User manual for Sarsim II

R. Lengenfelder

November 27, 1997

Contents

List of Figures

1. Introduction

1.1. Overview

This radar simulator was originally developed to produce Synthetic Range Profiles (SRPs) of complex models of aircraft.. It was then generalized to also generate SAR simulation files. The main idea behind writing this simulator was to have an easy-to-use, graphical interface which can show the results in real-time. This of course implied that some way of reducing the computation required had to be found. The solution is WYSIWIC (what you see is what is calculated). This means that the data is only calculated with a resolution depending on the screen resolution. Only when it is saved to disk, the data will be written with the required sampling rate. The exact method will be explained in detail in chapter 5.

The features of the program include :

- Chirp, monochrome or 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 RCS 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 such that features like e.g. using real-life antenna gain patterns can be loaded with ease.

Section 1.3 explains exactly what this simulator is intended for, giving some brief introduction to general radar theory.

Chapter 2 introduces the program to people who do not want to read the whole manual, but only want to do some very simple simulations.

Chapter 3 is a complete description of all windows and dialogs appearing in Sarsim. This chapter acts more like a reference and can be referred to for more complete covering.

The structure of the script files is explained in chapter 4. Script files can be used to automate certain tasks, they are effectively text file containing a description of the simulation setup.

In chapter 5 the formulas used in Sarsim are described in detail.

1.2. Background and System requirements

The graphical frontend of the simulator has initially been 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, which offers visual components combined with the flexibility and speed of C++. The program 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 acts like a compiler which reads the script files (text files) and writes the required simulation files to disk.

The program 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 colors, but 65k is preferable. A screen resolution of 800 by 600 or higher is necessary.

Only the file *SARSIM2.EXE* is necessary to run the program, all DLL files have been linked into it. However if some of the dialogs are displayed incorrectly, you might still have outdated libraries on your system. In this case copy the files in the **\dll** directory into the **\system32** subdirectory of your windows directory.

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

1.3. Radar Theory applied for a Simulator

This section will give a very brief overview of radar theory and how it was implemented for this simulator.

Radars are electromagnetic devices used for detection of targets by radiating electromagnetic energy and examing 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).

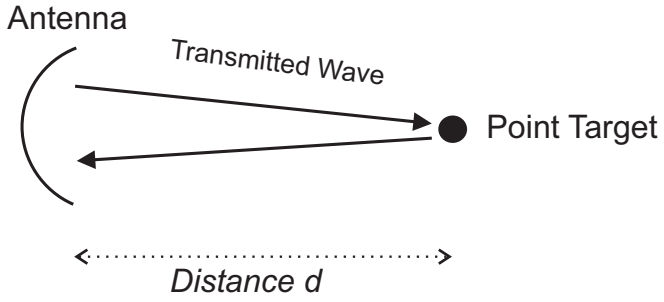

Figure 1.1: A simple radar setup

Most radars work in pulsed mode, i.e. they send out short burst of energy, with relative long delays in-between.. The period between these pulses is called Pulse Repetition Interval (PRI), but usually the reciprocal called Pulse Repetition Frequency (PRF) is specified.

Consider the setup shown in figure 1.1 with a single point target. A pulse is sent out at time t and the return is received after $\Delta t = \frac{2 \cdot d}{c}$ seconds where c is the speed of light. For this simulator the received signal will be a replica of the transmitted signal, although the power will be a fraction of the original pulse. The power of the reflected signal at the radar will be $\frac{P \cdot G \cdot \sigma}{(4 \cdot \pi \cdot d^2)^2}$, P being the transmitted power, G the gain of the antenna, 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 1.2.

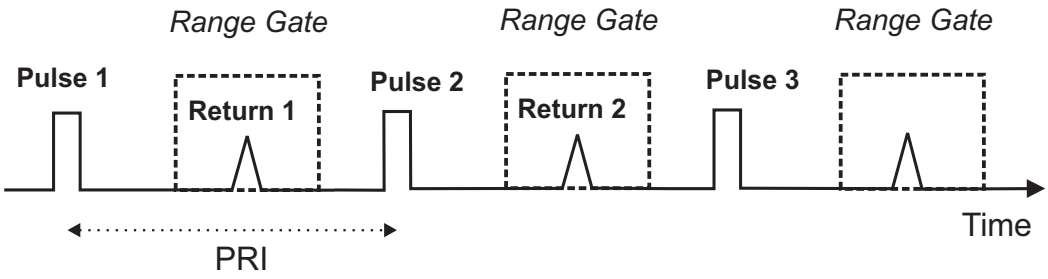

Figure 1.2: Pulsed operation

The point target will return an echo after some time Δt which depends on the distance of the target from the radar. 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 corresponds to the output of the radar at a certain time span. For this simulator the nomenclature of Synthetic Aperture Radars (SAR) 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 1.3.

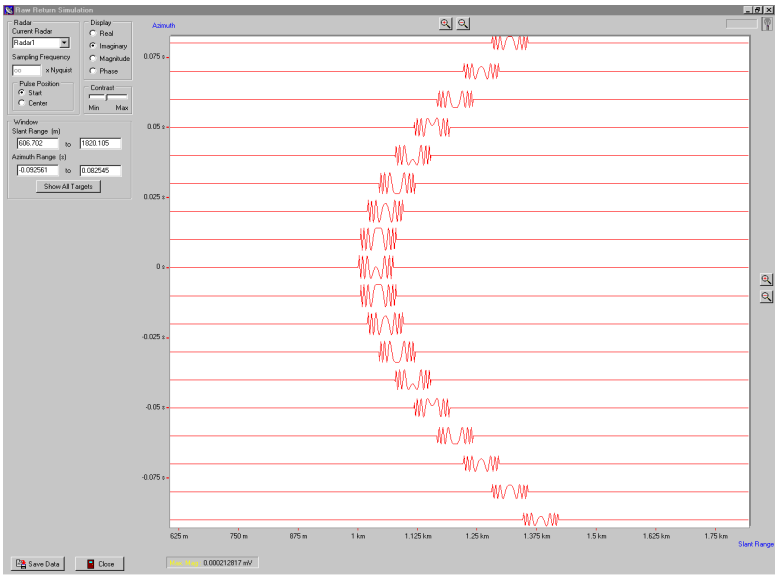

Figure 1.3: An example of a simulation output of a moving point target

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.

To summarize:

This radar simulator 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 in convolved with it's replica.

2. Getting Started

This chapter will introduce Sarsim by setting up a very simple simulation.

2.1. The start-up window

Start the program by executing 'SARSIM.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 2.1.

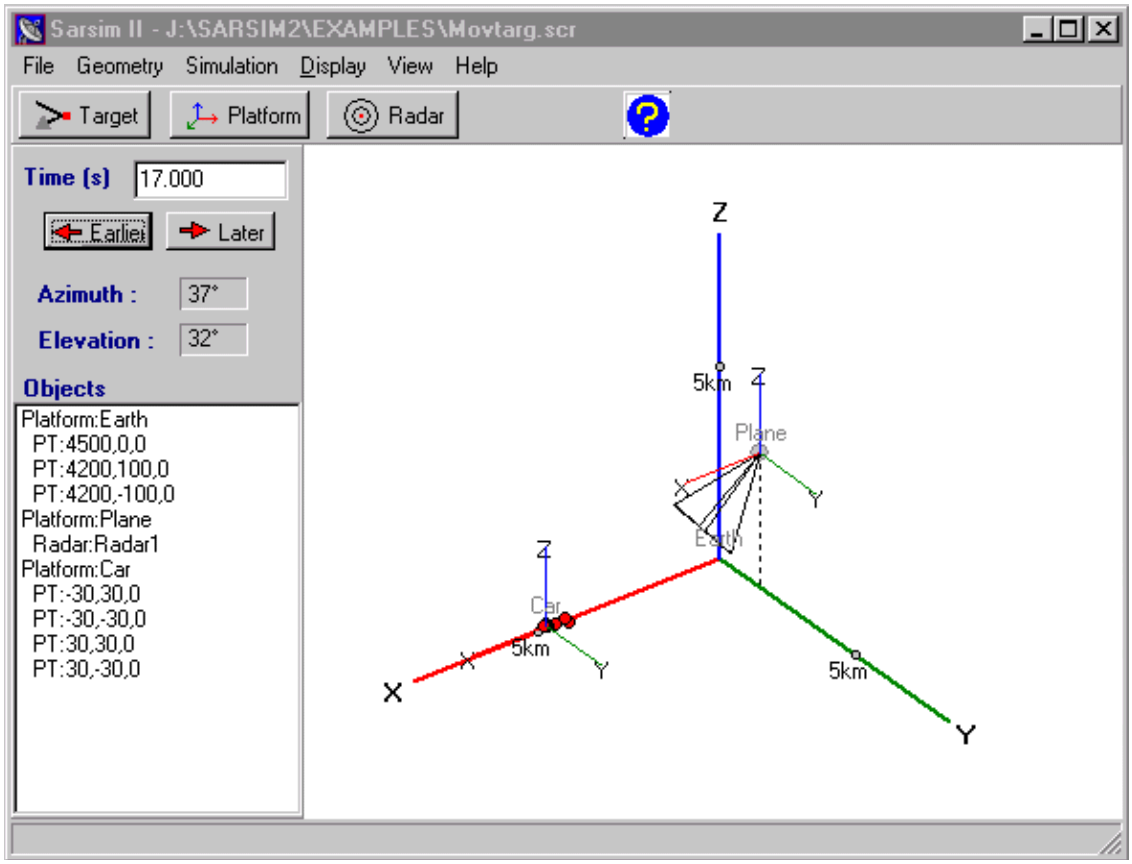

Figure 2.1: Main Window

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 just now. All point targets or radars defined on that platform will be stationary as seen from that coordinate system.

Point targets are infinite 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 3.

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

in chapter 3.

2.1.1. 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. The name of it is '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 labeled ends are pointing in the positive direction. There are distance marks on the axes (e.g. 10km) to give some sense of scale. Below the top border of the window, the azimuth and elevation angle of the current viewpoint are shown. There are three things one can do with the coordinate system, all of them are handled by the mouse.

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.

There is a way of changing the focus point (the point around which the coordinate system turns) - see section 3.9.

In the top-left corner the current simulation time is given, i.e. what one sees is a snapshot at the given time. This helps visualizing 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 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, e.g. all objects placed on a platform will appear (indented) just below it.

