The Design and Implementation of a Radar Simulator

Rolf Lengenfelder

A dissertation submitted to the Department of Electrical Engineering,
University of Cape Town, in fulfilment of the requirements
for the degree of Master of Science in Engineering.

Cape Town, September 1998

Declaration

I declare that this dissertation is my own, unaided work. It is being submitted for the degree of Master of Science in Engineering in the University of Cape Town. It has not been submitted before for any degree or examination in any other university.

Cape Town
September 1998

Abstract

This dissertation describes the design and implementation of a radar simulator called Sarsim2. The radar simulator was originally developed to produce *synthetic range profiles* (SRPs) of complex aircraft models. It was then expanded and upgraded to generate simulated *synthetic aperture radar* (SAR) data.

Over the last few years a substantial amount of work has been carried out by the *Radar Remote Sensing Group* (RRSG) at the *University of Cape Town* (UCT) to produce SRPs of aircraft targets using an L-Band search radar of Reutech Systems. The high range resolution that can be obtained from SRP processing makes it feasible to extract characteristic features from a profile obtained from an aircraft. The ultimate aim of producing SRPs is to use these extracted features for *non-cooperative target recognition* (NCTR), i.e. to be able to identify an aircraft type from the echo signal received by the radar. The radar simulator was written to produce SRPs of aircraft models, which could then be used to investigate the feasibility of various aircraft-identification algorithms.

The stepped-frequency processing required to obtain SRPs of aircraft targets has initiated further research in the RRSG into more efficient stepped-frequency processing techniques, and the radar simulator has been used extensively to generate simulated data.

The RRSG group is also actively involved with SAR processing techniques, and the radar simulator has been invaluable in providing necessary simulation data to test various processing algorithms.

One of the main objectives of this simulator was to have an easy-to-use graph-

ical interface, which can show results in real-time. This requirement makes it necessary to find some way of reducing the required computation. The solution implemented may be called WYSIWIC (what you see is what is calculated). This means that the data is only calculated to a resolution depending on the screen resolution. Only when the data is saved to disk will it be calculated and written with the required sampling rate.

Some of the features of the radar simulator include:

- Chirp, monochrome and user-defined pulse modulations
- Stepped-frequency implementation with constant or user-defined frequency increments
- Independent moving platforms with user-defined paths
- Generation of text script files
- Configurable A/D conversion
- Angle dependent *radar cross section* (RCS) of point targets
- Rotating antennas, spot mode SAR
- Point target and platform motion errors
- All user-defined functions can be imported by a separate text file
- Powerful image viewer which can display SAR files of practically any size

The program has been kept flexible so that features (for example using real-life antenna gain patterns) can be loaded with ease.

The graphical frontend of the simulator was initially written in C++ using the *Object Windows Library* (OWL) of Borland C++ 5.0, but was then rewritten with Borland C++ Builder, which made the development much easier. Borland C++ Builder (BCB) can be considered as the best *Rapid Application Development* (RAD) tool currently available, offering visual components combined with

the flexibility and speed of C++. The simulator exploits the protected memory model of 32 bit programs which has the advantages of crash protection and practically no memory restrictions. It therefore has to run under either Windows 95 or Windows NT 4.0. For portability reasons all the code (except the windows front end) has been written in ANSI C++. All processor intensive calculation routines have been written as threads. This makes other tasks running in the background more responsive and also enables the user to abort a calculation at any time.

A second program has been included, which is basically the same program without the graphical frontend. This program is portable as it is written in pure ANSI C++. It performs like a compiler which reads the script files (text files) and writes the required simulation files to disk.

Acknowledgements

I would like to thank the following people in the Radar Remote Sensing Group at UCT, who have all contributed to this dissertation in one way or another, be it through discussion, giving advice, finding bugs in the radar simulator, or simply just by providing a fun and stimulating environment to work in:

Mike Inggs
Jasper Horrell
Richard Lord
Andrew Wilkinson
Gavin Doyle
Yann Tremeac
Grant Carter
Alan Langman
Leon Alexander
Gordon Farquarson
Candace Rennie
Adam Isaacson

My sincerest thanks go to my supervisor, Prof. M.R. Inggs, for his guidance, advice and financial support, and ultimately for his encouragement to write up my simulator manual into this dissertation.

I would also like to give special thanks to Richard Lord, Jasper Horrell and Andrew Wilkinson for all their help, advice and discussions that enabled me to write, debug and enhance the radar simulator.

Contents

List of Figures

List of Tables

List of Symbols

B	—	Transmitted RF bandwidth
c	—	Speed of light
d	—	Distance between radar and target
f_c	—	Radar centre transmit frequency
G_t	—	Transmit antenna gain
G_r	—	Receive antenna gain
k	—	Boltzmann's constant ($= 1.38 \cdot 10^{-23}$ Joules/degree)
N	—	Noise power
P_{tx}	—	Transmitted power
P_{rx}	—	Received power
T	—	Equivalent receiver noise temperature in Kelvin
T_p	—	Pulse width
Δt	—	Time elapsed between transmitting and receiving a pulse
γ	—	Chirp rate of linear FM waveform
λ	—	Wavelength
σ	—	Cross section area of point target

Nomenclature

Azimuth—Angle in a horizontal plane, relative to a fixed reference, usually north or the longitudinal reference axis of the aircraft or satellite.

Beamwidth—The angular width of a slice through the mainlobe of the radiation pattern of an antenna in the horizontal, vertical or other plane.

Burst—Set of all frequencies required to produce a synthetic range profile.

C-band—The frequency range centered around 5 GHz.

Chirp—A pulse modulation method used for pulse compression, also called *linear frequency modulation.* The frequency of each pulse is increased or decreased at a constant rate throughout the length of the pulse.

Coherence—A continuity or consistency in the phases of successive radar pulses.

Corner reflector—A radar reflector that reflects nearly all of the radio frequency energy it intercepts back in the direction of the radar which is illuminating it.

Dilute—If individual scatterers on a target can be resolved, the target features are said to be *dilute.*

Doppler frequency—A shift in the radio frequency of the return from a target or other object as a result of the object's radial motion relative to the radar.

"Earth" platform—Stationary, unshifted, non-rotating coordinate system, identical to the visible axes on the main screen.

Encounter—Set of all profiles while target in sight, acquired over a number of scans.

Isotropic—Non-directional.

Ku-band—The frequency range centred around 14 GHz.

LSB—Least significant bit.

MTI—Moving target indication.

Nadir—The region directly below the satellite position.

NCTR—Non-cooperative target recognition.

NSCAT—NASA Scatterometer.

OWL—Object Windows Library.

Platform—A user-defined coordinate system which can move independently on any path seen relative to the "Earth" coordinate system. All point targets or radars defined on that platform will be stationary as seen from that coordinate system.

Point Target—Infinitely small point which reflects electromagnetic energy. The amount of reflection depends on its surface area and its directional vector.

PRF—Pulse repetition frequency.

PRI—Pulse repetition interval.

Profile—A single synthetic range profile of a target.

RAD—Rapid Application Development.

Radar—Actual energy-transmitting device. Can be positioned onto any platform, however the position is always at the origin of that platform.

Range—The radial distance from a radar to a target.

RCS—Radar cross section.

RRSG—Radar Remote Sensing Group.

Scan—Set of pulses received during illumination time.

Specular—Highly directive, i.e. the power returned from a specular reflector depends very much on the direction of illumination.

SRP—Synthetic range profile.

Swath—The area on earth covered by the antenna signal.

Synthetic Aperture Radar (SAR)—A signal-processing technique for improving the azimuth resolution beyond the beamwidth of the physical antenna actually used in the radar system. This is done by synthesising the equivalent of a very long sidelooking array antenna.

UCT—University of Cape Town.

Chapter 1

Introduction

1.1 Background and User Requirements

The radar simulator (also called Sarsim2) described in this dissertation was originally developed to produce *synthetic range profiles* (SRPs) of complex aircraft models. It was then expanded and upgraded to generate simulated *synthetic aperture radar* (SAR) data.

Over the last few years a substantial amount of work has been carried out by the *Radar Remote Sensing Group* (RRSG) at the *University of Cape Town* (UCT) to produce SRPs of aircraft targets using an L-Band search radar of Reutech Systems [14, 15, 16, 17]. The high range resolution that can be obtained from SRP processing makes it feasible to extract characteristic features from a profile obtained from an aircraft. The ultimate aim of producing SRPs is to use these extracted features for *non-cooperative target recognition* (NCTR), i.e. to be able to identify an aircraft type from the echo signal received by the radar [26]. The radar simulator was written to produce SRPs of aircraft models, which could then be used to investigate the feasibility of various aircraft-identification algorithms.

The stepped-frequency processing required to obtain SRPs of aircraft targets has initiated further research in the RRSG into more efficient stepped-frequency

processing techniques [9, 11, 20, 21, 32, 33], and the radar simulator has been used extensively to generate simulated data.

The RRSG group is also actively involved with SAR processing techniques [3, 4, 12, 13, 22, 24, 27], and the radar simulator has been invaluable in providing necessary simulation data to test various processing algorithms.

1.2 Simulator Features

One of the main objectives of this simulator was to have an easy-to-use graphical interface, which can show results in real-time. This requirement makes it necessary to find some way of reducing the required computation. The solution implemented may be called WYSIWIC (what you see is what is calculated). This means that the data is only calculated to a resolution depending on the screen resolution. Only when the data is saved to disk will it be calculated and written with the required sampling rate. The exact method will be explained in detail in Chapter 6.

Some of the features of the radar simulator include:

- Chirp, monochrome and user-defined pulse modulations
- Stepped-frequency implementation with constant or user-defined frequency increments
- Independent moving platforms with user-defined paths
- Generation of text script files
- Configurable A/D conversion
- Angle dependent *radar cross section* (RCS) of point targets
- Rotating antennas, spot mode SAR
- Point target and platform motion errors

- All user-defined functions can be imported by a separate text file
- Powerful image viewer which can display SAR files of practically any size

The program has been kept flexible so that features (for example using real-life antenna gain patterns) can be loaded with ease.

1.3 System requirements

The graphical front-end of the simulator was initially written in C++ using the *Object Windows Library* (OWL) of Borland C++ 5.0, but was then rewritten with Borland C++ Builder, which made the development much easier. Borland C++ Builder (BCB) can be considered as the best *Rapid Application Development* (RAD) tool currently available, offering visual components combined with the flexibility and speed of C++. The simulator exploits the protected memory model of 32 bit programs which has the advantages of crash protection and practically no memory restrictions. It therefore has to run under either Windows 95 or Windows NT 4.0. For portability reasons all the code (except the windows front end) has been written in ANSI C++. All processor intensive calculation routines have been written as threads. This makes other tasks running in the background more responsive and also enables the user to abort a calculation at any time.

A second program has been included, which is basically the same program without the graphical front-end.. This program is portable as it is written in pure ANSI C++. It performs like a compiler which reads the script files (text files) and writes the required simulation files to disk.

The simulator needs 16 Mb of RAM to work efficiently (32 Mb recommended with Windows NT), and it is recommended to use a graphics card displaying at least 256 colours, but 65k is preferable. A screen resolution of 800 by 600 or higher is necessary.

The following files are required to run the radar simulator:

1. sarsim2.exe
2. sarsimhlp.hlp
3. sarsimhlp.gid

The last two files are necessary to access online help. All the necessary *dll* files have been linked into the executable. However if some of the dialogs are displayed incorrectly when the simulator is executed, there might be outdated libraries on the computer system. In this case copy the files in the supplied **\dll** directory into the **\system32** subdirectory of the windows directory.

The geometry setup and all simulation parameters are saved as a text file with the extension "scr". External data files normally have the extension "dat". Note that long filenames and directory names are supported.

1.4 Development of Dissertation

This dissertation develops in the following way:

- Chapter 2 gives a brief introduction to general radar theory and discusses how this theory is applied in the radar simulator.
- Chapter 3 introduces the radar simulator by setting up and going through a very simple simulation.
- Chapter 4 gives a complete description of all windows and dialogs appearing in Sarsim2.
- The structure of the script files is explained in Chapter 5. Script files can be used to automate certain tasks. They are effectively text files containing a description of the simulation setup.
- In Chapter 6 the formulas used in Sarsim are described in detail.
- Chapter 7 describes various examples for different applications.

- Chapter 8 summarises conclusions reached and discusses future work.
- Appendix A describes various simulation examples to illustrate the functioning of the radar simulator. This work has been co-authored by Helvin Gunputh.

Chapter 2

Radar Theory applied to Simulator

This chapter gives a very brief overview of radar theory and how it is implemented in the simulator.

Radars are electromagnetic devices used for detection of targets by radiating electromagnetic energy and examining the reflected energy. A target can be described as an object which interferes with the transmitted wave and reflects part of its energy. For this simulator, targets occupy an infinitesimally small space, therefore they are called point targets (PTs).Most radars work in pulsed mode, i.e. they send out short burst of energy, with relatively long delays in-between. The period between these pulses is called the *pulse repetition interval* (PRI), but usually the reciprocal, namely the *pulse repetition frequency* (PRF), is specified.

Consider the setup shown in Figure 2.1 with a single point target. A pulse is sent out at time t_0 and the return is received after Δt seconds:

$$\Delta t = \frac{2 \cdot d}{c} \tag{2.1}$$

where c is the speed of light and d is the distance to the target. For this simulator the received signal will be a replica of the transmitted signal, although the power

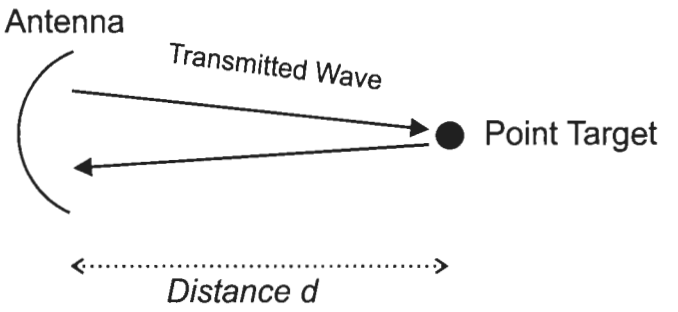

Figure 2.1: A simple radar setup

will be a fraction of the original pulse. The received power P_{rx} of the reflected signal at the radar will be

$$P_{rx} = \frac{P_{tx}\, G_t\, G_r\, \lambda^2\, \sigma}{(4\pi)^3\, d^4} \tag{2.2}$$

where P_{tx} is the transmitted power, G_t and G_r the transmit and receive antenna gain, λ the wavelength, σ the cross section area of the point target and d the distance between radar and target.

Normally many pulses are transmitted as shown in Figure 2.2.

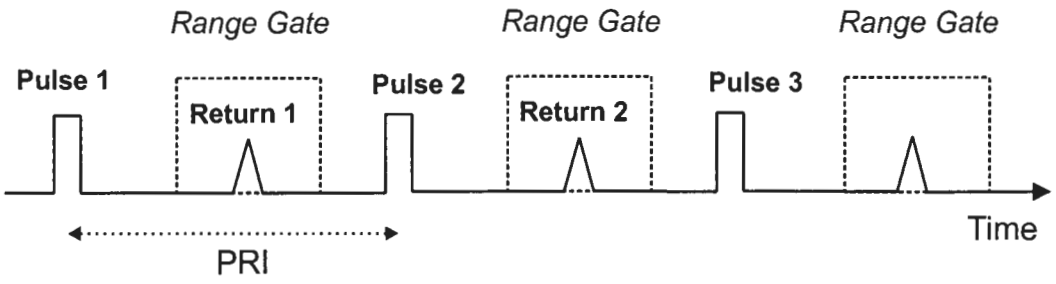

Figure 2.2: Pulsed operation

The point target will return an echo after some time Δt which depends on the distance of the target from the radar according to Equation 2.1. Usually one is not interested in the complete range between two pulses, but only a small fraction of it, which is set by a range gate. Obviously the range delay and returned power can be calculated by hand for simple stationary targets, but it

will get very tedious for moving targets, especially with the addition of noise and motion jitter.

This simulator is capable of generating an array of complex values which correspond to the output of the radar at a certain timespan. For this simulator the nomenclature of SAR literature has been used, so the interval for which data will be sampled (range gate) is called slant range. The time range for which pulses are sent out and the return is sampled is called azimuth range. An example of a single point target which moves with constant velocity past a radar is shown in Figure 2.3.

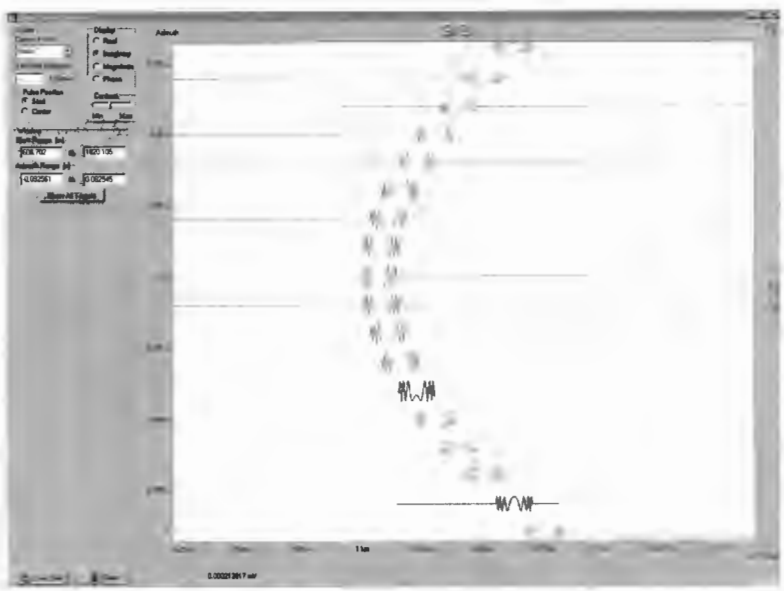

Figure 2.3: An example of a simulation output of a moving point target

A target moves past the radar, being closest at time $t = 0$. The x-axis represents distance from the radar, and the y-axis represents time. In this case chirp modulation has been used with a very low (unrealistic) chirp bandwidth. The simulation shows the raw return (no range compression applied). Note that the signal magnitude decreases as the distance to the radar increases.

This radar simulator therefore creates 2-dimensional arrays of complex numbers corresponding to the output of range versus time for a specific radar. It calculates

the relative distances between the radar(s) and point targets for the specified time period. Depending on these distances, scaled copies of the defined pulse are inserted into the output array. The raw return can be range compressed, i.e. every pulse is convolved with its replica.

Chapter 3

Getting Started

This chapter introduces the radar simulator by setting up a very simple simulation.

3.1 The start-up window

Start the program by executing "sarsim2.exe" either from the command prompt or by double-clicking on the icon in Windows. A screen with a coordinate-system on the right and a list of objects on the left will appear as shown in Figure 3.1.

This radar simulator uses three different kinds of objects: platforms, point targets and radars.

- **Platforms** are user-defined coordinate systems which can move independently on any path seen relative to the "Earth" coordinate system which will be explained below. All point targets or radars defined on that platform will be stationary as seen from that coordinate system.
- **Point targets** are infinitely small points which reflect electromagnetic energy. The amount of reflection depends on their surface area and their directional vector. This will be explained in more detail in Chapter 4.

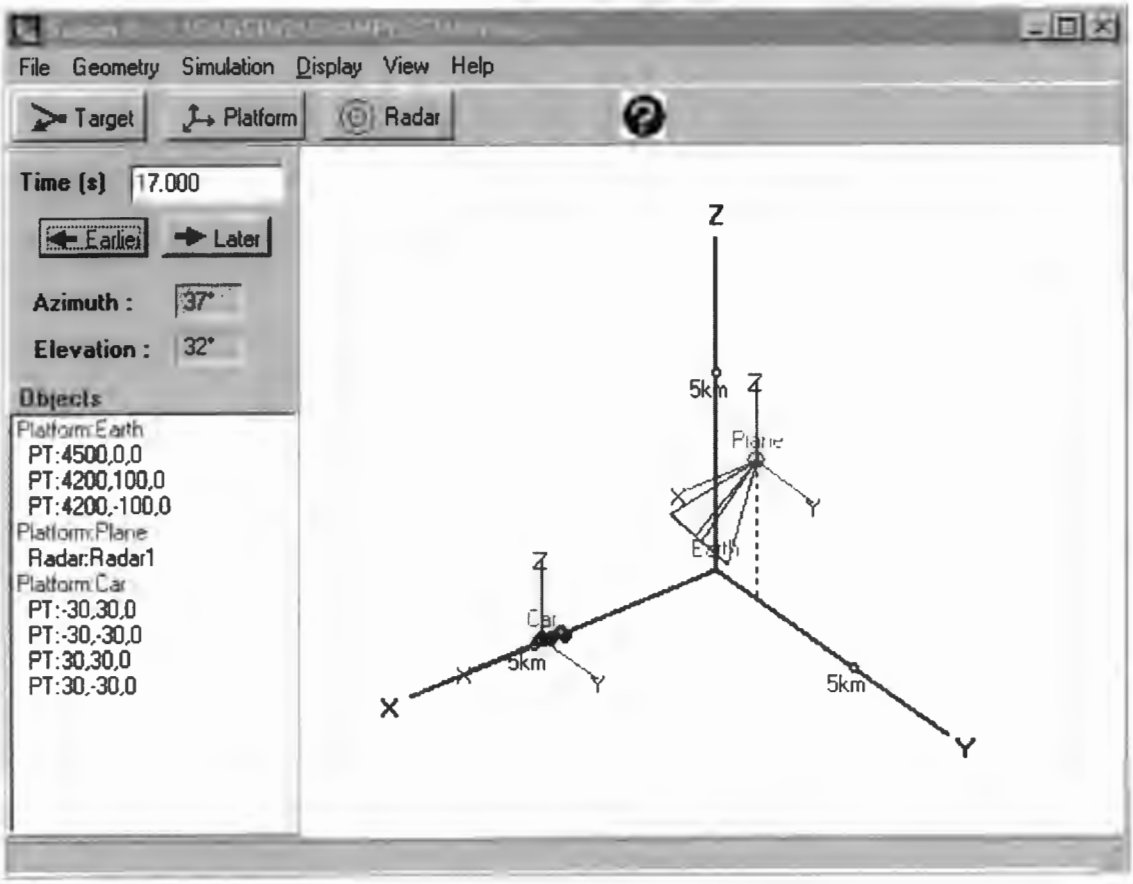

Figure 3.1: Main window

- **Radars** are the actual energy-transmitting devices. They can be positioned onto any platform (however their position is always at the origin of that platform), and there are numerous parameters to be set which will be explained in Chapter 4.

