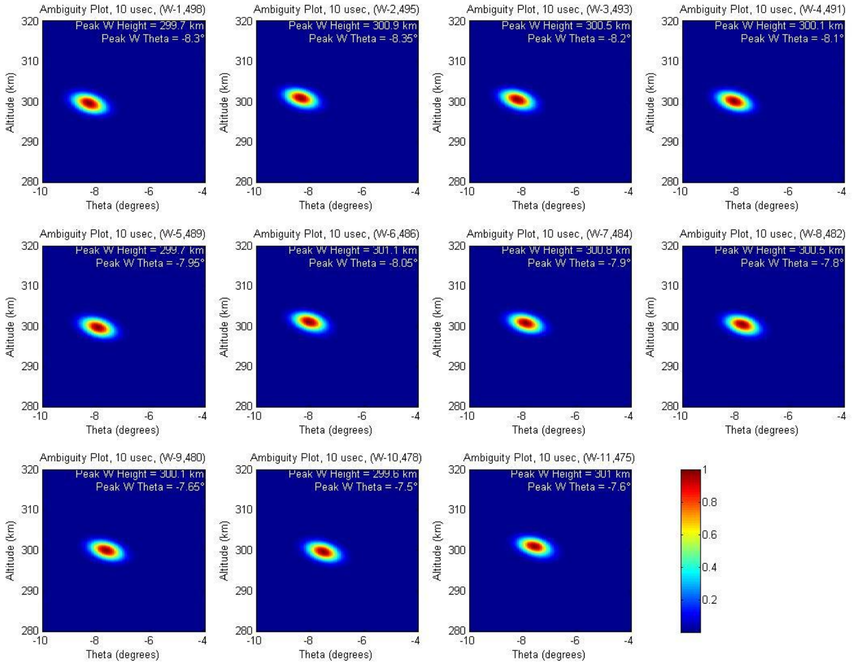
Ambiguity Plot (W-1,4053)
Peak W Height = 120 km
Peak W Theta = -0.65°
Ambiguity Plot (W-2,4033)
Peak W Height = 120 km
Peak W Theta = -0.5°
Ambiguity Plot (W-3,4013)
Peak W Height = 120 km
Peak W Theta = -0.4°
Ambiguity Plot (W-4,3993)
Peak W Height = 120 km
Peak W Theta = -0.25°
Ambiguity Plot (W-5,3973)
Peak W Height = 120 km
Peak W Theta = -0.15°
Ambiguity Plot (W-6,3952)
Peak W Height = 120 km
Peak W Theta = 0°
Ambiguity Plot (W-7,3932)
Peak W Height = 120 km
Peak W Theta = 0.1°
Ambiguity Plot (W-8,3913)
Peak W Height = 120 km
Peak W Theta = 0.25°
Ambiguity Plot (W-9,3893)
Peak W Height = 120 km
Peak W Theta = 0.4°
Ambiguity Plot (W-10,3873)
Peak W Height = 120 km
Peak W Theta = 0.5°
Ambiguity Plot (W-11,3853)
Peak W Height = 120 km
Peak W Theta = 0.65°
Altitude (km)
140
130
120
110
100
Theta (degrees)
-2
0
2
1
0.8
0.6
0.4
0.2

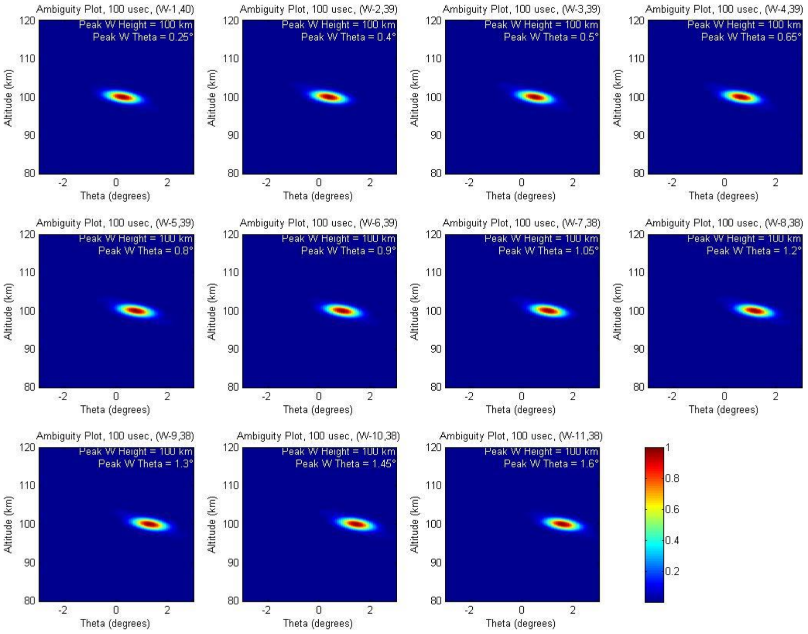
Appendix B: Change of ambiguity function for 1 µsec, 10 µsec and 100 µsec pulse widths at 100 km and 300 km