2.1.2. 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 lot 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 shortcut to create new objects is the toolbar on top of the screen :

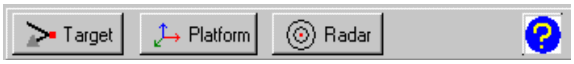

Figure 2.2: The toolbar

2.1.3. 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 (only point targets and radars - use the list to modify platforms).

2.1.4. 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.

2.2. A simple example

This section will show how to create a simple simulation with single point target. The simulation parameters are :

Chirp pulse, chirp *bandwidth 50 MHz*, *1 GHz carrier frequency*, *pulse width 5000 ns*, *1 kHz PRF*, *1 kW power output*, *isotropic* antenna, isotropic point target at distance *3000 m*, $3m^2$ cross section.

- 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 shown in figure 2.3 will come up.

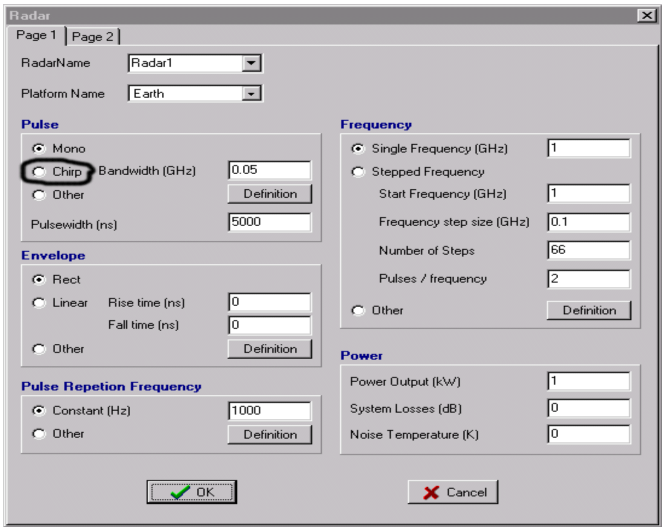

Figure 2.3: Radar Setup Window (1/2)

Change the pulse type to 'Chirp' by selecting the **Chirp** radio button (circled in figure 2.3). (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).

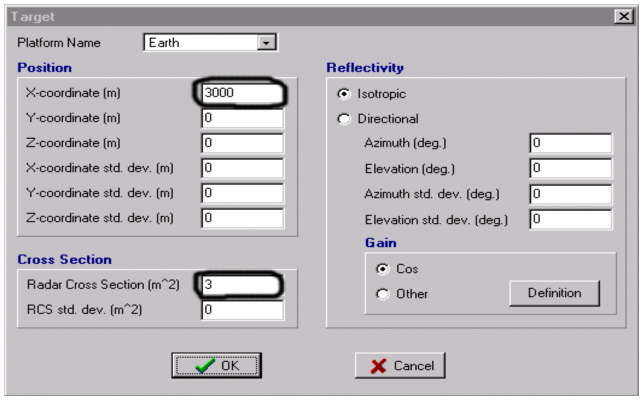

Figure 2.4: Point Target Setup Window

- Set the X-coordinate to 3000 m and the radar cross section to 3 m^2 as indicated in figure 2.4. 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 2.5.

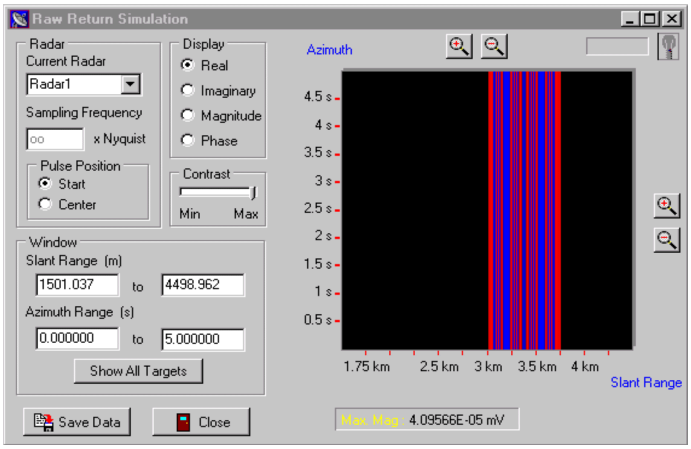

Figure 2.5: Return Signal of a single Point Target

Shown on the right-hand side of this window is a color-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 meter. The color palette 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 for 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:

- **Changing the values** of the slant/azimuth range **in the edit-boxes** on the left.
- **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.
- **Zooming out** by a factor of two in either direction by right-clicking anywhere on the image. The window will also update automatically.
- **Showing all targets** (i.e. selecting a slant range spanning from the closest to 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 2.6.

The time instance at which the shown pulses has been transmitted by the radar can then be easily identified, e.g. in this example the PRF is 1kHz, therefore a pulse will be sent 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 they have there 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:

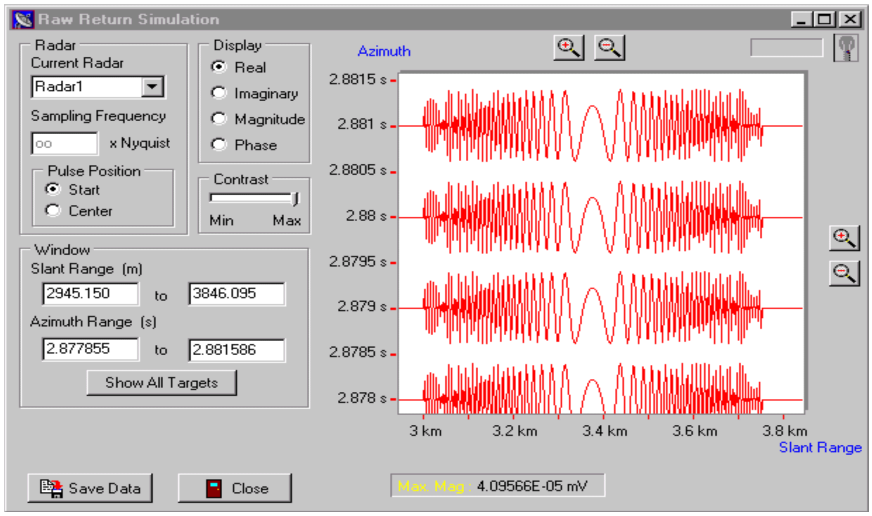

Figure 2.6: Raw Return after Zooming In

$$P_{received} = \frac{P_{transmitted} \cdot G^2 \cdot \lambda^2 \cdot \sigma}{(4 \cdot \pi)^3 \cdot R^4} = \frac{1000 \cdot 1 \cdot 0.3^2 \cdot 3}{(4 \cdot \pi)^3 \cdot 3000^4} = 1.67976 \cdot 10^{-15} W$$

and as the amplitude is show:

$$Magnitude = \sqrt{P_{received}} = 4.095 \cdot 10^{-5} mV$$

The whole window can be resized (or maximized) in the usual way, but note that the calculation time is proportional to the image size. On slow computers it might be advisable to make the window smaller than the default size.

To **save** the shown window, click onto the 'Save Data' button. This will be explained in more detail in section 3.9.

3. Command Summary

This chapter will explain all available commands.

3.1. Main Window

3.1.1. Overview

After starting Sarsim you will see a screen similar to the one shown in figure 3.1. 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 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 (PT), platforms and radars. They are sorted in respect to platforms, where below each platform (intended) a list of all object 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.

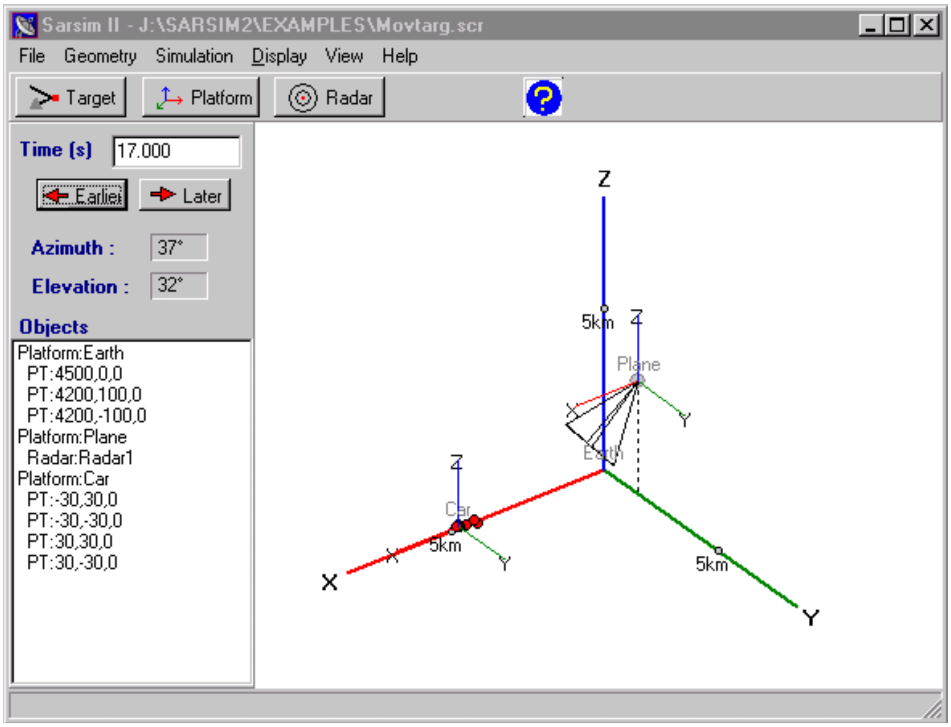

Figure 3.1: Main Window

3.1.2. 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 target, 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 with option to modify or delete the object will appear.

3.1.3. 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 4.

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 3.2.

New Platform - Creates a new platform. A detailed description is given in section 3.3.

New Radar - Creates a new radar. A detailed description is given in section 3.4.

Simulation

Raw Return - This option shows the received return signal mixed down to baseband. A detailed description is given in section 3.5.

Range Compressed - This options shows the range compressed return, also mixed down to baseband. A detailed description is given in section 3.5.

Previously Stored - The parameters of a simulation can be stored such 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 given.

3.2. New Target Window

Creating a new target brings up a dialog window shown in figure 3.2.

