Optimisation of Rotating, Range-Gated Fanbeam Scatterometer for Wind Retrieval

ESA ITT No. AO/1-3642/00/NL/DC

ESA Contract No. 14383/00/NL/DC

ESA Novation Agreement: ECT/NB/96.247

Task 2b Report:

Definition of Scatterometer Simulator

Change Record

Issue	Date	Page	Description
1	6.Dec.2001	all	Initial version reflection status at the RFSCAT Simulator Design Review (SDR) milestone
2	18.Mar.2002	all	Correction of errors found during implementation Update of S/W architectural design Update of pseudo level 1b file format Added detailed design (section 9)

Table of Contents

1 References

1 References

Applicable Documents

[AD1] ESA Novation Agreement: ECT/NB/96.247

[AD1] ESA Contract No. 14383/00/NL/DC

[AD3] Astrium GmbH Proposal No. A.2001-0122-0-1

[AD4] Statement of Work “Optimisation of Rotating, Range-Gated Fanbeam Scatterometer for Wind Retrievals APP-FP/99-10-222/CL/cl, 22.Nov.1999

Reference Documents

[RD1] Lin, C.C., Rommen, B., Wilson, J., Peter, P., An analysis of a rotating, range-gated, fanbeam spaceborne scatterometer concept, 2000

[RD2] Lin, C.C., Rommen, B. Rotating, range-gated, fanbeam radar – A new spaceborne scatterometer concept, Ocean Winds workshop, Ifremer, Brest, 19-22 June, 2000

[RD3] Task 1 Technical Report, Review of the Requirements and Scatterometer Concept, MPB, KNMI, IFARS , 17.Oct. 2000

[RD4] Optimisation of Rotating, Range-Gated Fanbeam Scatterometer for Wind Retrieval", Task 2a Report, GMF and wind field definition, and wind retrieval,KNMI, Dec.2001

[RD5] ERS Scatterometer System Simulator, Description of Simulation Model ER-MA-DSF-SY-0006 Issue 2A, 17.06.91

[RD6] ASCAT GPP, Level 1 Processor Algorithms, MO-RS-DOR-SC-0027, issue 1, 26.02.99

[RD7] ACAT GPP, Model Overview, MO-RS-DOR-SC-0024, issue 1, 31.7.98

[RD8] ASCAT Performance Algorithms, MO-TN-DOR-SC-0018, issue 2A, 30.04.01

[RD9] Press, W.H. et al., Numerical Recipes in C, Second Edition, Cambridge University Press, 1992, ISBN 0 521 43108 5

[RD10] “towards the real-time use of Quikscat winds", A. Stoffelen, J. de Vries and A. Voorrips, report: USP-2 00-26, Beleidscommissie Remote Sensing.

2 Introduction

2.1 Scope

This document defines the simulator to be used for end-to-end performance evaluation and optimisation of a rotating range gated fan-beam scatterometer. The definition covers the following subjects:

- Simulation Algorithm
- Architectural Software Design
- Definition of External Interfaces
- Software Verification Concept

The document is foreseen to be maintained through run of study, to reflect the actually implemented version of the software.

2.2 Structure of the Document

Section 1 lists the applicable and referenced documents.

Section 2 (this section) describes the scope and structure of this document. To make the document self-standing it furthermore contains a brief introduction of the instrument concept and simulator objectives.

Section 3 specifies the algorithms chosen to meet the simulator objective.

Section 4 defines the S/W architecture of the RFSCAT system simulator

Section 5 describes the S/W verification approach.

Section 6 lists the parameters that are modifiable input to RFSCAT system simulator.

Section 7 gives a detailed specification of the file formats used at the external interfaces of RFSCAT system simulator.

Section 8 contains the list of acronyms used in this document.

Section 9 contains the detailed design of the pseudo level 1 b generator.

2.3 Instrument Concept

The instrument being subject to this study is a rotating range-gated fanbeam scatterometer. It will be addressed as RFSCAT in following. The measurement principle for RFSCAT is shown in Figure 2-1.

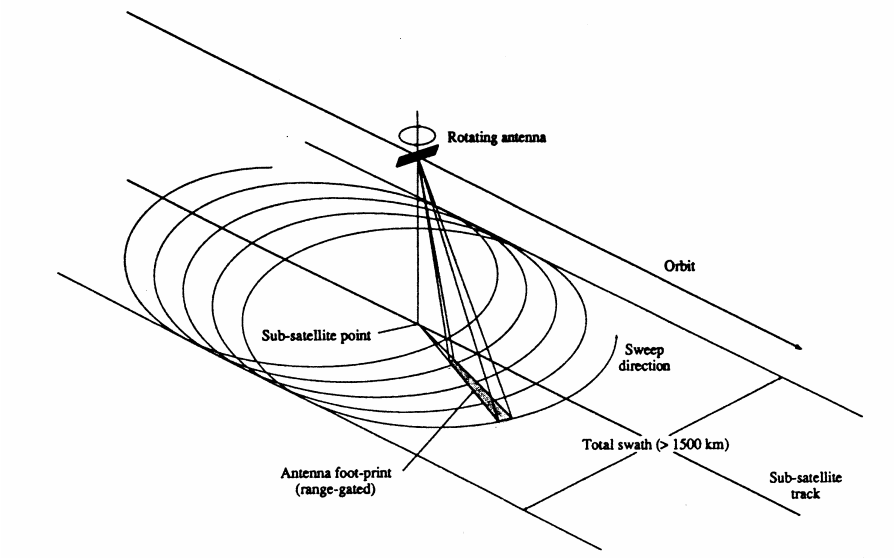


Figure 2-1 Rotating Fanbeam Scatterometer Concept

RFSCAT is a spaceborne system on a polar orbiting platform at a height of 725 km. The antenna rotates at a rate of 0.35 rad/s, while the satellite travels with a velocity of 7.49km/s, resulting in an epicyclical footprint on the earth′s surface with a total swath width of 1500 km. The incidence angle from a fanbeam varies from 28^0 to 51^0. The radar operates at C-band, 5.3 GHz, and transmits and receives vertically or horizontally polarised signals (VV, HV). The radar operates in a pulsed mode, so that each point of the echo profile can be attributed to a unique pixel position within the antenna footprint along the radial direction (range-gated). The instrument parameters listed in Table 2-1. The simulator will allow to vary all the listed parameters within reasonable ranges.

Parameter	Option 1	Option 2
Orbit / repeat cycle	725 km / 2 day	1075 km/2 days
Antenna scan rate	0.35 rad/s (3.3 rpm)	0.25 rad/s (2.4 rpm)
Total swath	1500 km	1600 km
Min. wind speed	≤ 4 m/s cross-wind	≤ 4 m/s cross-wind
Incidence angle	28^0 - 51^0	21^0 – 42.5^0
Footprint length	408.6 km	450.5 km
Frequency	5.3 GHz	5.3 GHz
Polarisation	VV	VV
Pulse bandwidth	1.06 MHz	1.39 MHz
Pulse compr. Ratio	200	200
Pulse rep. Frequency	239 Hz	179
Nrange	50 (HR)	50 (HR)
Nazimuth	13.7 (HR)	13.4 (HR)
Spatial resolution	15 km × 15 km (HR) 50 km x 50 km (wind)	15km x 15 km (HR) 50 km x 50 km (wind)
Radiometric resolution	≤ 0.3 dB (HR) ≤ 0.09 dB (wind)	≤ 0.3 dB ≤ 0.11 dB (wind)
Average RF power	145 W	83 W
Antenna size	3.6 m × 0.4 m	4.6 m x 0.4 m

Table 2-1: Parameter Sets for RFSCAT Option 1 and Option 2

2.3.1 Variation of the Polarisation and two Beam Antennas

The standard RFSCAT concept is an antenna, where send and receive signal are vertically polarised (VV). More advanced concepts, combining information of vertically and horizontally polarised signals are expected to cause a major improvement to wind retrieval. Therefore, the simulator will allow evaluation of the following scenarios:

- Option A: VV or HH, one beam (HH: send horizontally polarised, receive horizontally polarised).
- Option B: Alternating signal HH and VV, one beam.
- Option C: Two beams with 180^0 different look angle, one beam VV the other HH.
- Option D: Two beams, one for near swath, one for far swath. Option A produces many measurements with similar look-angles for the mid swath region. The idea of option D is to reduce the highly redundant information from this region and allow at the same time better adaptation of the antennas to near and far swath geometry.
- Option E: Polarimetric measurements (VV+VH). In this option, the antenna sends one vertically polarised signal and receives the vertically polarised echo and the horizontally polarised echo with two separate receivers at the same time.