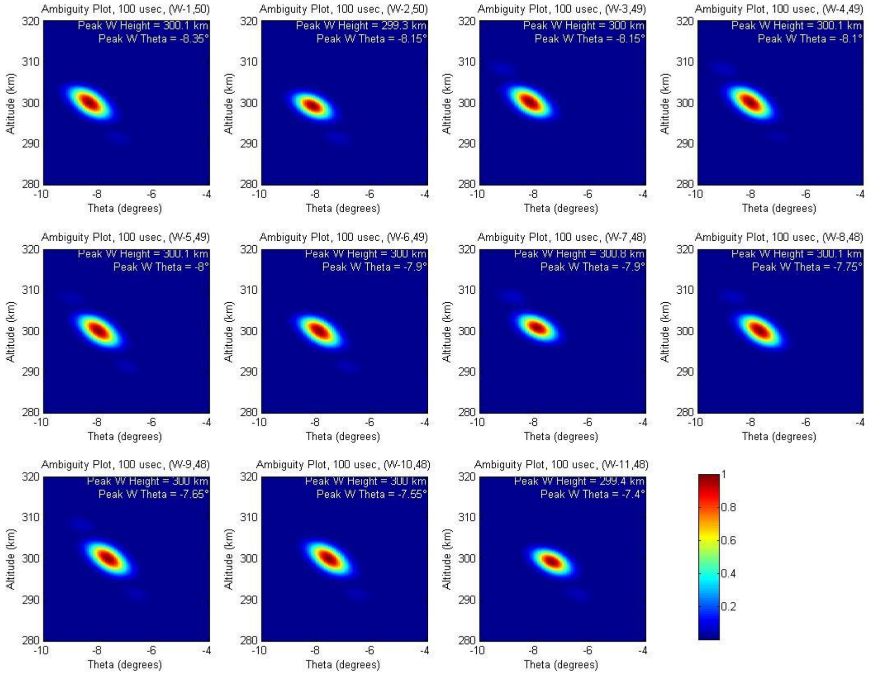





Ambiguity Plot, 10 usec, (W-1,396)
Peak W Height = 100.1 km
Peak W Theta = 0.2°
Ambiguity Plot, 10 usec, (W-2,394)
Peak W Height = 100.1 km
Peak W Theta = 0.35°
Ambiguity Plot, 10 usec, (W-3,392)
Peak W Height = 100.2 km
Peak W Theta = 0.45°
Ambiguity Plot, 10 usec, (W-4,390)
Peak W Height = 100.2 km
Peak W Theta = 0.6°
Ambiguity Plot, 10 usec, (W-5,388)
Peak W Height = 100.2 km
Peak W Theta = 0.7°
Ambiguity Plot, 10 usec, (W-6,386)
Peak W Height = 100.2 km
Peak W Theta = 0.85°
Ambiguity Plot, 10 usec, (W-7,384)
Peak W Height = 100.3 km
Peak W Theta = 0.95°
Ambiguity Plot, 10 usec, (W-8,382)
Peak W Height = 100.3 km
Peak W Theta = 1.1°
Ambiguity Plot, 10 usec, (W-9,380)
Peak W Height = 100.3 km
Peak W Theta = 1.2°
Ambiguity Plot, 10 usec, (W-10,378)
Peak W Height = 100.4 km
Peak W Theta = 1.3°
Ambiguity Plot, 10 usec, (W-11,376)
Peak W Height = 100.4 km
Peak W Theta = 1.45°
Altitude (km)
120
110
100
90
80
Theta (degrees)
-2
0
2
1
0.8
0.6
0.4
0.2


Ambiguity Plot, 10 usec, (W-1,498)
Peak W Height = 299.7 km
Peak W Theta = -8.3°
Ambiguity Plot, 10 usec, (W-2,495)
Peak W Height = 300.9 km
Peak W Theta = -8.35°
Ambiguity Plot, 10 usec, (W-3,493)
Peak W Height = 300.5 km
Peak W Theta = -8.2°
Ambiguity Plot, 10 usec, (W-4,491)
Peak W Height = 300.1 km
Peak W Theta = -8.1°
Ambiguity Plot, 10 usec, (W-5,489)
Peak W Height = 299.7 km
Peak W Theta = -7.95°
Ambiguity Plot, 10 usec, (W-6,486)
Peak W Height = 301.1 km
Peak W Theta = -8.05°
Ambiguity Plot, 10 usec, (W-7,484)
Peak W Height = 300.8 km
Peak W Theta = -7.9°
Ambiguity Plot, 10 usec, (W-8,482)
Peak W Height = 300.5 km
Peak W Theta = -7.8°
Ambiguity Plot, 10 usec, (W-9,480)
Peak W Height = 300.1 km
Peak W Theta = -7.65°
Ambiguity Plot, 10 usec, (W-10,478)
Peak W Height = 299.6 km
Peak W Theta = -7.5°
Ambiguity Plot, 10 usec, (W-11,475)
Peak W Height = 301 km
Peak W Theta = -7.6°
Altitude (km)
320
310
300
290
280
Theta (degrees)
-10
-8
-6
-4
1
0.8
0.6
0.4
0.2


Ambiguity Plot, 100 usec, (W-1,40)
Peak W Height = 100 km
Peak W Theta = 0.25°
Ambiguity Plot, 100 usec, (W-2,39)
Peak W Height = 100 km
Peak W Theta = 0.4°
Ambiguity Plot, 100 usec, (W-3,39)
Peak W Height = 100 km
Peak W Theta = 0.5°
Ambiguity Plot, 100 usec, (W-4,39)
Peak W Height = 100 km
Peak W Theta = 0.65°
Ambiguity Plot, 100 usec, (W-5,39)
Peak W Height = 100 km
Peak W Theta = 0.8°
Ambiguity Plot, 100 usec, (W-6,39)
Peak W Height = 100 km
Peak W Theta = 0.9°
Ambiguity Plot, 100 usec, (W-7,38)
Peak W Height = 100 km
Peak W Theta = 1.05°
Ambiguity Plot, 100 usec, (W-8,38)
Peak W Height = 100 km
Peak W Theta = 1.2°
Ambiguity Plot, 100 usec, (W-9,38)
Peak W Height = 100 km
Peak W Theta = 1.3°
Ambiguity Plot, 100 usec, (W-10,38)
Peak W Height = 100 km
Peak W Theta = 1.45°
Ambiguity Plot, 100 usec, (W-11,38)
Peak W Height = 100 km
Peak W Theta = 1.6°
Altitude (km)
120
110
100
90
80
-2
0
2
Theta (degrees)
1
0.8
0.6
0.4
0.2


Ambiguity Plot, 100 usec, (W-1,50)
Peak W Height = 300.1 km
Peak W Theta = -8.35°
Ambiguity Plot, 100 usec, (W-2,50)
Peak W Height = 299.3 km
Peak W Theta = -8.15°
Ambiguity Plot, 100 usec, (W-3,49)
Peak W Height = 300 km
Peak W Theta = -8.15°
Ambiguity Plot, 100 usec, (W-4,49)
Peak W Height = 300.1 km
Peak W Theta = -8.1°
Ambiguity Plot, 100 usec, (W-5,49)
Peak W Height = 300.1 km
Peak W Theta = -8°
Ambiguity Plot, 100 usec, (W-6,49)
Peak W Height = 300 km
Peak W Theta = -7.9°
Ambiguity Plot, 100 usec, (W-7,48)
Peak W Height = 300.8 km
Peak W Theta = -7.9°
Ambiguity Plot, 100 usec, (W-8,48)
Peak W Height = 300.1 km
Peak W Theta = -7.75°
Ambiguity Plot, 100 usec, (W-9,48)
Peak W Height = 300 km
Peak W Theta = -7.65°
Ambiguity Plot, 100 usec, (W-10,48)
Peak W Height = 300 km
Peak W Theta = -7.55°
Ambiguity Plot, 100 usec, (W-11,48)
Peak W Height = 299.4 km
Peak W Theta = -7.4°
Altitude (km)
320
310
300
290
280
-10
-8
-6
-4
Theta (degrees)
1
0.8
0.6
0.4
0.2

Appendix C: Partial derivatives of the error vector with respect to irregularity intensity and aspect angle width

$$S_{i,j} = \sum_{h,\theta} \omega_{i,j}\,(h,\theta) * I_k e^{-\left(\frac{h-h_{0_k}}{\Delta h}\right)^2} e^{-\left(\frac{\theta}{\Delta\theta_k}\right)^2}$$

Partial derivatives in terms of I_k parameters:

$$\frac{\partial\left(e_{i,j}\right)_m}{\partial I_k} = -\sum_{h,\theta} \omega_{i,j}\,(h,\theta) I_k e^{-\left(\frac{h-h_{0_k}}{\Delta h}\right)^2} e^{-\left(\frac{\theta}{\Delta\theta_k}\right)^2}$$

Partial derivatives in terms of $\Delta\theta_k$ parameters:

$$\frac{\partial\left(e_{i,j}\right)_m}{\partial \Delta\theta_k} = -\sum_{h,\theta} \frac{2\theta^2}{(\Delta\theta_k)^3}\,\omega_{i,j}\,(h,\theta) I_k\; e^{-\left(\frac{h-h_{0_k}}{\Delta h}\right)^2}\; e^{-\left(\frac{\theta}{\Delta\theta_k}\right)^2}$$