3.1.1 The "EARTH" coordinate system

The coordinate system shown on the start-up window is the most fundamental one and all other objects will be positioned relative to it. It is conveniently called "Earth". The origin of the "Earth" coordinate system is at $x = 0$, $y = 0$ and $z = 0$ with the red, green and blue lines representing the x-axis, y-axis

and z-axis respectively. The labelled ends are pointing in the positive direction. There are distance marks on the axes (for example 10 km) to give some sense of scale. Below the top border of the window, the azimuth and elevation angle of the current viewpoint are shown. Using the mouse, one can manipulate the coordinate system in three ways:

1. Rotate: To rotate the axes, push and hold the left mouse button and move the mouse in the required direction. Moving it up and down will change the elevation angle, moving it left and right changes the azimuth angle.

2. Zoom: To zoom in or out, push and hold the right mouse button and move it up or down. The distance marks on the axes should start moving.

3. There is a way of changing the focus point (the point around which the coordinate system turns). This is described in Section 4.9.

In the top-left corner the current simulation time is given, i.e. what one sees is a snapshot at that given time. This helps visualising the position of point targets and platforms at specific times. The time instance can be changed by entering a time into the edit-box or by clicking on the left/right scroll buttons just to the right of the edit-box.

On the left, a box containing a list of all objects is given. The order depends on the relative positioning, i.e. all objects placed on a platform will appear (indented) just below it.

3.1.2 Placing objects

To place a new object (point target, radar or platform), select the required object from the "Geometry" menu. A dialog with a whole set of parameters will appear. All of the parameters are set to some default value and most of them don't have to be changed for most of the simpler simulations. Use the **Tab** key or the mouse to navigate through the dialogs. By clicking on the **OK** button, the object will

appear on the screen (if in sight). A short-cut to create new objects is the toolbar on top of the screen (see Figure 3.2).

Figure 3.2: The toolbar

3.1.3 Editing objects

There are two ways to edit objects. One is to double-click the object on the list given on the left side, the other is to right-click the object itself in the coordinate system. This only applies to point targets and radars—use the list to modify platforms.

3.1.4 Deleting objects

To delete a point target or radar right-click it in the coordinate system and choose "Delete Point Target" / "Delete Radar" from the appearing menu. To delete a platform choose the required platform from the list.

3.2 A simple example

This section will show how to create a simple simulation with a single point target. The simulation parameters are given in Table 3.1.

- Clear (if necessary) the current simulation by selecting *File / New* on the menu.
- Create a new radar by selecting *Geometry / New Radar* (or by clicking on the **Radar** button). A window as shown in Figure 3.3 will appear.

Type of pulse modulation	chirp pulse
B	50 MHz
f_c	1 GHz
T_p	5000 ns
PRF	1 kHz
P_{tx}	1 kW
antenna gain	isotropic
d	3000 m
σ	3 m^2

Table 3.1: Simulation Parameters

Change the pulse type to "Chirp" by selecting the **Chirp** radio button (circled in Figure 3.3). (Radio buttons are the round circles with one having a black dot in it, meaning this option is selected). The remaining parameters will have the correct parameters by default. Click the **OK** button. A radar (dotted circle with a grey disc in it) will appear at the origin of the **Earth** coordinate system. On the list at the left a "Radar: Radar1" entry under "Platform: Earth" will appear.

- Create a point target by selecting *Geometry / New Target* (or the according **Target** button).

- Set the x-coordinate to 3000 m and the radar cross section to 3 m^2 as indicated in Figure 3.4. Click on **OK**. A point target (red dot with black circle around it) will appear.

- Select *Simulation / Raw return* from the menu. Another window will open, shown in Figure 3.5.

Shown on the right-hand side of this window is a colour-coded image of the received waveform displayed as slant range (x-direction) in meter versus azimuth time (y-direction) in seconds. As the point target is stationary, the position of the point target in slant range stays constantly at 3000 meters. The colour palette

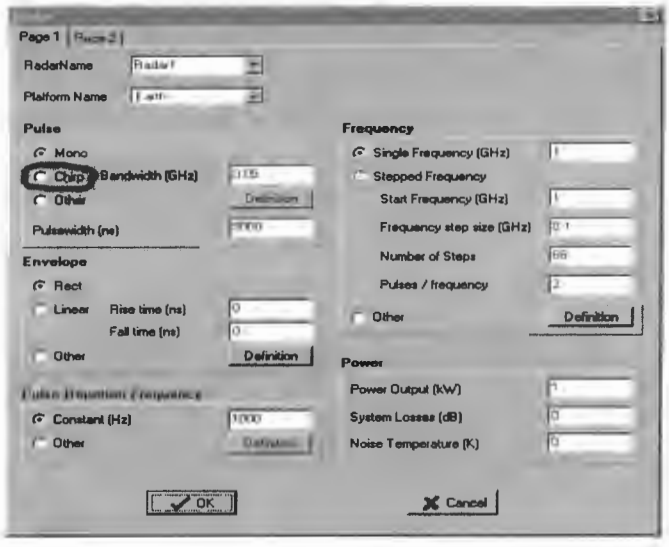

Figure 3.3: Radar setup window (1/2)

has been mapped such that light blue represents the highest positive value, black represents zero and light red the highest negative value. The waveform is shown at baseband, i.e. the frequency spectrum has been shifted such that the carrier frequency is removed. Initially the pulse shown does not resemble a chirp pulse, because aliasing takes place on the screen, i.e. it is necessary to zoom in, to see the details of the chirp pulse.

On the left side the current radar, the simulation window (slant range / azimuth) and the display type (real / imaginary / magnitude / phase) can be chosen.

To select a different radar, click on the desired radar on the list. For this simulation only a single radar is defined.

The simulation window size (slant range and azimuth) can be changed in four ways:

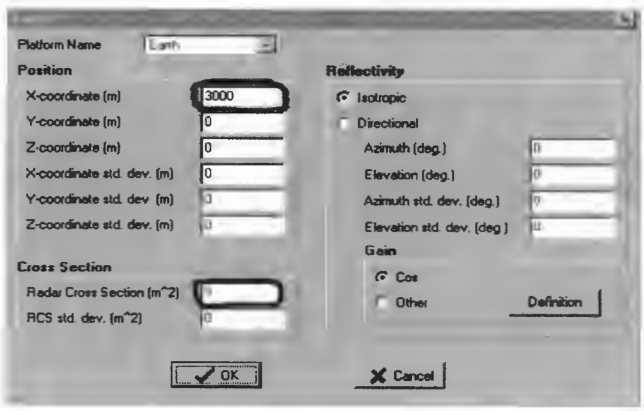

Figure 3.4: Point target setup window

1. Changing the values of the slant/azimuth range in the **edit-boxes** on the left.

2. Zooming in by selecting an area on the image with the left mouse button. This is done by pushing and holding the left mouse button on the desired corner and releasing it on the opposite corner. The window will update automatically.

3. Zooming out by a factor of two in either direction by right-clicking anywhere on the image. The window will also update automatically.

4. Showing all targets (i.e. selecting a slant range spanning from the closest to the furthest target) by clicking on the **Show all targets** button.

If the azimuth scale is such that two pulses are separate by at least 20 pixels, the actual waveform of the pulse will be displayed as a graph as shown in Figure 3.6.

The time instance at which the shown pulses has been transmitted by the radar can then be easily identified, for example in this example the PRF is 1 kHz, therefore a pulse will be transmitted every 0.001 seconds. Note that the graphs of the pulses do not use the azimuth time scale on the left, i.e. the amplitude of the graphs do not correspond to the scale in seconds given at the left, but

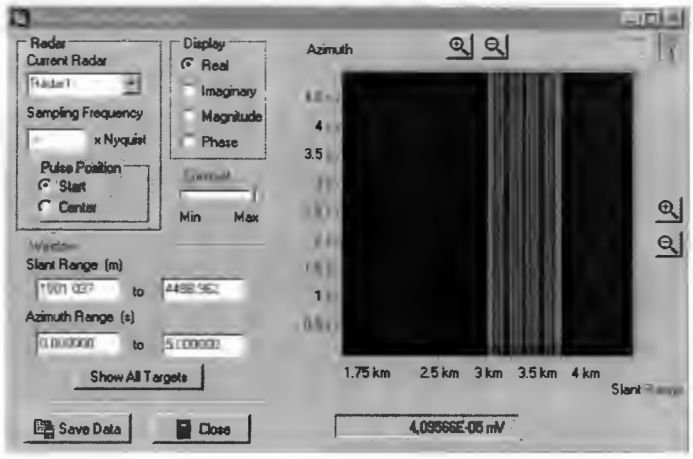

Figure 3.5: Return signal of a single point target

they have their own amplitude scale (not shown). Below the image, the highest absolute magnitude within the displayed image is shown in millivolt. Note that this value might not be accurate due to the sampling process (i.e. the waveform might have been sampled slightly off-peak). For the current simulation a value of $4.095 \cdot 10^{-5}$ mV is shown. For this simple simulation the radar equation reduces to:

$$P_{rx} = \frac{P_{tx} \cdot G^2 \cdot \lambda^2 \cdot \sigma}{(4 \cdot \pi)^3 \cdot d^4} \tag{3.1}$$

$$= \frac{1000 \cdot 1 \cdot 0.3^2 \cdot 3}{(4 \cdot \pi)^3 \cdot 3000^4} \tag{3.2}$$

$$= 1.67976 \cdot 10^{-15}\,\mathrm{W} \tag{3.3}$$

Therefore the amplitude is given by:

$$Magnitude = \sqrt{P_{rx}} \tag{3.4}$$

$$= 4.095 \cdot 10^{-5}\,\mathrm{mV} \tag{3.5}$$

The whole window can be resized (or maximised) in the usual way, but note that the calculation time is proportional to the image size. On slow computers

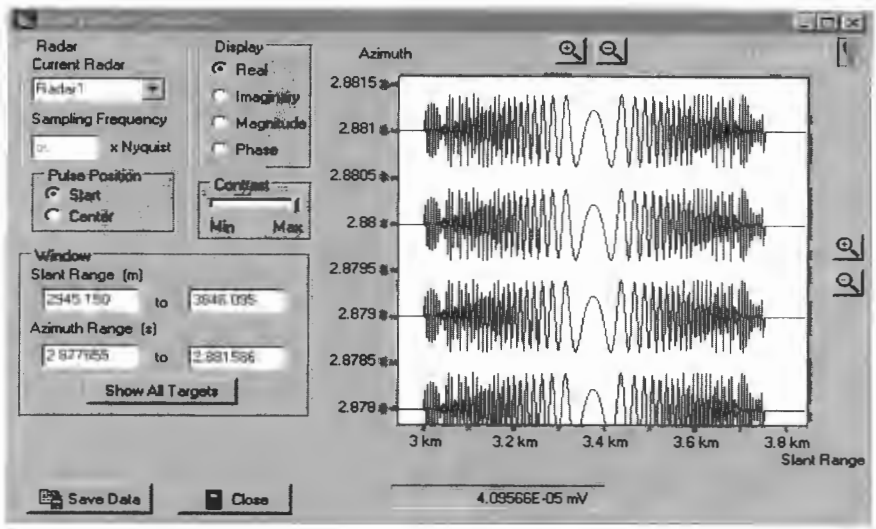

Figure 3.6: Raw return after zooming in

it might be advisable to make the window smaller than the default size.

To **save** the window displayed, click on the "Save Data" button. This function will be explained in more detail in Section 4.9.

Chapter 4

Command Summary

This chapter explains all available commands.

4.1 Main Window

4.1.1 Overview

After starting Sarsim you will see a screen similar to the one shown in Figure 4.1. In this specific example a file has been loaded for demonstration purposes. On the left side of the window, the current simulation time, the current look-angle and a list of all objects is given. The simulation time is useful in order to see how objects move with respect to time. It can be changed by pressing the **Earlier** or **Later** buttons, or by entering a specific time into the edit box. The look angle is given by an azimuth and elevation angle, where the azimuth angle is measured clockwise from the y-axis, and the elevation angle from the x-y plane. The object list shows all current objects, namely point targets (PTs), platforms and radars. They are sorted with respect to platforms, and below each platform an indented list of all objects relative to this platform is given. Objects can be altered by clicking or double-clicking the respective list item. On top of the screen a few speed buttons are placed. Clicking on them will create a new object

of the selected type.

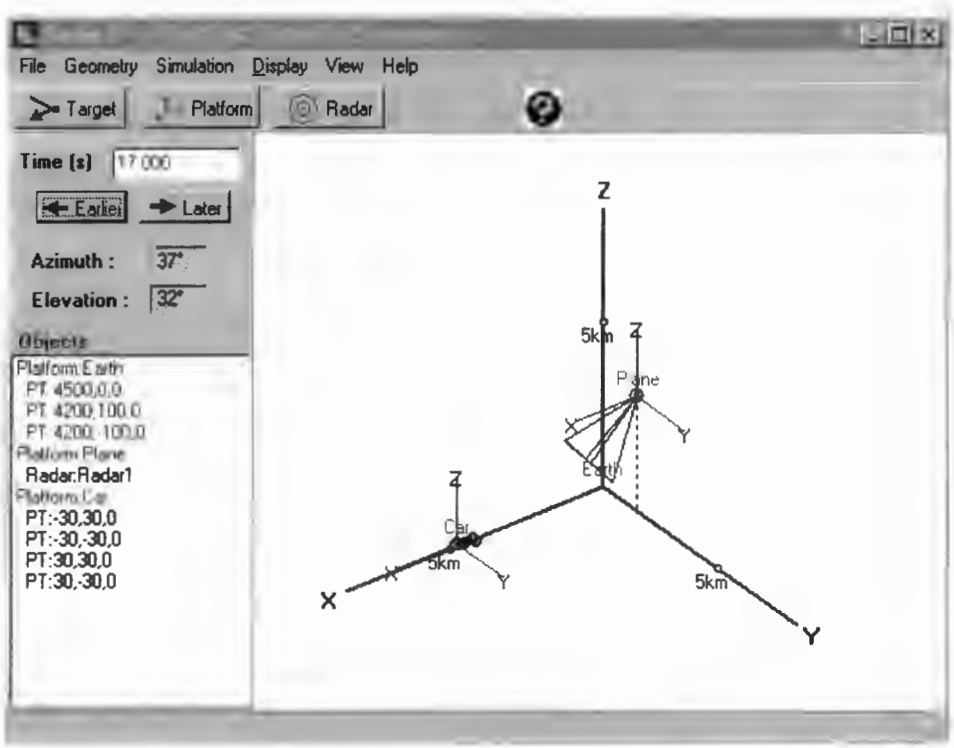

Figure 4.1: Main window

4.1.2 Mouse Commands

The mouse is used to rotate the coordinate system and to zoom in and out. It is also used to modify or delete objects.

- **ROTATION**: To rotate the axes, push and hold the left mouse button and move the mouse in the required direction. Moving it up and down will change the elevation angle, moving it left and right changes the azimuth angle.
- **ZOOM**: To zoom in or out, push and hold the right mouse button and move it up or down. The distance marks on the axes should start moving.

- **MODIFY / DELETE**: Point targets, platforms and radars can be modified by either selecting the required object on the list or right-clicking the required object in the coordinate system. A menu containing options to modify or delete the object will appear.

4.1.3 Menu Commands

File

- **New:** Creates a new simulation. If the current simulation has not been saved, the user will be asked if the current file should be saved.
- **Open:** Loads a script file from disk (default extension = ".scr"). The structure of the script files will be explained in detail in Chapter 5.
- **Save:** Saves the current simulation file to disk. If no name has been given yet, the user will be asked to enter a name.
- **SaveAs:** Saves the current file with a new name.
- **Exit:** Exits the program.

Geometry

- **New Target:** Creates a new point target. A detailed description is given in Section 4.2.
- **New Platform:** Creates a new platform. A detailed description is given in Section 4.3.
- **New Radar:** Creates a new radar. A detailed description is given in Section 4.4.

Simulation

- **Raw Return:** This option shows the received return signal mixed down to baseband. A detailed description is given in Section 4.5.1.
- **Range Compressed:** This options shows the range compressed return, also mixed down to baseband. A detailed description is given in Section 4.5.1.
- **Previously Stored:** The parameters of a simulation can be stored so that they can be recalled quickly. This includes the window range, sampling frequency, radar used, output file name and some more parameters. A list of all stored simulations will be given. An option to delete a simulation setup is also given.

4.2 New Target Window

Creating a new target brings up a dialog window shown in Figure 4.2.

The following parameters can be configured:

- **Platform name (up to 15 characters):** The platform to which the current point target belongs. The coordinates entered will be relative to the origin of this given platform.
- **Position x, y and z**: The coordinates of the point target in meters.
- **Standard deviation of x, y and z position**: Introduce Gaussian distributed random jitter.
- **Radar Cross Section**: The radar cross section in square meters.
- **Radar Cross Section deviation**: If the target scintillates, the standard deviation of the RCS can be specified. The distribution is Gaussian.

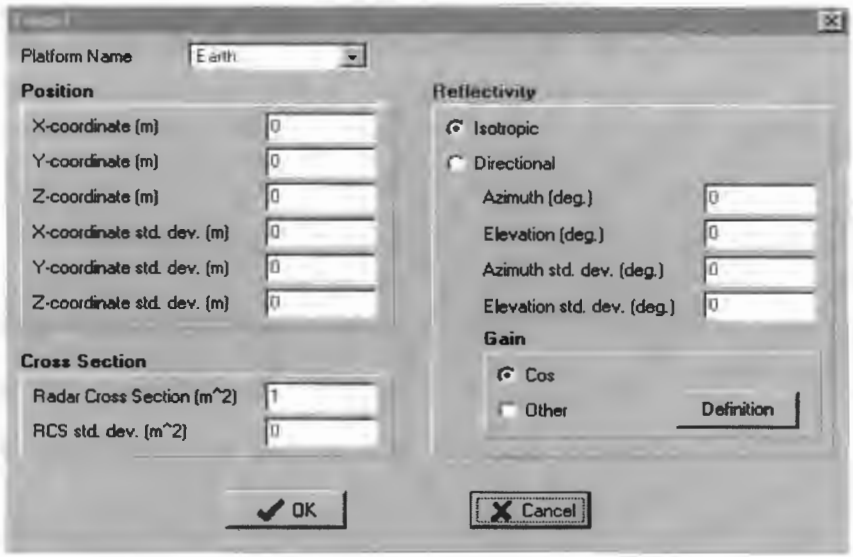

Figure 4.2: New target dialog

- **Isotropic reflectivity:** The incidence angle of the ray from the radar is irrelevant, i.e. the radar cross section will always be constant.
- **Directional reflectivity**: This imitates a surface for which the reflected energy is given as a function of the incidence angle (azimuth and elevation angle). If the gain is set to "Cos", the radar cross section changes exactly like the projected surface of a disc seen from a specific angle, i.e. effective RCS = RCS $\cdot$ cos(incidence angle). No reflection takes place on the backside. It is possible to define a specific gain pattern by clicking on the "Definition" button. This is explained in detail in Section 4.6.

4.3 New Platform Window

Creating a new platform brings up the dialog window shown in Figure 4.3.

The following parameters can be configured:

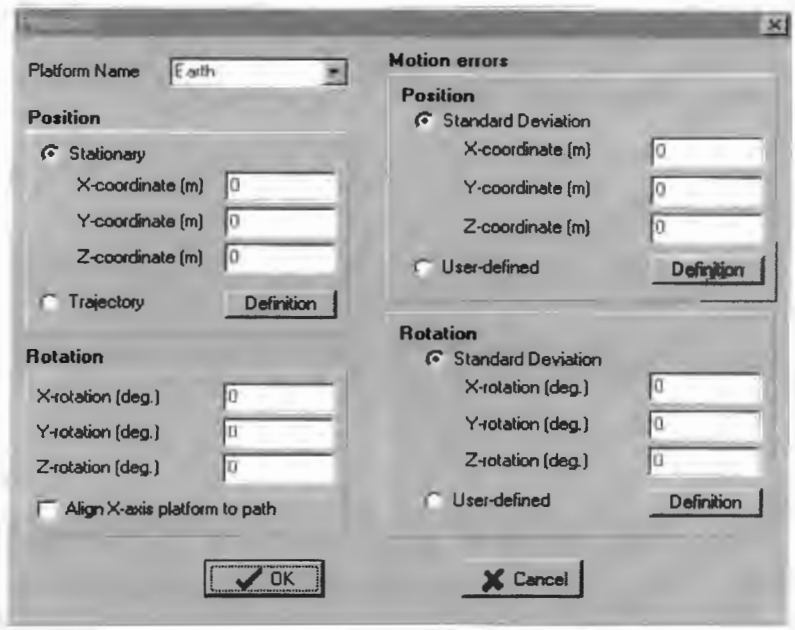

Figure 4.3: New platform dialog

- **Platform name**: The name of this platform (up to 15 characters) for future referencing.

- **Position, "Stationary" or "Trajectory"**: If the position of the platform is stationary, the x, y and z coordinates need to be entered. This will place the platform at the given coordinate on the "Earth" platform. If the position of the platform is defined as a trajectory, the position for every time instance can be defined. The given points will be interpolated (parabolic, linear or low-pass filtered). This will be explained in Section 4.6.

- **x, y and z-rotation**: This describes the rotation angles of the platform around the x, y and z axes seen relative to the "Earth" platform. Note that the platform can be "aligned".

- **Align x-axis to path**: If this box is provided with a check mark, the coordinate system will be rotated in such a way that the x-axis will coincide with the path at any given time instance. To express it differently, the

azimuth and elevation angle of any point on the path will be parallel to the x-axis of the platform. The y-axis will however still be parallel to the "Earth" platform. An example where this is useful is if a platform is used to describe the motion of an aircraft, then the nose of the aircraft will always point in the direction it is flying. An example is give in Chapter 7.

- The parameters in the box "Motion errors" describe any deviations from the ideal position or rotation. The distribution is Gaussian. There is the option to define motion errors by giving an integration envelope which will be convolved with Gaussian distributed noise, but this option is still in development.

4.4 New Radar Window

Creating a new radar brings up a dialog window shown in Figure 4.4.

The following parameters can be configured on this page:

- **Radar name**: For convenience there can be more than one radar in the simulation. The radar name has to be specified for future reference.
- **Platform name**: The name of the platform onto which the radar will be positioned. The radar is always positioned at the origin of the platform.
- **Pulse type**: Choose either monochrome, chirp pulse or user-defined pulses. If a chirp pulse is chosen, the chirp bandwidth needs to be specified.
- **Pulsewidth**: The pulsewidth in nanoseconds.
- **Envelope**: The pulse can be multiplied by an envelope function to model more realistic pulses as shown in Figure 4.5.
- **Pulse Repetition Frequency**: The PRF can be defined either as a constant or it can vary from pulse to pulse as shown in Figure 4.6. The data wraps around, such that PRI[x] =PRIArray[modulus(x, n)]. Note that all times are given in seconds.