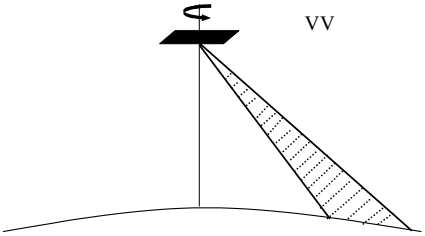


Figure 2-2 Option A, One Beam, VV

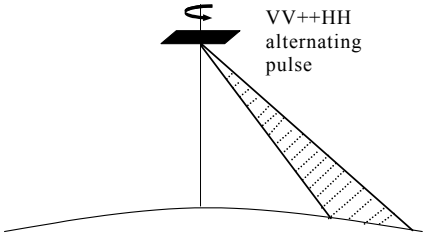


Figure 2-3 Option B, One Beam, VV+HH alternating

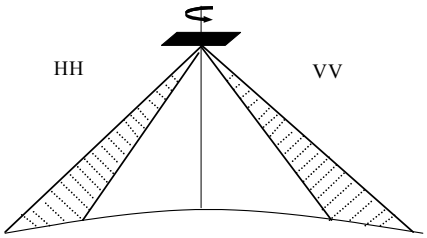


Figure 2-4 Option C, Two Beams, One VV, One HH

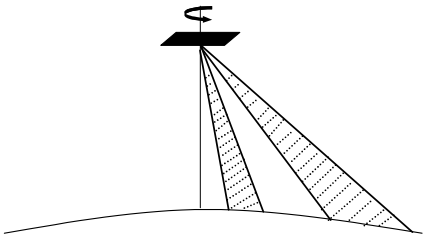


Figure 2-5 Option D, Two Beams, One Near Swath, One far Swath

2.4 Simulation Objectives

The overall goal of the RFSCAT study is the end-to-end performance analysis of windfield retrieval with the RFSCAT system. The simulation will start with an input windfield and a set of parameters defining the antenna system and the applied processing. The output will be a set of possible wind solutions for each grid point (WVC). The difference between input windfield and output windfield are expressed as a Figure-Of-Merit (FoM). The FoM will be the metric used for the optimisation the RFSCAT system.

3 Simulation Algorithm

For real existing scatterometer systems, the dataflow can be described by the following linear sequence:

1. A windfield generates surface waves on the ocean.
2. The scatterometer antenna sends a signal to the sea surface.
3. The signal is backscattered by the sea-surface.
4. The backscattered signal is received by the scatterometer antenna.
5. The received signal is downlinked to ground station as Level 0 data (on-board processing is not mentioned here for simplicity).
6. The level 0 data is transformed to physical units, calibrated and geolocated by the level 1 ground processor. The output is a level 1b product.
7. The level 2 processor computes the retrieved windfield from the data of the level 1 product. Output is a level 2 product.

A strict simulation of steps 2 to 6 would imply that all systematic errors of the measurements have to be introduced to the simulated level 0 data, and removed by the simulated level 1 processor afterwards. The objective of the RFSCAT system simulator is not the development of a level 1b processing algorithm, but the overall end-to-end performance evaluation. Therefore, a more efficient approach with respect to runtime will be implemented applying the approximation, that the level 1b processor is able to remove all significant systematic errors.

The simulation will replace steps 2 to 6 by the computation of the geometry, geolocation and variance of each measurement. These data will be stored in a so-called „Pseudo-level 1 b product“. Given a specific windfield, the theoretical backscatter coefficient for each measurement can be computed by the geophysical modelling function. After noise is added according to the variance, a level 1b product can be written and step 6 is reached.

The output windfield can then be computed by the level 2 processor (wind retrieval module).

The final processing step will then be the comparison of input windfield with output windfield, by the “Windfield Comparison Module”.

3 Simulation Algorithm

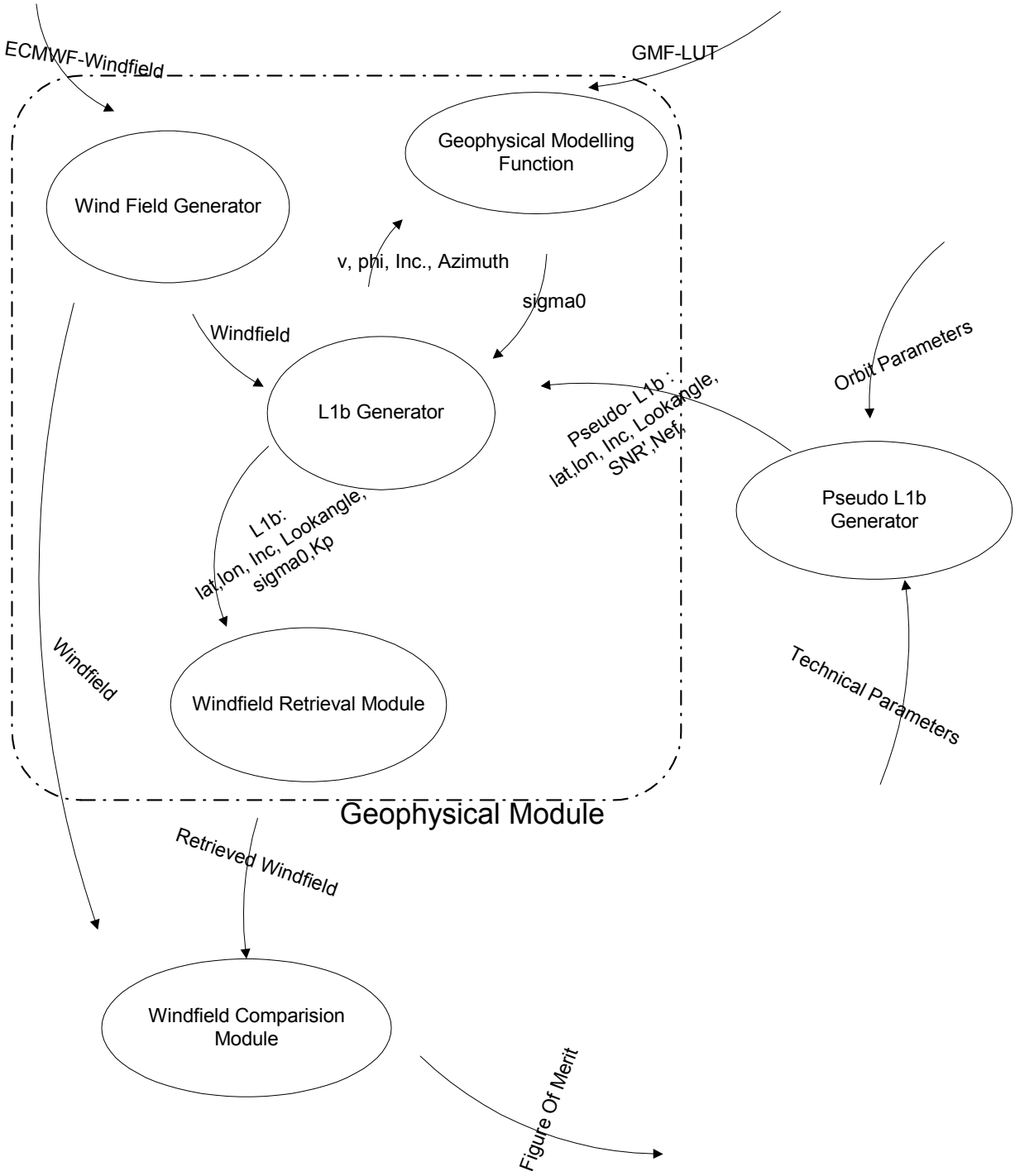


Figure 3-1 Overall Dataflow of the RFSCAT System Simulator

3 Simulation Algorithm

3.1 The Windfield Generator

For testing purposes, an artificial set of windvectors has been created. These have been chosen randomly from a distribution that is close to the distribution found on average in the real world.

To verify this approach a test will be done using a windfield produced by the ECMWF forecast model as input for this simulation.

To test the effect of the scatterometry measurement on coherent structures in the wind it is also possible to insert an artificial windfield (rotational, laminar, etc.).

The effect of geophysical noise (i.e. local variations of the wind inside the region of the WVC) has been added to the instrument noise Kp.

For details refer to the task 2a report [RD4].

3.2 The Geophysical Modelling Function

The Geophysical Model Function (GMF) is the function providing the radar backscatter coefficient. Its parameters are windspeed, relative look azimuth (compared to wind direction) and incidence angle.

The GMF is a function devised to describe the available experimental data, and differs for different wavelengths and different polarisation of the radar beam. It is used in the simulation programme in the form of a Look Up Table (LUT) that is read from file.

More details on the construction of these functions, and the LUT in which they are stored, are discussed in the task 2a report [RD4].

3.3 The Pseudo Level 1 b Generator

3.3.1 Definitions

3.3.1.1 Nodes

Nodes are areas on the earth surface located around the subsatellite track. Nodes are artificial quantities introduced by the level 1 processing. The idea of nodes is to average many measurements by spatial filtering to one quantity with a low variance. Following the ERS-Scat and ASCAT processing philosophy, the RFSCAT simulator nodes will be (nearly) equidistant squares (see Figure 3-2).