Appendix D: Code to produce synthetic data

raxfunc.f

```
C This is for multiple satellite locations

c     ******************************************************************
c     --getBoreVector
c     Subroutine to find the bore vector
c
c     inputs:
c           a     - azimuth
c           e     - elevation
c           d     - range
c           TM    - Time
c     outputs:
c           SX,SY,SZ - Coordinates of Vector pointing from Radar to Irregularity
c           unit vectors of  Local North,Vertical and East of Radar position

c     ------------------------------------------------------------------
      subroutine getBoreVector(a,e,d,TM,SX,SY,SZ,ULVx,ULVy,ULVz,ULNx,ULN
     cy,ULNz,ULEx,ULEy,ULEz)

c     ----Necessary Constants-------------------------------------------

c     Long and Lat of local North(N) and Vertical(V)
      real LNlon/32.53/,LNlat/24.85/,LVlon/-147.47/,LVlat/65.15/
c     Constant to convert degrees to radian and temporary unit radius
      real dtr/0.0174532925/, unitradius/1.0/
c     File Identifier
      integer u /1/
c     =================================================================

c     ----Variables used-----------------------------------------------

c     azimuth, elevation, distance and time
      real a,e,d,TM
c     X,Y,Z of local V, N and E
      real LVx,LVy,LVz,LNx,LNy,LNz,LEx,LEy,LEz
c     Unit vectors of local V,N and E
      real ULVx,ULVy,ULVz,ULNx,ULNy,ULNz,ULEx,ULEy,ULEz
c     Coordinates of vector S with respect to local V, N and E
      real SLV,SLN,SLE
c     X,Y,Z of vector S
      real SX,SY,SZ
c     ========================================================

c     ----Convert local vertical and north to cartesian-------
      call convCart(LVlat,LVlon,unitradius,LVx,LVy,LVz)
      call convCart(LNlat,LNlon,unitradius,LNx,LNy,LNz)

c     ----Find local east from cross product of local vertical and north
      LEx = (LVy*(-LNz))-(LVz*(-LNy))
      LEy = -((LVx*(-LNz))-(LVz*(-LNx)))
      LEz = (LVx*(-LNy))-(LVy*(-LNx))

c     ----Find the unit vectors of local vertical, north and east-
      call getUnitV(LVx,LVy,LVz,ULVx,ULVy,ULVz)
      call getUnitV(LNx,LNy,LNz,ULNx,ULNy,ULNz)
      call getUnitV(LEx,LEy,LEz,ULEx,ULEy,ULEz)

c     ----Find the components of vector S---------------------
      call convCart(e,a,d,SLN,SLE,SLV)
      SX = SLN*ULNx+SLV*ULVx+SLE*ULEx
      SY = SLN*ULNy+SLV*ULVy+SLE*ULEy
      SZ = SLN*ULNz+SLV*ULVz+SLE*ULEz

c     ----Print the values----------------------------------------
c      write(*,903)LVx,LVy,LVz,LNx,LNy,LNz,LEx,LEy,LEz,ULVx,ULVy,ULVz,ULN
c     Cx,ULNy,ULNz,ULEx,ULEy,ULEz

c     ----Print formats-------------------------------------------
 903  FORMAT('CALCULATED VALUES:'/
     C2X,'Cartesian coordinates of local Vertical, North and East:'/
     C4X,'LVx=',F9.4,3X,'LVy=',F9.4,3X,'LVz=',F9.4/
     C4X,'LNx=',F9.4,3X,'LNy=',F9.4,3X,'LNz=',F9.4/
     C4X,'LEx=',F9.4,3X,'LEy=',F9.4,3X,'LEz=',F9.4//
     C2X,'Unit vectors of local Vertical, North and East:'/
     C4X,'ULVx=',F9.4,3X,'ULVy=',F9.4,3X,'ULVz=',F9.4/
     C4X,'ULNx=',F9.4,3X,'ULNy=',F9.4,3X,'ULNz=',F9.4/
     C4X,'ULEx=',F9.4,3X,'ULEy=',F9.4,3X,'ULEz=',F9.4/)

      return

      end



c     ******************************************************************
c     --getAsp------
c     Subroutine to find the aspect angle
c
c     inputs:
c           IX,IY,IZ  - Cartesian Position vector of Irregularity
c           Ilat,Ilon - Elevation and Azimuth of Irregularity)
c           TX,TY,TZ  - Cartesian Position Vector of Satellite
c           SX,SY,SZ  - Cartesian Vector of S(pointing from radar to irregularity)
c           BT,BR,BP  - Spherical Coordinates of Magnetic field
c     Output:
c           aspect angle=theta(degrees)
c     ------------------------------------------------------------------

      subroutine getAsp(IX,IY,IZ,Ilat,Ilon,TX,TY,TZ,SX,SY,SZ,BT,BP,BR,as
     cpect)

c     ----Variables used-----------------------------------------------

c     X,Y,Z,latitude and longitude of irregularity location
      real IX,IY,IZ,Ilat,Ilon

c     X,Y,Z of satellite
      real TX,TY,TZ

c     spherical and cartesian components of magnetic field and aspect angle
      real BT,BP,BR,BX,BY,BZ,UBx,UBy,UBz,aspect
      real cbfieldrz, cbfieldrx, cbfieldry, cbfieldtx, cbfieldty, cbfiel
     cdtz, cbfieldpz, cbfieldpx, cbfieldpy

c     X,Y,Z of vector pointing from radar to irregularity
      real SX,SY,SZ

c     Cartesian Coordinates of Vector KR(the reflected beam from I to S)
      real KRx,KRy,KRz,UKRx,UKRy,UKRz

c     Cartesian Coordinates of Vector KO(pointing from I to R)
      real KOx,KOy,KOz,UKOx,UKOy,UKOz

c     Unit Vector of KB
      real UKBx,UKBy,UKBz

c     dot product of UKB and UB, magnitudes of UKB and UB
      real dotP,magUKB, magUB

c     Necessary constants
      real dtr/0.0174532925/
      real invert/-1.0/
c     ========================================================

c     --Compute Vector KR-----------------------------------------------
      KRx = TX - IX
      KRy = TY - IY
      KRz = TZ - IZ

c     --Compute Vector KO-----------------------------------------------
      KOx=SX*invert
      KOy=SY*invert
      KOz=SZ*invert

c     --Compute unit vectors of KR and KO-------------------------------
      call getUnitV(KRx,KRy,KRz,UKRx,UKRy,UKRz)
      call getUnitV(KOx,KOy,KOz,UKOx,UKOy,UKOz)

c     --Find the unit vector of KB--------------------------------------
      UKBx = UKRx+UKOx
      UKBy = UKRy+UKOy
      UKBz = UKRz+UKOz

c     --Process Magnetic field------------------------------------------
      cbfieldrz=Br*sin(Ilat*dtr);
      cbfieldrx=Br*cos(Ilat*dtr)*cos(Ilon*dtr);
      cbfieldry=Br*cos(Ilat*dtr)*sin(Ilon*dtr);

      cbfieldtz=-Bt*cos(Ilat*dtr);
      cbfieldtx=-Bt*sin(Ilat*dtr)*cos((180.+Ilon)*dtr);