The following parameters can be configured :

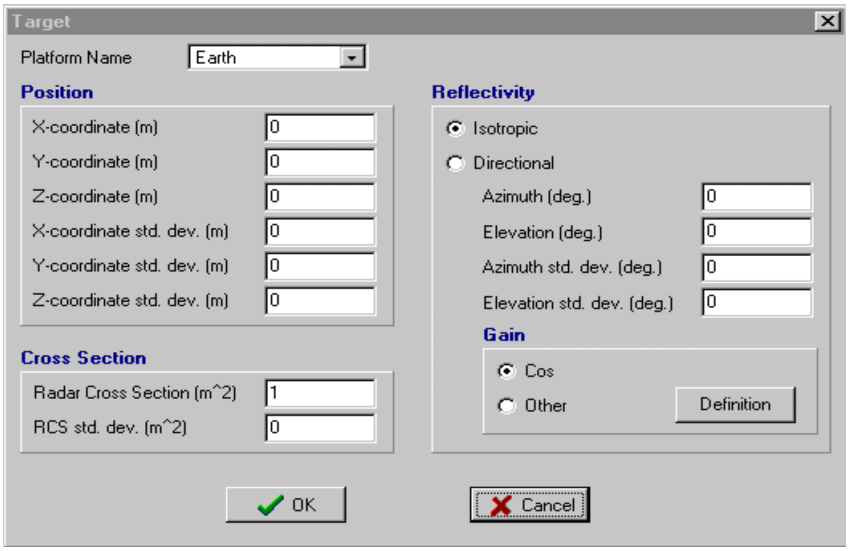

Figure 3.2: New Target Dialog

- Platform name (up to 15 characters)- the platform to which this 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 meter.
- Standard deviation of X, Y and Z position - introduce gaussian distributed random jitter.
- Radar Cross Section - the radar cross section in square meter.
- Radar Cross Section deviation - if the target scintillates the standard deviation of RCS can be specified. The distribution is gaussian.
- Isotropic reflectivity - the incidence angle of the ray from the radar is irrelevant, 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 x cos(incident 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 3.6.

3.3. New Platform Window

Creating a new platform brings up a dialog window shown in figure 3.3.

The following parameters can be configured :

- 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 coordinate needs 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 3.6.
- 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".

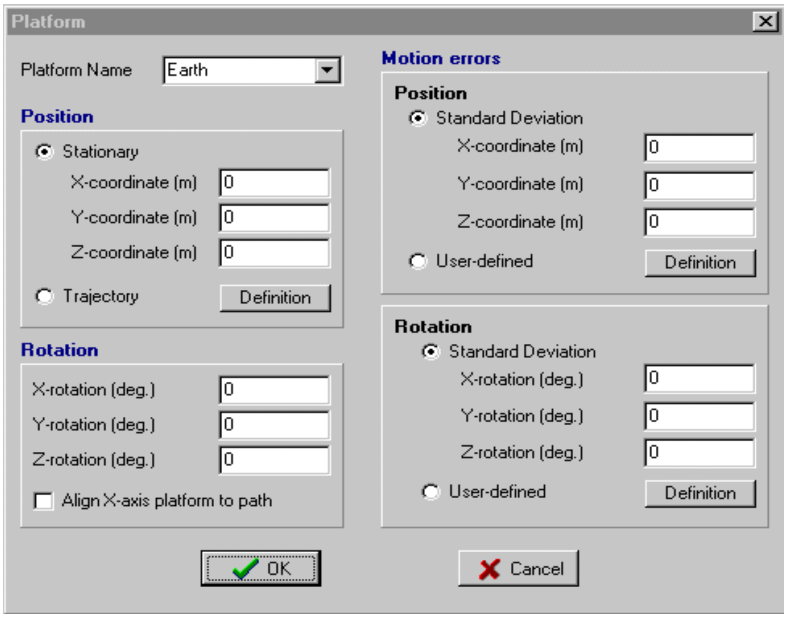

Figure 3.3: New Platform Dialog

- 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 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. As an example where this is useful is if a platform is used to describe the motion of an aircraft, the nose of the aircraft will always point in the direction it is flying to. An example is give in chapter 6.
- 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 error by giving a integration envelope which will be convolved with gaussian distributed noise, but this option is still in development..

3.4. New Radar Window

Creating a new radar brings up a dialog window shown in figure 3.4.

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.
- 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 modell more realistic pulses as shown in figure 3.5.
- Pulse Repetition Frequency - the PRF can be defined either as a constant or it can vary from pulse to pulse as shown in figure 3.6. The data wraps around such that $PRI[x] = PRIArray[modulus(x, n)]$. Note that all times are given in seconds.

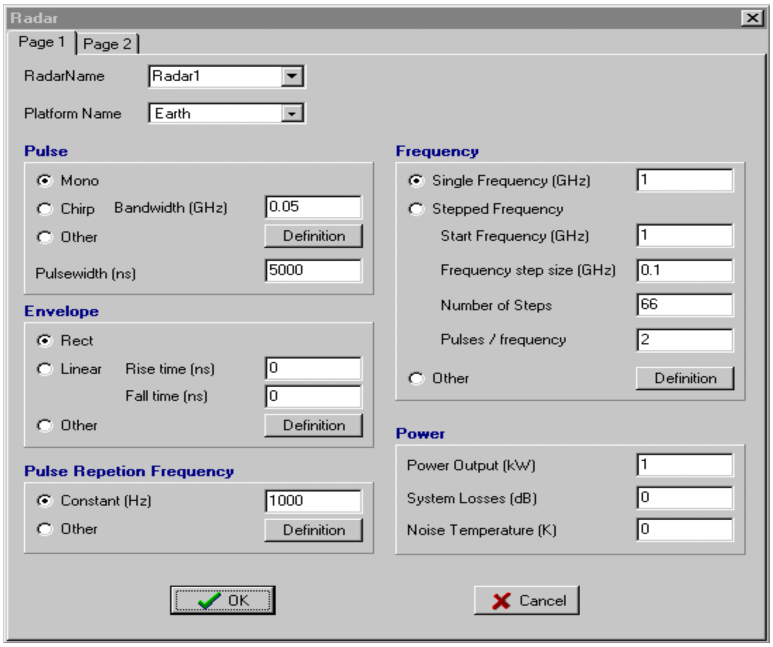

Figure 3.4: New Radar Dialog - Page 1

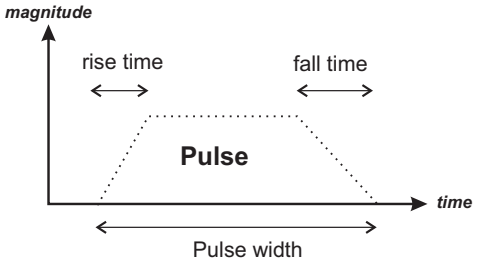

Figure 3.5: Pulse Envelope

- Center frequency - specify the center frequency of each pulse. It can be either constant, stepped or completly arbitary.
- 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 :

$$N = k \cdot T \cdot B$$

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 pulse = $\frac{1}{Pulsewidth}$). Let's assume 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 :

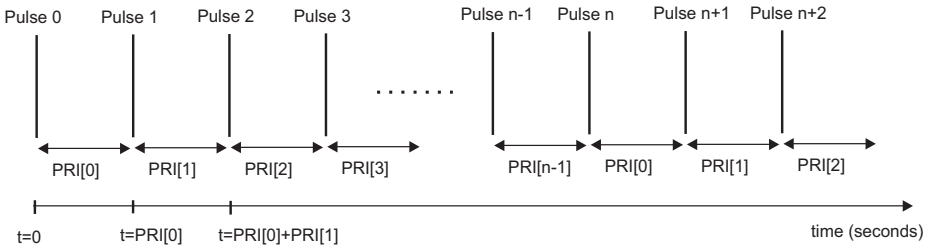

Figure 3.6: User-defined PRFs

$$\begin{aligned} IN &= I + GaussianNoise(\sqrt{N}) \\ QN &= Q + GaussianNoise(\sqrt{N}) \end{aligned}$$

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

Consider a simple simulation with a point target (RCS = $1m^2$) at a distance of 1 km and an a radar with power output of 1kW. The pulse should have monochrome modulation and have a width of 1000 ns. The noise temperature is set to 100 Kelvin, the center frequency is 1 GHz.

The signal power received would be :

$$P_{received} = \frac{P_{transmitted} \cdot \lambda^2 \cdot \sigma}{(4 \cdot \pi)^3 \cdot R^4} = \frac{1000 \cdot 0.3^2 \cdot 1}{(4 \cdot \pi)^3 \cdot 1000^4} = 4.53 \cdot 10^{-14} W$$

The noise power would be :

$$N = k \cdot T \cdot B = 1.38 \cdot 10^{-23} \cdot 100 \cdot \frac{1}{10^{-6}} = 1.38 \cdot 10^{-15} W$$

which gives a power S/N ratio of :

$$\frac{S}{N} = \frac{4.53 \cdot 10^{-14} W}{1.38 \cdot 10^{-15} W} = 32.8 = 15.2 dB$$

Converting to amplitudes (assuming a 1Ω resistor) :

$$V_{received} = \sqrt{P_{received}} = 2.13 \cdot 10^{-7} V$$

$$NoiseVoltage = 3.715 \cdot 10^{-8} V$$

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

A second page of configurable parameters can be selected. It is shown in figure 3.7.

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.

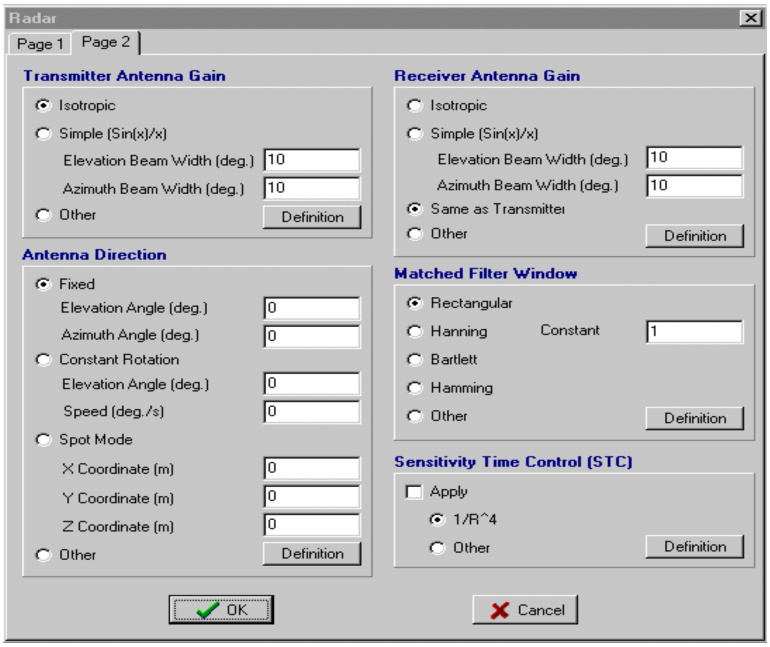

Figure 3.7: New Radar Dialog - Page 2

3.5. Simulation Window

3.5.1. 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 computational intensive the sampling rate can be set. Note that setting low sampling rates (less than three times Nyquist rate) can result in unfamiliar graphs.