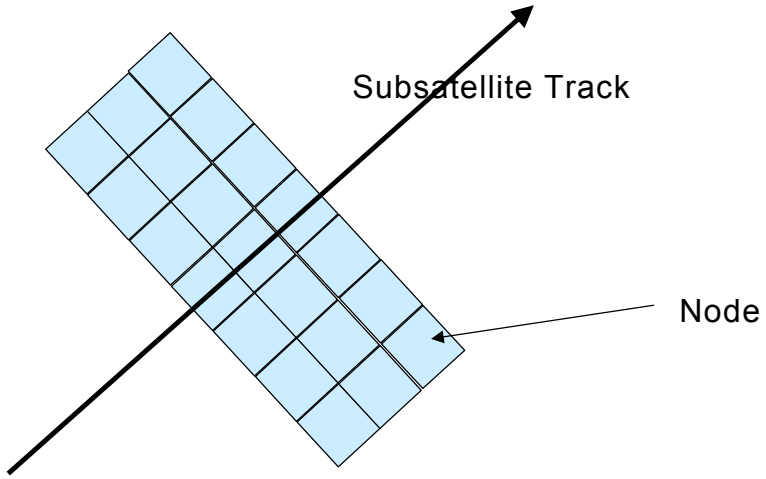


Figure 3-2 Nodes are equidistant Squares around the Subsatellite Track.

3.3.1.2 Samples

Samples represent measurements of the RFSCAT instrument (see Figure 3-3). The signal of each sample represents an area on ground of the size of the effective spatial resolution of the instrument. The effective spatial resolution in the plane perpendicular to the antenna beam axis (azimuth) is a function the antenna pattern and the measurement geometry.

The spatial resolution along the beam axis (range) is furthermore a function of pulse duration the chirp rate and the local Doppler rate. With baseline options (see Table 2-1) range processing of each pulse will result into approximately 2000 samples.

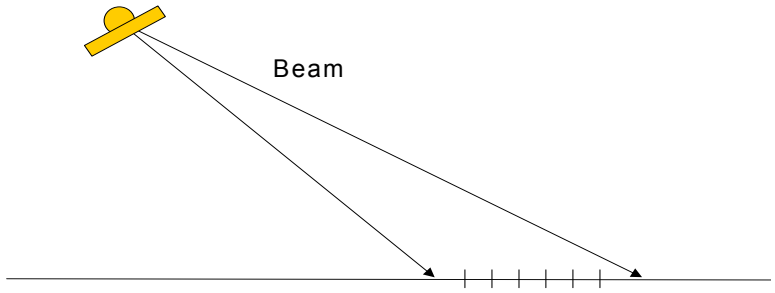


Figure 3-3 Samples are the output of range processing of the returned echo of a pulse.

3.3.1.3 Views

Views (see Figure 3-4) are a collection of samples with similar azimuth angle, belonging to one node. For each view, the level 1b processor will compute one sigma naught value by weighted averaging over its samples.

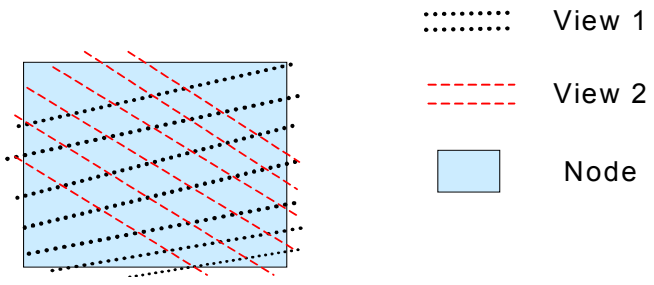


Figure 3-4 Views: The picture shows one node with two views.

3.3.1.4 Co-ordinate Systems

True Inertia System

The true inertia system is used by the orbit propagator. It is earth centred, but does not follow the earth rotation around the pole-axis. Note that the x-y-plane is in the equator plane of the earth.

Centre	Centre of Earth
x	to True Vernal Equinox
y	Complete to Right System
z	to Celestial Pole

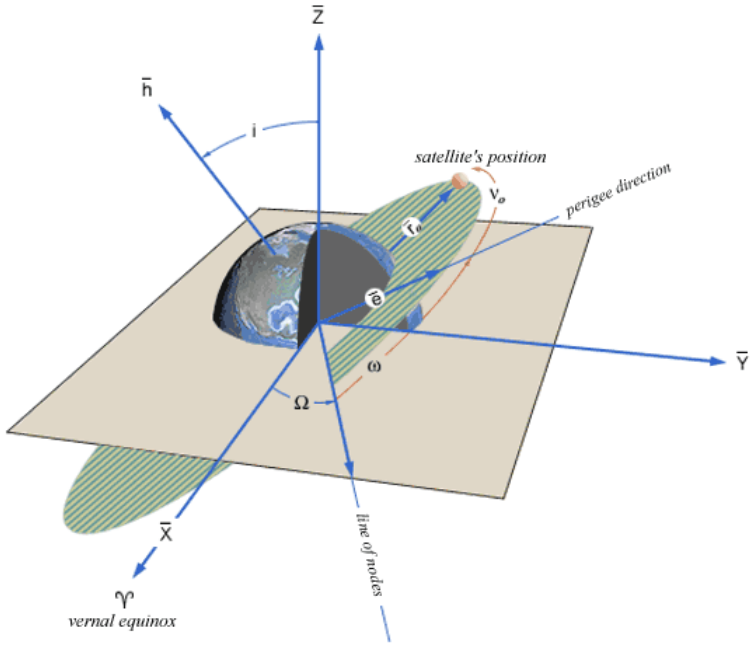
Earth-Fixed co-ordinates

This co-ordinate system is used in most parts of the algorithm. It is earth centred and rotates with the earth around the pole-axis:

Centre	Centre of Earth
x	from Earth Centre to Intersection between Equator and Greenwich Meridian
y	Complete to Right System
z	to Celestial Pole

Osculating Keplerian Elements

The orbit propagator defines the satellites location in phase-space using the osculating Keplerian elements.



a - defines the size of the orbit
e - defines the shape of the orbit
i - defines the orientation of the orbit with respect to the Earth's equator.
ω - defines where the low point, perigee, of the orbit is with respect to the Earth's surface.
Ω - defines the location of the ascending and descending orbit locations with respect to the Earth's equatorial plane.
ν - defines where the satellite is within the orbit with respect to perigee.

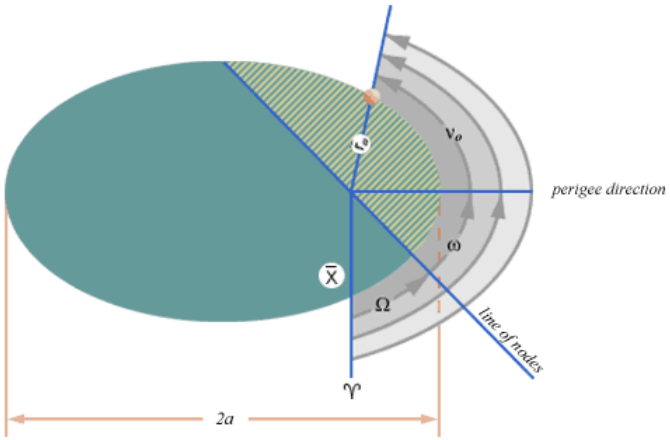


Figure 3-5 Visualisations of the Osculating Keplerian Elements

a	Semi-major axis of the orbit ellipse
e	Eccentricity of the orbit ellipse
ν	True anomaly: The angle between the vector to the spacecraft and the vector to the perigee
Ω	Right ascension of ascending node: The angle between the intersection of the orbit plane with the equatorial plane (ascending side of orbit) and the vector to true vernal equinox
i	Inclination: The angle between the equatorial plane and the orbit plane
ω	Argument of perigee: The angle between the intersection of the orbit plane with the equatorial plane (ascending side of orbit) and the vector to perigee

Satellite Fixed System

The satellite fixed system is used to compute the position of the rotating antenna. Its x,y,z axis are defined relative to the satellite position. The expression “Nadir” is the point on earth surface, which has the satellite in direction of normal vector of the tangent plane. Note that in general the nadir is not the intersection of the vector to the satellite with the earth surface.

Centre	Centre of Satellite
x	in the plane defined by the vector from satellite to nadir and the ground speed vector, perpendicular to z
y	Complete to Right System
z	from Satellite to Nadir

3.3.2 Top Level Control Flow

For a given σ^0 and view the standard deviation. k_p can be computed according to:

$$K_p = \sqrt{\frac{1}{N^{eff}}\left(1+\frac{1}{SNR'\cdot\sigma^0}\right)^2 + \frac{1}{N^{noise}}\left(\frac{1}{SNR'\cdot\sigma^0}\right)^2} \qquad (3.3.2\text{-}1)$$

with σ^0 being a function of wind-vector, look-angle and incidence-angle. As the pseudo Level 1b product processor shall be independent of windfield under investigation the output of the pseudo level 1b processor will be the quantities on right side of equation (3.3.2-1) for each view, except for σ^0. Furthermore, all geometrical quantities needed to compute σ^0 for a given windfield will be reported.