      cbfieldty=-Bt*sin(Ilat*dtr)*sin((180.+Ilon)*dtr);

      cbfieldpz= 0.0;
      cbfieldpx=+(-1.*Bp)*sin(Ilon*dtr)
      cbfieldpy=-(-1.*Bp)*cos(Ilon*dtr)

      BX = cbfieldrx +cbfieldtx+ cbfieldpx;
      BY = cbfieldry +cbfieldty+ cbfieldpy;
      BZ = cbfieldrz +cbfieldtz+ cbfieldpz;

      call getUnitV(BX,BY,BZ,UBx,UBy,UBz)

c     --Compute Aspect Angle--------------------------------------------
      dotP = (UKBx*UBx)+(UKBy*UBy)+(UKBz*UBz)
      magUKB = sqrt((UKBx*UKBx)+(UKBy*UKBy)+(UKBz*UKBz))
      magUB = sqrt((UBx*UBx)+(UBy*UBy)+(UBz*UBz))
      aspect = acos(dotP/(magUKB*magUB))/dtr

      return

      end



c     ******************************************************************
c     --getPathPosn--------
c     Subroutine to find the path length and new position of irregularity
c
c     inputs:
c           SX,SY,SZ - Cartesian Vector of S(pointing from radar to irregularity)
c           Theta,Phi,Rad - To locate the new irregularity location
c           Rx,Ry,Rz - Cartesina Coordinate of Radar
c           Tx,Ty,Tz - Cartesina Coordinate of Satellite
c     Output:
c           newIx,newIy,newIz - new location of irregularity
c           newSx,newSy,newSz - new S vector
c           path - path length
c     ------------------------------------------------------------------

      subroutine getPathPosn(SX,SY,SZ,Theta,Phi,Rad,Rx,Ry,Rz,Tx,Ty,Tz,ne
     cwIx,newIy,newIz,newSx,newSy,newSz,path)
```

```
c     ----Variables used-----------------------------------------------
c     Function Input Parameters
      real Theta,Phi,Rad,Rx,Ry,Rz,Tx,Ty,Tz,SX,SY,SZ

c     Function Output Parameters
      real newSx,newSy,newSz,newIx,newIy,newIz,ath

c     ----Other used variables
      real Stheta,Sphi,Srad

      real nSX,nSY,nSZ,nStheta,nSphi,nSrad

      real nIX,nIY,nIZ

      real LXx,LXy,LXz,uLXx,uLXy,uLXz

      real LYx,LYy,LYz,uLYx,uLYy,uLYz

      real LZx,LZy,LZz,uLZx,uLZy,uLZz,LZtheta,LZphi,LZrad

      real rLXx,rLXy,rLXz,rLX,rLY,rLZ,magLX

      real PNx,PNy,PNz

      real ITx, ITy, ITz

c     Necessary Constants
      real dtr/0.0174532925/
c     =========================================================

c     ----Re Compute the bore vector S pointing from radar to irreg.---
      call convSphe(SX,SY,SZ,Stheta,Sphi,Srad)
      Srad = Rad
      call convCart(Stheta,Sphi,Srad,SX,SY,SZ)

c     ----Find the new S-----------------------------------------------
      nStheta = Stheta + Theta
      nSphi   = Sphi
      nSrad   = Srad
      call convCart(nStheta,nSphi,nSrad,nSX,nSY,nSZ)

c     ----Find vector nI; pointing from I to I'------------------------
      nIX = nSX - SX
      nIY = nSY - SY
      nIZ = nSZ - SZ

c     ----Find the new local Z vector----------------------------------
      LZrad   = Srad - nSrad*cos(Theta*dtr)
      LZtheta = Stheta
      LZphi   = Sphi
      call convCart(LZtheta,LZphi,LZrad,LZx,LZy,LZz)
      call getUnitV(LZx,LZy,LZz,uLZx,uLZy,uLZz)

c     ----Find the new local X vector----------------------------------
      LXx = LZx + nIX
      LXy = LZy + nIY
      LXz = LZz + nIZ
      call getUnitV(LXx,LXy,LXz,uLXx,uLXy,uLXz)

c     ----Find the new local Y vector----------------------------------
      LYx = (LZy*LXz)-(LZz*LXy)
      LYy = -((LZx*LXz)-(LZz*LXx))
      LYz = (LZx*LXy)-(LZy*LXx)
      call getUnitV(LYx,LYy,LYz,uLYx,uLYy,uLYz)

c     ----Find the vector, rLX, representing local X, LX, rotated about Phi
      magLX = sqrt(LXx*LXx+LXy*LXy+LXz*LXz)

      rLX = magLX*Cos(Phi*dtr)
      rLY = magLX*Sin(Phi*dtr)
      rLZ = 0
      rLXx = rLX * uLXx + rLY *uLYx + rLZ*uLZx
      rLXy = rLX * uLXy + rLY *uLYy + rLZ*uLZy
      rLXz = rLX * uLXz + rLY *uLYz + rLZ*uLZz
c      rLXx = magLX*cos(Phi*dtr)*uLXx + magLX*sin(Phi*dtr)*uLYx
c      rLXy = magLX*cos(Phi*dtr)*uLXy + magLX*sin(Phi*dtr)*uLYy
c      rLXz = magLX*cos(Phi*dtr)*uLXz + magLX*sin(Phi*dtr)*uLYz

c     ----Find the vector PN, pointing from the bore sight to the new Irregularity locatoin
      PNx = rLXx - LZx
      PNy = rLXy - LZy
      PNz = rLXz - LZz

c     ----Find the vector newS, pointing from radar to new Irregularity location
      newSx = Sx + PNx
      newSy = Sy + PNy
      newSz = Sz + PNz

c     ----Find the position vector of the new Irregularity----------------------
      newIx = Rx + newSx
      newIy = Ry + newSy
      newIz = Rz + newSz

c     ----Find the vector,IT, pointing from new Irregularity location to satellite
      ITx = Tx - newIx
      ITy = Ty - newIy
      ITz = Tz - newIz

c     ----Find the total path----------------------------------------------------
      path = sqrt(newSx*newSx+newSy*newSy+newSz*newSz)+sqrt(ITx*ITx+ITy*
     cITy+ITz*ITz)

      return

      end


c     ******************************************************************
c     --getUnitV--------Subroutine that finds the unit vectors

      subroutine getUnitV(x,y,z,Ux,Uy,Uz)
      real x,y,z,Ux,Uy,Uz
      real mag
      mag = sqrt((x*x)+(y*y)+(z*z))
      Ux = x/mag
      Uy = y/mag
      Uz = z/mag
      return
      end


c     ******************************************************************
c     --convCart--------Subroutine that converts spherical to cartesian units
      subroutine convCart(theta,phi,r,x,y,z)
      real theta,phi,r,x,y,z
      real dtr/0.0174532925/
      x = r * cos(theta*dtr) * cos(phi*dtr)
      y = r * cos(theta*dtr) * sin(phi*dtr)
      z = r * sin(theta*dtr)
      return
      end

c     ******************************************************************
c     --convSphe--------Subroutine that converts cartesian to spherical units
      subroutine convSphe(x,y,z,theta,phi,r)
      real x,y,z,theta,phi,r
      real dtr/0.0174532925/
      theta = (atan2(z,(sqrt((x*x)+(y*y)))))/dtr
      phi = (atan2(y,x))/dtr
      r = sqrt((x*x)+(y*y)+(z*z))
      return
      end
```