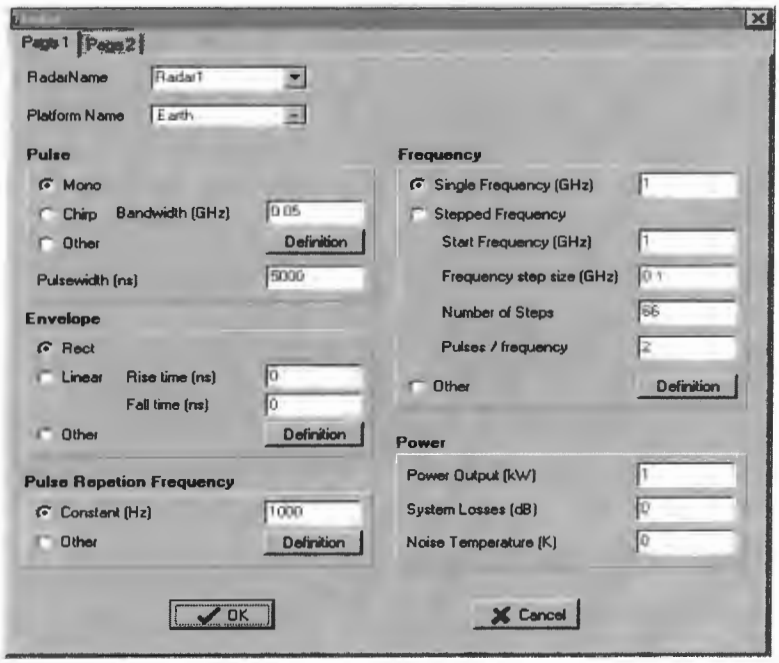

Figure 4.4: New radar dialog - Page 1

- **Centre frequency**: Specify the centre frequency of each pulse. It can be constant, stepped or completely arbitrary.
- **Power output**: Total power output of radar in kW.
- **System losses**: Total system losses in dB.
- **Noise temperature**: Noise temperature of system in Kelvin.

Noise is modelled in a simple way by specifying noise temperature of the receiver. The following example will show how to convert between the noise temperature and a required S/N ratio at a certain distance:

The noise power can be calculated by:

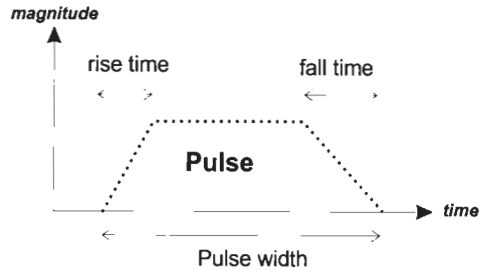

Figure 4.5: Pulse envelope

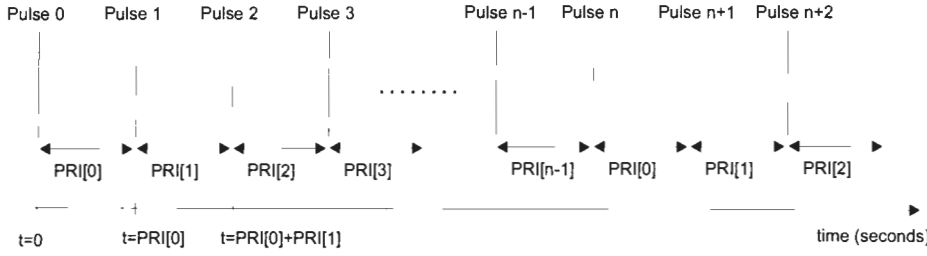

Figure 4.6: User-defined PRFs

$$N = k \cdot T \cdot B \tag{4.1}$$

where k is Boltzmann's constant ($= 1.38 \cdot 10^{-23}$ Joules/degree), T is the equivalent receiver noise temperature in Kelvin and B is the bandwidth of the pulse (for chirp pulses = Chirp bandwidth, for monochrome pulses = $\frac{1}{\text{Pulsewidth}}$). Assuming we want to add the noise to a given complex sample $S = I + j \cdot Q$, the values representing complex voltage. The resulting vector (signal + noise) $SN = IN + j \cdot QN$ is then calculate as follows:

$$IN = I + \text{GaussianNoise}\left(\sqrt{N}\right) \tag{4.2}$$

$$QN = Q + \text{GaussianNoise}\left(\sqrt{N}\right) \tag{4.3}$$

where GaussianNoise() returns a random value with mean = 0 and given standard deviation.

Consider a simple simulation with a point target (RCS = $1\,\text{m}^2$) at a distance of

1 km, and a radar with a power output of 1 kW. The pulse should have monochrome modulation and have a width of 1000 ns. The noise temperature is set to 100 Kelvin, the centre frequency is 1 GHz. The signal power received would be:

$$
\begin{aligned}
P_{rx} &= \frac{P_{tx} \cdot \lambda^2 \cdot \sigma}{(4 \cdot \pi)^3 \cdot d^4} && (4.4)\\
&= \frac{1000 \cdot 0.3^2 \cdot 1}{(4 \cdot \pi)^3 \cdot 1000^4} && (4.5)\\
&= 4.53 \cdot 10^{-14}\,\mathrm{W} && (4.6)
\end{aligned}
$$

The noise power would be:

$$
\begin{aligned}
N &= k \cdot T \cdot B && (4.7)\\
&= 1.38 \cdot 10^{-23} \cdot 100 \cdot \frac{1}{10^{-6}} && (4.8)\\
&= 1.38 \cdot 10^{-15}\,\mathrm{W} && (4.9)
\end{aligned}
$$

which gives a power S/N ratio of:

$$
\begin{aligned}
\frac{S}{N} &= \frac{4.53 \cdot 10^{-14}\,\mathrm{W}}{1.38 \cdot 10^{-15}\,\mathrm{W}} && (4.10)\\
&= 32.8 && (4.11)\\
&= 15.2\,\mathrm{dB} && (4.12)
\end{aligned}
$$

Converting to amplitudes (assuming a 1Ω resistor):

$$
\begin{aligned}
V_{\text{received}} &= \sqrt{P_{rx}} && (4.13)\\
&= 2.13 \cdot 10^{-7}\,\mathrm{V} && (4.14)
\end{aligned}
$$

and

$$\text{Noise Voltage} = 3.715 \cdot 10^{-8}\,\text{V} \tag{4.15}$$

The signal to noise ratio with respect to amplitudes would be 5.73.

A second page of configurable parameters can be selected. It is shown in Figure 4.7.

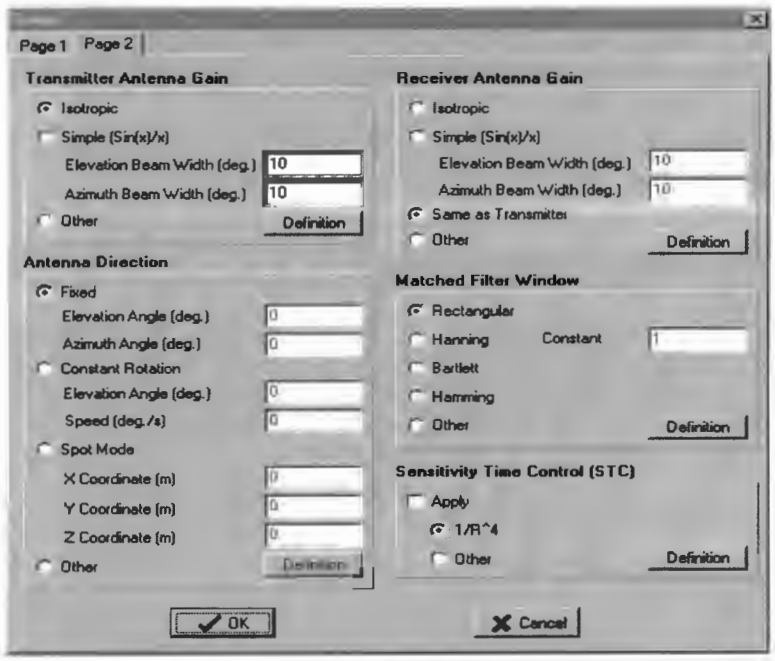

Figure 4.7: New radar dialog - Page 2

The following parameters can be configured on this page:

- **Transmitter and receiver antenna gain**: Either isotropic, a typical $\sin(x)/x$ pattern, or user-defined.
- **Antenna Direction**
- **Matched Filter Window**

- **Sensitivity Time Control**

The radar is positioned at the origin of the given platform.

4.5 Simulation Window

4.5.1 Overview

There are two processing options available:

- **A raw return simulation**: In this case the transmitted signal is not range-compressed (i.e. convolved with a replica). The sampling rate is automatically set such that all detail is shown at the current screen resolution.
- **A range-compressed simulation**: The return gets range-compressed. Because range-compression is computationally intensive, the sampling rate can be set. Note that setting low sampling rates (less than three times Nyquist rate) may result in unfamiliar looking graphs.

A typical simulation window is shown in Figure 4.8.

The simulation window shows the received return signal mixed down to baseband. On the right hand side a window with slant range (in meters) versus azimuth (in seconds) is shown. The maximum amplitude (in mV) is shown at the bottom. This is only an approximation based on the current window. In the top left corner the name of the selected radar is shown. The returned signals have been calculated for this specific radar. Select a different radar to display returns for a different radar.

To change the simulation window, edit the range and azimuth values in the window box. It is possible to zoom in interactively by left-clicking and dragging on the simulation window. If the simulation window is right-clicked, it will zoom out by a factor of two in the slant range and azimuth directions.

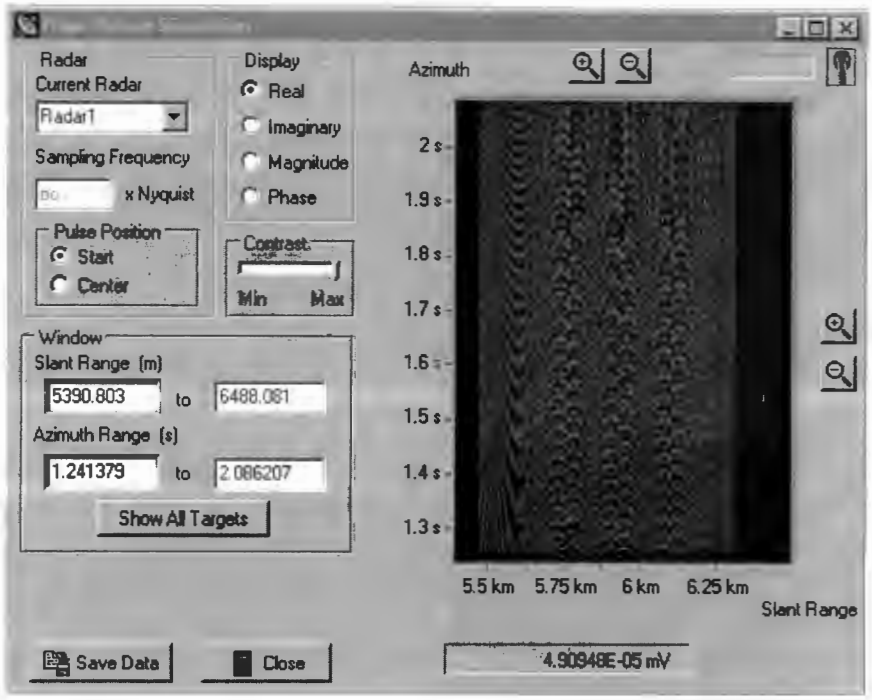

Figure 4.8: A typical simulation window

The "Display" box selects whether the real, imaginary part or the phase or magnitude is displayed. The display of the pulses is colour-coded, where blue represents positive, black zero and red negative values. In case of a graphics card supporting 256 or more colours, the colours will change smoothly according to the amplitude. The contrast can be changed by moving the slider. If one zooms in such that successive pulses are separated by more than 10 screen pixels in the azimuth direction, the pulses will not be colour-coded anymore, but shown as proper graphs.

There is a important point which needs to be clarified: The simulation window shows slant range in the x-direction and azimuth in the y-direction. However what is displayed in reality is what the A/D converter will see, i.e. the scale in slant range is measured as range delay. The difference between distance and range delay is a factor of two, because the pulse has to travel twice the distance. So the simulation window actually uses range delay $= \frac{c}{2d}$ (in seconds) as the x-

axis, but shows a scale which corresponds to the actual distance to the target. An example: a pulse of length 1 us is 300 m long. If the A/D frequency is 500 MHz, the returning pulse will cover over 500 samples. One sample corresponds to 0.3 m $\left(\frac{c}{2 \cdot f_{ad}}\right)$ in distance and therefore the pulse will have a length of $500 \cdot 0.3$ m = 150 m on the display (and not the actual 300 m). This might sound confusing, but the factor of 2 simply originates from the fact that the beam has to travel twice the actual distance (to the target and back). If the return is converted into range by multiplying by the speed of light, the target will appear at twice the distance where it actually is, and therefore the range (and the pulse length) is divided by a factor of 2 to overcome this problem.

4.5.2 Saving Data

After choosing the required window in range and azimuth, the data can be saved in either text or binary format. The dialog shown in Figure 4.9 will come up.

The dialog is divided into three sections:

Simulation Window

The simulation window determines the range in time (azimuth) and distance (slant range) which will be saved to disk. The actual values were set in the previous window (simulation window) and cannot be changed in this dialog. The slant range is given in meters and represents the distance from the radar. It can be converted into time (range delay) by using the simple relationship $t = \frac{2d}{c}$. The azimuth range is measured in seconds and corresponds to the time span for which the data will be recorded. If the PRF is constant, the number of pulses can be determined by using the following equation:

$$\text{number of pulses} = (\text{EndTime} - \text{StartTime}) \cdot \text{PRF} + 1 \tag{4.16}$$

In non-SAR applications the term azimuth range is equivalent to the time window (slow time) for which the radar is capturing data.

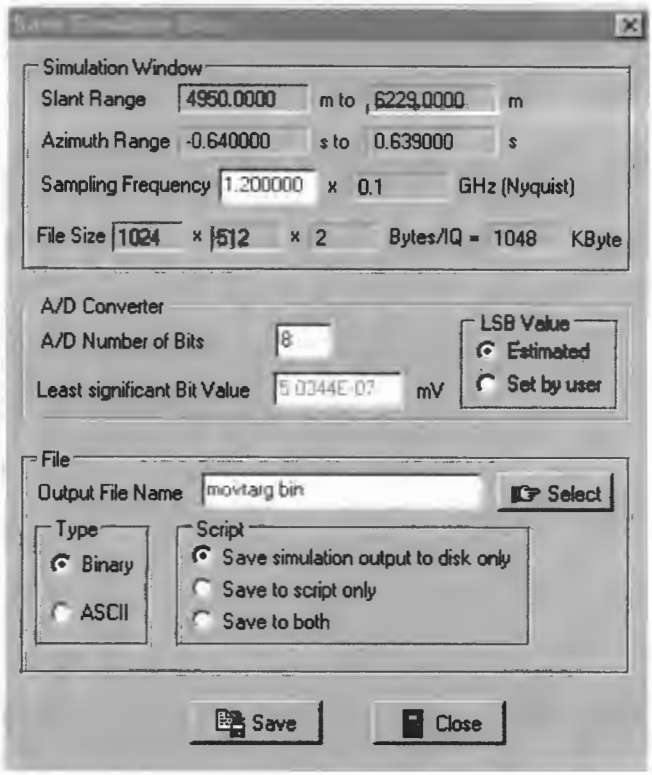

Figure 4.9: "Save Data" dialog

The sampling frequency determines the sampling rate of the received signal. By default the sample frequency is set to the Nyquist frequency which is calculated from either the pulse length or the chirp bandwidth for monochrome and chirp pulses respectively. The sampling frequency can be altered by changing the multiplication factor. Undersampling, i.e. inserting values below unity, is accepted.

Below that, the size of the file is estimated. In this case (see Figure 4.9), there are 512 sample points in slant range (1279 m in range $\triangleq \frac{2 \cdot 1279}{c} = 8.5352\,us$, $8.5352\,us \cdot 120\,\text{MHz} = 1024$) and 512 pulses in azimuth (1.279 seconds at 400 Hz PRF (inclusive)). For ASCII files, the program can only estimate the actual space needed for each I/Q pair, as trailing zeroes are not saved. However if the file is saved in binary format, the size given will be precise.

A/D Converter

The next box requires two inputs, namely the A/D sampling accuracy in bits and the value of the least significant bit (LSB) in millivolt. If the value is not divisible by 8, the remaining bits will be zero-padded. The LSB value is estimated from the maximum magnitude occurring in the simulation window. It will not be absolutely accurate since the sampling frequency of the screen window might be less than the sampling frequency used for saving the file. It is possible to change the LSB value by selecting the "Set by user" check box. After the file has been saved, the program will show how much dynamic range has actually been used.

File

The last box contains the file name and the format in which the data should be saved in. If the file is saved in binary format, each I and Q value is represented by an integer number of bytes with zero-padding applied if necessary. For example 12 bit A/D accuracy will require 2 bytes for each I and Q value, with the four least-significant bits set to 0. The offset is $(2^{x-1} - 1)$ with x being the number of A/D bits (for example 8 bits correspond to an offset of 127, i.e. 0 Volts = 127). If the file is saved in text (ASCII) format the number (as calculated before) will be written as a proper number.

An example is given now:

In the example above the maximum value in the simulation window was $6.39369 \cdot 10^{-5}$ mV. Using 8 bits per value, this means the least significant value will be $\frac{6.39369 \cdot 10^{-5}}{2^{8-1}-1} = \frac{6.39369 \cdot 10^{-5}}{127} = 5.0344 \cdot 10^{-7}$ mV as is also shown in Figure 4.9. If for example the I-value at some point is $1.2 \cdot 10^{-5}$ mV, then the value written to disk will be $(2^{8-1}-1)+\text{round}\left(\frac{1.2 \cdot 10^{-5}}{5.0344 \cdot 10^{-7}}\right) = 127+37 = 164$, where the first part of the formula represents the offset value. In binary mode the character corresponding to 164 will be written to file, while in ASCII mode the three separate digits "1", "6" and "4" will be saved.

There is the option to save the current simulation window (i.e. the window range,

sampling frequency, etc.) in the script file so that it can be recalled quickly at a later stage. This is done by setting the "Save to script only" or the "Save to both" check box. If it is set to "Save to script only" it means the file will be not written to disk, however the simulation parameters will be added to the script file. This will be explained in more detail in Chapter 5. If it is set to "Save simulation output to disk only", no simulation entry will be added to the script file. "Save to both" adds the simulation to the script file and also saves the actual data to disk.

After selecting the "Save" option, the file will be saved, showing a progress bar. The saving process can always be interrupted by pressing the cancel button.

4.5.3 Data file structure

The data is saved in a complex format where each sample point is represented by an inphase component (I) and a quadrature component (Q). The structure is shown in Figure 4.10.

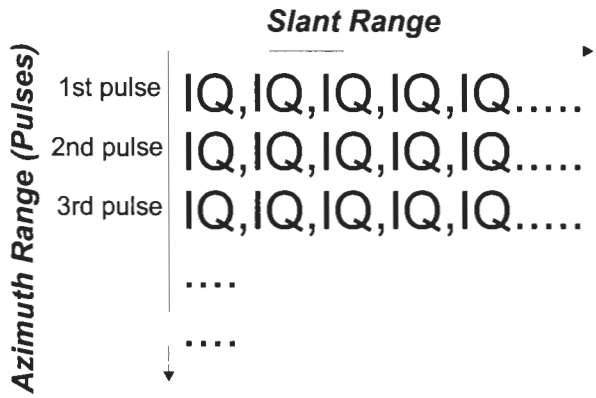

Figure 4.10: Data file structure

The I and Q values are written in turns. Each line in the file corresponds to the return of one pulse, with the "earliest" pulse written first. Within each line the sampled values are written as IQ pairs, the "earlier" (closer to the radar) samples written first. These pairs are separated by commas if the file is saved

as text, but have no separation character if the file is saved in binary format. In text mode after each pulse a "new line" character is inserted. For the parameters shown in Figure 4.9 there would be 1024 IQ pairs (2048 numbers) for each line and a total of 512 lines.

4.6 User-defined Functions

The radar simulator is highly configurable. There are 12 functions which can be defined completely arbitrarily, for example the pulse shape, the centre frequency of every pulse, the PRI (pulse repetition interval) between any 2 pulses, the trajectory of moving platforms, the antenna gain patterns, matched filter windows and some more. These functions can either be defined by entering the sample points straight into the simulator, or by specifying an external data file.

4.6.1 The data entry window

A standardised input screen is used for all of them, which is shown in Figure 5.1.

In this case the window for defining a trajectory of a platform is shown. There are three variables to be defined for the trajectory, i.e. the x, y and z coordinates vs time. To define a function, at least one point (coordinate vs time) has to be specified for each of the three variables. The program will interpolate between the given points. If only one point is specified (for example 500 m at $t = 0$ seconds), the coordinate will stay constant (500 m) and for all given times the graph will be a (horizontal) straight line passing through $(0, 500)$. If two points are given, the graph will also be a straight line passing through both points, but will not necessarily be horizontal. For more than 2 points, the program will use the selected interpolation method. Note that the points do not have to be defined with constant intervals for the x-coordinate, for example $(1, 3)$, $(2, 5)$ and $(20, 10)$ is perfectly acceptable.

There are currently 3 interpolation methods implemented:

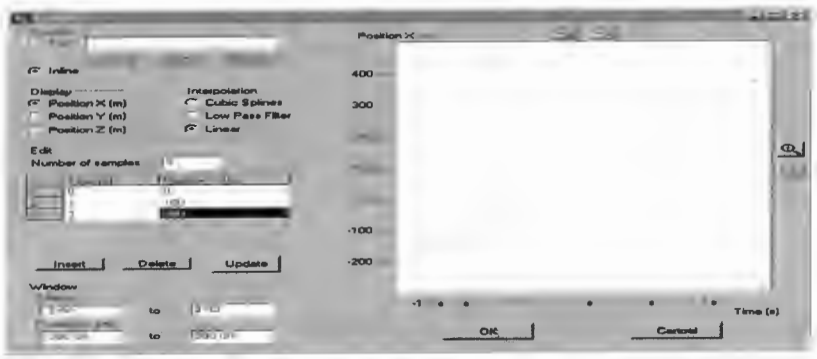

Figure 4.11: A user-defined function

1. **Cubic splines:** This will use cubic splines to interpolate between points. This interpolation is computationally intensive and should be used only for less than 1000 points or so.

2. **Low pass filter:** In this case the defined points can be considered as impulses to a low-pass filter with a highest frequency response given by the inverse of the average distance between 2 successive points. Note that the function might not touch all given points if their x-interval is not constant.

3. **Linear:** A straight line interpolation between 2 successive points is implemented, as shown in Figure 5.1.

A comparison between the different methods is given in Figure 4.12.

For the example in Figure 5.1, 3 samples are given. To create a platform moving with constant velocity it would be sufficient to specify two samples.