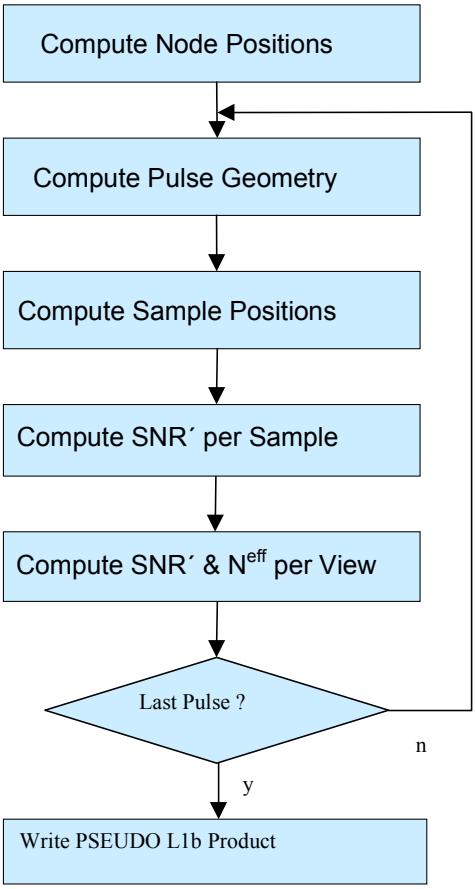


Figure 3-6 Top Level Control Flow of the Pseudo Level 1 b Generator

To achieve this goal the Pseudo-Level 1 b processor performs the operations outlined in Figure 3-6).

First the positions of all nodes are computed. Then processor enters a loop over all pulses within the selected time interval. For each pulse the pulse specific quantities are computed (satellite position, antenna position, range resolution as function of the Doppler rate…). Using these quantities the position of each sample and its normalised signal to noise ratio SNR'=SNR/σ^0 can be computed. Using the sample location and the corresponding pulse azimuth angle, it can be decided which view (or views) a sample belongs to. The view specific data SNR` and N^{eff} is updated accordingly. The view data is computed by weighted averaging over the corresponding sample quantities:

$$X_{view} = \frac{\sum_{samples} w_{sample} X_{sample}}{\sum_{samples} w_{sample}}$$

Storing the numerator and the denominator separately will allow continuous updates of the view within the loop over the pulses. After processing of the last pulse the pseudo level 1b product can be written.

3.3.3 Computation of Node Positions

This algorithm computes a grid of nodes on the earth surface. The middle nodes are located (nearly) equidistant along the subsatellite track. The other nodes are located left and right of the middle nodes on the line perpendicular to the locale sub-satellite track velocity with the same distance such that the full swath is covered by nodes. The algorithm uses the orbit propagator to compute the sub satellite positions and the ground velocities. The line for the side nodes is computed as the intersection of the plane with normal to the ground-velocity vector and the earth ellipsoid. The intersection is an ellipse. The nodes are finally computed by variation of the angle between the ellipse semi-major axis and the vector to the nodes-centre position. The proposed algorithm assumes low eccentricity of the orbit.

Input

All orbit parameters PL1-10 to PL1-130 and

ID	Name	Symbol	Unit	Dim.	Type	Description
PL1-380	p_Dnode	d^{Node}	m	0	double	Distance of Nodes
PL1-390	p_TotalSwath	d^{Swath}	m	0	double	Total width of swath

Output

Symbol	Unit	Dim.	Type	Description
$N^{NodesPerRow}$	m	0	double	Number of Nodes in on across-track line
$\vec{N}_{i,j}$	m	0	double	Position of centre of Nodes in earth fixed co-ordinates. The first index increases with time (along track) starting from 0, the second index is from the left most node to the right most node across-track

Detailed Algorithm

i. Compute the number of nodes in one across-track row :

$$N^{NodePerRow} = \text{int}(\frac{d^{Swath}}{2 \cdot d^{Node}}) \cdot 2 + 1$$

ii. Compute average ground speed $| v_{Ground}^{avg} |$ using the orbit propagator

iii. Compute time increment $\Delta t = \frac{d^{Node}}{| v_{Ground}^{avg} |}$

iv. Compute the time vector $t_i = \Delta t \cdot i$ for $i = 0..\frac{T^{dur}}{\Delta t}$

v. Compute the middle node position $\vec{G}_i$ as the nadir (local normal) point of the satellite t_i for all i

vi. Compute normal vector of the ground speed numerically $\vec{v}_n{}^G = \frac{\vec{G}_i - \vec{G}_{i+dt}}{dt \cdot | \vec{v}_n{}^G |}$

vii. Compute for each t_i the vectors $\vec{A}_i$ (semi-major axis), $\vec{B}_i$ (semi-minor axis) and $\vec{C}_i$ (centre) of the ellipse that is defined by the intersection of the earth ellipsoid and the plane with the normal vector $\vec{v}_n{}^G$

viii. Compute the vector from ellipse centre to the central node: $\vec{N}^C{}_i = \vec{G}_i - \vec{C}_i$

ix. Compute the angle β_i such that $\vec{N}^C{}_i = \vec{A}_i \cos(\beta_i) + \vec{B}_i \sin(\beta_i) + \vec{C}_i$, i.e.

$$\beta_i = \arctan 2(\frac{\vec{B}_i \cdot \vec{N}_i^C}{\vec{B}_i^2}, \frac{\vec{A}_i \cdot \vec{N}_i^C}{\vec{A}^2{}_i})$$

x. Compute the ellipse angles to the side nodes:

$$\alpha_{i,j} = \frac{(j - \frac{N^{NodesPerRow} - 1}{2}) \cdot d^{Node}}{\sqrt{\vec{A}^2{}_i \cdot \cos^2 \beta_i + \vec{B}_i^2 \cdot \sin^2 \beta_i}} \text{ with } j=0 \ldots N^{NodesPerRow}$$

xi. Compute the node positions as $\vec{N}_{i,j} = \vec{A}_i \cos(\alpha_{i,j}) + \vec{B}_i \sin(\alpha_{i,j}) + \vec{C}_i$

3.3.4 Computation of the Intersection of a Plane with the Earth Ellipsoid

This function computes the intersection of a plane with the earth ellipsoid. The plane is defined by one point that is on the earth surface and its normal vector. The result of the intersection is an ellipse.

Input

Symbol	Unit	Dim.	Type	Description
$\vec{G}_{E,g}$	m	1	double	Vector from Centre of earth to Nadir, i.e. one point on the ellipsoid and in the plane
$\dot{\vec{G}}_{E,g}$	m	1	double	Normal vector to the plane, here the normal vector in direction of the ground track velocity

Output

Symbol	Unit	Dim.	Type	Description
$\vec{C}_{E,g}$	m	1	double	Vector to centre of ellipse
$\vec{A}_{E,g} \cdot a_E$	m	1	double	Semi-major axis of ellipse
$\vec{B}_{E,g} \cdot b_E$	m	1	double	Semi-minor axis of ellipse

Detailed Algorithm

This result ellipse is determined by the following parameters:

- the centre $\vec{C}_{E,g}$
- the unit vectors $\vec{A}_{E,g}$, $\vec{B}_{E,g}$ in the direction of the semi-major and semi-minor axis
- and the lengths a_E and b_E of the semi-major axis and semi-minor axis.

The ellipse centre $\vec{C}_{E,g}$ is given by:

$$\vec{C}_{E,g} = \frac{x_{\vec{G}_g} \cdot x_{\vec{G}_g} + y_{\vec{G}_g} \cdot y_{\vec{G}_g} + z_{\vec{G}_g} \cdot z_{\vec{G}_g}}{A^{Earth^2} \cdot x^2_{\vec{G}_g} + A^{Earth^2} \cdot y^2_{\vec{G}_g} + B^{Earth^2} \cdot z^2_{\vec{G}_g}} \cdot \begin{pmatrix} A^{Earth^2} \cdot x_{\vec{G}_g} \\ A^{Earth^2} \cdot y_{\vec{G}_g} \\ B^{Earth^2} \cdot z_{\vec{G}_g} \end{pmatrix}$$

The unit vectors $\vec{A}_{E,g}$, $\vec{B}_{E,g}$ in the direction of the semi-major and semi-minor axis are:

$$\vec{A}_{E,g} = \frac{1}{\sqrt{x^2_{\vec{G}_g} + y^2_{\vec{G}_g}}} \cdot \begin{pmatrix} y_{\vec{G}_g} \\ -x_{\vec{G}_g} \\ 0 \end{pmatrix}$$

and

$$\vec{B}_{E,g} = \frac{1}{|\vec{G}_g| \cdot \sqrt{x^2_{\vec{G}_g} + y^2_{\vec{G}_g}}} \cdot \begin{pmatrix} x_{\vec{G}_g} \cdot z_{\vec{G}_g} \\ y_{\vec{G}_g} \cdot z_{\vec{G}_g} \\ -x^2_{\vec{G}_g} - y^2_{\vec{G}_g} \end{pmatrix}$$

For the semi-major axis a_E we obtain

$$a_E = \left| \frac{-B - \sqrt{B^2 - 4 \cdot A \cdot C}}{2 \cdot A} \right|$$

where

$$A = \frac{x_{\vec{A}_{E,g}}^2 + y_{\vec{A}_{E,g}}^2}{A^{Earth^2}} + \frac{z_{\vec{A}_{E,g}}^2}{B^{Earth^2}},$$

$$B = 0,$$

$$C = \frac{x_{\vec{C}_{E,g}}^2 + y_{\vec{C}_{E,g}}^2}{A^{Earth^2}} + \frac{z_{\vec{C}_{E,g}}^2}{B^{Earth^2}} - 1.$$

For the semi-minor axis b_E we obtain

$$b_E = \left| \frac{-B - \sqrt{B^2 - 4 \cdot A \cdot C}}{2 \cdot A} \right|$$

where

$$A = \frac{x_{\vec{B}_{E,g}}^2 + y_{\vec{B}_{E,g}}^2}{A^{Earth^2}} + \frac{z_{\vec{B}_{E,g}}^2}{B^{Earth^2}},$$

$$B = 0,$$

$$C = \frac{x_{\vec{C}_{E,g}}^2 + y_{\vec{C}_{E,g}}^2}{A^{Earth^2}} + \frac{z_{\vec{C}_{E,g}}^2}{B^{Earth^2}} - 1.$$

3.3.5 Usage of the Orbit Propagator

Input

All orbit parameters PL1-10 to PL1-130 and vector of times where the output shall be computed.