ambiguity.f

```
C Provides data to plot ambiguity function at different locations.

c     RAX Data Inversion Project
c     ******************************************************************
c     ----ambiguity-----
c     This program takes in the elevation, azimuth, range and satellite
c     geodetic coordinates to provide the data for the point spread function (W).
c     Provides all the posible data for a particular altitude

c     The program outpute one file for each satellite location which contains 4 columns
c         -- dV * RadarPowerFactor
c         -- altitude
c         -- aspect angle
c         -- j index
c         -- bore altitude
c         -- i index

c     ------------------------------------------------------------------

      program ambiguity

c     ----Necessary Constants------------------------------------------

c     Degree to radian to conversion
      real dtr/0.0174532925/
c     Speed of light in km/sec
      real c/299792.458/
c     Long, Lat and alt of Radar (Geocentric)
      real Rlon/-147.47/,Rlat/64.9829/,Ralt/6360.6782/

      integer Wcount/0/, u/1/,index/0/,v/2/
c     ==================================================================

c     ----Variables used--------------------------------------

c     vector S
      real SX,SY,SZ, Stheta,Sphi,Srad, newSx,newSy,newSz

c     Vector R(Posn Vector Radar)
      real RX,RY,RZ
c     Unit vectors of Local Vertical, North and East at the radar position
      real ULVx,ULVy,ULVz,ULNx,ULNy,ULNz,ULEx,ULEy,ULEz

c     vector I(Posn Vector Irregularity)
      real Ilat,Ilon,Irad, Ix,Iy,Iz

c     vector T(position vector of satellite)
      real TX,TY,TZ, Tlat,Trad
c     Tlon,Tgdlat,Tgdalt

c     Magnetic field components
      real BR,BT,BP,B

c     Angle of Dip, aspect angle theta, altitude in km, volume, total path in km and timedelay in
micro second
      real DIP,aspect,alt,dVol,totalpath,timeDelay

c     Power Factor, Reference index
      real Pr, ReferenceIndex/0/

c     keep track of the location count
      integer count

      real tempRad/0/,tempTheta/0/,tempPhi/0/

c     ----The source file for power factor
      character*20 PowerFile
      data PowerFile/"radarPower.txt"/
```

```
c     ===================================================================

c     Necessary Parameters to be filled>>>>>>>>>>>>>>>>>

c     azimuth, elevation, distance and time
      real a/0.0/,e/50.0/,d/300.0/,TM/2000/

c     step for the changing the radius, elevation and azimuth
      real deltaR/0.1/,deltaT/0.3/,deltaP/10.0/

c     variables for the radius, elevation and azimuth loop
      data Tcount/0/,Pcount/0/,Rcount/0/
      data ThetaStart/0/,ThetaEnd/30/,ThetaStep/1/
      data PhiStart/0/,PhiEnd/360/,PhiStep/10/

c     TFactor is 0.1 as our count starts from 0 to 30 with step 1.
c     however, actualy we are supposed to go from 0 to 3 with step 0.1
c     Thus we multiply our data with Rfactor
c     Same reasoning for TFactor and PFactor
      data TFactor/0.1/

c     ----Geodetic location os satellites
      real Tglat(1)/63.5/
      real Tlon(1)/-148/
      real Tgalt(1)/665.95000/

      integer LocationStart/1/,LocationEnd/1/,LocationStep/1/
c     -------

c     Tell at which altitude to center the data
      real boreRange(1)/300.0/
      real boreStart/1/,boreEnd/1/,boreStep/1/

c     ----This is the divider that determines whether data is in 1 micro, 10 micro, 100 micro and
so on
c     ----1 for 1 micro second, 10 for 10 microsecond, 100 for 100micro....
c      real timerange/100/
      real timerange/1/

c     ----The file to store restults
      character*15 BoreOut
      data BoreOut/"Scatter_90.txt"/
c     ==================================================================



      deltaT = deltaT*dtr
      deltaP = deltaP*dtr

      d = d/sin(e*dtr)

      write(*,*)'Processing...'

c     ----Find the X, Y, Z components of the radar----------------------
      call convCart(Rlat,Rlon,Ralt,RX,RY,RZ)

c     ----Get Vector of Bore Sight -------------------------------------
      call getBoreVector(a,e,d,TM,SX,SY,SZ,ULVx,ULVy,ULVz,ULNx,ULNy,ULNz
     c,ULEx,ULEy,ULEz)

c     ----Open bore file for storing the volumes------------------------
      open(u, file = BoreOut, status = "replace")


c     ====Deduce volume, altitude and aspect angle======================

c     ----First loop for loation----
      do 5 count = LocationStart,LocationEnd,LocationStep
          call convrt(1,Tglat(count),Tgalt(count),Tlat,Trad)
          call convCart(Tlat,Tlon(count),Trad,TX,TY,TZ)
          write(*,*) count
```

```
c     ----Second loop for Bore Radius----
      do 10 Rcount = boreStart,boreEnd,boreStep
        open(v, file = PowerFile, status = "old")
        tempRad = (boreRange(Rcount))/sin(e*dtr)

c       ----Third loop for Theta------
       do 20 Tcount = ThetaStart,ThetaEnd,ThetaStep
          tempTheta = Tcount*Tfactor
c         ----Fourth loop for Phi-------
         do 30 Pcount = PhiStart,PhiEnd,PhiStep

c           ----Determine the volume------
            read(v,*)Pr
            dVol=Pr*tempRad*tempRad*deltaR*sin(tempTheta*dtr)*deltaP*del
     ctaT
c           ----If volume is not zero-----
            if(dVol.NE.0.0)then
c           ----Get the path vector-------
            call getPathPosn(SX,SY,SZ,tempTheta,Pcount,tempRad,Rx,Ry,Rz,
     cTx,Ty,Tz,Ix,Iy,Iz,newSx,newSy,newSz,totalpath)
c           ----Find the magnetic field components at irreg. location---
            call convSphe(Ix,Iy,Iz,Ilat,Ilon,Irad)
            call geobfield(TM,Irad,(90- Ilat)*dtr,Ilon*dtr,BR,BT,BP,B)
            DIP = atan2(Br,sqrt(Bt*Bt+Bp*Bp))/dtr
c           ----Find the Aspect Angle-----------------------------------
            call getAsp(IX,IY,IZ,Ilat,Ilon,TX,TY,TZ,newSX,newSY,newSZ,BT
     C,BP,BR,aspect)

c           ----Find the altitude---------------------------------------
            alt = ((ULEZ*ULNX-ULEX*ULNZ)*(ULEY*newSX-ULEX*newSY)-(ULEY*U
     CLNX-ULEX*ULNY)*(ULEZ*newSX-ULEX*newSZ))/((ULEY*ULVX-ULEX*ULVY)*(UL
     CEZ*ULNX-ULEX*ULNZ)-(ULEY*ULNX-ULEX*ULNY)*(ULEZ*ULVX-ULEX*ULVZ))

c           ----Calculate the time delay-----
            timeDelay = (totalpath/c)*1000000

c           ----Calculate the index----------
            index = int((timeDelay/timerange)+0.5)

c           ----Store the information--------
            if((Tcount.eq.1).AND.(Pcount.eq.0))then
c              this tells to pick up data with index same as that of the bore
               write(*,*)dVol,alt,aspect,index,boreRange(Rcount)
               ReferenceIndex = index
            endif

            if((ReferenceIndex.ne.0).AND.(index.eq.ReferenceIndex))then
               write(u,100)dVol,alt,aspect,index,boreRange(Rcount),count
            endif

c           ----End of if volume is not zero----
            endif

c     ----End of the fourth loop----
 30   continue

c     ----End of the third loop-----
 20   continue

      close(v)

c     ----End of the second loop----
 10   continue

c     ----End of the first loop-----
 5    continue
      close(u)

 100  FORMAT(F11.10,',',F10.4,',',F10.4,',',I6,',',F10.4,',',I3)
      end
```