A detailed description of every part of the window is given below:

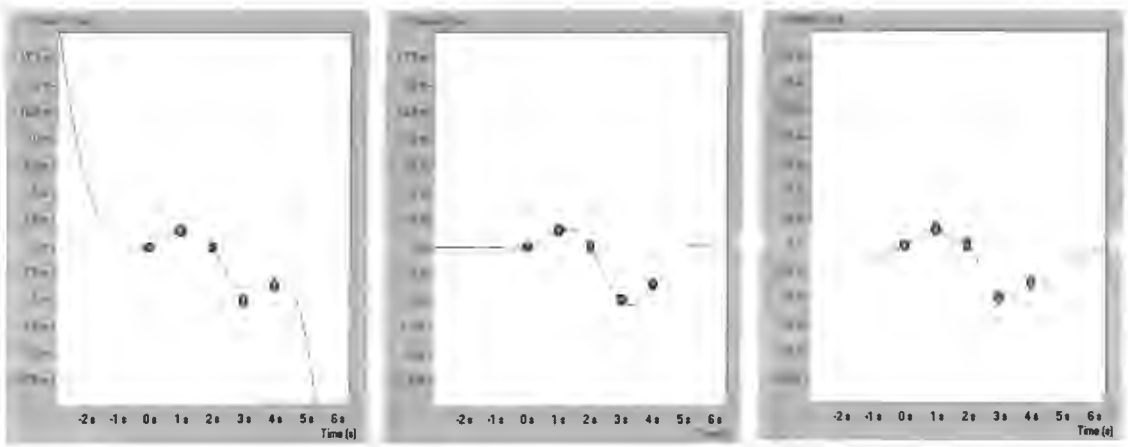

Figure 4.12: The different interpolation methods

- **Source:** The data can be either stored in a separate file or inline (within the script file). For large data sets it might be more practical to use external data files. However for simple simulations the "inline" option is more compact. The structure is described in Chapter 5.

- **Display:** Depending on the data type, there are up to three variables which can be defined simultaneously. For the above example three coordinates need to be defined. If there is more than one variable, one can switch between them by clicking on the specific radio button.

- **Edit:** The data can be entered or modified in this box. The **Number of samples** edit box defines the total number of samples specified for the given variable (they might be different for another variable, for example you might want to specify 5 points for the x-coordinate, but only 1 for the y and z coordinate). Below that a list of x and y coordinates is given for the number of specified points. After entering the values press **Update** to update the graph on the right. When entering more than 20 values or so it is more practical to read external text files. Points can be deleted or added at the current cursor position.

- **Window:** Here the current range of the graph shown can be set. Some

data types are limited in range.

- It is possible to **zoom in** by selecting an area on the graph with the left mouse button. This is done by pushing and holding the left mouse button on the desired corner and releasing it on the opposite corner.
- **Zooming out** by a factor of two in either direction is achieved by right-clicking anywhere on the graph.
- Points can be **dragged** to a new location on the graph by pushing and holding the left mouse button on the desired point and moving it to a new position.

All of the data types for which the ordinate increases in fixed steps of 1, the ordinate will be ignored. The graph will not be a continuous line but a bar graph. One example is the definition of the centre frequency for each pulse. In this case the ordinate just represents the pulse number and will be identical to the current sample number.

4.6.2 External data files

If the data is specified to be an external file, the structure of the data depends on the functions to be defined, but is of the following general format:

```
[Number of samples for function 1], [X1], [Y1], [X2], [Y2], [X3], [Y3] ....[Xn],[Yn]
[Number of samples for function 2], [X1], [Y1], [X2], [Y2], [X3], [Y3] ....[Xn],[Yn]
[Number of samples for function 3], [X1], [Y1], [X2], [Y2], [X3], [Y3] ....[Xn],[Yn]
```

First the number of samples is given, then the x and y coordinates for all the samples are given. This is repeated up to 3 times depending on the specific data type (for example position requires 3 coordinates, others only require 1 or 2). Comments can be inserted by using an exclamation mark after which all text until the next line break is ignored. Commas or spaces can be used as separators. As an example the data file for the points given in Figure 4.12 is given:

```
! Data file for : Position X (m) vs Time (s), Position Y (m) vs Time (s), Position Z (m) vs Time (s)
! Structure : [No. Samples] [Time (s) 1] [Position X (m) 1] [Time (s) 2] [Position X (m) 2] ...
! [No. Samples] [Time (s) 1] [Position Y (m) 1] [Time (s) 2] [Position Y (m) 2] ...
! [No. Samples] [Time (s) 1] [Position Z (m) 1] [Time (s) 2] [Position Z (m) 2] ...
5
0 0, 1 1.5, 2 0, 3 -5, 4 -3.5
1
0 0
1
0 0
```

5 points have been define for the x coordinate, and 1 for the y and z coordinate. Data files can also be edited as described above and the changes can be written to disk by clicking the **Save** button.

4.7 Viewing Simulation Files

A powerful image viewer has been integrated into Sarsim. It can be found under **Display / Simulation Output**. The features of this viewer include:

- No limit in the file size of the image to be viewed. The file is loaded "on-the-fly" and only the displayed portion is kept in memory.
- Import of Sun-Raster Files, Sarsim simulation files and user-defined files.
- Thumbnail overview image shown.
- Slices in range or azimuth.
- Several colour palettes available with slider controls for brightness, contrast and saturation.
- Automatic display of I,Q, phase and magnitude for selected sample.
- Usual zooming and panning controls.

The dialog displayed in Figure 4.13 appears after selecting the menu option **File / Open**.

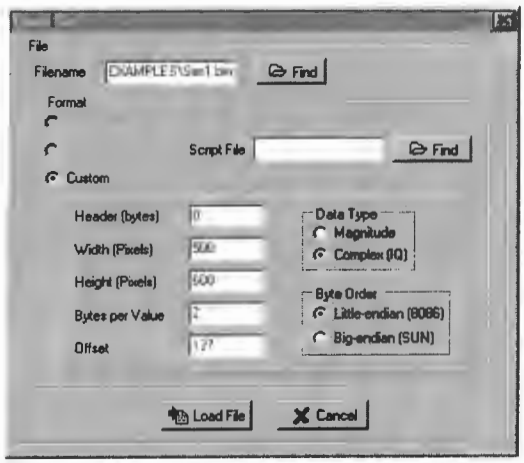

Figure 4.13: Load file dialog

- **Filename**: The filename of the image file. Use the **Find** button to browse for a file.
- **Format**: Several formats are supported:

  1. Sun-Raster-File format: These files have a header defining their size and other parameters.
  2. Sarsim simulation: If there is a **$SIMULATION** entry in the script file with the given image filename as output name, it will use the specified parameters.
  3. Custom: A number of parameters have to be specified for custom files:
     - Header: Header size in bytes, data in header will be ignored.
     - Width: Width of image in pixel (corresponds to slant range)
     - Height: Height of image in pixel (corresponds to azimuth range)
     - Bytes per Value: Number of bytes for each pixel. Note: If samples are complex this will be the number of bytes for both I and Q values together.
     - Offset: The offset which will be subtracted from each sample (to get negative numbers). For example single byte values have a range of 0 to 255. An offset of 127 will map this to -127 to $+128$.

- Data Type: Either magnitude (single values) or complex. Complex numbers need to have the I value first, i.e. IQIQ.
- Byte Order: Specify if least-significant byte comes first or not.

After successfully loading the file, the window will look similar to the one shown in Figure 4.14.

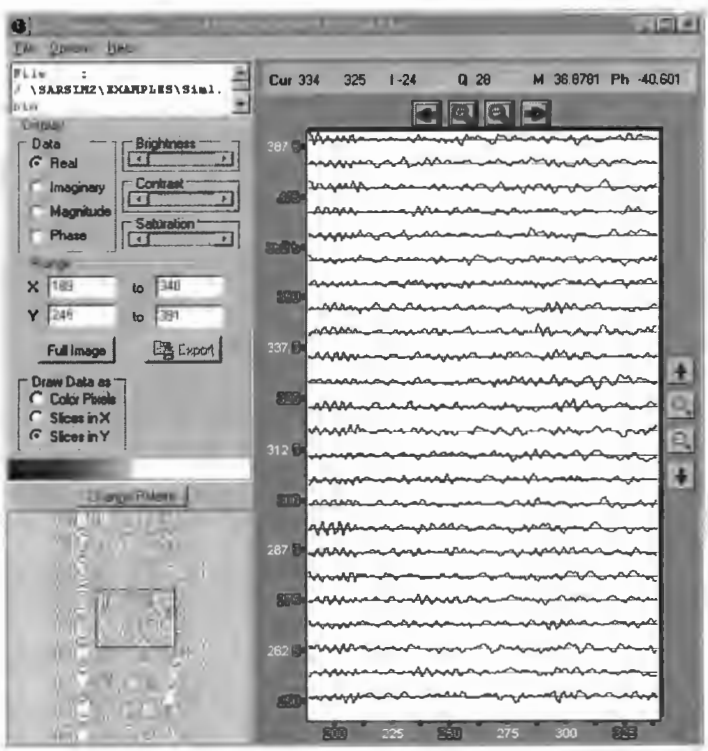

Figure 4.14: The image viewer window (showing slices in azimuth)

On the left, bottom side the complete image is shown. A white rectangle marks the zoomed section, shown on the right. The data can be displayed as:

- Colour pixels: Each value corresponds to a certain colour (see colour palette)
- Slices in x: This will show the data as graphs, ranging from top to bottom.

- Slices in y: This will show the data as graphs, ranging from left to right.

4.8 Looking at Script Files

Script files are used to describe the complete geometry of the simulation and they also store the parameters of files written to disk. Sometimes it is useful to quickly take a look at this script file. It can be displayed by selecting **Display / Script File**. The file displayed will be identical to the file which will be stored on disk after selecting **File / Save** on the main window. An example is shown in Figure 4.15.

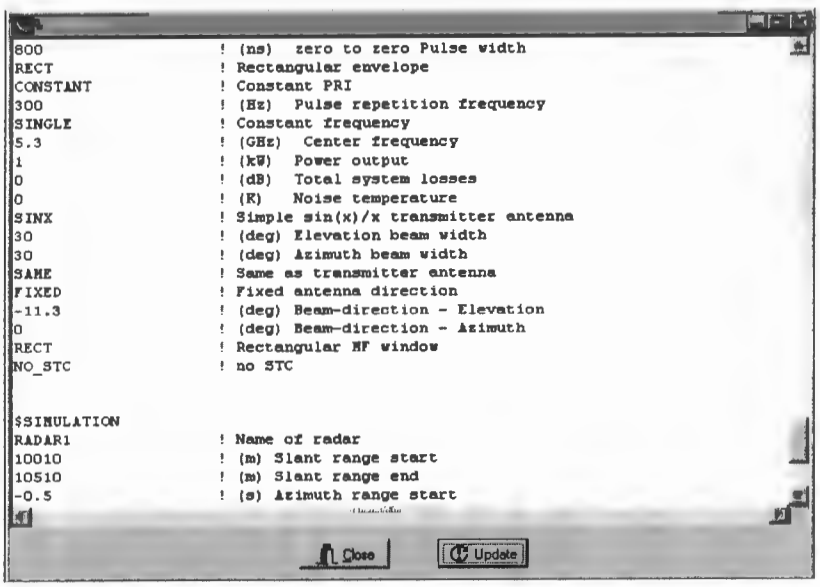

Figure 4.15: The script file viewer

4.9 Changing Focus

It is possible to change the focus point of the main screen from the origin of the "Earth" coordinate system to somewhere else. Either a point other than the

Pulse No.	Pulse transmittion time (s)	PT1 Distance (m)	PT1 Radial Velocity (m/s)	PT2 Distance (m)	PT2 Radial Velocity (m/s)	PT3 etc
0	1.200	32.323	2.312	34.411	1.212	etc.
1	1.202	32.343	2.323	34.422	1.222	etc.
2	1.202	32.363	2.334	34.433	1.232	etc.

Table 4.1: Distance and radial velocity for each point target

origin can be selected or the focus can follow a different platform. This is useful if one needs to see how an object moves seen relative to some platform.

4.10 Investigating the Geometry for every Pulse

By selecting **Geometry** on the main menu it is possible to have a look at the internal variables used for each simulation. This data can also be saved in text format, so that it can be imported into other data analysis programs. It is also a very useful "debugging" tool.

The purpose of this command is to show the relative locations and distances between the radar and all given point targets for each pulse for the given simulation period. An example will clarify this. Suppose you want to model your own (very complicated) pulse shape and want to use Sarsim only to do the geometry calculations, i.e. distance and radial velocity for each point target seen by the radar, this is the tool to use. The output format will be a table of the general form shown in Table 4.1

Note that in Table 4.1 only a small subset of the variables have been shown. A detailed list of all variables is given below.

- Pulse No.: This gives each transmitted pulse a unique number. Pulse 0 (zero) will always be sent at simulation time $t = 0$, the next pulse will be 1 etc. If a simulation extends over negative time, the pulse numbers will be negative.

- Pulse transmitting time (seconds): Displays the exact time when the pulse is transmitted (for example pulse 0 will be transmitted at time $t = 0$, pulse 1 at time $t =$ PRI, 2 at $t = 2 \cdot$ PRI, and so on, assuming constant PRIs).
- Pulse frequency (GHz): Displays the exact frequency for each pulse given in Gigahertz.
- Platform position x, y, z (m): Displays the exact position in meters for each defined platform at the instance of time where the pulse was sent. The coordinates are relative to the world coordinate system. Note that a radar will always reside on the origin of the platform, so the given position will be identical to the position of a radar (if any) situated on this specific platform.
- Platform velocity x, y, z (m/s): Displays the exact velocity in meters per second for each defined platform at the instance of time where the pulse was sent. The velocity vector is relative to the world coordinate system.
- Platform rotation x, y, z (degrees): Displays the exact rotation of the platform around the 3 axes at the instance of time when the pulse was sent.

4.11 Help

A slightly modified version of this entire dissertation is available under **Help / Contents**.

Chapter 5

Script Files

There are only 4 commands used in the script files, namely:

1. **$TARGET**
2. **$PLATFORM**
3. **$RADAR**
4. **$SIMULATION**

After each of these commands, a series of parameters need to be given. They can be separated by spaces or commas. The number of parameters depend on the parameters themselves and can vary considerably. This approach has been taken to provide flexibility if the descriptions are complex, and to provide better readability if the descriptions are simple. Comments can be made with an "!" (exclamation mark); all text behind it will be ignored until the next line break. The parameters follow the input dialogs closely, so if there should be any doubts it might be helpful to look at the corresponding dialog.

The order in which the objects are defined in a script file is important. They should be in the following order: platforms, targets, radars and then simulations.

5.1 User-defined functions

The flowcharts for all 4 functions will be shown. Some of them contain a block called "Array Definition." These blocks contain data on user-defined functions and the flow diagram is shown below in Figure 5.1. Up to three functions can be defined in one block, for example x-, y- and z-coordinates versus time. For each function an interpolation method needs to be specified (cubic, matched-filter or linear).

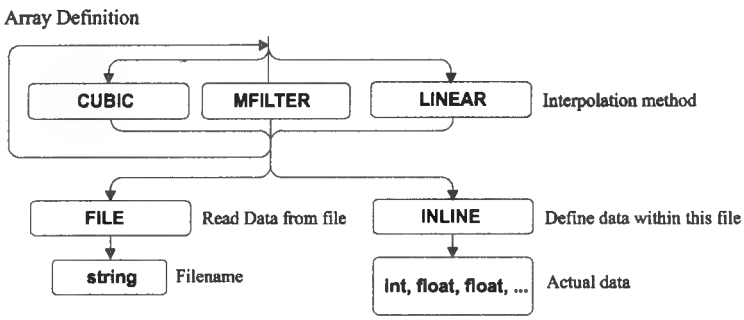

Figure 5.1: Flow diagram for user-defined function block

User-defined functions can be defined either inline or via an external data file. Inline means that the actual values for the function(s) are included in the parameter list of the command (compact solution). This is useful if there are only a small number of samples. With a larger number of samples it is more practical to use external data files. In this case it is only necessary to specify the filename. The files should be in the same directory as the script file. Note that the filename string should not contain commas or spaces. The structure of the data (either inline or in a file) depends on the functions to be defined but are of the following pattern:

```
[Number of samples for function 1], [X1], [Y1], [X2], [Y2], [X3], [Y3] ....[Xn][Yn]
[Number of samples for function 2], [X1], [Y1], [X2], [Y2], [X3], [Y3] ....[Xn][Yn]
[Number of samples for function 3], [X1], [Y1], [X2], [Y2], [X3], [Y3] ....[Xn][Yn]
```

First the number of samples is given, then the x- and y-coordinates for all the samples are given, then it starts again with the next function (if there is one).

For example specifying the path of a platform needs 3 functions, one for each coordinate versus time. An example of an "INLINE" user-defined trajectory function is given below (the comments are not necessary):

```
INLINE
2 ! Number of samples for Position X (m) vs Time (s)
-1, -100 1, 100
1 ! Number of samples for Position Y (m) vs Time (s)
0, 10000
1 ! Number of samples for Position Z (m) vs Time (s)
0, 2000
```

The equivalent in a "FILE" would look identical (ignoring the comments):

```
! Data file for : Position X (m) vs Time (s), Position Y (m) vs Time (s), Position Z (m) vs Time (s)
! Structure : [No. Samples] [Time (s) 1] [Position X (m) 1] [Time (s) 2] [Position X (m) 2] ...
! [No. Samples] [Time (s) 1] [Position Y (m) 1] [Time (s) 2] [Position Y (m) 2] ...
! [No. Samples] [Time (s) 1] [Position Z (m) 1] [Time (s) 2] [Position Z (m) 2] ...
2
-1 -100, 1 100
1
0 10000
1
0 2000
```

From the given samples the program will interpolate the requested values as shown in Figure 4.12.

5.2 The $TARGET command

A flowchart of the "$TARGET" command is shown in Figure 5.2.

5.3 The $PLATFORM command

A flowchart of the "$PLATFORM" command is shown in Figure 5.3.

5.4 The $RADAR command

The flowcharts of the "$RADAR" command are shown in Figure 5.4 and in Figure 5.5 (Split up due to size considerations).

5.5 The $SIMULATION command

A flowchart of the "$SIMULATION" command is shown in Figure 5.6.