Output

Symbol	Unit	Dim.	Type	Description
$\vec{G}_{E,g}$	m	1	double	Vector from centre of earth to nadir for all input time points in earth-fixed co-ordinates
$\vec{r}_E^{Sat}$	m	1	double	Vector to spacecraft for each time point in earth fixed co-ordinates

Detailed Algorithm

Usage of the orbit propagator from the STARS library is performed in several steps:

i. Compute Kepler elements for each time step t (STARS_KeplerJ2Orbit)

ii. Transform the osculating Kepler elements for each time step into Cartesian co-ordinates of the true inertia system (STARS_Cartesian_Kepler)

iii. Compute the modified Julian date corresponding to each time step (STARS_addModJulDat)

iv. Compute the Greenwich Mean Sideral Time (Earth rotation angle) for each modified Julian date (STARS_gmst)

v. Compute the transformation matrix EI from true inertia to earth fixed co-ordinates for each time step (STARS_EQ_Matrix).

vi. Apply the matrix EI to compute the Satellite location in earth fixed co-ordinates $\vec{r}_E^{Sat}$

vii. Compute the local normal vector (from spacecraft to nadir) $\vec{r}_E^{LN}$ for each spacecraft position (STARS_localNormal)

viii. Compute the vector from centre of earth to nadir : $\vec{G}_{E,g} = \vec{r}_E^{Sat} + \vec{r}_E^{LN}$

3.3.6 Loop over all Pulses

As displayed in Figure 3-6 the processing steps of pseudo level 1 generator after the computation of the node positions are performed in a loop over all pulses. This approach was selected, because it is not be feasible to store all pulse and sample specific data in RAM at the same time. With a pulse repetition frequency of 240 Hz and an orbit duration of 100 min the RFSCAT system will produce about 864000 Pulses per orbit, each with about 2000 samples.

The loop skips every f_{noise}-th. pulse, as the corresponding receive window is used for noise measurements.

Input

ID	Name	Symbol	Unit	Dim.	Type	Description
PL1-70	k_duration	T^{Dur}	s	0	int	Duration of simulated period
PL1-170	i_prf	f^{pulse}	Hz	0	double	Pulse Repetition Frequency
PL1-180	i_NoiseRatio	f_{noise}	-	0	int	Frequency of noise measurements relative to the pulse repetition frequency. Every f_{noise}-th, pulse is skipped and a noise measurement takes place instead.

Input used by the called algorithms is listed there.

Output

Symbol	Unit	Dim.	Type	Description
N^{Pulse}	-	0	double	Number of pulses during the simulated period
t_i	s	0	double	Time of each pulse starting with t=0

The top-level output computed as the view specific data is listed in the corresponding algorithm definition.

Detailed Algorithm

i. Compute the number of pulses $N^{Pulse} = T^{Dur} \cdot f^{pulse}$

for $i = 0$ to $i = N^{Pulse} - 1$ do

if (i mod $f_{noise} \neq 0$) then

ii. $t_i = \frac{i}{f^{pulse}}$

iii. Compute the pulse specific geometrical quantities for t_i

iv. Compute sample positions for all samples of the current pulse

v. Compute SNR´ for each sample

vi. Update the view specific quantities for all affected views

end if not a noise measurement

end loop over all pulses

3.3.7 Computation of Pulse Geometry

This algorithm computes the position of the satellite and the rotating antenna for given time point t. For the satellite this is simple done by the usage of the orbit propagator, for the antenna a co-ordinate transformation has to applied. The antenna rotates with constant angular speed with respect to the satellite. The satellite is assumed to be nadir (local normal) pointing to avoid that swath moves relative to satellite. Furthermore, it is assumed that yaw steering is performed. This might not be strictly necessary, but it has the impact that a-cross track azimuth line will always have zero Doppler shift. The function may easily be modified to switch off yaw steering in order to evaluate the impact on system performance.

The transformation matrix from earth fixed co-ordinates to the satellite fixed co-ordinates can be computed by feeding the STARS_IN_Matrix routine with the vector from centre of earth to the satellite and the vector of ground speed velocity, both vectors in earth fixed co-ordinates. If no yaw-steering shall be assumed the same routine may be used, but it has to be fed with the nadir vector in the true inertia system and the satellite velocity in the true inertia system.

Input

ID	Name	Symbol	Unit	Dim.	Type	Description
PL1-150	i_phi0	$\varphi^{ant}{}_0$	rad	0	double	Angle of antenna beam wrt. satellite velocity at t=0
PL1-160	i_scanRate	ω^{scan}	rad/s	0	double	Antenna scan rate
		t_i	s	0	double	Time of the pulse, for which the quantities shall be computed

Output

Symbol	Unit	Dim.	Type	Description
$\vec{G}_{E,g}$	m	1	double	Vector from centre of earth to nadir for all input time points in earth-fixed co-ordinates
$\dot{\vec{G}}_{E,g}$	m/s	1	double	Ground velocity vector
$\vec{r}_E^{\,Sat}$	m	1	double	Vector to spacecraft for each time point in earth fixed co-ordinates
$\vec{v}_E^{\,Sat}$	m	1	double	Vector of spacecraft velocity for each time point in earth fixed co-ordinates
$\vec{r}_E^{\,Ant}$	m	1	double	Normal vector to of antenna axis in earth-fixed co-ordinates
$\vec{r}_E^{\,Beam}$	m	1	double	Normal vector in direction of the antenna beam in the rotation plane of the antenna. $\vec{r}_E^{\,Ant}$ and $\vec{r}_E^{\,Beam}$ are in the same plane, but with a phase shift of 90 degrees.

Detailed Algorithm

i. Compute the unit vector $\vec{r}_{SatS}^{Ant}$ in antenna direction at time t_i in the satellite fixed co-ordinates

$$\varphi_i = \varphi_0 + \omega^{Scan} \cdot t_i$$

$$\vec{r}_{SatS}^{Ant} = \begin{pmatrix} \cos\varphi_i \\ \sin\varphi_i \\ 0 \end{pmatrix}$$

ii. Compute the unit vector projection of the beam direction $\vec{r}_{SatS}^{Beam}$ into the satellite system x-y plane. Without loss of generality this is assumed to be perpendicular to the antenna vector:

$$\vec{r}_{SatS}^{Beam} = \begin{pmatrix} \sin\varphi_i \\ -\cos\varphi_i \\ 0 \end{pmatrix}$$

iii. Compute satellite position $\vec{r}_E^{Sat}$ satellite velocity $\vec{v}_E^{Sat}$, nadir position $\vec{G}_{E,g}$ and ground velocity $\dot{\vec{G}}_{E,g}$ at time t_i in earth fixed co-ordinates using the orbit propagator. The velocities are computed numerically, by calling the orbit propagator for t'=t+dt.

iv. Compute the transformation Matrix from satellite system to earth fixed system using the Stars_IN_Matrix function :

$$\vec{u}_z = \frac{\vec{G}_{E,g} - \vec{r}^{\,Sat}}{|\vec{G}_{E,g} - \vec{r}^{\,Sat}|} \qquad \vec{u}_y = \frac{\vec{u}_z \times \dot{\vec{G}}_{E,g}}{|\vec{u}_z \times \dot{\vec{G}}_{E,g}|} \qquad \vec{u}_x = \vec{u}_y \times \vec{u}_z$$

$$IN = (\vec{u}_x, \vec{u}_y, \vec{u}_z)^T$$

v. Apply the transformation matrix IN to obtain antenna and beam unit vectors $\vec{r}_E^{Ant}, \vec{r}_E^{Beam}$ in earth fixed co-ordinates

3.3.8 Computation of the Position of one Sample

This algorithm computes the position of one sample on the earth surface, which corresponds to a certain pulse and a receive time that is defined relative to time of transmit. The sample satisfied three conditions:

- It is on a sphere's surface around the satellite position. The radius of the sphere is defined by the time of reception. The signal has to travel the radius two times (forward and backward) until reception.
- It is on the earth-ellipsoid.
- It is in the plane defined by the antenna beam.