syntheticData.f

```
c     RAX Data Inversion Project
c     *****************************************************************
c     ----syntheticData-----
c     This program takes in the elevation, azimuth, range and satellite
c     geodetic coordinates to provide the data for the point spread function (W).
c     Provides all the posible data for a given radius range, elevation range and azimuth range

c     The program outpute one file for each satellite location which contains 4 columns
c         -- dV * RadarPowerFactor
c         -- altitude
c         -- aspect angle
c         -- j index

c     -----------------------------------------------------------------

      program syntheticData

c     ----Necessary Constants-------------------------------------------

c     Degree to radian to conversion
      real dtr/0.0174532925/
c     Speed of light in km/sec
      real c/299792.458/
c     Long, Lat and alt of Radar (Geocentric)
      real Rlon/-147.47/,Rlat/64.9829/,Ralt/6360.6782/

      integer Wcount/0/, u/1/,index/0/,v/2/
c     =================================================================

c     ----Variables used---------------------------------------

c     vector S
      real SX,SY,SZ, Stheta,Sphi,Srad, newSx,newSy,newSz

c     Vector R(Posn Vector Radar)
      real RX,RY,RZ
c     Unit vectors of Local Vertical, North and East at the radar position
      real ULVx,ULVy,ULVz,ULNx,ULNy,ULNz,ULEx,ULEy,ULEz

c     vector I(Posn Vector Irregularity)
      real Ilat,Ilon,Irad, Ix,Iy,Iz

c     vector T(position vector of satellite)
      real TX,TY,TZ, Tlat,Trad
c     Tlon,Tgdlat,Tgdalt

c     Magnetic field components
      real BR,BT,BP,B

c     Angle of Dip, aspect angle theta, altitude in km, volume, total path in km and timedelay in 
micro second
      real DIP,aspect,alt,dVol,totalpath,timeDelay

      real Pr

c     keep track of the location count
      integer count

      real tempRad/0/,tempTheta/0/,tempPhi/0/

c     ----The source file for power factor
      character*20 PowerFile
      data PowerFile/"radarPower.txt"/
c     ==================================================================

c     Necessary Parameters to be filled>>>>>>>>>>>>>>>>>

c     azimuth, elevation, distance and time
```

```
      real a/0.0/,e/50.0/,d/300.0/,TM/2000/

c     step for the changing the radius, elevation and azimuth
      real deltaR/0.1/,deltaT/0.3/,deltaP/10.0/

c     variables for the radius, elevation and azimuth loop
      data Tcount/0/,Pcount/0/,Rcount/0/
      data ThetaStart/0/,ThetaEnd/30/,ThetaStep/1/
      data PhiStart/0/,PhiEnd/360/,PhiStep/10/
      data RadStart/1000/,RadEnd/5000/,RadStep/1/

c     RFactor is 0.1 as our count starts from 1000 to 5000 with step 1.
c     however, actualy we are supposed to go from 100 to 500 with step 0.1
c     Thus we multiply our data with Rfactor
c     Same reasoning for TFactor and PFactor
      data RFactor/0.1/,TFactor/0.1/,PFactor/1/

c     Geodetic location os satellites
      real Tglat(11)/59.824833,59.872217,59.919600,59.966983,60.014367,6
     c0.061750,60.109133,60.156517,60.203900,60.251283,60.298667/

      real Tlon(11)/-157.515667,-157.440733,-157.365800,-157.290867,-157
     c.215933,-157.141000,-157.066067,-156.991133,-156.916200,-156.84126
     c7,-156.766333/

      real Tgalt(11)/665.950000,665.965000,665.980000,665.995000,666.010
     c000,666.025000,666.040000,666.055000,666.070000,666.085000,666.10/

      integer LocationStart/1/,LocationEnd/11/,LocationStep/1/

c     This is the divider that determines whether data is in 1 micro, 10 micro, 100 micro and so
on
c     1 for 1 micro second, 10 for 10 microsecond, 100 for 100micro....
      real timerange/1/

c     ----The file to store restults for each satellite location
      character*9 file(11)
      data file/'W_01.txt','W_02.txt','W_03.txt','W_04.txt','W_05.txt','
     cW_06.txt','W_07.txt','W_08.txt','W_09.txt','W_10.txt','W_11.txt'/
c     ==================================================================

      deltaT = deltaT*dtr
      deltaP = deltaP*dtr

      write(*,*)'Processing...'

c     ----Find the X, Y, Z components of the radar----------------------
      call convCart(Rlat,Rlon,Ralt,RX,RY,RZ)

c     ----Get Vector of Bore Sight -------------------------------------
      call getBoreVector(a,e,d,TM,SX,SY,SZ,ULVx,ULVy,ULVz,ULNx,ULNy,ULNz
     c,ULEx,ULEy,ULEz)


c     ====Deduce volume, altitude and aspect angle======================

c     ----First loop for loation----
      do 5 count = LocationStart,LocationEnd,LocationStep
          call convrt(1,Tglat(count),Tgalt(count),Tlat,Trad)
          call convCart(Tlat,Tlon(count),Trad,TX,TY,TZ)
          write(*,*) count

c     ----Open file for storing the data------------------------
          open(u, file = file(count), status = "replace")

c     ----Second loop for Radius----
          do 10 Rcount = RadStart,RadEnd,RadStep
             tempRad = Rcount * RFactor

             open(v, file = PowerFile, status = "old")
```

```
c     ----Third loop for Theta------
             do 20 Tcount = ThetaStart,ThetaEnd,ThetaStep
                tempTheta = Tcount * TFactor

c     ----Fourth loop for Phi-------
                do 30 Pcount = PhiStart,PhiEnd,PhiStep

c     ----Determine the volume------
                   read(v,*)Pr
                   dVol=Pr*tempRad*tempRad*deltaR*sin(tempTheta*dtr)*del
     c                  taP*deltaT

c     ----If volume is not zero-----
                   if(dVol.NE.0.0)then

c     ----Get the path vector-------
            call getPathPosn(SX,SY,SZ,tempTheta,Pcount,tempRad,Rx,Ry,Rz,
     cTx,Ty,Tz,Ix,Iy,Iz,newSx,newSy,newSz,totalpath)

c           ----Find the magnetic field components at irreg. location---
            call convSphe(Ix,Iy,Iz,Ilat,Ilon,Irad)
            call geobfield(TM,Irad,(90- Ilat)*dtr,Ilon*dtr,BR,BT,BP,B)
            DIP = atan2(Br,sqrt(Bt*Bt+Bp*Bp))/dtr

c           ----Find the Aspect Angle-----------------------------------
            call getAsp(IX,IY,IZ,Ilat,Ilon,TX,TY,TZ,newSX,newSY,newSZ,BT
     C,BP,BR,aspect)


c           ----Find the altitude---------------------------------------
            alt = ((ULEZ*ULNX-ULEX*ULNZ)*(ULEY*newSX-ULEX*newSY)-(ULEY*U
     CLNX-ULEX*ULNY)*(ULEZ*newSX-ULEX*newSZ))/((ULEY*ULVX-ULEX*ULVY)*(UL
     CEZ*ULNX-ULEX*ULNZ)-(ULEY*ULNX-ULEX*ULNY)*(ULEZ*ULVX-ULEX*ULVZ))


c           ----Calculate the time delay-----
            timeDelay = (totalpath/c)*1000000

c           ----Calculate the index----------
            index = int((timeDelay/timerange)+0.5)

c           ----Store the information--------
            write(u,100)dVol,alt,aspect,index


c           ----End of if volume is not zero----
            endif

c     ----End of the fourth loop----
 30   continue

c     ----End of the third loop-----
 20   continue

      close(v)

c     ----End of the second loop----
 10   continue

      close(u)

c     ----End of the first loop-----
 5    continue

 100  FORMAT(F11.10,',',F10.4,',',F10.4,',',I6)

      end
```