A typical simulation window is shown in figure 3.8.

The simulation window shows the received return signal mixed down to baseband. On the right hand side a window with slant range (in meter) versus azimuth (in seconds) is shown. The maximum amplitude (mV) is shown on 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 left-click and drag on the simulation window. If the simulation window is right-clicked it will zoom out by a factor of two in the slant and azimuth range.

The “Display” box selects whether the real, imaginary part or the phase or magnitude is displayed. The display of the pulses is color-coded where blue represents positive, black zero and red negative values. In case of a graphics card supporting 256 or more colors, the colors 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 azimuth direction, the pulses will be not color-coded anymore but shown as a proper graphs.

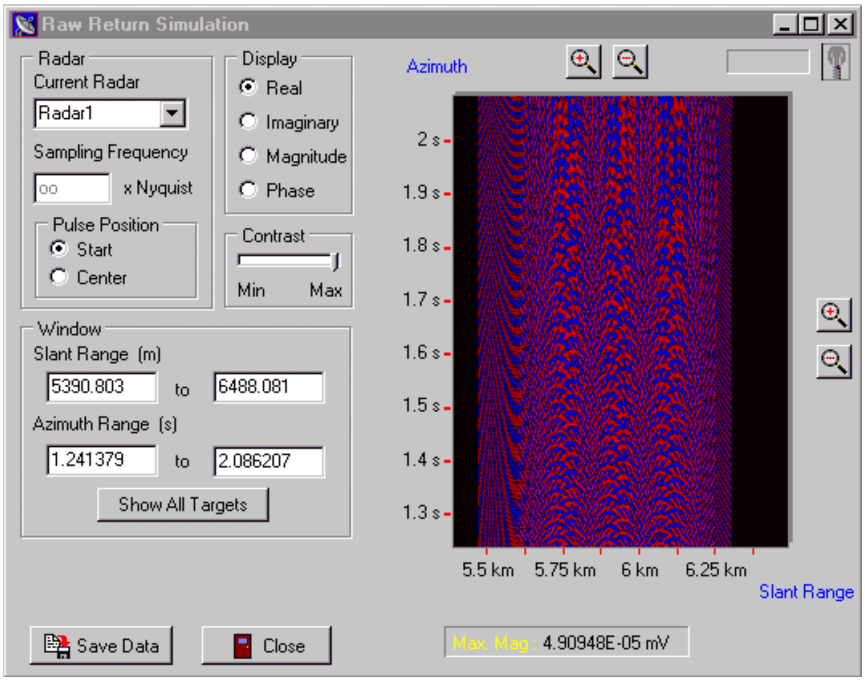

Figure 3.8: A typical Simulation Window

There is a important points which needs to be clarified : The simulation windows 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 ($=\frac{c}{2 \cdot dist}$ (in seconds)) as x-axis, but shows a scale which corresponds to the actual distance to the target. An example: a pulse of length $1us$ is 300 meter long. If the A/D frequency is 500 MHz, the returning pulse will range over 500 samples. One sample corresponds to 0.3 m ($\frac{c}{2 \cdot f_{A/D}}$) in distance and therefore the pulse will have a length of $500 \cdot 0.3m = 150m$ on the display (and not the actual 300m). 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 there it actually is, and therefore the range (and the pulse-length) is divided by a factor of 2, to overcome this problem.

3.5.2. 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 3.9 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 meter and represents the distance from the radar. It can be converted into time (range delay) by the simple relationship $t = \frac{2d}{c}$, the symbols having their usual meanings. The azimuth range is measured in seconds and corresponds to the time span for which the data will be recorded. If the PRF is a constant value, the number of pulses can be determined by the simple formula : $pulses = (EndTime - StartTime) \cdot PRF + 1$. In non-SAR applications the term azimuth range is equivalent to the time window (slow time) for which the radar is capturing data.

The sampling frequency determines the sampling rate of the received signal. By default the sample frequency is set

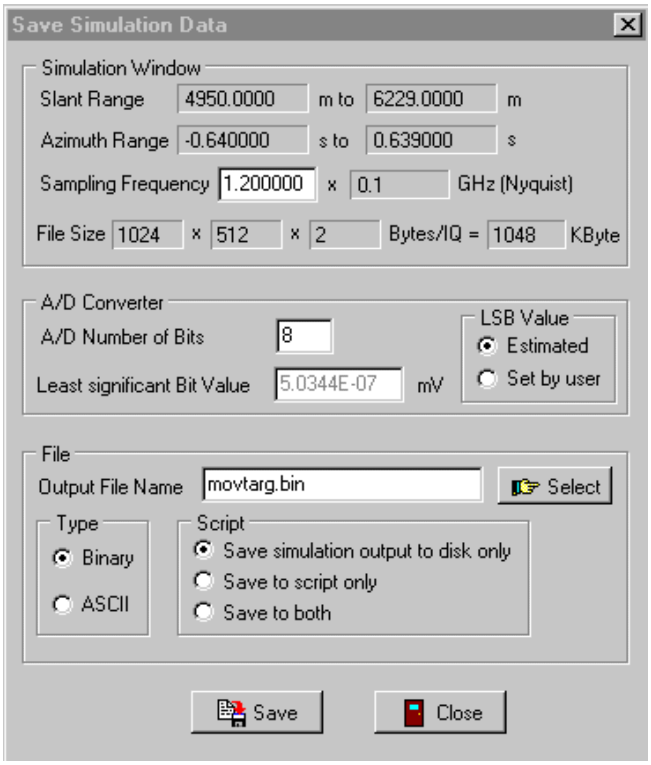

Figure 3.9: "Save Data" dialog

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. values below one are accepted.

Below that, the size of the file is estimated. In this case (figure 3.9), there are 512 (1279 m in range $\triangleq \frac{2 \cdot 1279}{c} = 8.5352us$, $8.5352us \cdot 120Mhz = 1024$) sample points in slant range 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, 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 as 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 been actually 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 (e.g. 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 also shown in figure 3.9. Say the I-value at some point is $1.2 \cdot 10^{-5}$ mV. The value written to disk will then be:

$(2^{8-1}-1) + round(\frac{1.2 \cdot 10^{-5}}{5.0344 \cdot 10^{-7}}) = 127 + 37 = 164$, 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 such that it can be recalled quickly. This is done by setting the selecting the 'Save to script only' or '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 4. 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.

3.5.3. 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 3.10.

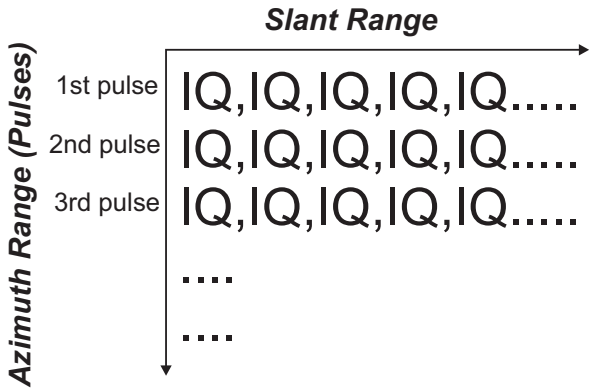

Figure 3.10: Data file structure

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 3.9 there would be 1024 IQ pairs (2048 numbers) for each line and a total of 512 lines.

3.6. User-defined Functions

This radar simulator is highly configurable. There are 12 functions which can be defined completely arbitrarily, e.g. the pulse shape, the center 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.

3.6.1. The data entry window

A standardized input screen is used for all of them which is shown in figure 4.1.

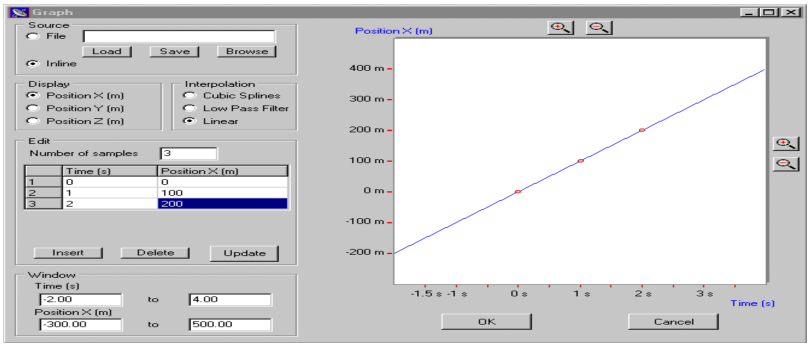

Figure 3.11: A User-defined Function

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 coordinate 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 (e.g. 500 m at t = 0 seconds), the coordinate will stay constant (500 m) 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 (not necessarily horizontal). For more than 2 points, the program will use the selected interpolation method. Note that the points does not have to be defined with constant intervals for the x-coordinate, e.g. (1,3),(2,5) and (20,10) is perfectly acceptable.

There are currently 3 interpolation methods implemented :

1. **Cubic splines** - this will use cubic splines to interpolate between points. This interpolation is computational 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 as shown in figure 4.1.

A comparison of the different methods is given in figure 3.12.

For the example in figure 4.1, 3 samples are given. To create a platform moving with constant velocity it would be sufficient to specify two samples.

A detailed description of the every part of the window will be given now:

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 4.