Hence, the sample position has to be computed as the intersection of sphere with a plane and an ellipsoid. As this algorithm is to be performed for every sample of every pulse special attention needs to be drawn on runtime efficiency.

Input

Symbol	Unit	Dim.	Type	Description
t^{Rec}	s	1	double	Receive time, relative to transmission of pulse
$\vec{r}_E^{\,Sat}$	m	1	double	Vector to spacecraft at time of transmit for in earth fixed co-ordinates
$\vec{r}_E^{\,Ant}$	m	1	double	Normal vector to of antenna axis in earth-fixed co-ordinates
$\vec{r}_E^{\,Beam}$	m	1	double	Normal vector in direction of the antenna beam projected into the rotation plane of the antenna

Output

Symbol	Unit	Dim.	Type	Description
$\vec{r}_E^{\,Sample}$	m	1	double	Position of Sample in earth fixed co-ordinates

Detailed Algorithm

i. Compute the distance from sample to satellite $d = \frac{c \cdot t^{Rec}}{2}$

ii. Solve the non-linear equation system:

(1) $(\vec{r}_E^{\,sample} - \vec{r}_E^{\,Sat})^2 - d^2 = 0$

(2) $\frac{x_E^{Sample^2}}{A^{Earth^2}} + \frac{y_E^{Sample^2}}{A^{Earth^2}} + \frac{z_E^{Sample^2}}{B^{Earth^2}} - 1 = 0$

(3) $(\vec{r}_E^{\,sample} - \vec{r}_E^{\,Sat}) \cdot \vec{r}_E^{\,Ant} = 0$

iii. Choice the real solution where projection on the beam direction $(\vec{r}^{\,sample} - \vec{r}^{\,Sat}) \cdot \vec{r}^{\,Beam}$ has the bigger value.

The equation system can be solved iterative using Newton's method (as implemented in [RD9]). A good starting point for the iteration can be computed by replacing equation (2) with a sphere of earth radius at nadir:

(2') $\vec{r}_E^{\,Sample^2} = \vec{G}_{E,g}^2 = R_0^2$

The angle between the vector to the satellite and the sample can be computed using the law of cosine (see Figure 3-7)

$$-\cos\vartheta = \frac{d^2 - \vec{r}_E^{\,Sat^2} - \vec{G}_{E,g}^2}{2 \cdot |\vec{G}_{E,g}| \cdot |\vec{r}_E^{\,Sat}|}$$

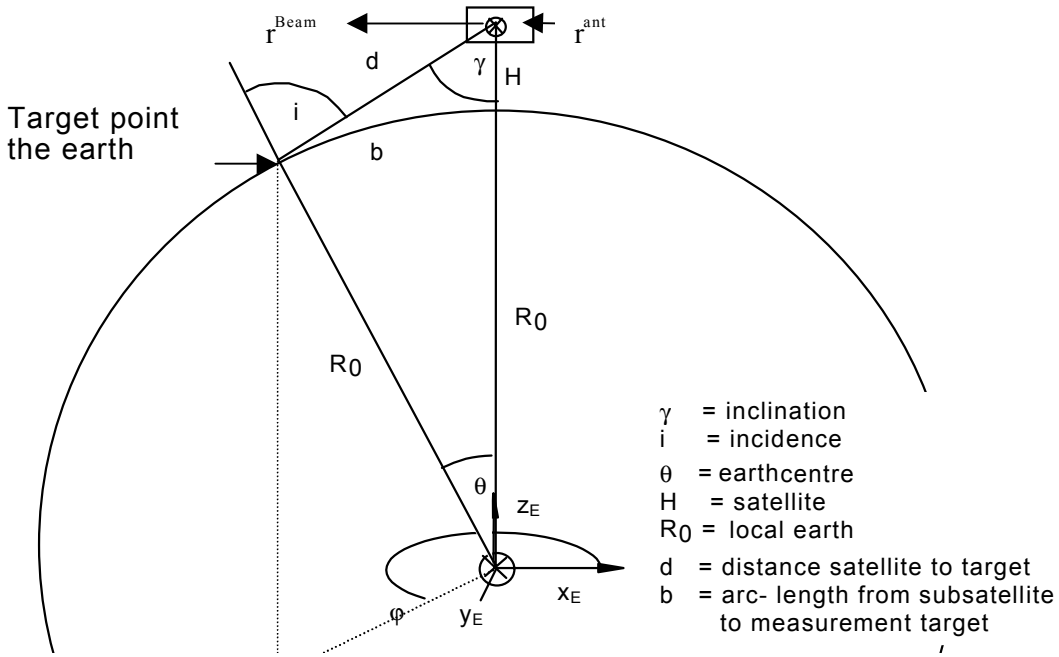


Figure 3-7 Geometry of Sample Measurements in Local Sphere Approximation.

With the approximation that $\vec{r}^{\,Beam}$ vector is perpendicular to the $\vec{r}^{\,Sat}$ (in reality it is perpendicular to the nadir vector $\vec{G}_{E,g}$) the starting point can be computed as:

$$\vec{r}^{\,sample} \approx \left(\frac{\vec{r}_E^{\,Sat}}{|\vec{r}_E^{\,Sat}|} \cdot \cos\vartheta + \frac{\vec{r}_E^{\,Beam}}{|\vec{r}_E^{\,Beam}|} \cdot \sin\vartheta \right) \cdot |\vec{G}_{E,g}|$$

Further runtime optimisations are possible when all samples of a pulse are to be computed, like using the preceding sample vector as starting point or linear interpolation between the first and the last sample vector.

3.3.9 Computation of the Doppler Shift for a Sample

This algorithm defines how to compute the Doppler shift caused by the relative movement of the satellite wrt. the sample.

Input

Symbol	Unit	Dim.	Type	Description
$\vec{r}_E^{Sample}$	m	1	double	Position of the sample in earth-fixed co-ordinates
$\vec{r}_E^{Sat}$	m	1	double	Position of spacecraft in earth fixed co-ordinates
$\vec{v}_E^{Sat}$	m/s	1	double	Velocity of spacecraft in earth fixed co-ordinates
λ	m	0	double	Free space wavelength (c/<RFSCAT centre freq.>), parameter number PL1-230

Output

Symbol	Unit	Dim.	Type	Description
$f_{Doppler}$	Hz	1	double	Doppler shift

Detailed Algorithm

i. Compute the unit vector pointing from satellite to sample in earth fixed co-ordinates:

$$\vec{u}_E^{target} = \frac{\vec{r}_E^{Sample} - \vec{r}_E^{Sat}}{|\vec{r}_E^{Sample} - \vec{r}_E^{Sat}|}$$

ii. Compute the relative velocity from satellite to the sample

$$v^{rel} = \vec{u}_E^{\,target} \cdot \vec{v}_E^{\,Sat}$$

iii. Compute the Doppler Shift corresponding to the relative velocity

$$f_{Doppler} = \frac{2 \cdot v^{rel} \cdot}{\lambda}$$

3.3.10 Computation of Sample Positions

This algorithm defines how the positions of the samples of a pulse are computed. For the correct performance estimation, the spatial resolution of measurement has to be used, which is inverse of the effective bandwidth. The Doppler spread modifies the effective bandwidth and hence the spatial resolution. If a real instrument would sample the received pulse with a higher resolution the variance would not decrease, therefore the real sampling rate of the instrument is not relevant for this algorithm, as long as it is sufficient.

Input

ID	Name	Symbol	Unit	Dim.	Type	Description
PL1-250	i_tauD	τ_D	s	0	double	Duration of Transmit Pulse
PL1-260	i_chirpRate	$\frac{\partial f_{Chirp}}{\partial t}$	$1/s^2$	0	double	Chirp Rate
PL1-360	p_Toffset	T^{offset}	s	0	double	Time offset from sending of pulse to begin of receive window
PL1-370	p_TauRec	τ^{rec}	s	0	double	Length of receive window

Output

Symbol	Unit	Dim.	Type	Description
$N^{Samples}$	-	0	double	Number of samples of the current pulse
B^{eff}	Hz	0	double	Effective bandwidth for this pulse
$\vec{r}_{E,j}^{\,Sample}$	m	2	double	Position of all samples **j** of the current pulse in earth fixed co-ordinates

Detailed Algorithm

i. Compute the position of the first and the last sample of the receive window

ii. Compute the Doppler shift for the first and the last sample of the receive window

iii. Compute Doppler Rate: $\frac{\partial f_{Doppler}}{\partial t} = \frac{f_{Doppler}(t^{Offset} + \tau^{rec}) - f_{Doppler}(t^{Offset})}{\tau^{rec}}$

iv. Compute the effective bandwidth for this pulse:

$$B^{eff} = \left(\frac{\partial f_{Chirp}}{\partial t} + \frac{\partial f_{Doppler}}{\partial t} \right) \cdot \tau^{D}$$

v. Compute the number of samples of the current pulse $N^{Samples} = round(B^{eff} \cdot \tau^{rec})$

vi. Compute the samples positions for the times $t_j = t^{Offset} + \frac{j \cdot \tau^{rec}}{N^{Samples}} \quad \forall j = 0..\left(N^{Samples}\right)$

3.3.11 Computation of SNR' for all Samples

This algorithm computes the normalised signal to noise ration SNR' for each sample. The steps defined here are to be performed in a loop over all on samples of a pulse.