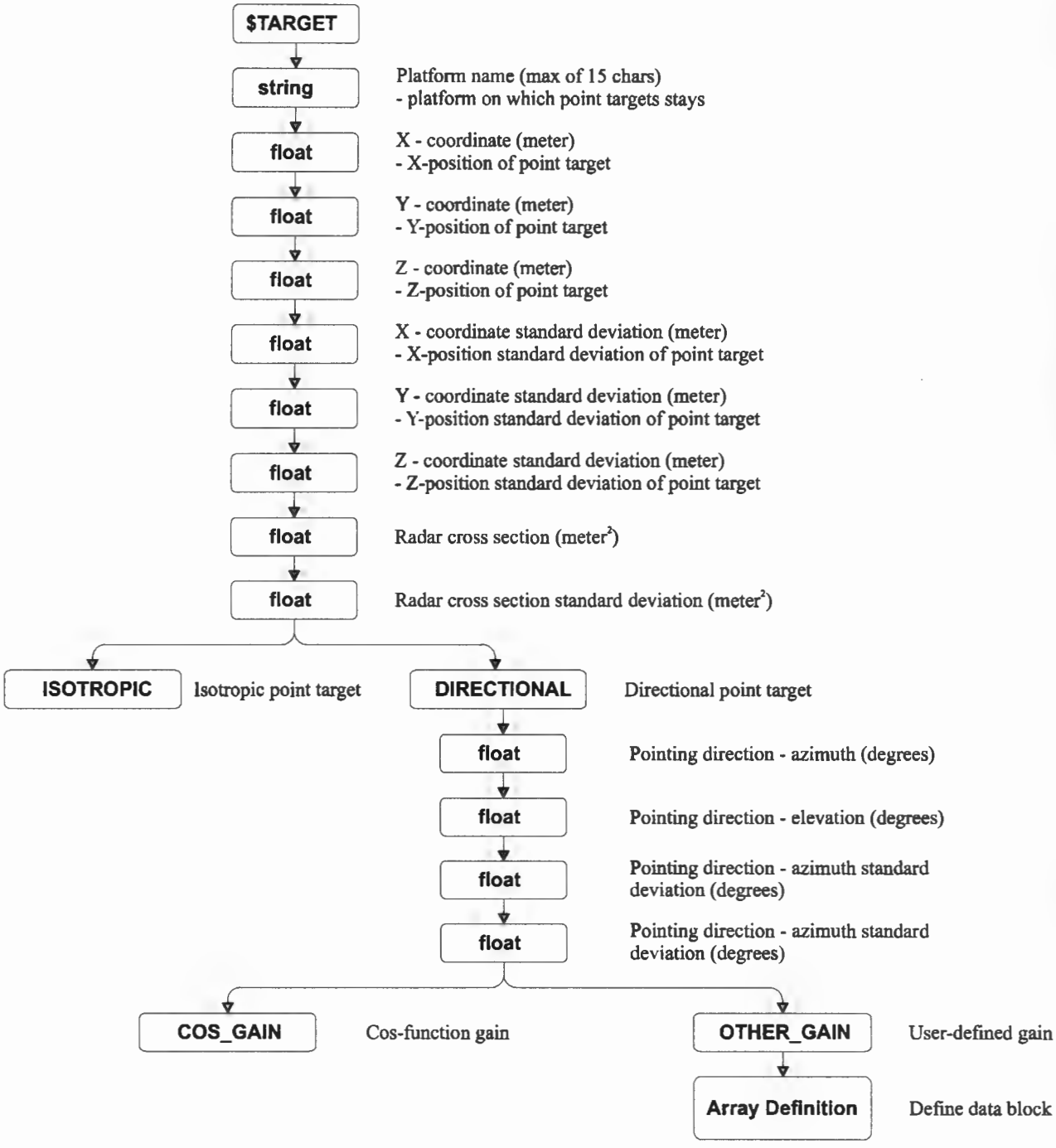

Figure 5.2: Flow diagram of the $TARGET command

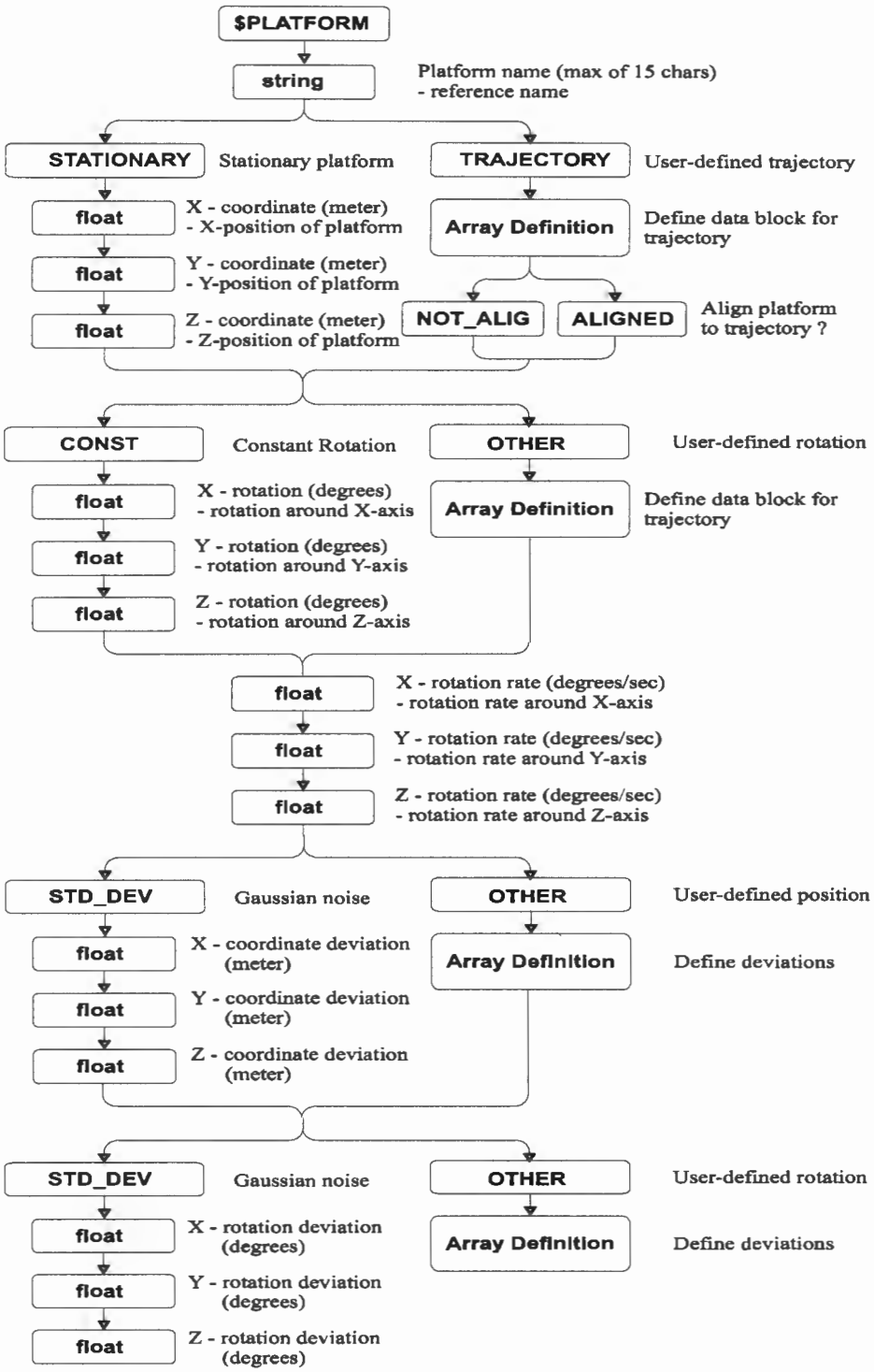

Figure 5.3: Flow diagram of the $PLATFORM command

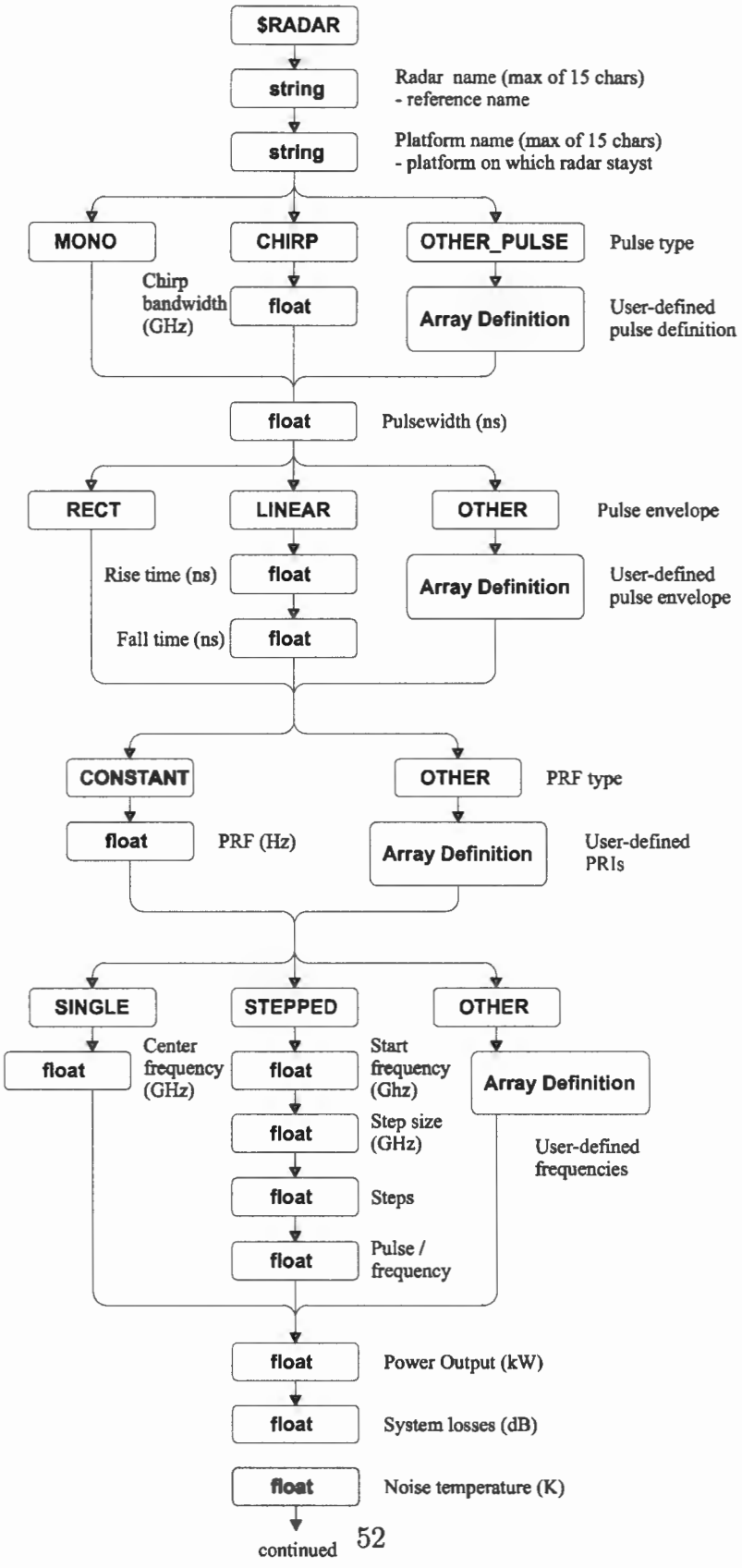

Figure 5.4: Flow diagram of the $RADAR command (1st part)

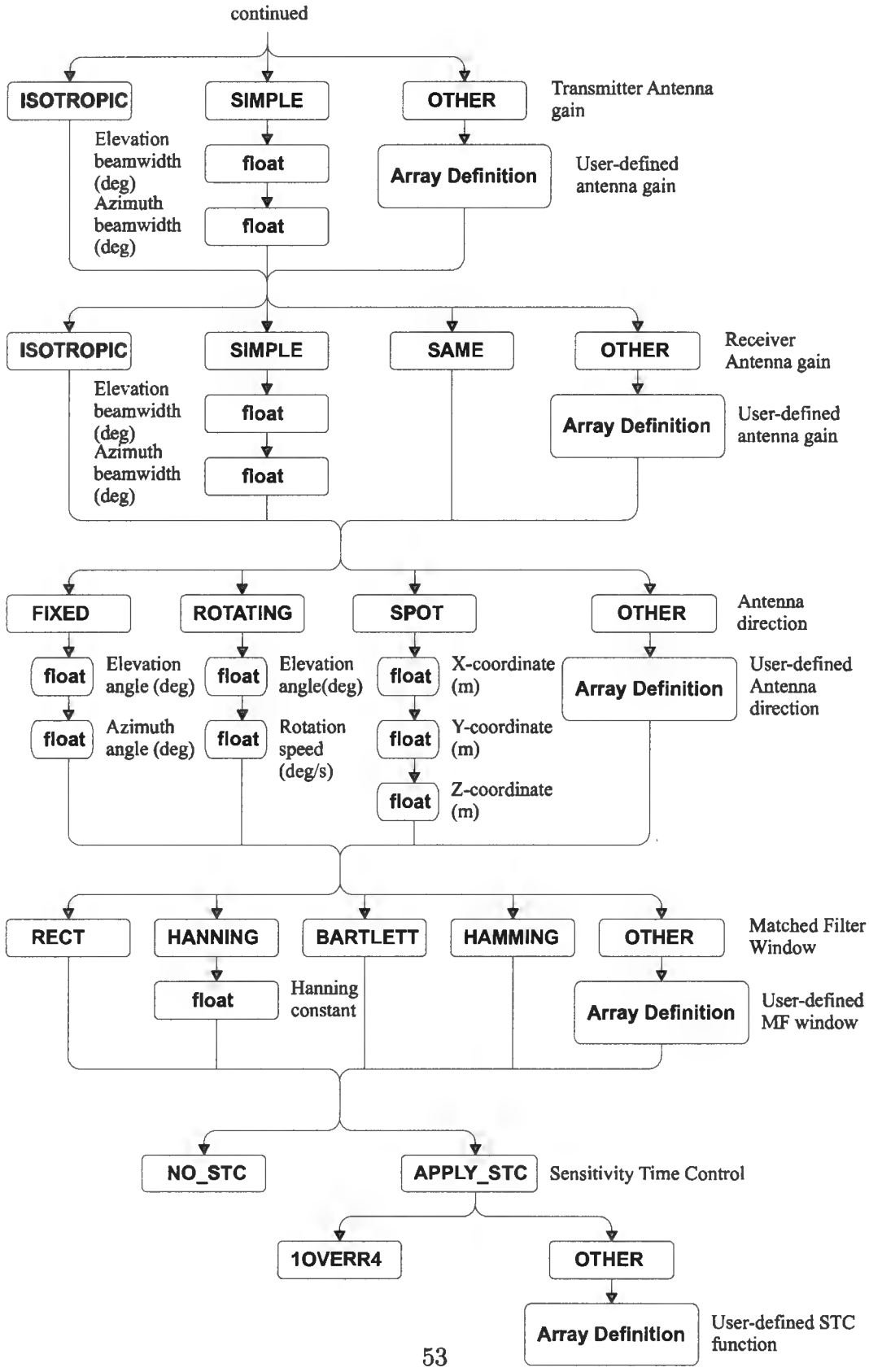

Figure 5.5: Flow diagram of the $RADAR command (2nd part)

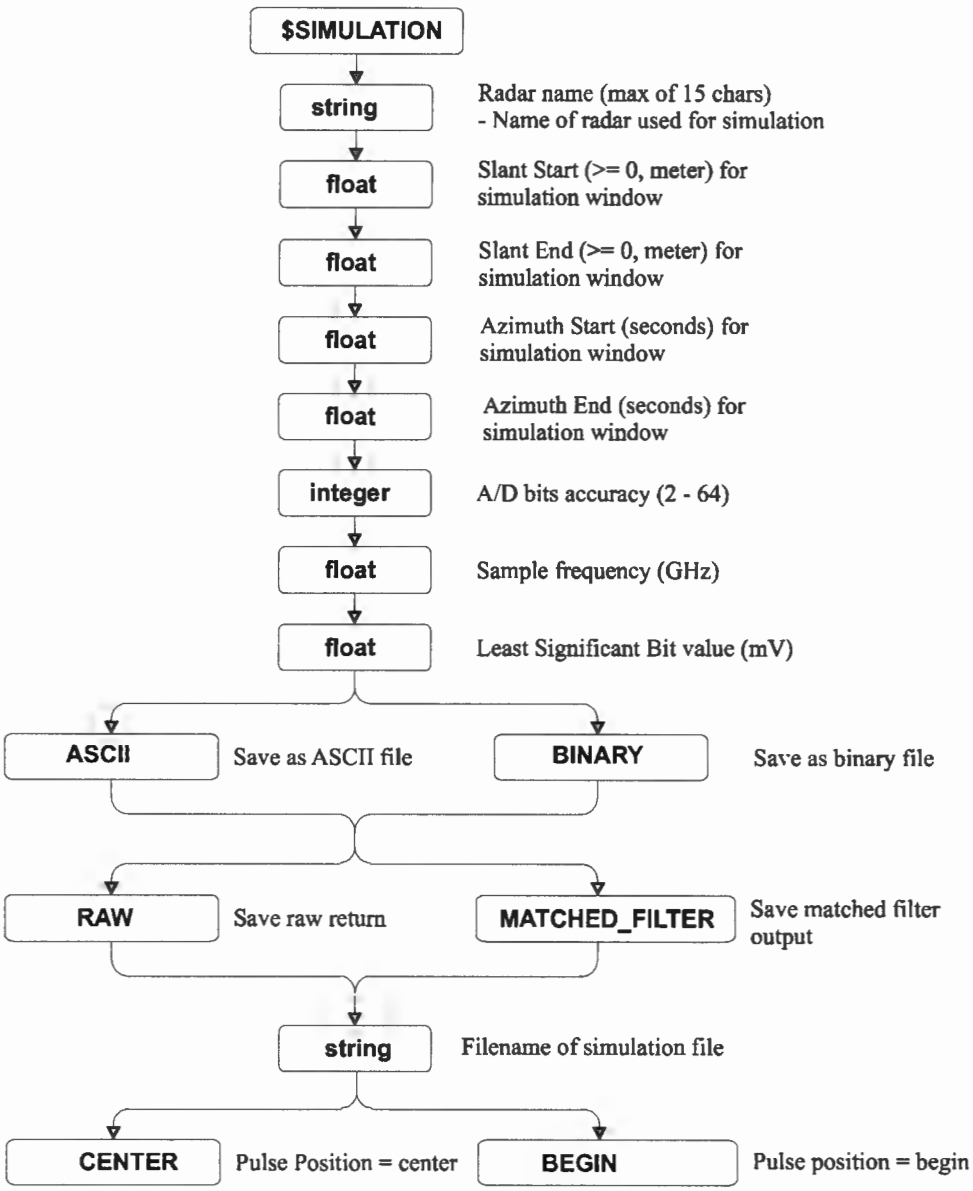

Figure 5.6: Flow diagram of the $SIMULATION command

Chapter 6

Sarsim Internals

In this chapter the formulas behind Sarsim will be explained in detail. The functions which depend on each other will be presented in order of dependence.

6.1 General Function and Variable Definitions

PNo = pulse number (integer $\geqq 0$)

TNo = target number (integer $\geqq 0$)

$PFNo$ = platform number (integer $\geqq 0$)

$floor(x)$ = integer part of number

fmod(x, y) = the remainder f, where $x = ay+f$ for some integer a, and $0 < f < y$ (modulus)

$$\text{RotMatrix(a1,a2,a3)} = \begin{bmatrix} \cos(a2)\cos(a3) & -\cos(a2)\sin(a3) & \sin(a2) \\ \cos(a1)\sin(a3)+\sin(a1)\sin(a2)\cos(a3) & \cos(a1)\sin(a3)+\sin(a1)\sin(a2)\cos(a3) & -\sin(a1)\cos(a2) \\ \sin(a1)\sin(a3)-\cos(a1)\sin(a2)\cos(a3) & \sin(a1)\cos(a3)+\cos(a1)\sin(a2)\sin(a3) & \cos(a1)\cos(a2) \end{bmatrix}$$

6.2 Pulse Calculations

Let us assume we want to calculate the return for the following window in azimuth and slant range as shown in Figure 6.1.

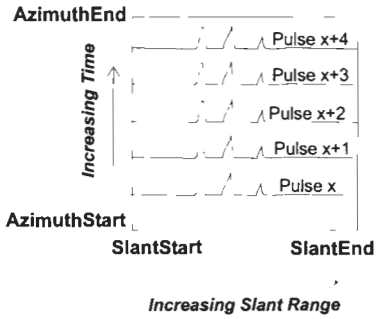

Figure 6.1: Simulation window

The variables *AzimuthStart* and *AzimuthEnd* are given in seconds, while *SlantStart* and *SlantEnd* are given in meters.

6.2.1 FindPulseSendTime Function

This function is defined in `engine.cpp`. It returns the time (in seconds) at which the given pulse is sent (i.e. the beginning of the pulse). Pulse 0 is always sent at time $t = 0$. For a constant PRF the function is :

$$PulseSendTime = \frac{PulseNo}{\text{PRF}} \tag{6.1}$$

For user-defined PRIs, the function is more complicated. The slightly modified (for clarity reasons) source code is given here:

```
long i,FracPRINo;
double frac, ipart, FracPRI;
double SumPRI=0; // sum of all defined PRI's
// find sum of all PRI's defined (i.e. PRI0+PRI1+PRI2 etc.)
for (i=0;i<n;i++)
SumPRI += PRIArray[i];
// find out how many complete PRI cycles there are in the given
// PulseNo (1 cycle = sum of all defined PRI's)
frac = modf(double(PulseNo)/double(n), &ipart);
```

```
// calculate the number of remaining PRI's
FracPRINo = long(round(frac * double(n)));
// if PulseNo <0 goto the next lower integral time (ipart -= 1)
// and add positive PRI's from there
if (FracPRINo < 0)
FracPRINo += n;
ipart -= 1;
FracPRI = 0;
for (i=0;i<FracPRINo;i++)
FracPRI += PRIArray[i];
return (ipart * SumPRI + FracPRI);
```

6.2.2 FindPulsesInRange Function

First the exact time when each pulse is sent out needs to be calculated. This is important for calculating the relative distances between the radar and the point targets. Pulses are numbered such that pulse 0 will be sent at simulation time $t = 0$, pulse 1 will be sent at time $t = 0 + \text{PRI}[0]$, pulse 2 will be sent at time $t = 0 + \text{PRI}[0] + \text{PRI}[1]$ etc. Negative times (and pulse numbers) are possible, for example pulse -1 will be sent at time $t = 0 - \text{PRI}[n-1]$. In Sarsim it is possible to define the interval between any two pulses. Let us assume this data is given in the array called $PRIArray[x]$, where x ranges from 0 to $(n-1)$, where n is the number of defined PRIs. The array wraps around such that $\text{PRI}[x] = PRIArray\,[\text{modulus}(x, n)]$. Note that all times are given in seconds. Figure 6.2 will clarify what was explained above.

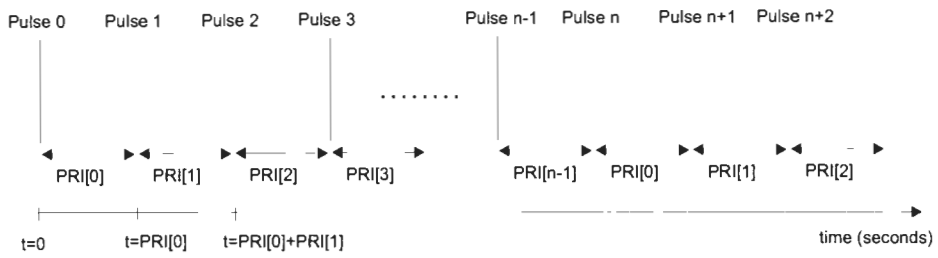

Figure 6.2: The "PulseSendTime"

There are two cases:

1. PRF is constant:

$$FirstPulse = \text{ceil}\,(AzimuthStart \cdot \text{PRF}) \tag{6.2}$$

$$LastPulse = \text{floor}\,(AzimuthEnd \cdot \text{PRF}) \tag{6.3}$$

2. User-defined PRIs: The function is more complicated. The slightly modified (for clarity reasons) source code is given here:

```
long i;
// user defined PRI
double SumPRI=0; // sum of all defined PRI's
// find sum of all PRI's defined (i.e. PRI0+PRI1+PRI2 etc.)
for (i=0;i<n;i++)
SumPRI += PRIArray[i];
// estimate what number the first pulse will have
FirstPulse = (floor((TimeStart / SumPRI)+ROUNDERROR)*n)-1;
// and now find the exact one
while (FindPulseSendTime(FirstPulse) < TimeStart)
{
FirstPulse++;
}
LastPulse = (floor((TimeEnd / SumPRI)+ROUNDERROR)*n)-1;
while (FindPulseSendTime(LastPulse) < TimeEnd)
}
LastPulse++;
}
// overshot by one, so subtract one again
if (*LastPulse > 0) (*LastPulse)--;
```

6.2.3 FindPlatformPosition

This function finds the platform position at a give time t.

6.2.4 FindPlatformVelocity

This function finds the platform velocity at a give time t.

6.2.5 FindPlatformRotation

This function finds the platform rotation at a give time t.

6.2.6 The frequency of each pulse

Single frequency case:

$$PulseFreq[PNo] = \text{specified in RADAR dialog 1} \tag{6.4}$$

Stepped frequency case:

$$\begin{aligned} PulseFreq[PNo] &= StartFreq \\ &\quad + floor\,(fmod\left(\frac{PNo}{FreqSteps \cdot PulsesPerFreq}, 1\right) \\ &\quad \cdot FreqSteps + ROUNDERROR) \cdot StepSize \end{aligned} \tag{6.5}$$

Note that $ROUNDERROR$ (a small number) needed to be included to overcome rounding problems, for example sometimes $\frac{12}{6}$ would give 1.999999etc which is incorrectly rounded to 1.

User-defined case:

$$PulseFreq[PNo] = DataArray[fmod[PNo], arraysize] \tag{6.6}$$

6.2.7 The time when each pulse is transmitted

Constant PRF:

$$PulseSendTime[PNo] = \frac{PNo}{PRF} \tag{6.7}$$

User-defined PRIs:

$$PulseSendTime[PNo] = \text{see source code} \tag{6.8}$$

6.2.8 Range delay

$$RangeDelay[TNo][PNo] = \frac{2 \cdot TargetDist}{\text{LIGHT\_SPEED} - TargetRadialVel[TNo][PNo]} \tag{6.9}$$

6.2.9 CalcGeometry Function

This function calculates the signal amplitude and range delay for the return of all pulses and point targets of interest contained in the simulation window.