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

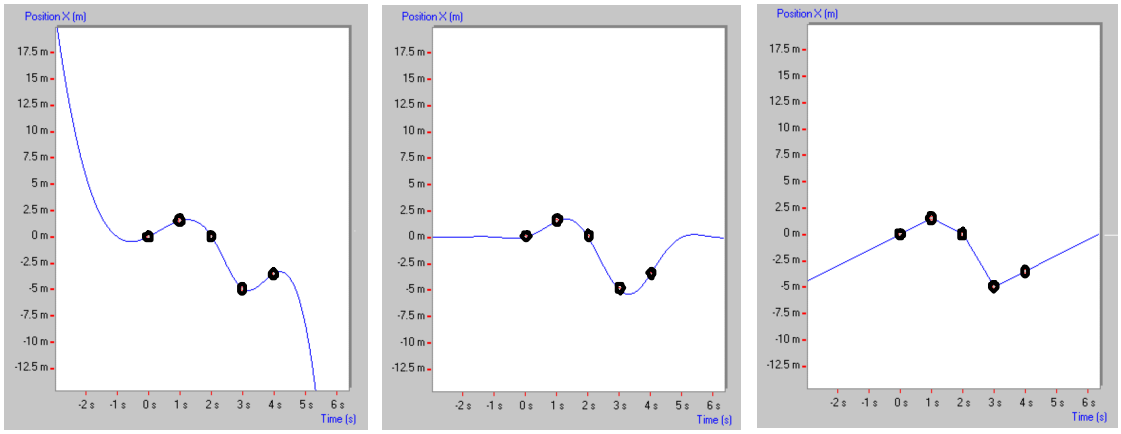

Figure 3.12: The different interpolation methods

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, i.e. you might want to specify 5 points for the X-coordinate, but only 1 for the Y- an Z - coordinate). Below that a list of x and y coordinates are given for the number of specified points.After entering the values press **Update** to update the graph on the right. For 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 type 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 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 center frequency for each pulse. In this case the ordinate just represents the pulse number and will be identical to the current sample number.

3.6.2. 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 are 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 (e.g. position require 3 coordinates, other 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 3.12 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.

3.7. 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.
- Thumb-nail overview image shown.
- Slices in range or azimuth.
- Several color palettes available with slider controls for brightness, contrast and saturation.
- Automatic display of I,Q, phase and magnitude for selected sample.
- Usual zooming and paning controls.

The dialog displayed in figure 3.13 appears after selecting the menu option **File / Open**.

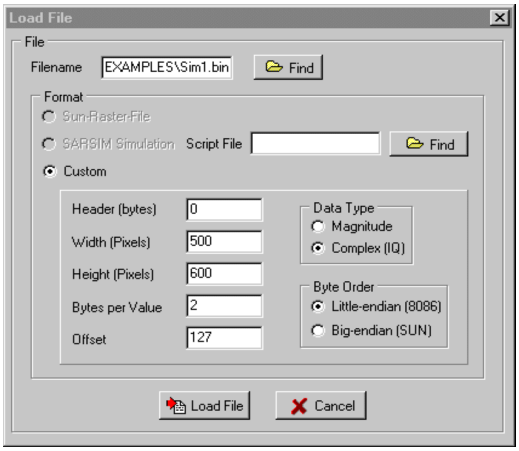

Figure 3.13: Load File Dialog

- 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). e.g. 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 3.14.

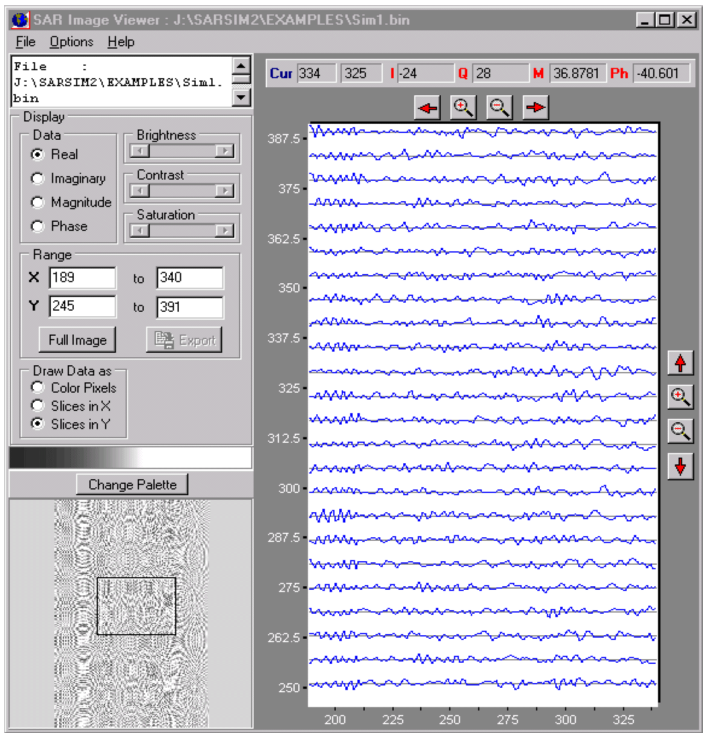

Figure 3.14: The Image Viewer Window (showing slices in azimuth)

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 :

- Color pixels - each value corresponds to a certain color (see color 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.

3.8. Looking at Script Files

Script files are used to describe the complete geometry of the simulation and they also store the paramaters of files written to disk. Sometimes it 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 3.15.

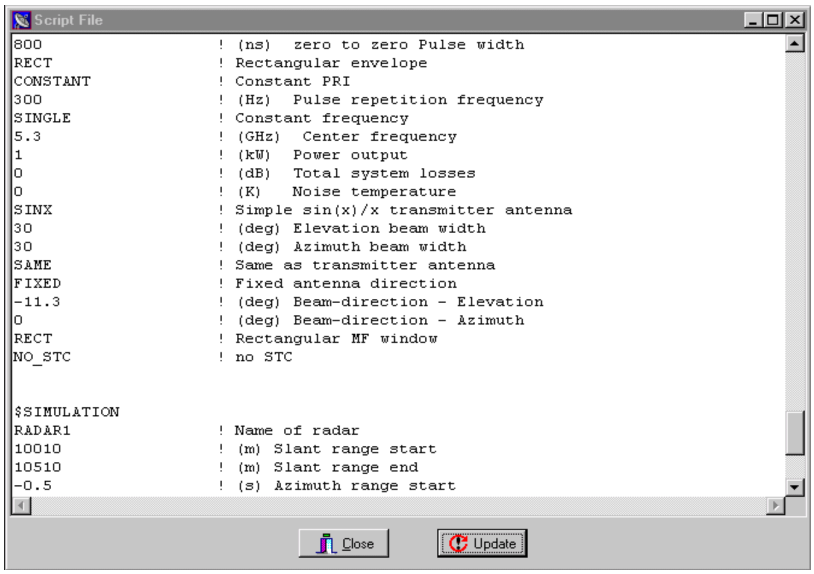

Figure 3.15: The script file viewer

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

3.10. Help

This complete manual is available under **Help** / **Contents**.

4. Script Files

There are only 4 commands used in the script files, they are : **$TARGET**, **$PLATFORM**, **$RADAR** and **$SIMULATION.** After the command, 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 better readable 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 simulations as last.

4.1. User-defined functions

The flowcharts of 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 4.1. Up to three functions can be defined in one block e.g. X-, Y- and Z-coordinate versus time. For each function a interpolation method needs to be specified (Cubic, Matched-Filter or Linear).

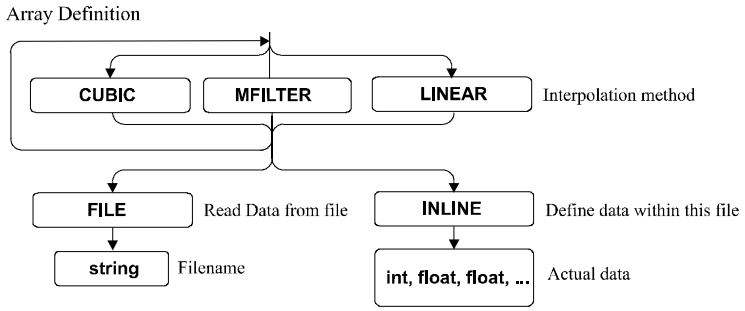

Figure 4.1: Flow diagram for user-defined function block

User-defined functions can be defined either inline or via a 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 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 are given, then the x- and y-coordinates for all the samples are given, then starts again with the next function (if there is). For example specifying the path of a platform needs 3 functions, one for each coordinate versus time. An example for an 'INLINE' user-defined trajectory function is given now (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 3.12.

4.2. The $TARGET command

A flowchart of the '$TARGET' command is shown in figure 4.2.

4.3. The '$PLATFORM' command

A flowchart of the '$PLATFORM' command is shown in figure 4.3.

4.4. The '$RADAR' command

The flowcharts of the '$RADAR' command is shown in figure 4.4 and figure 4.5 (Split up due to size reasons).

4.5. The '$SIMULATION' command

A flowchart of the '$SIMULATION' command is shown in figure 4.6.