Input

Symbol	Unit	Dim.	Type	Description
$N^{Samples}$	-	0	double	Number of samples of the current pulse
B^{eff}	Hz	0	double	Effective bandwidth for this pulse
$\vec{r}_E^{\,Sat}$	m	1	double	Position of spacecraft in earth fixed co-ordinates
$\vec{r}_E^{\,Beam}$	m	1	double	Normal vector in direction of the antenna beam projected into the rotation plane of the antenna (earth-fixed co-ordinates)
$\vec{r}_{E,j}^{\,Sample}$	m	2	double	Position of all samples **j** of the current pulse in earth fixed co-ordinates

and the instrument parameters:

ID	Name	Symbol	Unit	Dim.	Type	Description
PL1-180	i_NoiseRatio	f_{noise}	-	0	int	Frequency of Noise measurement relative to the pulse repetition frequency. NoiseRatio= n means: Every n-th, Pulse is skipped and a noise measurement takes place instead.
PL1-190	i_ptx	P_{TX}	W	0	double	Transmit power at the HPA output
PL1-200	i_nGain	N^{Gain}	-	0	int	Number of elements of vector PL1-180

PL1-210	i_gain2	G^2_{Ant}	-	1	double	Vector containing the tabulated product of transmit and receive antenna gain. The vector index corresponds to an inclination angle (angle between z-axis of body system (nadir) and antenna beam line).
PL1-220	i_incGain	i^{Gain}	rad	1	double	Vector containing the inclination angles for PL1-210.
PL1-230	i_lamba	λ	m	0	double	Free space wavelength. (c/<RFSCAT centre freq.>)
PL1-240	i_LTxRx	L_{TxRx}		0	double	Losses within instrument
PL1-250	i_tauD	τ_D	s	0	double	Duration of Transmit Pulse
PL1-270	i_aziWidth	$\theta^{azi}_{-1.5dB}$	rad	0	double	-1.5 dB width in azimuth of antenna pattern
PL1-280	i_frot	$f_{rotation}$	-	0	double	Correction factor for azimuth width of antenna resolution, to consider signal damping caused by antenna rotation
PL1-290	i_Latm	$L_{atm/km}$	1/km	0	double	Atmospheric loss per db/km
PL1-300	i_Lrain	L_{rain}	1/km	0	double	Rain loss per db/km
PL1-310	i_Hatm	$H_{atm/km}$	km	0	double	Atmospheric height in km
PL1-320	i_Hrain	Hrain	km	0	double	Rain height in km
PL1-330	i_NoiseFig	n_f	Hz	0	double	Noise figure, used to compute the noise power
PL1-340	i_Tref	T_{ref}	K	0	double	Reference temperature used to compute the noise power

Output

Symbol	Unit	Dim.	Type	Description
SNR'	-	0	double	Normalised signal to noise power of all samples **j** of the current pulse
i_j	[rad]	0	double	Incidence angle of all samples **j** of the current pulse

Detailed Algorithm

for j=0 .. N^{sample}-1 do

i. Compute the vector normal to the sample tangent plane $\vec{N}^{Sample}$ (see section 3.3.15)

ii. Compute the incidence angle $i_j = \arccos\left(\frac{\left(\vec{r}_E^{\,Sat} - \vec{r}_{E,j}^{\,Sample}\right) \cdot \vec{N}^{Sample}}{|\left(\vec{r}_E^{\,Sat} - \vec{r}_{E,j}^{\,Sample}\right)| \cdot |\vec{N}^{Sample}|} \right)$

iii. Compute the inclination angle $\gamma = \frac{\pi}{2} - \arccos\left(\frac{\left(\vec{r}_{E,j}^{\,Sample} - \vec{r}_E^{\,Sat}\right) \cdot \vec{r}_E^{\,Beam}}{|\left(\vec{r}_{E,j}^{\,Sample} - \vec{r}_E^{\,Sat}\right)| \cdot |\vec{r}_E^{\,Beam}|} \right)$

iv. Compute the Atmospheric Losses $L_{atm} = \frac{10^{\frac{2\cdot(L_{atm/km}\cdot H_{atm} + L_{rain}\cdot H_{rain})}{10}}}{\cos(i_j)}$

v. Compute the measurement target length in along beam direction $\Delta x = \frac{c}{2 \cdot B^{eff} \cdot \sin(i_j)}$

vi. Compute the distance d from sample to spacecraft $d_j = |\vec{r}_{E,j}^{\,Sample} - \vec{r}_E^{\,Sat}|$

vii. Compute the measurement target length in across beam direction $\Delta y = \theta_{-1.5db}^{azi} \cdot d \cdot f_{rotation}$

viii. Compute the antenna transmit-receive gain $G_{Ant}^2(\gamma)$ for inclination angle γ by linear interpolation from the setting PL1-200 and PL1-210.

ix. Compute the normalised energy of the signal $\frac{S}{\sigma^0} = \frac{P_{Tx} \cdot G_{Ant}^2(\gamma) \cdot \lambda^2 \cdot \Delta x \cdot \Delta y \cdot \tau_D}{(4\pi)^3 \cdot d_j{}^4 \cdot L_{TxRx} \cdot L^2{}_{atm}}$

x. Compute the noise energy $N = k_B \cdot n_f \cdot T_{ref}$

xi. Compute the normalised signal to noise ratio $SNR'_j = \frac{S}{\sigma^0 \cdot N}$

end loop over all samples

3.3.12 Update of the View Specific Data

This algorithm determines which samples shall be collected to a view and how the view specific data shall be computed. In order to avoid storage of the sample data until a view is completed the weighted averages

$$X_{view} = \frac{\sum_{samples} w_{sample} X_{sample}}{\sum_{samples} w_{sample}}$$

are computed by separate accumulation the numerator and the denominator. For the incidence angle and look angle associated with view several different algorithm options are possible:

(a) Direct weighted averaging on the quantity

(b) Weighted averaging over sample position vector and the corresponding satellite position vector and computation of the angles with resulting vectors

(c) Computation of the angles from the sample next to the centre of node

The differences in results are expected to be small. Therefore option (c) is chosen, because it is the most efficient wrt. runtime.

Input

Symbol	Unit	Dim.	Type	Description
$\vec{r}_E^{Sat}$	m	1	double	Position of spacecraft in earth fixed co-ordinates
$\vec{r}_E^{Beam}$	m	1	double	Normal vector in direction of the antenna beam projected into the rotation plane of the antenna (earth-fixed co-ordinates)
$\vec{r}_{E,j}^{Sample}$	m	2	double	Position of all samples **j** of the current pulse in earth fixed co-ordinates
SNR'	-	0	double	Normalised signal to noise power of all samples **j** of the current pulse
i_j	[rad]	0	double	Incidence angle of all samples **j** of the current pulse
$\vec{N}_{i,j}$	m	0	double	Position of centre of nodes in earth fixed co-ordinates. The first index increases with time (along track) starting from 0, the second index is from the left most node to the right most node across-track

and the processing parameters:

ID	Name	Symbol	Unit	Dim.	Type	Description
PL1-400	p_Nfilter	N^{filter}	-	0	double	Number of elements used for the spatial filter vector
PL1-410	p_dist	r^{filter}	m	1	double	x axis for tabulated spatial filter. The values correspond to the distance form the centre of node.
PL1-420	p_FLUT	w^{filter}	-	1	double	Tabulated spatial filter to be used along axis defined by PL1-390
PL1-430	p_dtview	Δt^{View}	s	0	double	Maximum allowed time difference for samples to be collected to one view

Output

All the output quantities are specific per view. The indices allowing the mapping of the data to a view are not shown for simplicity.