scatterAngles.f

```
c     RAX Data Inversion Project
c     ******************************************************************

c     Program: GetScatteringAngle
c     ===========================
c     This program provides the scattering angle for a particular elevation,
c     azimuth and radius of bore sight at different satellite locations
c     ------------------------------------------------------------------

      program GetScatteringAngle

c     ----Necessary Constants-------------------------------------------

c     Degree to radian to conversion,Speed of light in km/sec
      real dtr/0.0174532925/,c/299792.458/

c     Long, Lat and alt of Radar (Geocentric)
      real Rlon/-147.47/,Rlat/64.9829/,Ralt/6360.6782/

c     File identifiers, inverter
      integer u/1/,v/2/
      real invert/-1/
c     ==================================================================

c     ----Variables used--------------------------------------

c     vector S(pointing from radar to irregularity)
      real SX,SY,SZ

c     Vector R(Posn Vector Radar)
      real RX,RY,RZ

c     Unit vectors of Local Vertical, North and East at the radar position
      real ULVx,ULVy,ULVz,ULNx,ULNy,ULNz,ULEx,ULEy,ULEz

c     vector I(Posn Vector Irregularity)
      real Ix,Iy,Iz

c     vector T(position vector of satellite)
      real TX,TY,TZ, Tlat,Trad

c     Cartesian Coordinates of Vector KR(the reflected beam from I to S)
      real KRx,KRy,KRz,UKRx,UKRy,UKRz

c     Cartesian Coordinates of Vector KO(pointing from I to R)
      real KOx,KOy,KOz,UKOx,UKOy,UKOz

c     dot product of UKR and UKO, magnitudes of UKR and UKO, Scattering Angle
      real dotP,magUKR,magUKO,ScattteringAngle,tempRad

c     simple counter
      integer Lcount,Rcount/0/
c     ===================================================================

c     Necessary Parameters to be filled>>>>>>>>>>>>>>>>>

c     azimuth, distance and time
      real a/0.0/,d/150.0/,TM/2000/

c     elevation
      real elevation(3)/15,50,75/
      integer ElevationStart/1/,ElevationEnd/3/,ElevationStep/1/

c     Geodetic location of satellites
c      real Tglat(1)/64.9829/
c      real Tlon(1)/-152.28/
c      real Tgalt(1)/665.95/
      real Tglat(1)/67.5/
      real Tlon(1)/-142/
      real Tgalt(1)/665.95/
      integer LocationStart/1/,LocationEnd/1/,LocationStep/1/
c     Tell at which altitude to center the data
      real boreAlt(1)/300.0/
      real boreStart/1/,boreEnd/1/,boreStep/1/

c     ----The file to store restults
      character*15 BoreOut
      data BoreOut/"Bore_0_deg.txt"/
c     ==================================================================
c     ACTUAL COMPUTATION BEGINS HERE>>
C     ==============================
      write(*,*)'Processing...'

c     ----Find the X, Y, Z components of the radar----------------------
      call convCart(Rlat,Rlon,Ralt,RX,RY,RZ)

c     ----Open bore file for storing the volumes------------------------
      open(u, file = BoreOut, status = "replace")

c     ----First loop for loation----------------------------------------
      do 5 Lcount = LocationStart,LocationEnd,LocationStep
         call convrt(1,Tglat(Lcount),Tgalt(Lcount),Tlat,Trad)
         call convCart(Tlat,Tlon(Lcount),Trad,TX,TY,TZ)
         write(*,*) Lcount

c     ----Second loop for Radius----------------------------------------
         do 10 Rcount = boreStart,boreEnd,boreStep

c     ----Third loop for elevation--------------------------------------
            do 15 Ecount = ElevationStart,ElevationEnd,ElevationStep

c     ----Get Vector of Bore Sight -------------------------------------
               tempRad = (boreAlt(Rcount))/sin(Elevation(Ecount)*dtr)
               call getBoreVector(a,Elevation(Ecount),tempRad,TM,SX,SY,S
     c              Z,ULVx,ULVy,ULVz,ULNx,ULNy,ULNz,ULEx,ULEy,ULEz)

c     ----Get the irregularity position vector--------------------------
               IX = RX + SX
               IY = RY + SY
               IZ = RZ + SZ

c     ----Get the vector of the reflected beam--------------------------
               KRx = TX - IX
               KRy = TY - IY
               KRz = TZ - IZ

c     ----Compute Vector KO, reverse of the incident beam---------------
               KOx=SX*invert
               KOy=SY*invert
               KOz=SZ*invert

c     ----Compute the scattering angle----------------------------------
               call getUnitV(KRx,KRy,KRz,UKRx,UKRy,UKRz)
               call getUnitV(KOx,KOy,KOz,UKOx,UKOy,UKOz)
               dotP = (UKRx*UKOx)+(UKRy*UKOy)+(UKRz*UKOz)
               magUKR = sqrt((UKRx*UKRx)+(UKRy*UKRy)+(UKRz*UKRz))
               magUKO = sqrt((UKOx*UKOx)+(UKOy*UKOy)+(UKOz*UKOz))
               ScatteringAngle = acos(dotP/(magUKR*magUKO))/dtr
               write(*,*)ScatteringAngle

c     ----End of the third loop------------------------------------------
 15         continue
c     ----End of the second loop-----------------------------------------
 10      continue
c     ----End of the first loop------------------------------------------
 5    continue
      close(u)
 100  FORMAT(F11.10,',',F10.4,',',F10.4,',',I4,',',F10.4,',',I3)
      end
```