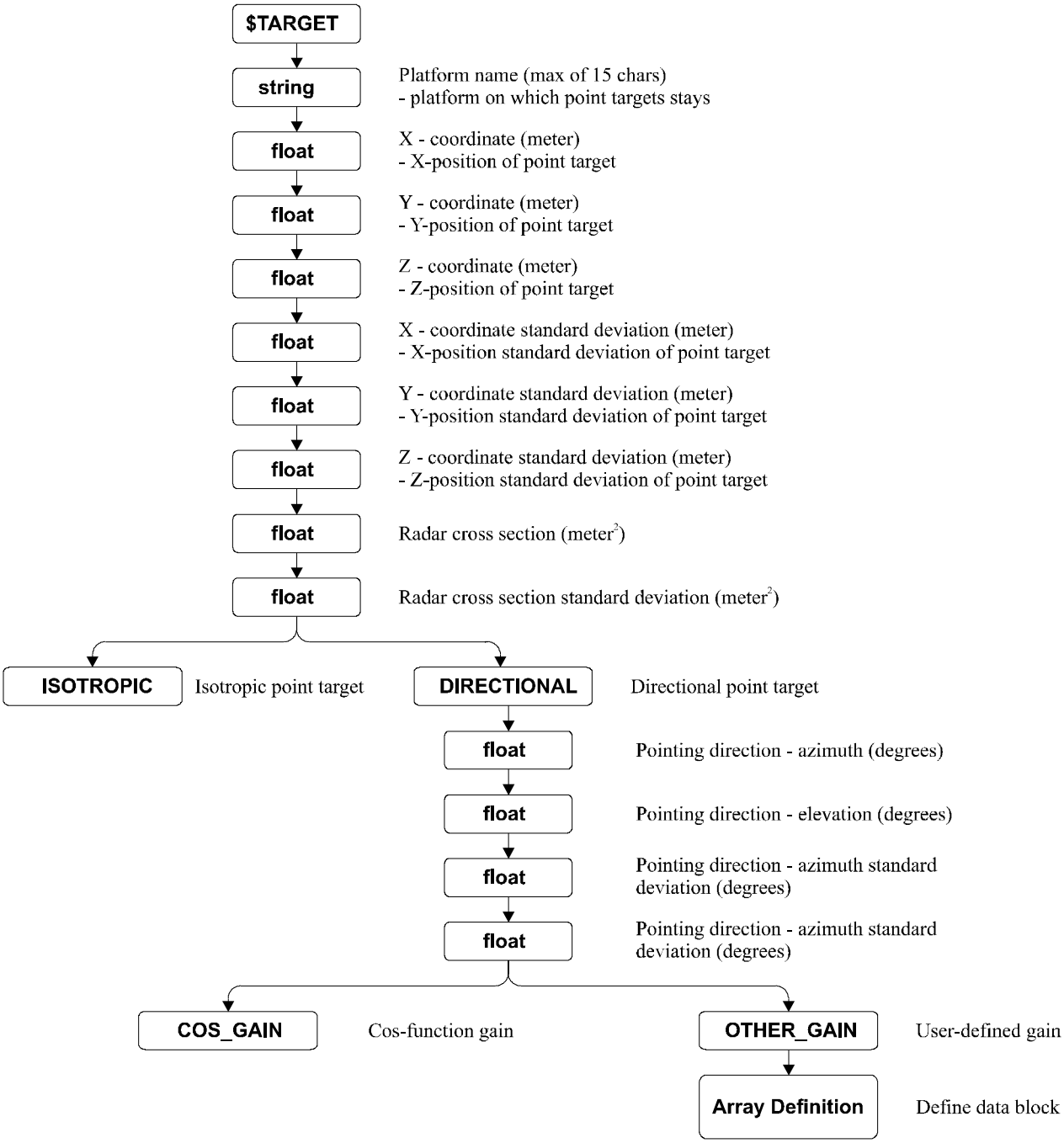

Figure 4.2: Flow diagram of the $TARGET command

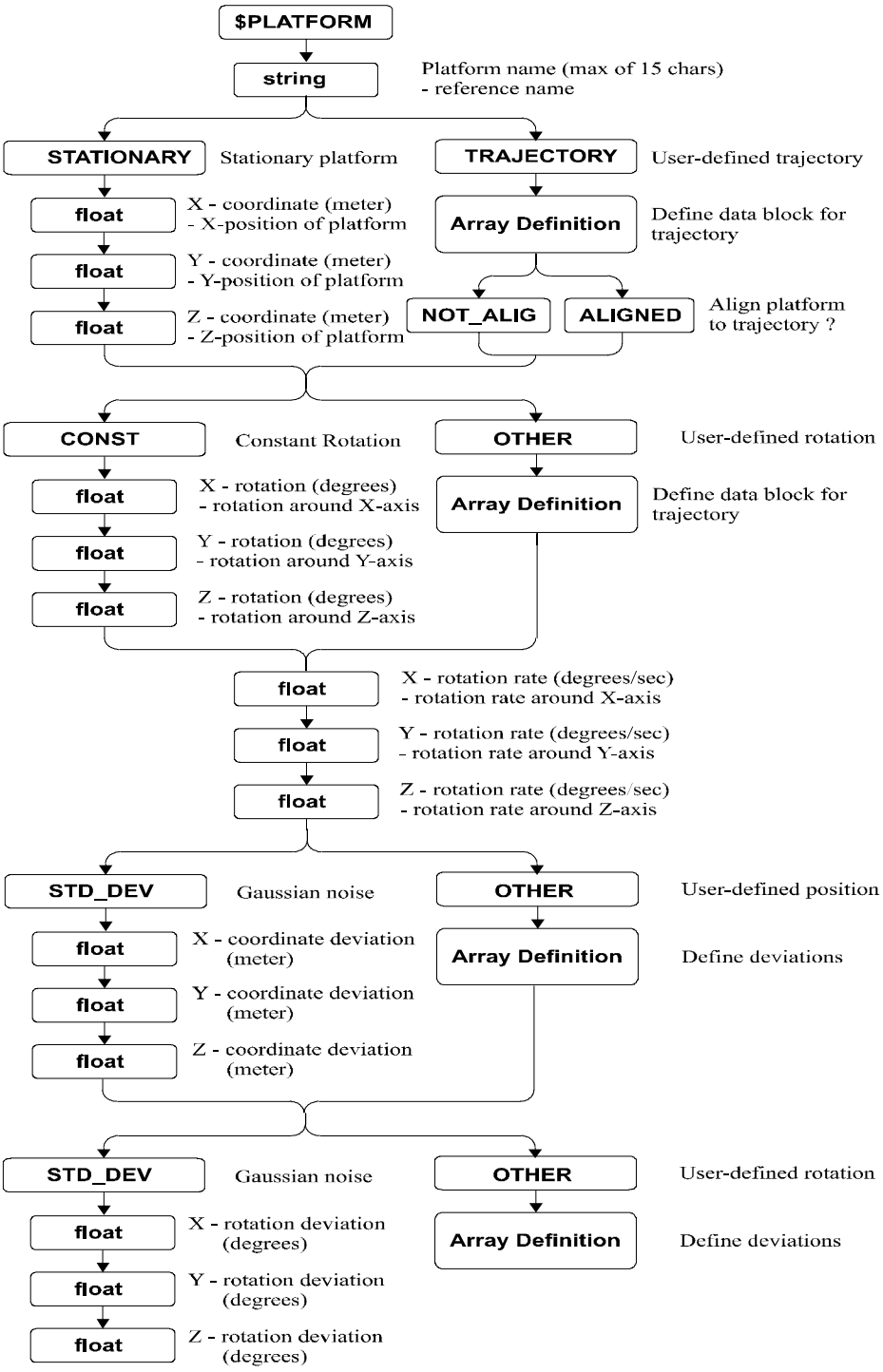

Figure 4.3: The flow diagram of the $PLATFORM command

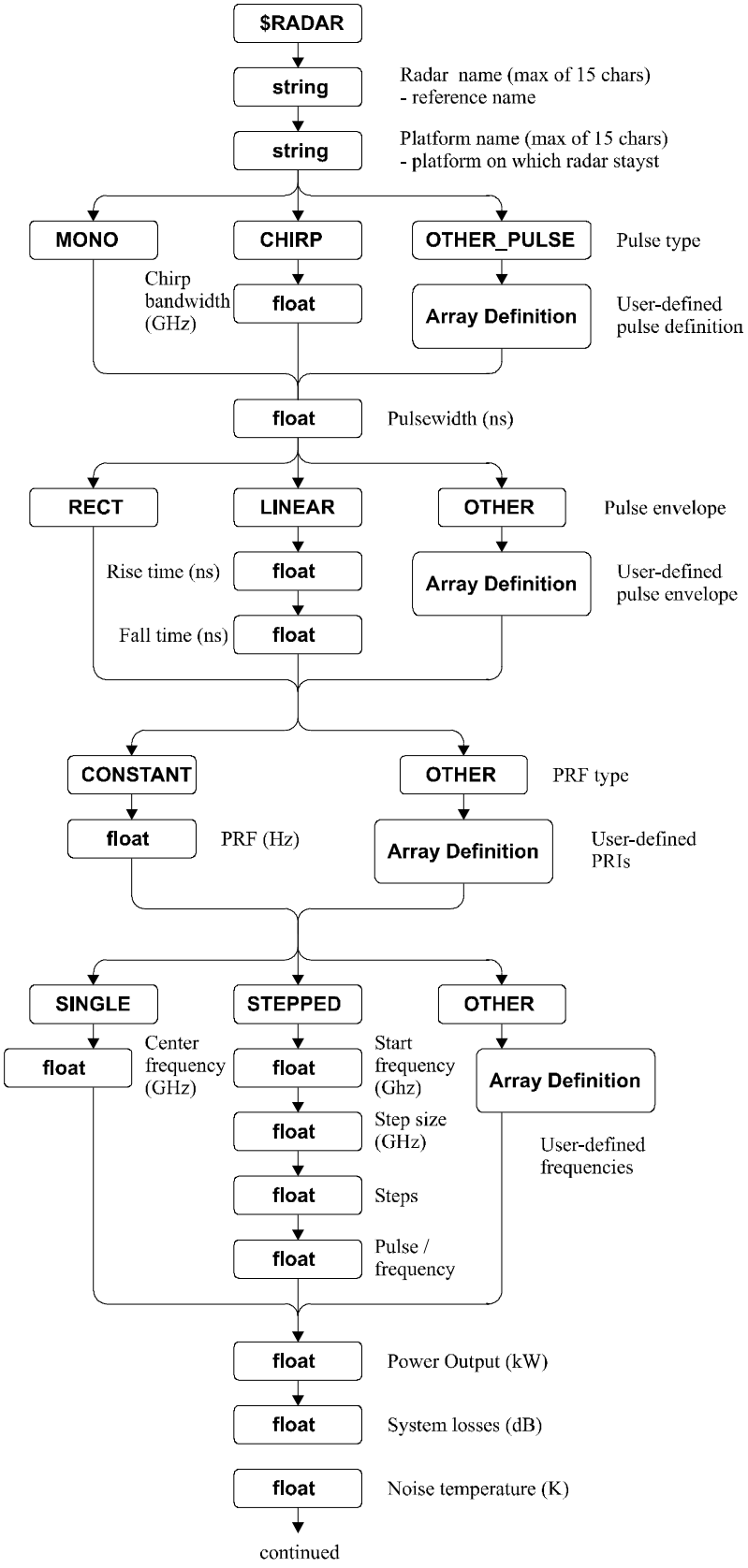

Figure 4.4: Flow diagram of the $RADAR command (1st part)