- Find *Firstpulse* and *LastPulse* by using function $FindPulsesInRange(AzimuthStart, AzimuthEnd)$
- Calculate the number of pulses which need to be calculated:

$$PulseNo = (LastPulse - FirstPulse) + 1 \tag{6.10}$$

- Create array which contains the times for which each pulse is sent:

$$PulseSendTime[PNo] = FindPulseSendTime(PNo + FirstPulse) \tag{6.11}$$

- Create 2D arrays which contain the position, velocity and rotation for all platforms for all pulses:

$$\begin{aligned} \text{PlatformPos[PFNo][PNo]} &= \text{FindPlatformPosition(PFNo,PulseSendTime[PNo])} \\ \text{PlatformVel[PFNo][PNo]} &= \text{FindPlatformVelocity(PFNo,PulseSendTime[PNo])} \\ \text{PlatformRot[PFNo][PNo]} &= \text{FindPlatformRotation(PFNo,PulseSendTime[PNo])} \end{aligned} \tag{6.12}$$

- Compute the return gain factor independent of time:

$$GainFactor = \frac{\frac{c}{Radar->StartFreq} \cdot \sqrt{Radar->PowerOutput}}{(4 \cdot \pi)^{1.5} \cdot \sqrt{Radar->Losses}} \tag{6.13}$$

with Radar->StartFreq being the centre frequency given in Hz, Radar->PowerOutput given in Watt and Radar->Losses given as a unitless factor.

- For a sinusoidal antenna the gain for a certain offset angle can be calculated using function *SinAntennaGain*.

- The combined antenna gain is given by $\sqrt{AntennaGainT \cdot AntennaGainR}$ where $AntennaGainT$ is transmitter antenna gain and $AntennaGainR$ is the receiver antenna gain which can be calculated from:

$$\mathrm{AntennaGain'T} = \begin{cases} 1 \text{ for isotropic antennas} \\ \mathrm{SinAntennaGain(OffsetAzi, AziBeamWidthT)} \\ \cdot \mathrm{SinAntennaGain(OffsetElev, ElevBeamWidthT)} \text{ for sinusoidal antennas} \end{cases} \tag{6.14}$$

- The return amplitude can be calculated with the formula

$$ReturnAmp[TNo][PNo] = \frac{GainFactor \cdot AntennaGain \cdot \sqrt{RCS}}{TargetDist^2} \tag{6.15}$$

6.2.10 CalcOnePulse Function

Chirp Modulation

The chirp rate (= DelaySlope, Hz/s) is calculated as follows:

$$DelaySlope = \frac{ChirpBandWidth}{PulseWidth} \tag{6.16}$$

Monochrome Modulation

For monochrome pulses the DelaySlope would be zero.

$$DelaySlope = 0 \tag{6.17}$$

The range delay is the time (in seconds) needed for the pulse travelling forth and back to the point target and is given by:

$$RangeDelay = \frac{2 \cdot d}{c} \tag{6.18}$$

where d is the distance to the target in meters and c is the speed of light (= 299792500 m/s).

The position of the pulse relative to the target is specified by

$$PulseCentre = \begin{cases} 0 & \text{if the pulse is at the beginning of the point target} \\ \frac{PulseWidth}{2} & \text{if the pulse is at the centre of the point target} \end{cases} \tag{6.19}$$

and explained in Figure 6.3. For real radars you would receive the pulse "after" the point target location, however for simulations it is sometimes more convenient to have the point target in the centre. All it really means is that the output array will be shifted by half a pulsewidth in range.

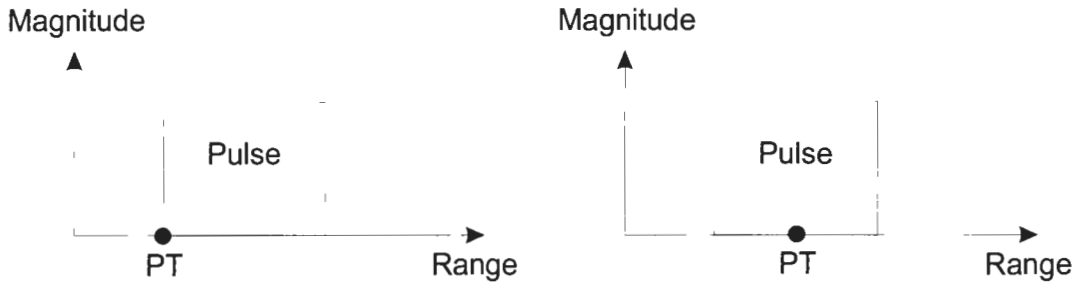

Figure 6.3: Point target at the start of the pulse and at the centre of the pulse

For the following calculations it is assumed that the point target position corresponds to the beginning of the pulse. A certain time range needs to be sampled denoted by $SlantStartTime$ and $SlantStartEnd$, both measured in seconds. The sampling frequency is f_s. The pulse is situated from $RangeDelay$ to $RangeDelay + Pulsewidth$, both variables given in seconds. The time axis is shifted by an amount of $\left(RangeDelay - \frac{1}{2}Pulsewidth\right)$ as shown in Figure 6.4, such that the variable t goes from $-\frac{1}{2}Pulsewidth$ to $+\frac{1}{2}Pulsewidth$ over the pulse range.

The frequency modulation (chirp rate for chirp pulses) can be calculated as:

$$Mod = DelaySlope \cdot \frac{1}{2} \cdot t^2 \tag{6.20}$$

For monochrome pulses this value would be zero (DelaySlope = 0).

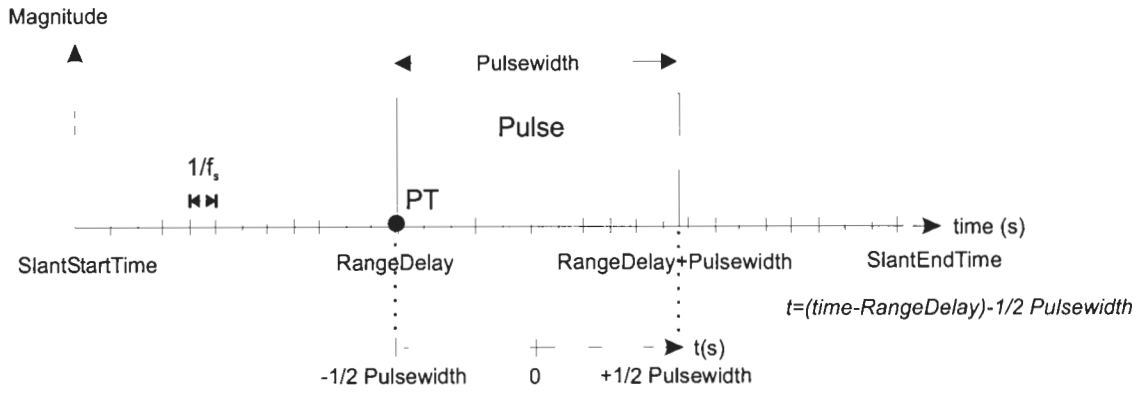

Figure 6.4: Positioning of pulse in range

The instantaneous frequency of the returned pulse at some point t ($t = 0$ at beginning of pulse as shown in Figure 6.4) is:

$$Freq(t) = \underbrace{DelaySlope \cdot t}_{\text{Modulation}} \underbrace{- \frac{2 \cdot RadVel}{c} \cdot (Freq + DelaySlope \cdot t)}_{\text{Frequency shift due to Doppler}} \tag{6.21}$$

The phase of the returned pulse is the integral with respect to time of the frequency and can be calculated as follows:

$$Phase(t) = 2 \cdot \pi \cdot (\underbrace{Mod}_{\text{modulation}} \underbrace{- (PulseFreq \cdot RangeDelay)}_{\text{phase shift due to range}} \underbrace{- \frac{2 \cdot RadVel}{c} \cdot (Freq \cdot t + Mod)}_{\text{phase shift due to Doppler}} \tag{6.22}$$

$Freq$ stands for the $PulseFreq[PNo][TNo]$ and specifies the centre frequency of that specific pulse sent out, $RangeDelay$ is defined above and $RadVel$ specifies the radial velocity of the target.

From here the inphase and quadrature values are calculated simply by:

$$I(t) = ReturnAmp \cdot \cos(Phase(t)) \tag{6.23}$$

$$Q(t) = ReturnAmp \cdot \sin(Phase(t)) \tag{6.24}$$

Chapter 7

Example Files

Various examples for different applications will be described in this chapter. The script files for all examples are available in the **EXAMPLES** subdirectory.

7.1 A typical C-band SAR Application

For this SAR application a plane is flying in a straight line at some height over an area, which has to be investigated. The radar is mounted on the plane with a sideways looking beam. The simulation file can be found as "**sar_r.scr**" in the "**examples**" directory.

This example will model the following scenario:

A plane is flying at 200 m/s parallel to the y-axis at a height of 2000 meters and 10 km horizontal distance from the area of interest as shown in Figure 7.1.

A sidelooking radar is mounted on the plane, the depression angle of the beam is 11.3 degrees. There have been 24 point targets placed in an "R" shape, extending over 200 x 400 m as shown in Figure 7.2.

The radar parameters are given in Table 7.1.

First the PLANE platform is created. This is done by creating a new platform

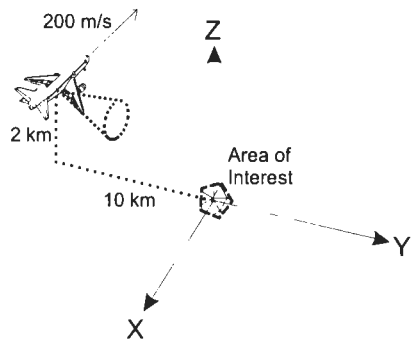

Figure 7.1: Geometry setup

pulse waveform	chirp pulse
B	100 MHz
f_c	5.3 GHz
T_p	800 ns
PRF	300 Hz
P_{tx}	1 kW
losses	none
noise	none
beamwidth	30 degrees

Table 7.1: Radar Parameters

with the name "PLANE" and defining a trajectory for it. The x-coordinate decreases linearly by 200 m/s, the y-coordinate stays constant at -10000 m and the z-coordinate stays constant at 2000 m. Therefore we define the following:

- Position $x = 100$ m at time $= 0$ s
- Position $x = -100$ m at time $= 1$ s
- Position $y = -10000$ m at time $= 0$ s (one point suffices)
- Position $z = 2000$ m at time $= 0$ s (one point suffices)

The plane will move 200 meters in 1 second, being closest to the earth origin at 0.5 seconds.

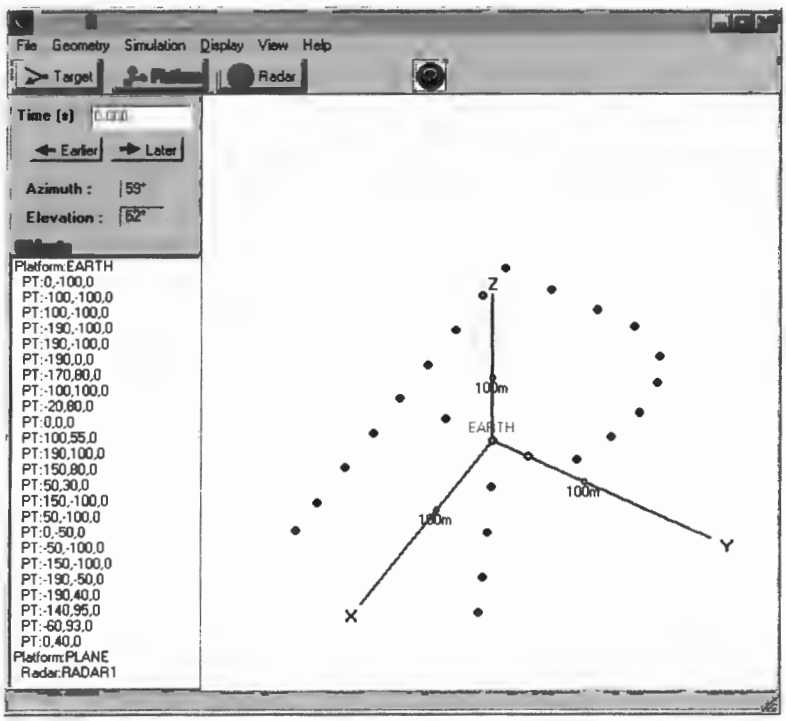

Figure 7.2: Point target setup

The next step is to create the radar on the PLANE platform, the given parameters need to be entered in the dialogs.

The last step would be to create the point target setup as shown in Figure 7.2. For this setup a simulation has been stored already and can be recalled by selecting **Simulation / Previous**. The window shown in Figure 7.3 will appear.

Because the return signals of the point targets interfere with each other, the original R-shape cannot be seen. However after saving this simulation window and performing range and azimuth compression on it (in this specific example the Chirp Scaling Algorithm has been applied), the image can be restored as shown in Figure 7.4. Some artifacts appear on the upper and lower border due to the fact that the azimuth range should have covered more time.

The simulation script file will look as shown in Figure 7.5.

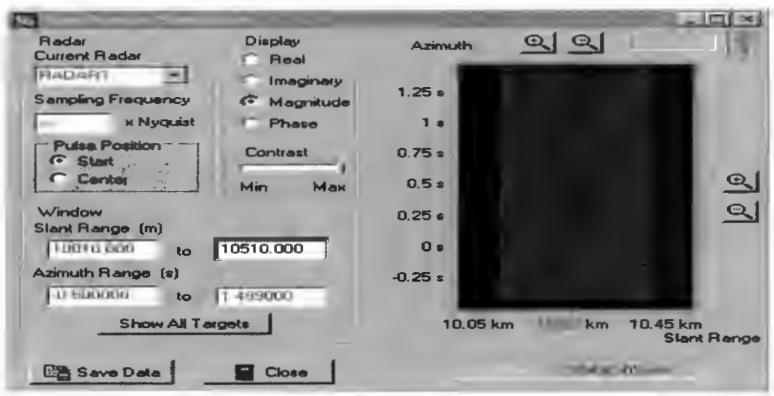

Figure 7.3: Simulation window

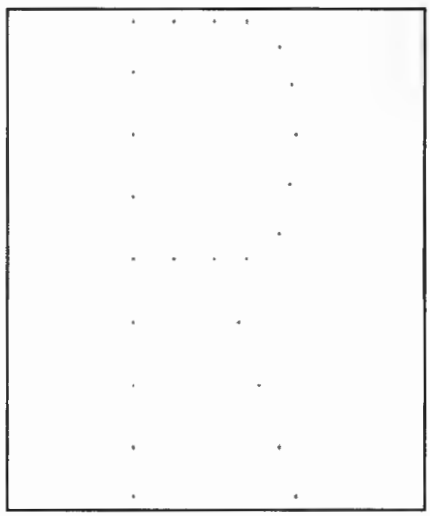

Figure 7.4: Image after processing

```
! Radar Simulator SARSIM version B28 (c) 1997 R.L.


$PLATFORM  EARTH
STATIONARY
0          ! (m) X-Position
0          ! (m) Y-Position
0          ! (m) Z-Position
0          ! (deg) X-axis Rotation
0          ! (deg) Y-axis Rotation
0          ! (deg) Z-axis Rotation
STDDEV
0          ! (m) X-Position standard deviation
0          ! (m) Y-Position standard deviation
0          ! (m) Z-Position standard deviation
STDDEV
0          ! (deg) X-axis Rotation standard deviation
0          ! (deg) Y-axis Rotation standard deviation
0          ! (deg) Z-axis Rotation standard deviation

$PLATFORM  PLANE
TRAJECTORY
CUBIC        .
CUBIC
CUBIC
INLINE
2          ! Number of samples for Position X (m) vs Time (s)
0, 100  1, -100
1          ! Number of samples for Position Y (m) vs Time (s)
0, -10000
1          ! Number of samples for Position Z (m) vs Time (s)
0, 2000
NOT_ALIGNED
0          ! (deg) X-axis Rotation
0          ! (deg) Y-axis Rotation
0          ! (deg) Z-axis Rotation
STDDEV
0          ! (m) X-Position standard deviation
0          ! (m) Y-Position standard deviation
0          ! (m) Z-Position standard deviation
STDDEV
0          ! (deg) X-axis Rotation standard deviation
0          ! (deg) Y-axis Rotation standard deviation
0          ! (deg) Z-axis Rotation standard deviation
```

```
$TARGET EARTH  ! Platform
0, -100, 0   ! Position X,Y,Z (m)
0, 0, 0   ! Position standard deviation X,Y,Z (m)
1, 0  ! Radar cross section (m*m), RCS std. dev. (m*m)
ISOTROPIC
(Many more PTs omitted here .....)

$RADAR
RADAR1       ! Name of radar
PLANE        ! Platform name of radar
CHIRP
0.1          ! (GHz) Chirp bandwidth
800          ! (ns)  zero to zero Pulse width
RECT         ! Rectangular envelope
CONSTANT     ! Constant PRI
300          ! (Hz)  Pulse repetition frequency
SINGLE       ! Constant frequency
5.3          ! (GHz)  Center frequency
1            ! (kW)  Power output
0            ! (dB)  Total system losses
0            ! (K)   Noise temperature
SINX         ! Simple sin(x)/x transmitter antenna
30           ! (deg) Elevation beam width
30           ! (deg) Azimuth beam width
SAME         ! Same as transmitter antenna
FIXED        ! Fixed antenna direction
-11.3        ! (deg) Beam-direction - Elevation
0            ! (deg) Beam-direction - Azimuth
RECT         ! Rectangular MF window
NO_STC       ! no STC


$SIMULATION
RADAR1       ! Name of radar
10010        ! (m) Slant range start
10510        ! (m) Slant range end
-0.5         ! (s) Azimuth range start
1.499        ! (s) Azimuth range end
8            ! A/D bit accuracy
0.15         ! (GHz) Sample frequency
1.68257E-07  ! (mV) Least Significant Bit value
ASCII        ! Save file format
RAW          ! Processing
SIM1.ASC     ! File Name of output file
             ! Size : 600 (Azimuth) x 250 (Slant Range)
BEGIN        ! Point Target position relative to pulse
```

Figure 7.5: Shortened script file for C-band SAR example

Chapter 8

Conclusions

The radar simulator described in this dissertation has proven to be extremely useful in providing simulated data used for SAR processing applications, for stepped-frequency processing applications and also for aircraft and ship recognition analysis. The graphical user-interface makes it easy to use, since one can see and analyse the returned waveforms before writing them to disk. Furthermore, due to efficient programming, it is extremely fast, since only the appropriate computations (corresponding to the current screen resolution) are executed.

8.1 Future work

As with any software developed, there is always scope for improvement. The following items are suggestions for future work:

- Implementation of surfaces. At the moment only point target simulations are possible, however for proper SAR and Interferometric SAR applications it would be of great benefit to implement proper surface simulations.
- Bandlimiting of returned (and transmitted) waveforms to reduce aliasing. Theoretical radar pulses are never truly bandlimited, due to the sharp

cutoffs at the signal boundaries. This can lead to aliasing problems, even if the returned waveform is sampled well above the Nyquist rate.

- Looking at a JAVA implementation, in order to be able to distribute a large simulation over a number of machines, and to make the simulator platform independent, i.e. the graphical interface in particular.
- Including satellite ephemeris platform trajectories, to facilitate spaceborne SAR simulation.
- Separate motion scenario generator, to provide motion files for radar and targets in complex airborne radar environments.

Appendix A

Examples for Sarsim II

A.1 Introduction

The discrete analysis of radar signals, which are nothing else than electromagnetic waves sent in a specific pattern, demands very powerful computers. To undertake the task of writing a simulator is always a compromise between accuracy and execution time. Many simulators have been written to model the return of complex objects like aircraft or ships by approximating them by a large number of surfaces. Although these simulations (taking multiple reflections into account) give fairly reasonable approximations for simple bodies, calculation times are usually measured in hours.

A short description of the methods used in this case will be given:

The simulator used here models the reflective surface by means of very small perfect 'mirrors' called point targets. In theory, everything could be modelled by an extremely large number of these point targets and the reflected signal would just be a linear combination of these. However for most simulation purposes a small number of point targets will be used to get a clear picture of what the effects of changing certain conditions entail.

A radar is considered to be the transmitter and receiver of electromagnetic pulses.

Usually short bursts of energy are transmitted in regular intervals at a certain carrier frequency. Sometimes the pulse is not modulated (monochrome pulse), however more often it is. A very common linear frequency modulation is called 'chirp'. The purpose of modulation is to increase the pulse length (less peak power required) without sacrificing resolution. After reception the signals are mixed to baseband, in other words the carrier frequency component is removed. For the purpose of this simulator, the output will always be at baseband, due to the simple reason that sampling at typical carrier frequencies would require an enormous number of samples. (In practice the signal would be mixed to some intermediate frequency). The simulator therefore can only do simulations which require baseband signals.

As the reflected pulse will be a replica of the transmitted pulse (although scaled in amplitude and shifted in phase), the simulator does a rather simple task. It calculates the distance and the relative radial velocity of all given point targets and constructs a sampled return array by adding 'pulses' at calculated positions within the array. Power losses introduced by range and antenna gains are taken into account. Effectively the simulator takes the burden of calculating the geometry setup at any given time. For very simple simulations, this might be trivial, however for non-linear moving targets or special antenna gain patterns for example, a computer simulation is the only practical option.

The following pages will describe simple scenarios which will be simulated. The results will be verified by applicable formulas.

The next section will describe the use of Excel, Matlab and IDL to view the simulation results saved in the ASCII files.

A.1.1 Simulation results on Excel

When the simulation is run, the raw return will display the actual waveform pulse as shown in Figure A.1. To see the individual pulses, edit the range and azimuth values in the window box. In this example, the slant range is from 0 to 2000 m and the azimuth range is from 0 to 1.5 ms.

To view the graph on Microsoft Excel, follow the steps below:

- Save file in ASCII format (for this example, the LSB was $1.67572 \cdot 10^{-6}$ mV).
- Open file in Excel. The 'Text Import Wizard' window will appear.
- Select option 'delimited' in the dialog box.
- Click 'Next' and select 'Tab' and 'Space' in the 'Delimiters' window.
- Click 'Finish'.

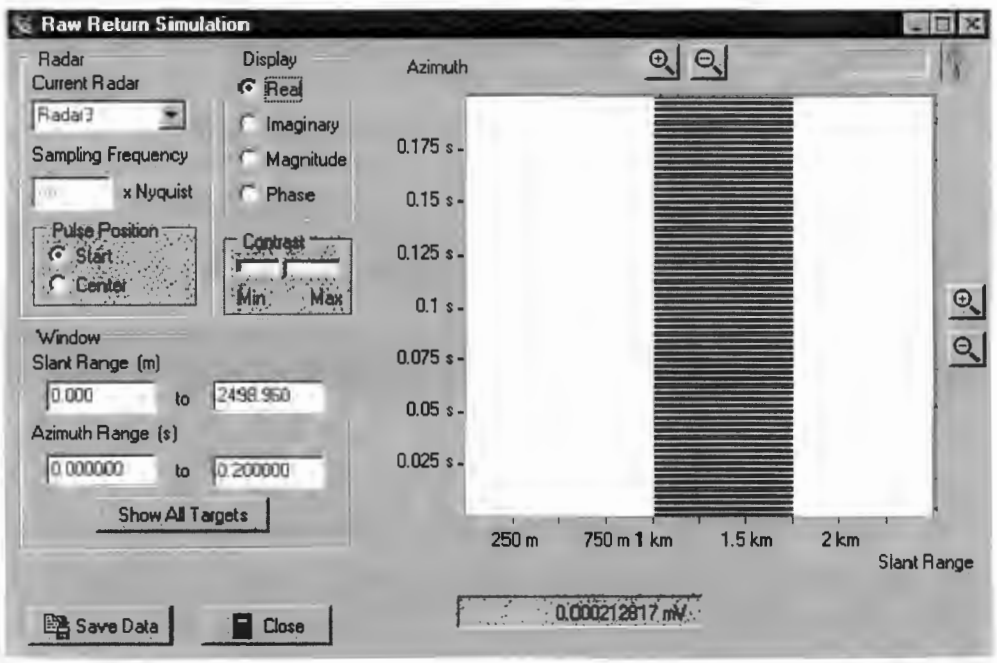

Figure A.1: Raw Return Window

Figure A.2 shows some of the values obtained from the simulation.