loci.f

```
c     RAX Data Inversion Project
c     ******************************************************************

c     Program: GetLociOfPerpendicularity
c     =================================
c     This program provides the the aspect angle for different satellite
c     positions in order to plot the locai of perpendicularity
c     It produces three files
c       1. File containing aspect angles
c       2. File containing geodetic latitude
c       3. File contianing geodetic longitude
c
c     The file of aspect angle needs to be processed by another C program 'aspectCSV.c' to
c     Generated a nice csv file
c     ------------------------------------------------------------------

      program getLocaiOfPerpendicularity

c     ----Necessary Constants-------------------------------------------

c     Long, Lat and alt of Radar (Geocentric)
      real Rlon/-147.47/,Rlat/64.9829/,Ralt/6360.6782/

c     Degree to radian to conversion
      real dtr/0.0174532925/

c     File identifiers
      integer u/1/,v/2/,w/3/
c     ==================================================================

c     ----Variables used--------------------------------------

c     Vector R(Posn Vector Radar)
      real RX,RY,RZ

c     Unit vectors of Local Vertical, North and East at the radar position
      real ULVx,ULVy,ULVz,ULNx,ULNy,ULNz,ULEx,ULEy,ULEz

c     vector T(position vector of satellite)
      real TX,TY,TZ,Tlat,Trad,Tlon,Tgdlat,Tgdalt

c     vector S(pointing from radar to irregularity)
      real SX,SY,SZ, Stheta,Sphi,Srad, newSx,newSy,newSz

c     vector I(Posn Vector Irregularity)
      real Ilat,Ilon,Irad, Ix,Iy,Iz

c     Magnetic field components
      real BR,BT,BP,B

c     counter for latitude loop and longitude loop, aspect angle
      real latcount,loncount,aspect

c     radar azimuth, elevation, radial distance, IGRF time
      real a,e,d,TM

c     the file to store aspect angles
      character*25 aspectFile

c     the file to store geodetic latitudes
      character*25 latFile

c     the file to store geodetic longitudes
      character*25 lonFile

c     parameters for the latitude loop
      real latStart,latEnd,latStep,latMin,latMultiplier
```

```
c     parameters for the longitude loop
      real lonStart,lonEnd,lonStep,lonMin,lonMultiplier

c     ==================================================================

c     ----Change these parameters to change outcome of program----------

      data aspectFile /"aspect_e_50.txt"/
      data latFile /"gdlat_e_50.txt"/
      data lonFile /"gdlon_3_50.txt"/

c     azimuth,elevation,radial distance, and time
      a = 0.0
      e = 50.0
      d = 300
      TM = 2000

c     satellite altitude
      Tgdalt=665.95

      d = d/sin(e*dtr)


c     latMin and lonMin will specify where to start counting and the loop will addup to the final
value
      latMin = 60
      lonMin = -180

c     these parameters define the latitude and longitude range
      latStart = 0
      latEnd   = 500
      latStep  = 1

      lonStart = 0
      lonEnd   = 500
      lonStep  = 1

c     a Multiplier here is given by min value - max value divided by the range
c     so if you want latitude from 40 to 80 with a range of 200 values then
c     multiplier = (80-40)/200
      latMultiplier = 0.05
      lonMultiplier = 0.11
c     ==================================================================

c     Actual Computation begins here>>
c     ==============================

c     ----Find the X, Y, Z components of the radar----------------------
      call convCart(Rlat,Rlon,Ralt,RX,RY,RZ)

c     ----Get Vector of Bore Sight -------------------------------------
      call getBoreVector(a,e,d,TM,SX,SY,SZ,ULVx,ULVy,ULVz,ULNx,ULNy,ULNz
     c,ULEx,ULEy,ULEz)

c     ----Get position vector of Irregularity---------------------------
      IX = RX + SX
      IY = RY + SY
      IZ = RZ + SZ
      call convSphe(Ix,Iy,Iz,Ilat,Ilon,Irad)

c     ----Find the magnetic field components at irreg. location---------
      call geobfield(TM,Irad,(90- Ilat)*dtr,Ilon*dtr,BR,BT,BP,B)

c     ----Open necessary files for writing data-------------------------
      open(u,file = aspectFile,status="replace")
      open(v,file = latFile,status="replace")
      open(w,file = lonFile,status="replace")

c     ----First loop is for latitude------------------------------------
      do 5 latcount = latStart,latEnd,latStep
         Tgdlat = latMin + latcount * latMultiplier
```

```
c     ----Second loop is for longitude---------------------------------
         do 10 loncount = lonStart,lonEnd,lonStep
            Tlon = lonMin + loncount * lonMultiplier

c     ----Convert satellite vector(T) to geocentric and then cartesian--
            call convrt(1,Tgdlat,Tgdalt,Tlat,Trad)
            call convCart(Tlat,Tlon,Trad,TX,TY,TZ)

c     ----Get the aspect angle-----------------------------------------
            call getAsp(IX,IY,IZ,Ilat,Ilon,TX,TY,TZ,SX,SY,SZ,BT,BP,BR,as
     c           pect)

            aspect = abs(aspect-90)

c     ----write the aspect angle
            write(u,*)aspect

c     ----write the latitude
            if(latcount.eq.0)then
               write(v,100)Tlon
            endif

c     ----End of first loop--------------------------------------------
 10      continue

c     ----write the longitude
         write(w,100)Tgdlat

c     ----End of second loop-------------------------------------------
 5    continue

c     ----Close the open files-----------------------------------------
      close(w)
      close(v)
      close(u)

 100  FORMAT(F10.4)

      end
```

aspectCSV.c

```
#include <stdio.h>

int main()
{
  FILE *fs;   //File for input
  FILE *fs2;  //File for output
  float temp; //dummy variable for reading data from file
  int RowCount =0,ColCount2 = 0; //counters to keep track of row number an column number
  int totalLat; //this tells how many value of latitude are present.
  int totalLon; //this tells how many value of longitude are present.


  //*******Change these parameters according to need*********
  totalLat = 501;
  totalLon = 501;

  //need to only change I/O file name
  fs = fopen("aspect_e_50.txt","r");       //open the file from where data is read
  fs2 = fopen("aspect_e_50_out.txt","w");  //open the file where data will be stored
  //*********************************************************

  while(!feof(fs))
  {

    //Manage the indices
    if(ColCount==totalLat)
    {
      RowCount++;
      ColCount = 1;
      fprintf(fs2,"\n");
    }
    else ColCount++;

    //read and write data
    fscanf(fs,"%f",&temp);
    fprintf(fs2,"%f,",temp);
  }
  fclose(fs);
  fclose(fs2);

  return 0;
}
```