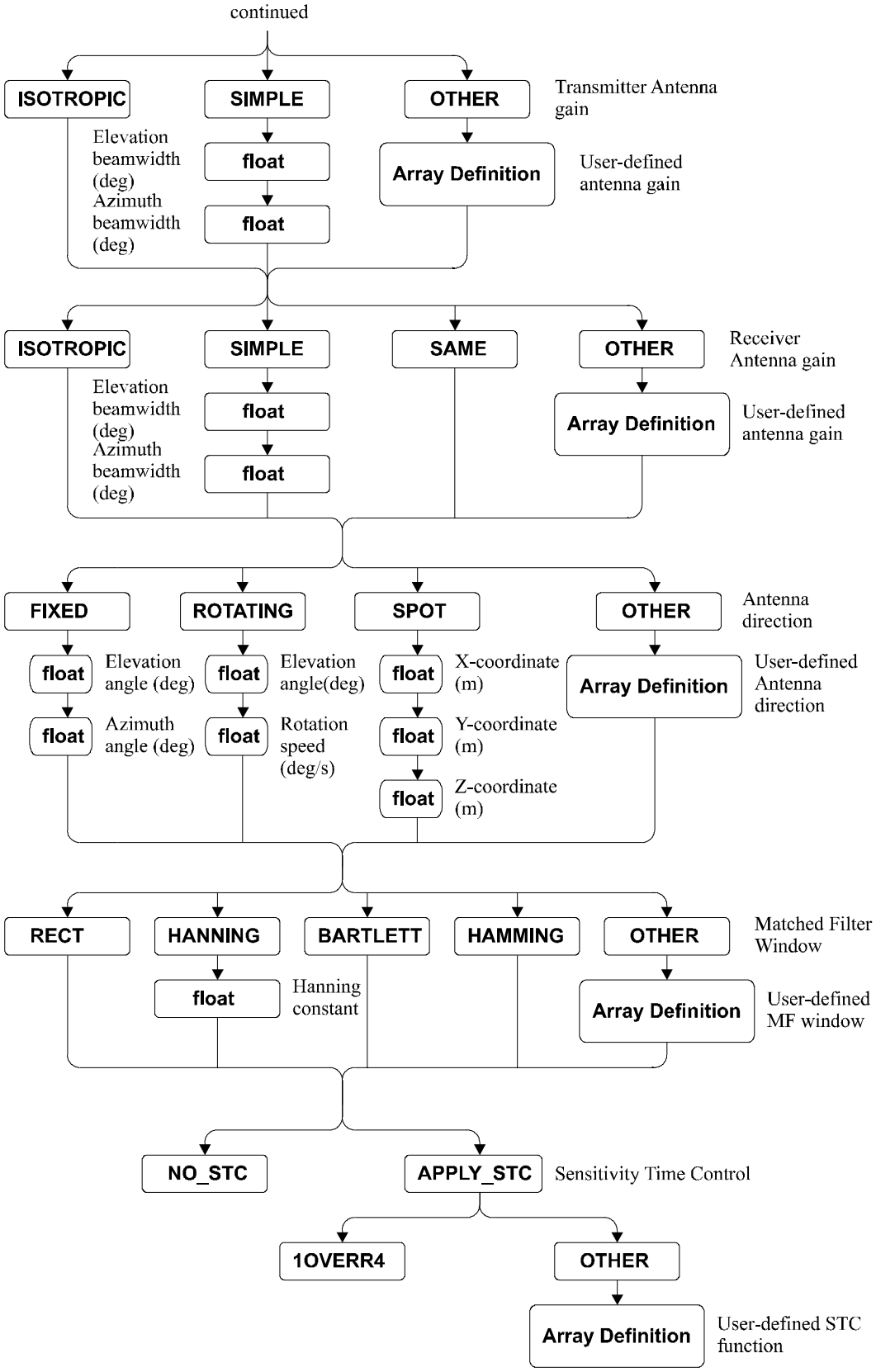

Figure 4.5: Flow diagram of the $RADAR command (2nd part)

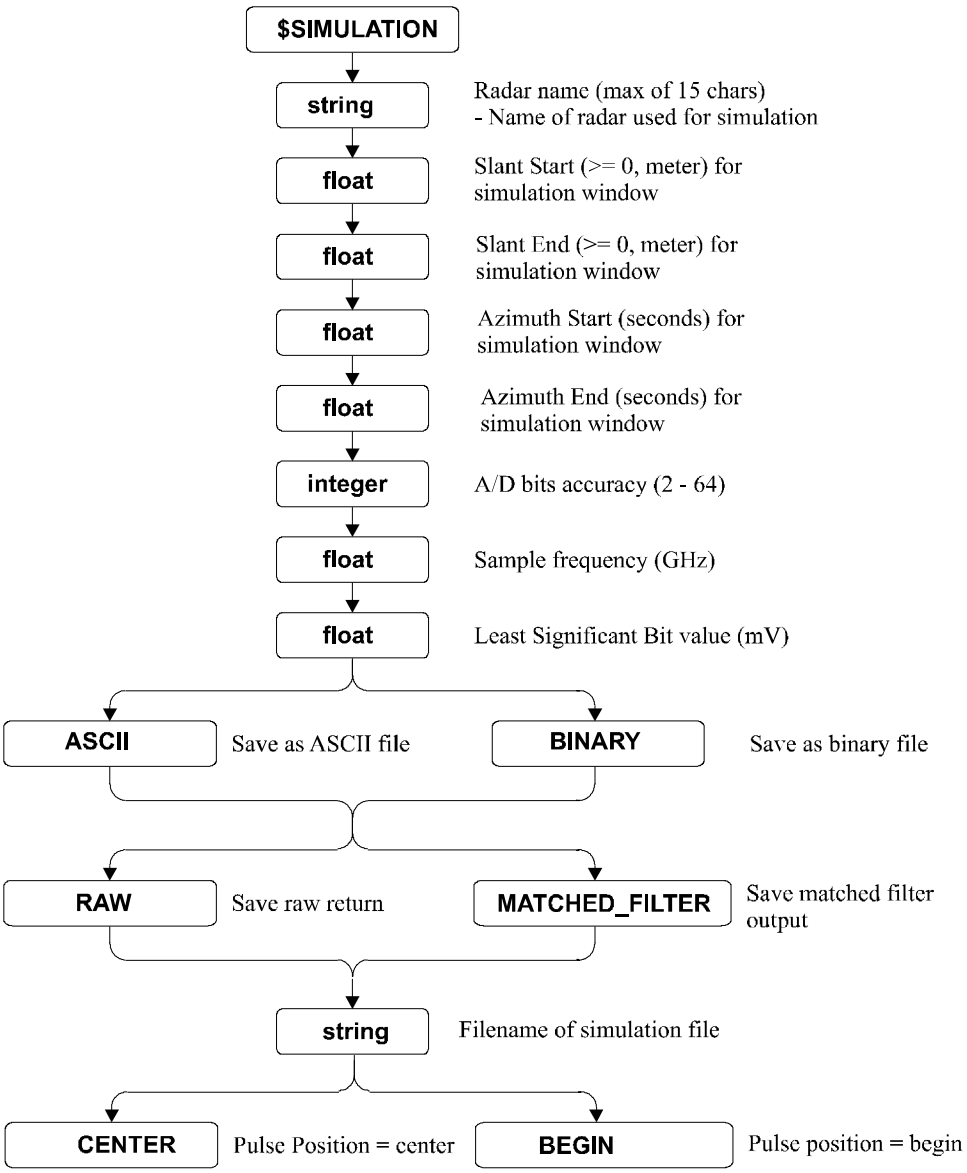

Figure 4.6: Flow diagram of the $SIMULATION command

5. Sarsim Internals

In this chapter the formulas behind Sarsim with lots of explanations will be revealed. The functions which depend on each other will presented in order of dependence.

5.1. 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)

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

5.2. Pulse Calculations

Let's assume we want to calculate the return for the following window in azimuth and slant range as shown in figure 5.1.

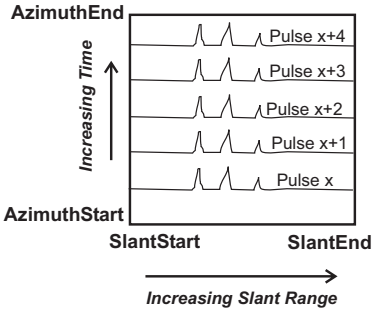

Figure 5.1: Simulation Window

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

5.2.1. 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 = 0. For a constant PRF the function is :

$$PulseSendTime = PulseNo/PRF$$

For a 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);
```

5.2.2. FindPulsesInRange Function

At 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 $t = 0+PRI[0]$, pulse 2 will be sent at time $t = 0+PRI[0]+PRI[1]$ etc. Negative times (and pulse numbers are possible), e.g. pulse -1 will be sent at time $t = 0 - PRI[n-1]$. In Sarsim it is possible to define the interval between any two pulses. Let's 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 $PRI[x] = PRIArray[modulus(x, n)]$. Note that all times are given in seconds. Figure 5.2 will clarify what was explained above.

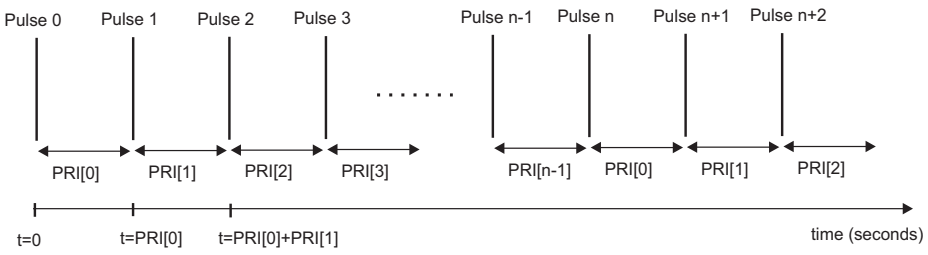

Figure 5.2: The 'PulseSendTime'

There are two cases :

- PRF is constant :

$$\begin{aligned} FirstPulse &= ceil(AzimuthStart \cdot PRF) \\ LastPulse &= floor(AzimuthEnd \cdot PRF) \end{aligned}$$

- 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)--;
```

5.2.3. FindPlatformPosition

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

5.2.4. FindPlatformVelocity

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

5.2.5. FindPlatformRotation

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

5.2.6. The frequency of each pulse

Single frequency case : $PulseFreq[PNo]$ = specified in RADAR dialog 1

Stepped frequency case : $PulseFreq[PNo] = StartFreq + floor(fmod(\frac{PNo}{FreqSteps \cdot PulsesPerFreq}, 1) \cdot FreqSteps + ROUNDERROR) \cdot StepSize$

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

User-defined case : $PulseFreq[PNo] = DataArray[fmod[PNo], arraysize]$

5.2.7. The time when each pulse is transmitted

Constant PRF : $PulseSendTime[PNo] = \frac{PNo}{PRF}$

User-defined PRIs : $PulseSendTime[PNo]$ = see source code

5.2.8. Range delay

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

5.2.9. 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 needs to be calculate :

$$PulseNo = (LastPulse - FirstPulse) + 1$$

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

$$PulseSendTime[PNo] = FindPulseSendTime(PNo + FirstPulse)$$

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

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

- Compute return gain factor independent of time :

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

with Radar->StartFreq being the center 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:

$$AntennaGainT = \left[\begin{array}{c} 1_{\text{for isotropic antennas}} \\ SinAntennaGain(OffsetAzi, AziBeamWidthT) \cdot SinAntennaGain(OffsetElev, ElevBea \end{array} \right.$$

- The return amplitude can be calculated with the formula

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

5.2.10. CalcOnePulse Function

Chirp Modulation

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

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

Monchrome Modulation

For monochrome pulse the DelaySlope would be zero.

$$DelaySlope = 0$$

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

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

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

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

$$PulseCenter = \begin{array}{l} 0 \text{ if Pulse is at the beginning of the point target} \\ \frac{PulseWidth}{2} \text{ if the pulse is at the center of the point target} \end{array}$$

and explained in figure 5.3. For real radars you would receive the pulse 'after' the point target location, however for simulations it is sometimes more convinient to have the point target in the center. All it really means is that the output array will be shifted by half a pulsewidth in range.