	A	B	C	D	E	F
1	I	Q	N	Slant Range	Magnitude	Phase
2	0	0	1	37.47	0	0
3	0	0	2	74.95	0	0
4	0	0	3	112.42	0	0
5	0	0	4	149.90	0	0
6	0	0	26	974.33	0	0
7	-25	-125	28	1049.27	127.47549	-101.31
8	-25	-125	29	1086.75	127.47549	-101.31
9	-25	-125	30	1124.22	127.47549	-101.31
10	-25	-125	44	1648.86	127.47549	-101.31
11	-25	-125	45	1686.33	127.47549	-101.31
12	-25	-125	46	1723.81	127.47549	-101.31
13	0	0	48	1798.76	0	0
14	0	0	49	1836.23	0	0
15	0	0	50	1873.70	0	0
16	0	0	51	1911.18	0	0
17	0	0	52	1948.65	0	0
18	0	0	53	1986.13	0	0

Figure A.2: Excel Worksheet for Simulation

The spreadsheet will give a series of I and Q values. The number of values will depend on the sampling frequency and the slant range used. For this example, the sampling frequency was 20 times the Nyquist rate (0.2 MHz) and the slant range was from 0 to 2000 m. The number of values obtained for one single pulse was 53. The magnitude, phase and slant range can be calculated from these values as follows:

$$\text{Magnitude} = \sqrt{\mathrm{I}^2 + \mathrm{Q}^2} \tag{A.1}$$

$$\text{Phase} = \left[\text{atan}\left(\frac{\mathrm{Q}}{\mathrm{I}}\right) - \frac{180}{\pi}\right] \pm n\pi \text{ where n is an integer} \tag{A.2}$$

$$\text{Slant range} = \frac{\mathrm{c}}{\mathrm{f}_s} \tag{A.3}$$

Figure A.3 displays the graphs of Magnitude and Phase against Slant Range.

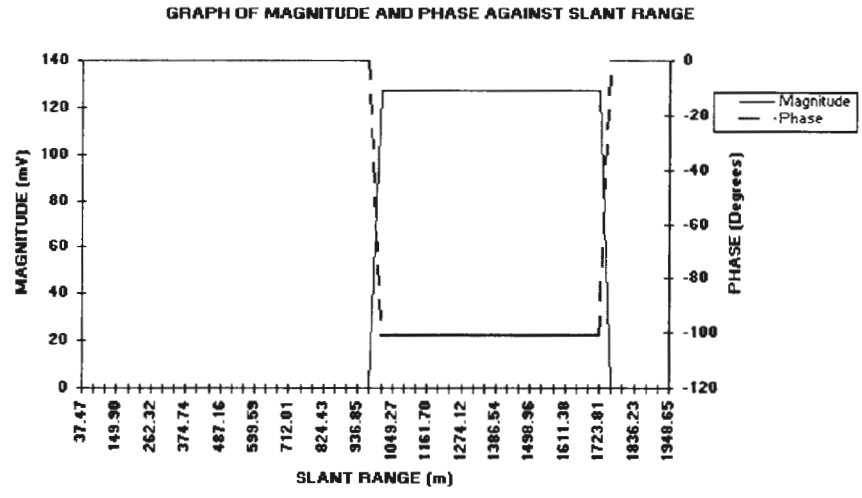

Figure A.3: Graph of Magnitude and Phase against Slant Range

A.1.2 Simulation Results in Matlab

The program to view the magnitude of the pulse is given below:

```
load F:\filename.asc   %loads the I and Q values from the ASCII file into a
                       %matrix called 'filename'

X = filename (:,1);    %X is a vector representing the I values of 'filename'
Y = filename (:,2);    %Y is the vector representing the Q values of
                       %'filename'

Z = X + i*Y;           %Z is a vector of complex numbers
B = abs(Z);            %B is a vector representing the magnitude of Z
k = length(B);         %assigns the length of the vector to a variable k
```

```
d=1:k;

W=(299792500/8E6)*d;  %converts from number of pulses to range in meters

plot(B(1:k))          %plots the graph of B against W

title('Graph of Magnitude against Slant Range')  %title of graph
xlabel('Slant Range (m)')                        %labelling of x-axis
ylabel('Magnitude (mV)')                         %labelling of y-axis
```

The graph is shown in Figure A.4.

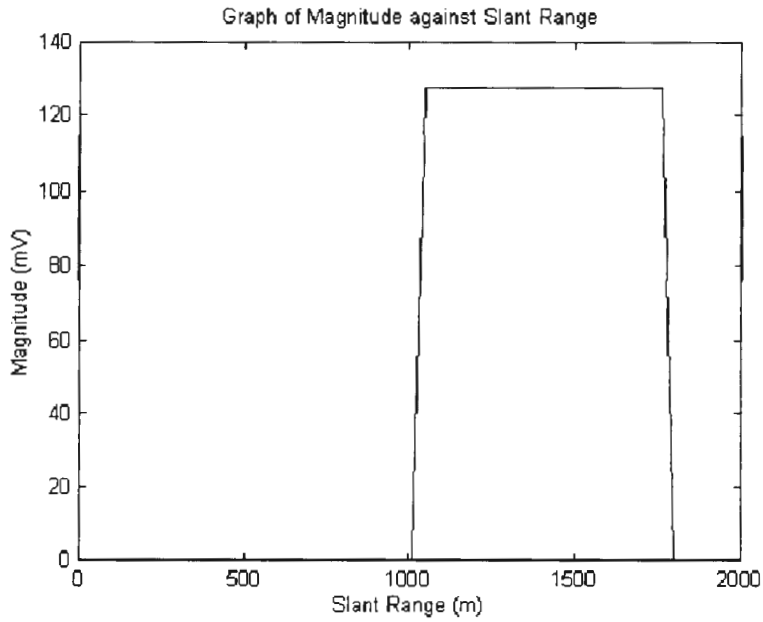

Figure A.4: Graph of Magnitude against Slant Range

A.1.3 Simulation Results in IDL

The program to view the Magnitude and the Phase of the pulse is given below:

```
filename = 'sim11.asc'
arrlength = 53              ;assigns the number of values in the file to a
                            ;variable called arrlength
A = complexarr(arrlength)   ;creates an array of complex numbers of size 53
X = fltarr(54)              ;creates an array of floating-point numbers
line = ''                   ;creates a string called 'line'
realpart = 0.0              ;initializes the real part of the complex array
imagpart = 0.0              ;initializes the imaginary part of the complex arra

Openr, 1, filename          ;opens the file 'filename' for reading only

For i = 0,(arrlength - 1) Do Begin
readf, 1, line              ;reads each line of the file filename into the
                            ;string called 'line'
reads, line, realpart, imagpart ;the components of the line is read into
                                ;variables 'realpart' and 'imagpart'
A(i) = complex(realpart, imagpart);stores the values into the complex array
Endfor
Close, 1                        ;closes the file 'filename'

B = abs(A)                      ;returns the magnitude
C = atan(imaginary(A), float(A));returns the phase
D = C*57.3                      ;converts the phase from radians to degrees

For j = 0,(arrlength) Do Begin
X(j) = (299792500/8E6)*j        ;converts the number of pulses into
                                ;range(m)and stores them into an array
Endfor
```

```
;Routine to plot the graphs of Magnitude and Phase against Slant Range
;In order to plot the two graphs, which have different min values and
;max values, the Oplot and Yrange procedures are used.

Plot,X,B,Yrange = [-120,150], $ ;plots the graph of Magnitude against
                                ;Slant Range with the range of the Y-axis
                                ;between (-120,150)

Title='Graph of Magnitude and Phase against Slant Range',$ ;title of the graph
Xtitle = 'Slant Range (m)'                    ;title for the X-axis
Ytitle = 'Magnitude (mV) and Phase (Degrees) ;title for the Y-axis

Oplot,X,D                       ;plots the graph of Phase against Slant
                                ;Range on the same axes as the previous graph
End
```

The graph is shown in Figure A.5.

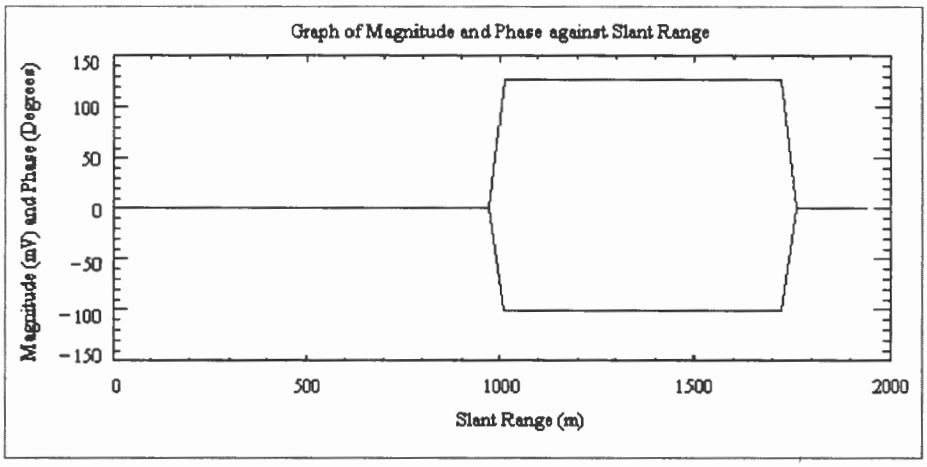

Figure A.5: Graph of Magnitude and Phase against Slant Range

A.2 A stationary point target

The purpose of this simulation is to demonstrate:

- the phase shift of the reflected pulse due to range delay,
- power loss due to distance,
- and pulse compression by using matched filters.

A.2.1 Setup

This very simple example consists of a radar located at the origin and a stationary point target on the x-axis at a distance of 1 kilometre..

The radar parameters are as follows:

- Radar Position: origin of the 'Earth' coordinate system
- Pulse Type: monochrome
- Pulsewidth: $T_p = 5000$ ns
- Pulse Repetition Frequency: 1000 Hz
- Carrier Frequency: 1 GHz
- Output Power: 1 kW
- Target cross-section: 1 m^2

Figure A.6 shows the geometrical setup of the radar and the point target.

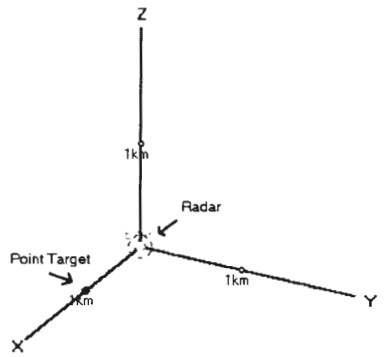

Figure A.6: Geometry Setup

A.2.2 Results

A typical simulation window is shown in Figure A.7. Slow time (azimuth) is presented on the y-axis (increasing from bottom to top), while fast time (shown as slant range) is displayed on the x-axis.

Several facts can be observed from Figure A.7. The target is stationary, positioned at 1 km as specified. The pulse extends over 750 m in range. The actual pulse length is twice that ($c \cdot T_P = 1500$ m), the discrepancy is due to the fact that the range and not the range delay is shown on the x-axis. In other words the time scale has been divided by a factor of two, as the pulse has to travel back and forth. The pulse repetition interval (PRI) is 1 ms.

After saving the data the phase and magnitude of the pulse can be determined. Usually data is sampled with 8 bit or 12 bit accuracy. For this example 8 bits will be used. The least significant bit value is chosen such that the signal makes optimum use of the dynamic range without saturation occurring. (In reality there will always be some saturation occurring due to noise.)

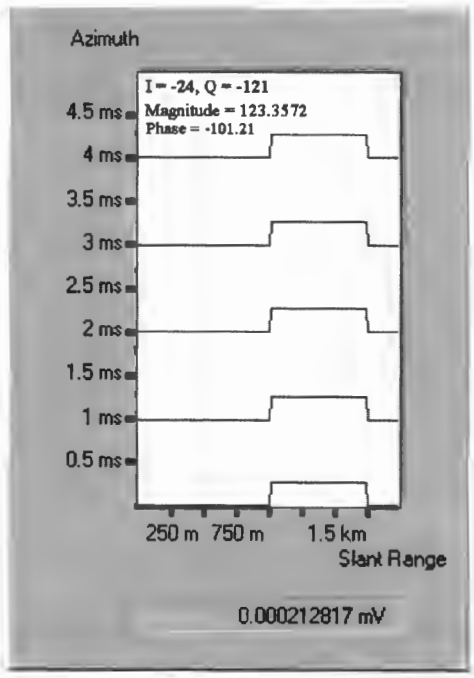

Figure A.7: Magnitude against Slant Range for Raw Return

For the given data the following values were obtained:

Least significant bit value $= 1.6757 \cdot 10^{-6}$ mV

$I = -24 = (-24 - 127) \cdot 1.6757 \cdot 10^{-6}$ mV = -253 μV (in-phase component)

$Q = -121 = (-121 - 127) \cdot 1.6757 \cdot 10^{-6}$ mV = -415.6 μV (quadrature component)

If the data is saved in binary format an offset of $(2^{n-1} - 1)$ is added, where n specifies the number of bits used.

A.2.3 Analysis

Sampling spacing

In this specific simulation the data has been sampled at 20 times the Nyquist rate, therefore $f_s = 20 \cdot 2 \cdot \frac{1}{T_P} = 8$ MHz.

For the given range window of 2 km shown in Figure A.7, 53 samples were taken. Each sample corresponds to $\frac{c}{2f_s}$ meters in range, where $c = 299792500\ \frac{\text{m}}{\text{s}}$.

Power loss

Although the return pulse is of the same shape as the radiated one, the magnitude is significantly less than the radiated pulse. The received power is given by the following equation:

$$\mathrm{P}_{received} = \frac{\mathrm{P}_{transmitted} \cdot G^2 \cdot \lambda^2 \cdot \sigma}{(4 \cdot \pi)^3 \cdot R^4} \tag{A.4}$$

where

- σ = target cross-section
- λ = wavelength of pulse
- R = distance to target
- G = Antenna gain

For the given scenario the received power would be:

$$\begin{aligned} \mathrm{P}_{received} &= \frac{\mathrm{P}_{transmitted} \cdot G^2 \cdot \lambda^2 \cdot \sigma}{(4 \cdot \pi)^3 \cdot R^4} && \text{(A.5)} \\ &= \frac{1000 \cdot 1 \cdot 0.2998^2 \cdot 1}{(4 \cdot \pi)^3 \cdot 1000^4} && \text{(A.6)} \\ &= 45.293269 \cdot 10^{-15}\ \text{W} && \text{(A.7)} \end{aligned}$$

which gives a corresponding amplitude (assuming the ubiquitous 1 Ω resistor) of:

$$\text{Magnitude} = \sqrt{\mathrm{P}_{received}} = 2.1282 \cdot 10^{-4}\ \text{mV} \tag{A.8}$$

The simulated value gives a magnitude of $\sqrt{I^2+Q^2} = 2.128 \cdot 10^{-4}$ mV, which agrees with the above result.

Phase shift

The phase of the return pulse is determined by the wavelength and the distance to the radar and is given by:

$$\text{Phase} = 2 \cdot \pi \cdot \left(-2 \cdot d \cdot \frac{f}{c}\right) \tag{A.9}$$

where c is the velocity of light and d is the radar-target distance. As the phase wraps around every 2π radians, only the remainder can be determined. For a point target at a distance of 1000 m:

$$\begin{aligned}
\text{Phase} &= \text{mod}\left[2 \cdot \pi \cdot \left(-2 \cdot d \cdot \frac{f}{c}\right), 2 \cdot \pi\right] && \text{(A.10)}\\
&= \text{mod}\left[2 \cdot \pi \cdot \left(-2 \cdot 1000 \cdot \frac{10^9}{299792500}\right), 2 \cdot \pi\right] && \text{(A.11)}\\
&= -1.7653 && \text{(A.12)}\\
&= 101.15\,\text{deg} && \text{(A.13)}
\end{aligned}$$

The simulation values give the same value: Phase $= -\frac{\pi}{2} - \arctan(\frac{-24}{-121}) = -101.22\,\text{deg}$. The slight discrepancy is due to the 8 bit quantisation of the samples.

Range Compression

In pulse compression, the return pulse from the raw return is compressed.This is achieved by using a matched filter. The concept of range compression will be explained in more detail in the following section.

The choice of a radar waveform will affect radar performance in terms of range, Doppler, and angle measurements, as well as the system's detection performance. For example, a short monochromatic pulse of timelength t_p will allow a resolution of

$$\mathrm{R}_s = \frac{c \cdot t_p}{2} \tag{A.14}$$

Therefore, targets that are closer in distance than R_s cannot be easily "resolved" or separated, because the returns will overlap. The important parameter for range resolution is bandwidth; thus, a monochrome pulse has an effective bandwidth $\mathrm{B}_p = \frac{1}{t_p}$. The effective bandwidth for a monochrome pulse waveform is the bandwidth of the pulse spectrum. By making a pulse shorter, or the effective bandwidth larger, range resolution is improved. However, the amount of energy contained in the pulse diminishes as the pulse shortens.

Improved detection is achieved by proper filtering or processing of the received signal. The use of a matched filter optimises performance in the presence of white, Gaussian noise. The return pulse is convolved with a conjugate replica of itself. Since the pulse is rectangular in shape, a convolution with itself will give a triangular pulse.

As can be seen from Figure A.8, the resultant pulse is triangular in shape. Since range-compression is computationally intensive, the sampling rate can be set. Using low values for the sampling rate (less than the Nyquist frequency) should be avoided due to aliasing. In this example, the sampling rate used is a hundred (100) times the Nyquist frequency. It is to be noted that neither the magnitude nor the phase changes when the return pulse is range compressed.

The importance of pulse compression will be demonstrated in more detail in the next example.

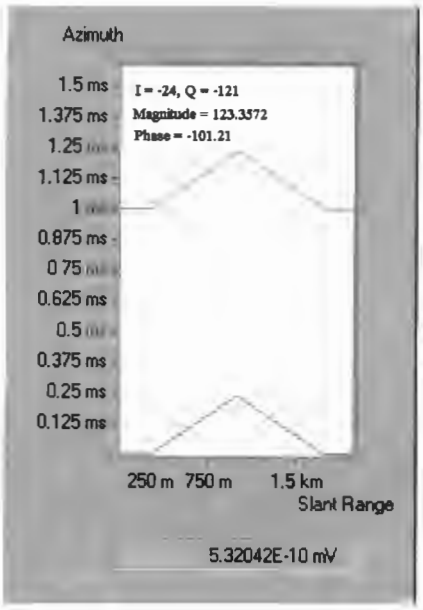

Figure A.8: Magnitude against Slant Range for Range Compression

A.3 A moving point target

The previous simulation demonstrated fixed point targets. For this simulation the target will move relative to the radar. The purpose is to demonstrate:

- moving point targets
- chirp modulation
- matched filters
- range resolution
- Doppler frequency shift
- and the concept of moving platforms.

A platform is a convenient way of grouping targets, which are not moving relative to each other, together. Their position and velocity can then be set by just defining the trajectory of that specific platform.

A.3.1 Setup

The radar parameters are as follows:

- Radar position: origin of the 'Earth' coordinate system
- Pulse Type: chirp
- Bandwidth: 0.1 GHz
- Pulsewidth, T_p: 2500 ns
- Pulse Repetition Frequency: 1 kHz
- Carrier Frequency: 0.5 GHz
- Output power: 1 kW
- Target Cross-section: 1 m^2

There are two targets that are 8 meters apart. They have been grouped on a platform and the parameters for the platform are:

- $x = 300$ m and moving at a speed of 50 ms^{-1} in the x-direction
- $y = 0$
- $z = 0$.

The geometrical setup is given in Figure A.9.

A.3.2 Results

Raw Return

The pulse type used in this example is a chirp pulse. For chirp waveforms, the frequency f_0 is not kept constant throughout the pulse, but is linearly modulated from f_0 to $(f_0 + \Delta f)$. Δf can be positive (up chirp) or negative (down chirp).

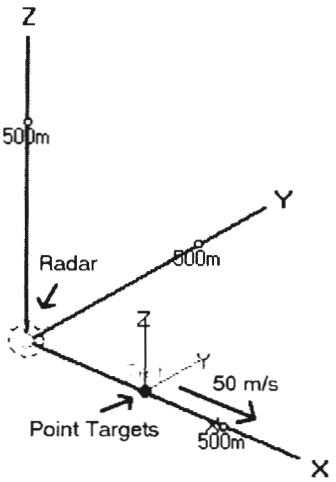

Figure A.9: Geometry Setup

The result is a waveform which has a bandwidth that is independent of the pulse length T_p. Thus, a pulse with large T_p and large B can be constructed. The reason for using a chirp waveform, is to have a high energy pulse which increases the range resolution capability of the radar. The transmit waveform is generated with a frequency versus time domain waveform as shown in Figure A.10.

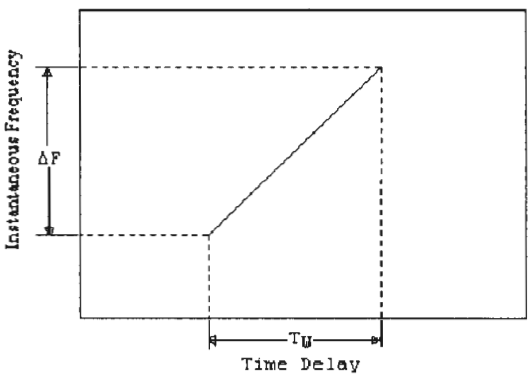

Figure A.10: Frequency against time domain waveform

Figure A.11 shows the raw return for the simulation (only one pulse is shown here for clarity). Due to the fact that there are two targets that are close together, it is very difficult to separate the two targets from each other. The received pulse consists of the return pulse from the two targets interfering with each other.

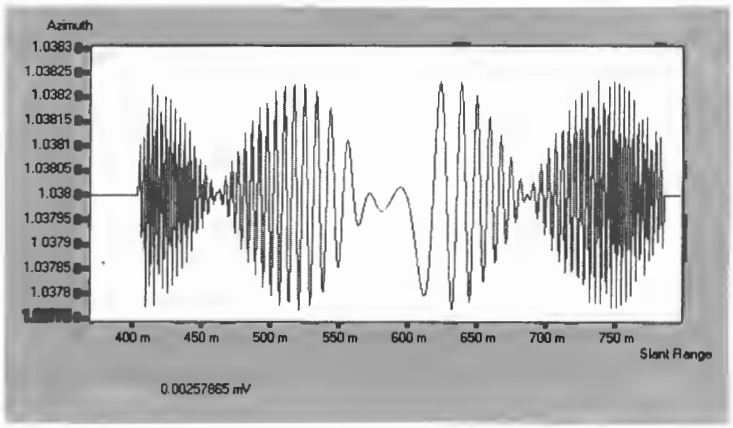

Figure A.11: Simulation Window for Real Part of Raw Return

Range Compression

In order to be able to distinguish between the two targets, range compression is used. This is shown in Figure A.12. From the figure, it can be seen that the first target is at 300 m and the second one is at 308 m.

The theoretical resolution of the radar is determined by the using the following equation:

$$\Delta R_s = \frac{c}{2 \cdot B} \tag{A.15}$$

where B = bandwidth of the chirp pulse.

For the given bandwidth of 100 MHz, the resolution will be 1.5 meters.

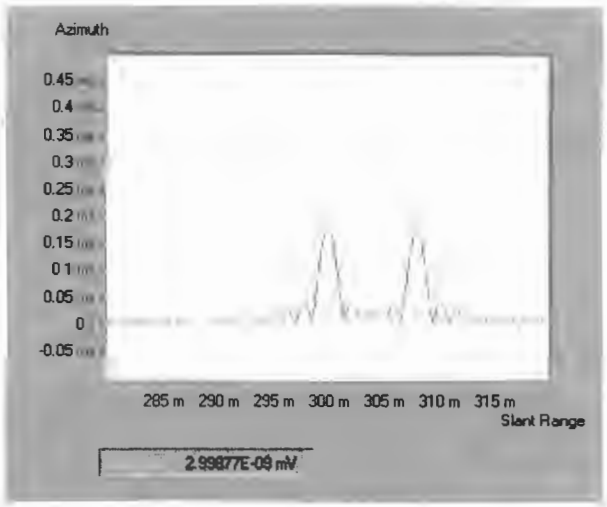

Figure A.12: Magnitude of Signal after Range Compression

Phase Shift

The first return of the first 9 pulses (the first pulse is sent at time $t = 0$ seconds) were sampled (after range compression), and the corresponding phase values are shown in Figure A.13. As the platform is moving into positive x-direction, this will introduce a phase shift of the return array for every pulse. The PRF $= 1$ kHz, therefore the target will move 0.05 m/pulse. This corresponds to a phase shift of $2 \cdot \pi \left(\frac{-2 \cdot d}{\lambda}\right) = -1.0472$ radians, as verified in Figure A.13. Point targets 1 and 2 are positioned at 300 m and 308 m respectively at time $t = 0$ seconds.

A.4 Search Radar

This example consists of a rotating antenna beam with a certain antenna gain pattern an a single point target. A measure of the ability of an antenna to concentrate energy in a particular direction is called the gain. Two different, but related, definitions of antenna gain are the directive gain and the power gain. The directive gain is descriptive of the antenna pattern.

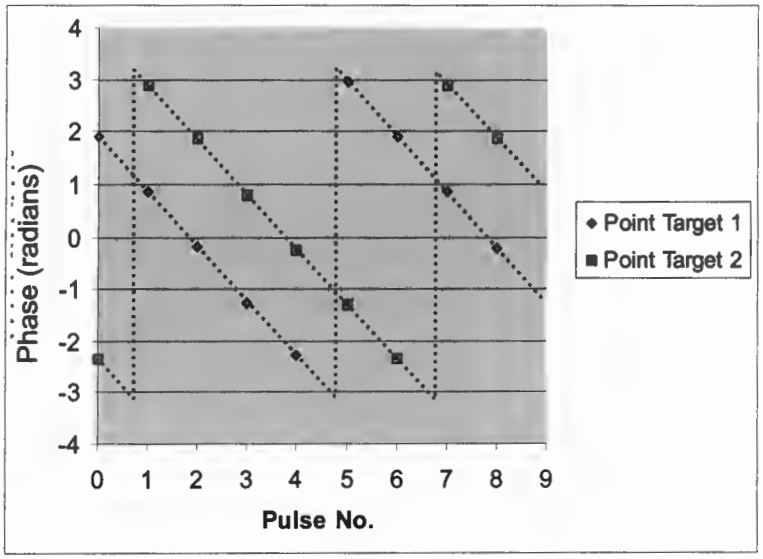

Figure A.13: Phase shift

The directive gain of a transmitting antenna is defined as the ratio

$$G_D = \frac{\text{maximum radiation intensity}}{\text{average radiation intensity}} \tag{A.16}$$

The radiation intensity is the power per unit solid angle in the direction (η, ϵ) and is denoted by $\mathrm{P}(\eta, \epsilon)$. The two-angle coordinates commonly used with ground-based antennas are azimuth angle η and elevation angle ϵ.

A simple approximation to typical antenna gain patterns is the $\frac{\sin(x)}{x}$ shape. This can be analytically calculated by considering the current distributions across a circular aperture. The power radiation pattern is a $\left(\frac{\sin(x)}{x}\right)^2$ pattern. Figure A.14 shows an example for a beamwidth of 10 degrees.

The half-power points would be at an offset of 5 degrees off the beam centre. The amplitude gain is calculated by:

$$\text{Gain} = \frac{\sin(x)}{x}, \; x = 0.88 \cdot \pi \cdot \frac{\sin(\theta)}{\text{Beamwidth}} \tag{A.17}$$

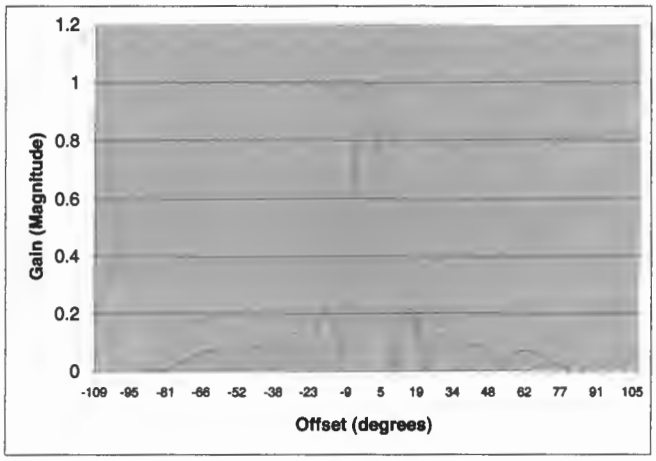

Figure A.14: Magnitude gain pattern for a beamwidth of 10 degrees

A.4.1 Setup

This example consists of a search radar and a stationary point target.The simulation parameters for the radar are as follows:

- Monochrome pulse modulation, pulsewidth 5000 ns
- Pulse repetition frequency: 400 Hz
- Carrier frequency: 1 GHz
- Output power: 1 kW
- $\sin(x)/x$ transmitter antenna gain, elevation beamwidth 30 degrees, azimuth beamwidth 1 degrees
- Rotation rate 24 deg/s, beam elevation angle 30 degrees
- Target offset : $x = 1000$ m, $y = 1000$ m, $z = 816$ m (distance = 1632.74 m)

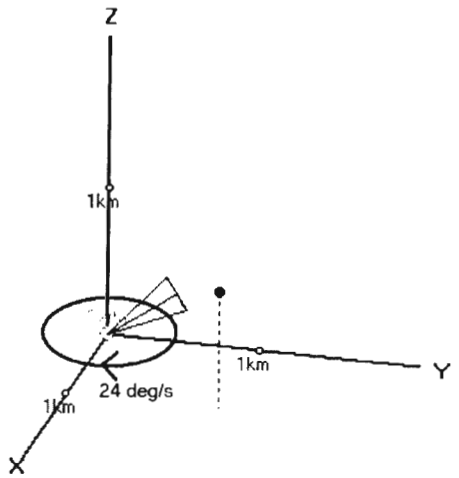

Figure A.15: Geometry setup

A.4.2 Results

The transmitter antenna gain pattern is a simple $\frac{\sin(x)}{x}$, the power radiation pattern of the return pulse is a $\left(\frac{\sin(x)}{x}\right)^2$. As can be seen from Figure A.16, the return pulse has a mainlobe and sidelobes of decreasing magnitude.

Power Loss

The transmitted power is 1 kW and the received power at the beam centre is calculated as follows:

$$\begin{aligned} P_{received} &= \frac{P_{transmitted} \cdot G^2 \cdot \lambda^2 \cdot \sigma}{(4 \cdot \pi)^3 \cdot R^4} && \text{(A.18)} \\ &= \frac{1000 \cdot 1 \cdot 0.2998^2 \cdot 1}{(4 \cdot \pi)^3 \cdot (1632.745)^4} && \text{(A.19)} \\ &= 6.37324 \cdot 10^{-15} \text{ W} && \text{(A.20)} \end{aligned}$$

$$\text{Magnitude} = \sqrt{P_{received}} = 7.9833 \cdot 10^{-5} \text{ mV} \qquad \text{(A.21)}$$

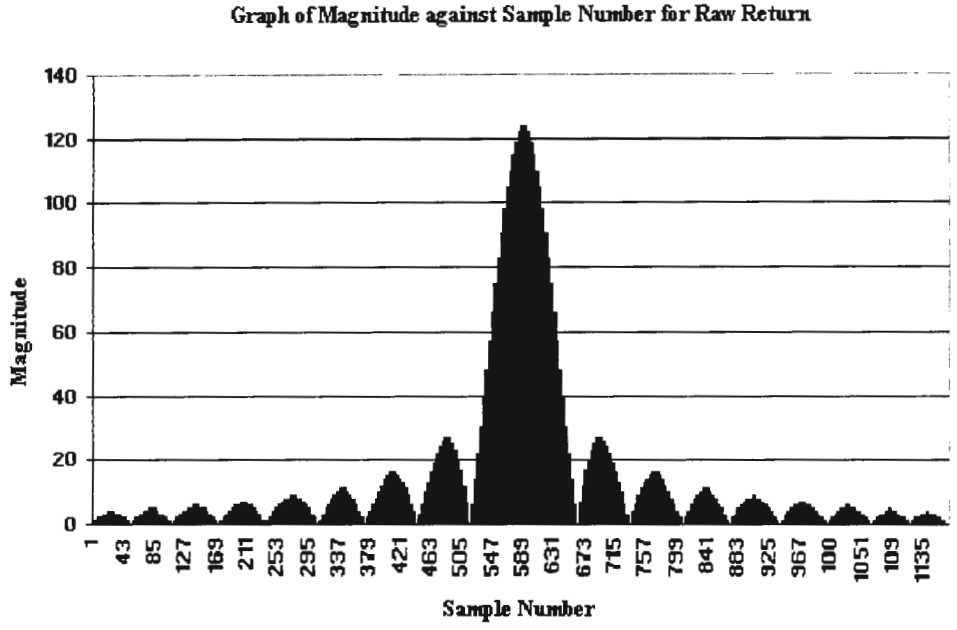

Figure A.16: Graph of Magnitude against Sample Number for Raw Return

The value obtained from the simulation is $7.97376 \cdot 10^{-5}$ mV.

Phase Calculation

The (x, y, z) coordinates for the target are (1000, 1000, 816).

The distance of the target from the radar is $d = 1632.744928$ m, $f = 1$ GHz and $c = 299792500$ ms^{-1}.

$$Phase = \mathrm{mod}\left(2 \cdot \pi \cdot \left(-2 \cdot d \cdot \frac{f}{c}\right), 2 \cdot \pi\right) \tag{A.22}$$

$$\approx -\pi \tag{A.23}$$

The phase is -180 degrees and from the simulation values, it can be seen that the Q values are all zero and the I values are negative.

A.4.3 Range Compression

The range compression result is shown in Figure A.17. This was obtained by convolving the raw return pulse with a conjugate replica of itself.

Resolution

The theoretical resolution of the radar is

$$\Delta R_s = \frac{c}{2 \cdot B} = \frac{299792500 \cdot T_p}{2} = 749.5 \tag{A.24}$$

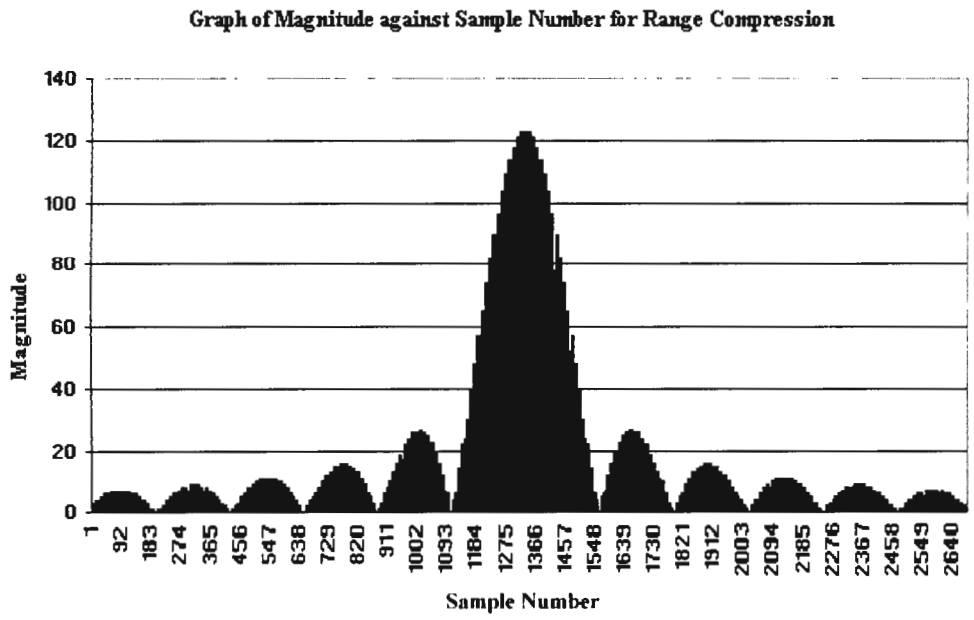

Figure A.17: Magnitude against Sample Number for Range Compression.

The difference between the mainlobe and the first sidelobe levels is 13.5 dB. With filtering, the sidelobes can be reduced at the expense of resolution.

Appendix B

Software Source Code

The Sarsim2 executable and the corresponding source code has been written on a CD and is either attached to this dissertation, or can be obtained from the Radar Remote Sensing Group at UCT.

Bibliography

[1] S. R. J. Axelsson. Frequency and Azimuthal Variations of Radar Cross Section and Their Influence Upon Low-Frequency SAR Imaging. *IEEE Transactions on Geoscience and Remote Sensing*, 33(5):1258–1265, September 1995.

[2] R. H. Clarke and J. Brown. *Diffraction Theory and Antennas.* Ellis Horwood Limited, Chichester, 1980.

[3] J. C. Crespo. The Extended Chirp Scaling Processor for the Experimental SAR System of DLR, E-SAR. In *Proc. European Conference on Synthetic Aperture Radar, EUSAR'96*, pages 353–356, Königswinter, Germany, March 1996. VDE-Verlag GMBH, Berlin and Offenbach.

[4] I. Cumming, F. Wong, and K. Raney. A SAR Processing Algorithm with no Interpolation. In *Proc. IEEE Geosci. Remote Sensing Symp., IGARSS'92*, pages 376–379, Clear Lake, TX, May 1992.

[5] J. C. Curlander and R. N. McDonough. *Synthetic Aperture Radar Systems and Signal Processing.* John Wiley and Sons, New York, Chichester, Brisbane, Toronto, Singapore, 1991.

[6] G. W. Davidson, F. Wong, and I. Cumming. The Effect of Pulse Phase Errors on the Chirp Scaling SAR Processing Algorithm. *IEEE Transactions on Geoscience and Remote Sensing*, 34(2):471–478, March 1996.

[7] C. Elachi. *Spaceborne Radar Remote Sensing: Applications and Techniques.* The Institute of Electrical and Electronics Engineers, Inc., New York, 1988.

[8] J. P. Fitch. *Synthetic Aperture Radar.* Springer-Verlag, New York, 1988.

[9] G. S. Gill. Step Frequency Waveform Design and Processing for Detection of Moving Targets in Clutter. In *Proceedings of the IEEE 1995 International Radar Conference, RADAR'95*, pages 573–578, Alexandria, Virginia, May 1995.

[10] S. A. Hovanessian. *Radar System Design and Analysis.* Artech House, Norwood, MA 02062, 1984.

[11] Y. Huang, Z. Ma, and S. Mao. Stepped-frequency SAR System Design and Signal Processing. In *Proc. European Conference on Synthetic Aperture Radar, EUSAR'96*, pages 565–568, Königswinter, Germany, March 1996. VDE-Verlag GMBH, Berlin and Offenbach.

[12] W. Hughes, K. Gault, and G. J. Princz. A Comparison of the Range-Doppler and Chirp Scaling Algorithms with reference to RADARSAT. In *Proc. IEEE Geosci. Remote Sensing Symp., IGARSS'96*, volume 2, pages 1221–1223, Lincoln, Nebraska, June 1996.

[13] M. R. Inggs. The SASAR VHF Sensor. In *Proc. European Conference on Synthetic Aperture Radar, EUSAR'96*, pages 317–320, Königswinter, Germany, March 1996. VDE-Verlag GMBH, Berlin and Offenbach.

[14] M. R. Inggs, J. Hurwitz, and A. Langman. Synthetic Range Profile Measurements of Aircraft. In *IEEE Proceedings of the 1993 South African Communications and Signal Processing Symposium, COMSIG'93*, pages 204–209, September 1993.

[15] M. R. Inggs, A. Knight, and P. Smit. Synthetic Range Profile Measurements with a Pulse Compression Radar. In *IEEE Proceedings of the 1992 South African Communications and Signal Processing Symposium, COMSIG'92*, pages 7–10, Cape Town, South Africa, September 1992.

[16] M. R. Inggs and R. T. Lord. SRP Phase VI Status Report. Technical report, University of Cape Town, Radar Remote Sensing Group, February 1996.

[17] M. R. Inggs, M. W. van Zyl, and A. Knight. A Simulation of Synthetic Range Profile Radar. In *IEEE Proceedings of the 1992 South African Communications and Signal Processing Symposium, COMSIG'92*, pages 1–6, Cape Town, South Africa, September 1992.

[18] J. A. Legg, A. G. Bolton, and D. A. Gray. SAR Moving Target Detection using a Nonuniform PRI. In *Proc. European Conference on Synthetic Aperture Radar, EUSAR'96*, pages 423–426, Königswinter, Germany, March 1996. VDE-Verlag GMBH, Berlin and Offenbach.

[19] R. T. Lord and M. R. Inggs. High Resolution VHF SAR Processing Using Synthetic Range Profiling. In *Proc. IEEE Geosci. Remote Sensing Symp., IGARSS'96*, volume 1, pages 454–456, Lincoln, Nebraska, June 1996.

[20] R. T. Lord and M. R. Inggs. High Range Resolution Radar using Narrowband Linear Chirps offset in Frequency. In *IEEE Proc. of the South African Symp. on Communications and Signal Processing, COMSIG'97*, pages 9–12, Grahamstown, South Africa, September 1997.

[21] R. T. Lord and M. R. Inggs. High Resolution SAR Processing Using Stepped-Frequencies. In *Proc. IEEE Geosci. Remote Sensing Symp., IGARSS'97*, volume 1, pages 490–492, Singapore, August 1997.

[22] J. Mittermayer, R. Scheiber, and A. Moreira. The Extended Chirp Scaling Algorithm for ScanSAR Data Processing. In *Proc. European Conference on Synthetic Aperture Radar, EUSAR'96*, pages 517–520, Königswinter, Germany, March 1996. VDE-Verlag GMBH, Berlin and Offenbach.

[23] S. Mobley and M. Maier. Synthetic Aperture Radar-Systems Processing with a Non-Uniform Pulse Repetition Interval. In *Proc. European Conference on Synthetic Aperture Radar, EUSAR'96*, pages 407–410, Königswinter, Germany, March 1996. VDE-Verlag GMBH, Berlin and Offenbach.

[24] A. Moreira, J. Mittermayer, and R. Scheiber. Processing of SAR and ScanSAR Imaging Modes using the Extended Chirp Scaling Algorithm. In *Proc. European Conference on Synthetic Aperture Radar, EUSAR'98*, pages

557–560, Friedrichshafen, Germany, May 1998. VDE-Verlag GMBH, Berlin and Offenbach.

[25] A. V. Oppenheim. *Applications of Digital Signal Processing.* Prentice-Hall, Englewood Cliffs, NJ, 1978.

[26] A. D. Robinson and M. R. Inggs. Correlation Filters Applied to Synthetic Range Profiles of Aircraft Targets. In *IEEE Proc. of the South African Symp. on Communications and Signal Processing, COMSIG'94*, October 1994.

[27] H. Runge and R. Bamler. A novel high precision SAR focussing algorithm based on chirp scaling. In *Proc. IEEE Geosci. Remote Sensing Symp., IGARSS'92*, pages 372–375, Clear Lake, TX, May 1992.

[28] O. Seger, M. Herberthson, and H. Hellsten. Real Time SAR Processing of Low Frequency Ultra Wide Band Radar Data. In *Proc. European Conference on Synthetic Aperture Radar, EUSAR'98*, pages 489–492, Friedrichshafen, Germany, May 1998. VDE-Verlag GMBH, Berlin and Offenbach.

[29] V. Shteinshleiger, A. Dzenkevich, G. Misezhnikov, and L. Mel'nikov. On the Possibility of Designing a High-Resolution Space-Borne VHF-Band SAR for Remote Sensing of the Earth. In *Proc. European Conference on Synthetic Aperture Radar, EUSAR'96*, pages 321–324, Königswinter, Germany, March 1996. VDE-Verlag GMBH, Berlin and Offenbach.

[30] G. W. Stimson. *Introduction to Airborne Radar.* Hughes Aircraft Company, El Segundo, California, 1983.

[31] M. Suess, M. Völker, J. J. W. Wilson, and C. H. Buck. Superresolution: Range Resolution Improvement by Coherent Combination of Repeat Pass SAR Images. In *Proc. European Conference on Synthetic Aperture Radar, EUSAR'98*, pages 565–569, Friedrichshafen, Germany, May 1998. VDE-Verlag GMBH, Berlin and Offenbach.

[32] L. M. H. Ulander. Performance of Stepped-Frequency Waveform for Ultra-Wideband VHF SAR. In *Proc. European Conference on Synthetic Aperture*

Radar, EUSAR'98, pages 323–326, Friedrichshafen, Germany, May 1998. VDE-Verlag GMBH, Berlin and Offenbach.

[33] D. R. Wehner. *High Resolution Radar.* Artech House, Norwood, MA 02062, 1987.