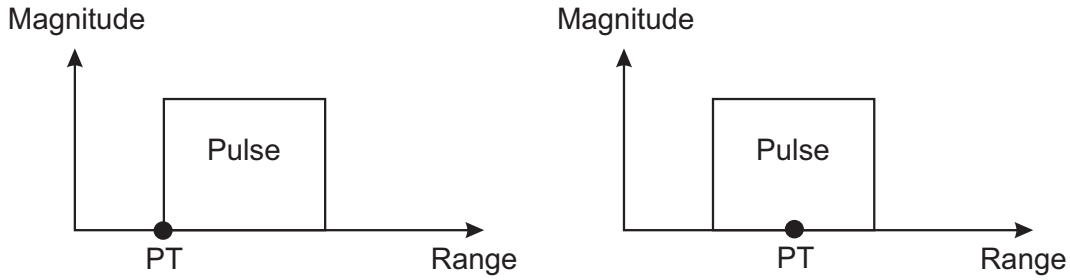

Figure 5.3: Point Target at the start of the pulse and at the center of the pulse

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 $(RangeDelay - \frac{1}{2}Pulsewidth)$ as shown in figure 5.4, such that the variable t goes from $-\frac{1}{2}Pulsewidth$ to $+\frac{1}{2}Pulsewidth$ over the pulse range.

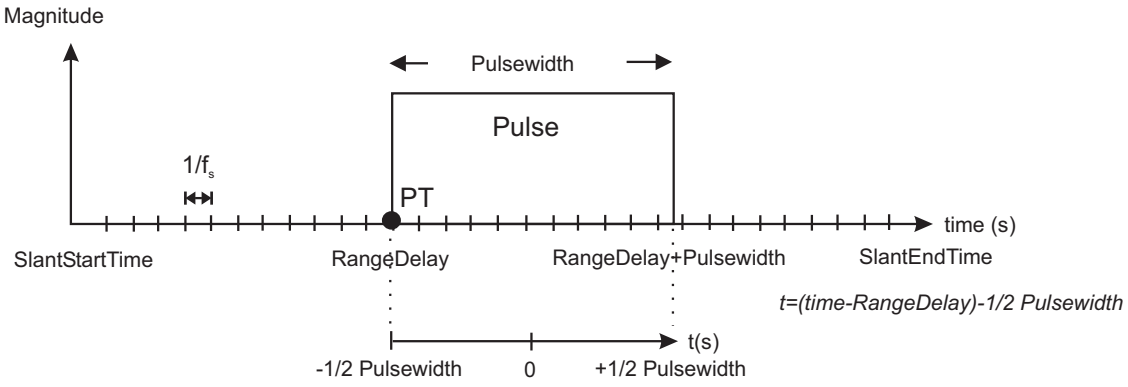

Figure 5.4: Positioning of pulse in range

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

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

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

The instantaneous frequency of the returned pulse at some point t (t = 0 at beginning of pulse as shown in figure 5.4) is :

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

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

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

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

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

$$\begin{aligned} I(t) &= ReturnAmp \cdot \cos(Phase(t)) \\ Q(t) &= ReturnAmp \cdot \sin(Phase(t)) \end{aligned}$$

6. Example Files

Various examples for different applications will be given now. The script files for all examples are available in the **EXAMPLES** subdirectory.

6.1. 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 6.1.

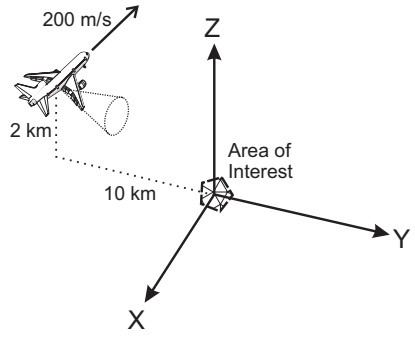

Figure 6.1: Geometry Setup

A sidelooking radar is mounted on the plane, the depression angle of the beam is 11.3 degrees. There have been 24 point target placed in an 'R' shape, extending over 200 x 400 m as shown in figure 6.2.

The radar parameters are as follows: chirp pulse with 100 MHz bandwidth at 5.3 GHz center frequency , 800 ns pulse width, 300 Hz PRF, 1 kW output power, no losses or noise, 30 degrees beamwidth.

First the PLANE platform is created. This is done by creating a new platform with the name 'PLANE' and defining a trajectory for it. The X-coordinate decreases linearly by 200 m/s therefore let's define 1) Position X = 100 m for Time = 0 s and 2) Position X = -100 m at time = 1 s. The Y-coordinate stays constant at -10000 m, therefore it is only necessary to define one point, e.g. Position X = -10000 m at time = 0 s. For the Z-coordinate (height) one point suffices, e.g. Position Z = 2000 m at time = 0 s. The plane will move with 200 meters in 1 second, being closest to the earth origin at 0.5 seconds.

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 6.2. For this setup a simulation has been stored already and can be recalled by selecting **Simulation / Previous**. The window shown in figure 6.3 will appear.

Obviously because the return signals of the point targets interfer 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

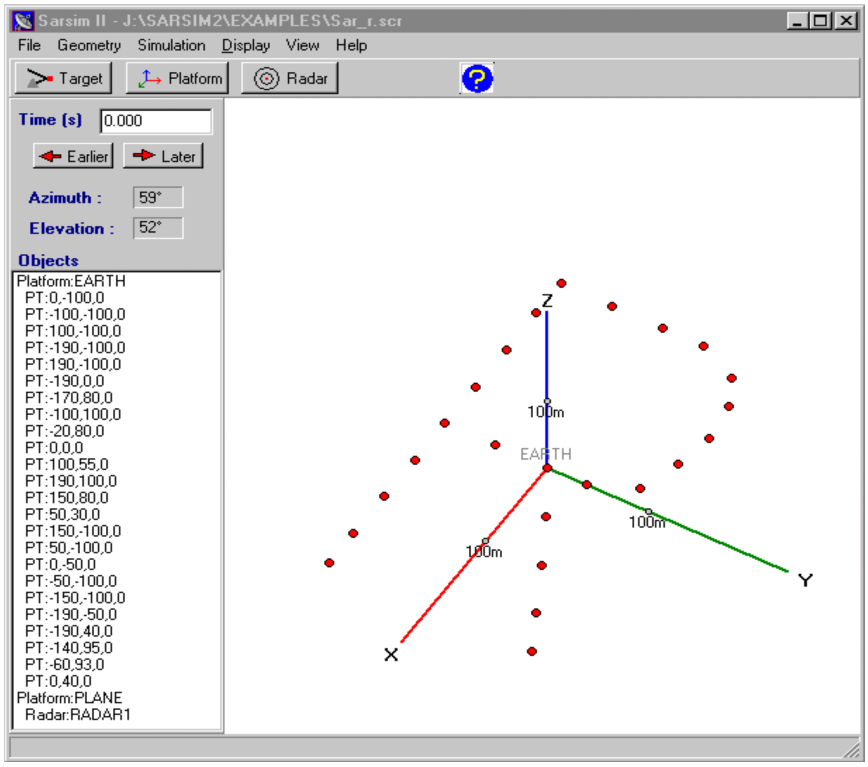

Figure 6.2: Point Target Setup

specific example the Chirp Scaling Algorithm has been applied) the image can be restored as shown in figure 6.4. 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 6.5 :

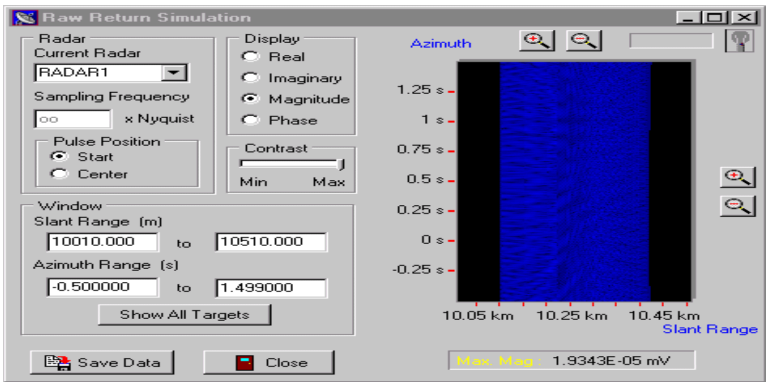

Figure 6.3: Simulation Window

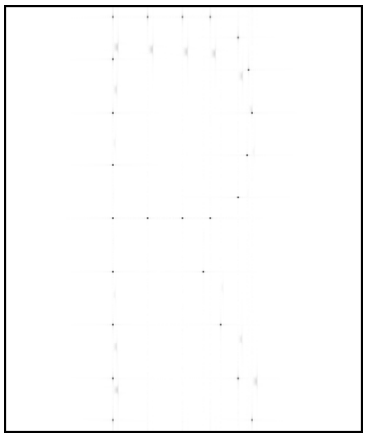

Figure 6.4: Image after processing

```
! 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 6.5: Shortened Script file for C-band SAR example

7. Glossary

- **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.
- **'Earth'** platform - a stationary, unshifted, non-rotated coordinate system, i.e. identical to the visible axes on the main screen
- **Point targets** are infinite 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 5.
- **Radars** are the actual energy-transmitting devices. They can be positioned onto any platform (however their position is always at origin of that platform), and there are numerous parameters to be set which will be explained in chapter 5.