Symbol	Unit	Dim.	Type	Description
w^{Sum}	-	0	double	Sum of all weights applied to the averaged data of this view
$w2^{Sum}$	-	0	double	Sum of the squares of all weights applied to the averaged data this view. This number will be used to compute the number of effective samples.
N^{Sum}	-	0	double	Total number of samples associated this view.
SNR^{Sum}	-	0	double	Sum of the normalised SNR of all samples associated to this view
$\vec{r}_{\min}^{\,Sample}$	m	1	double	Vector of sample position of the sample nearest to centre of node for this view
$\vec{r}_{\min}^{\,Sat}$	m	1	double	Vector of satellite position corresponding to the sample nearest to centre of node for this view
$t_{\min}^{Sample}$	s	0	double	Time of the pulse of the sample nearest to centre of node for this view

Detailed Algorithm

i. Find the nodes within the radius of the spatial filter (biggest element in LUT) of the first and the last sample of the pulse (using the time of the nodes and the time of the pulse allows to restrict the search to small number of nodes). All samples of the current pulse must be within the rectangle defined by the lowest and highest row and column indices of the found nodes.

for j=0 .. N^{sample}-1 do

ii. Find the node(s) within the radius of the spatial filter for the current sample j

for all nodes found

if a view $t_j \le t_{first} + \Delta t^{View}$ exists

iii. make this view to the current view

endif else

iv. create a new view and initialise it with the following view specific data

v. set $t_{first} = t_j$

vi. set d^{min}= distance of current sample to centre of node

vii. set $\vec{r}_{\min}^{Sat} = \vec{r}^{Sat}$

viii. set $\vec{r}_{\min}^{Sample} = \vec{r}_{\min}^{Sample}$

ix. set $t_{\min}^{Sample} = t_j$

x. set $w^{Sum}=w2^{Sum}=N^{Sum}=SNR^{Sum}=0$

endelse

Accumulate the view specific data:

xi. $w^{Sum}=w^{Sum}+w_j$

xii. $w2^{Sum}=w2^{Sum}+w_j^2$

xiii. $SNR^{Sum}=SNR^{Sum}+w_j\,SNR'_j$

xiv. $N^{Sum}=N^{Sum}+1$

if the distance of the current sample to centre of node < d^{min}

xv. set d^{min}= distance of current sample to centre of node

xvi. set $\vec{r}_{\min}^{Sat} = \vec{r}_E^{Sat}$

xvii. set $\vec{r}_{\min}^{Sample} = \vec{r}_{E,j}^{Sample}$

xviii. set $t_{\min}^{Sample} = t_j$

endif

end loop over all nodes found

end loop over all samples

3.3.13 Finalisation of View Specific Data

This algorithm defines the final computations to be performed on the accumulated view specific data in order to obtain the averaged view specific data to be written in the pseudo level 1b product.

Input

Symbol	Unit	Dim.	Type	Description
w^{Sum}	-	0	double	Sum of all weights applied to the averaged data of this view
$w2^{Sum}$	-	0	double	Sum of the squares of all weights applied to the averaged data this view. This number will be used to compute the number of effective samples.
N^{Sum}	-	0	double	Total number of samples associated this view.
SNR^{Sum}	-	0	double	Sum of the normalised SNR of all samples associated to this view
$\vec{r}_{\min}^{\,Sample}$	m	1	double	Vector of sample position of the sample nearest to centre of node for this view
$\vec{r}_{\min}^{\,Sat}$	m	1	double	Vector of satellite position corresponding to the sample nearest to centre of node for this view

Output

Symbol	Unit	Dim.	Type	Description
<i>	[rad]	0	double	Incidence angle associated to this view
<α>	[rad]	0	double	Look-angle associated to this view
N^{eff}	-	0	double	Number of effective samples of this view
<SNR´>	-	0	double	Averaged normalised SNR of this view

Detailed Algorithm

for all nodes

for each view of the current node

i. Compute the vector normal to the tangent plane at the sample $\vec{N}_{\min}^{Sample}$

ii. Compute the incidence angle of the view

$$< i >= \arccos\left(\frac{\left(\vec{r}_{\min}^{Sat} - \vec{r}_{\min}^{Sample}\right) \cdot \vec{N}_{\min}^{Sample}}{|\left(\vec{r}_{\min}^{Sat} - \vec{r}_{\min}^{Sample}\right)| \cdot |\vec{N}_{\min}^{Sample}|} \right)$$

iii. Compute the look angle using $\vec{r}_{\min}^{Sat}$ and $\vec{r}_{\min}^{Sample}$ according to section 3.3.14.

iv. Compute number of effective samples $N^{eff} = \frac{w2^{Sum}}{w^{Sum}} \cdot N^{Sum}$

v. Compute the normalised average $< SNR' >= \frac{SNR^{Sum}}{w^{Sum}}$

end loop over all views of a node

end loop over all nodes

3.3.14 Computation of the Look-Angle

The look angle is defined as the angle between the projection of the vector from the satellite to a sample onto the tangent plane of the sample and the local north vector of the sample. The computation of the local north and west vector is defined in section 3.3.15.

Input

Symbol	Unit	Dim.	Type	Description
$\vec{r}_E^{\,Sat}$	m	1	double	Position of spacecraft in earth fixed co-ordinates
$\vec{r}_E^{\,Sample}$	m	2	double	Position of a sample on the earth ellipsoid

Output

Symbol	Unit	Dim.	Type	Description
α	[rad]	0	double	Look angle: Angle between the projection of the vector from the satellite to a sample onto the tangent plane of the sample and the local north vector of the sample [-180 ,180]

Detailed Algorithm

Compute the local north unit vector at the position of the sample $\vec{u}'_{north} = \frac{\vec{r}^{\,s}_{north}}{|\vec{r}^{\,s}_{north}|}$

Compute the local west unit vector at the position of the sample $\vec{u}'_{west} = \frac{\vec{r}^{\,s}_{west}}{|\vec{r}^{\,s}_{west}|}$

Compute the vector from the sample to the satellite $\vec{r}_E^{\,look} = \vec{r}^{\,Sat} - \vec{r}_E^{\,Sample}$

Compute the look angle $\alpha = \arctan 2(\vec{r}_E^{\,look} \cdot \vec{u}_{west}, \vec{r}_E^{\,look} \cdot \vec{u}_{north})$

3.3.15 Elementary Mathematics of the Ellipsoid

The section does not correspond to a dedicated part of the algorithm, but defines some elementary formulas related to ellipsoids. They will be applied in various places of the RFSCAT algorithm.

Normal Vector at the a Point of the Earth Surface

Let

$$\vec{r}^{\,S} := \begin{pmatrix} x^S \\ y^S \\ z^S \end{pmatrix}$$

be a point on the Earth ellipsoids surface in an equatorial right-handed co-ordinate system (i.e. z-axis pointing to north pole, x,y-axis in the equatorial plane).

All points $\vec{r}^{\,S}$ satisfy the equation:

$$f^{surface}(x,y,z) = \frac{x^2}{A^{Earth^2}} + \frac{y^2}{A^{Earth^2}} + \frac{z^2}{B^{Earth^2}} - 1 = 0$$

The normal vector $\vec{N}^S$ at the point $\vec{r}^{\,S}$ is defined by:

$$\vec{N}^S = \vec{\nabla} f^{surface}(x,y,z)\Big|_{\vec{r}^{\,S}} = \begin{pmatrix} \frac{2x^S}{A^{Earth^2}} \\ \frac{2y^S}{A^{Earth^2}} \\ \frac{2z^S}{B^{Earth^2}} \end{pmatrix}$$

Of course, the common factor 2 may be neglected in RFSCAT as only the direction this vector is of interest.

Local North Vector of a Point at Earth Surface

The points on the earth ellipsoid may also be expressed through the parametric equation:

$$\vec{r}^{\,Surface} = \begin{pmatrix} A^{Earth} \sin\vartheta\cos\varphi \\ A^{Earth} \sin\vartheta\sin\varphi \\ B^{Earth} \cos\vartheta \end{pmatrix}$$

in spherical co-ordinates with ϑ the angle towards the z-axis, and φ the azimuth angle in the x-y-plane.

For a point $\vec{r}^{\,S}$ the vector in the tangent plane pointing to the north pole is called the "local north" vector $\vec{r}_{north}^{\,S}$. It is defined as the direction of decreasing angle ϑ

$$\vec{r}_{north}^{\,s} = -\left.\frac{\partial \vec{r}^{\,Surface}}{\partial \vartheta}\right|_{\vec{r}^{\,S}} = \begin{pmatrix} -A^{Earth}\cos\vartheta\cos\varphi \\ -A^{Earth}\cos\vartheta\sin\varphi \\ B^{Earth}\sin\vartheta \end{pmatrix}$$

Multiplication with a factor $\frac{\sin\vartheta}{B^{Earth}}$ and usage of the parametric ellipsoid equation leads to:

$$\vec{r}_{north}^{\,s} = \begin{pmatrix} -\frac{x^S \cdot z^S}{B^{Earth^2}} \\ -\frac{y^S \cdot z^S}{B^{Earth^2}} \\ \frac{(x^S)^2 + (y^S)^2}{A^{Earth^2}} \end{pmatrix}$$

For the north pole and the south pole $\sin\vartheta = x^s = y^S = 0$. Per definition the local north vector will be defined as the null-vector $\vec{0}$.

Local West Vector of a Point at Earth Surface

The vector to local west is defined as the direction of decreasing azimuth angle:

$$\vec{r}_{west}^{\,s} = \left.\frac{\partial \vec{r}^{\,Surface}}{\partial \varphi}\right|_{\vec{r}^{\,S}} = \begin{pmatrix} A^{Earth}\sin\vartheta\sin\varphi \\ -A^{Earth}\sin\vartheta\cos\varphi \\ 0 \end{pmatrix} = \begin{pmatrix} y^s \\ -x^s \\ 0 \end{pmatrix}$$

At the poles the local west vector is defined as the null-vector $\vec{0}$.

Note that

$$\vec{u}'_x = \frac{\vec{r}_{north}^{\,s}}{|\vec{r}_{north}^{\,s}|}, \vec{u}'_y = \frac{\vec{r}_{west}^{\,s}}{|\vec{r}_{west}^{\,s}|}, \vec{u}'_y = \frac{\vec{N}^S}{|\vec{N}^S|}$$

form an ortho-normal right-handed co-ordinate system.

3.4 The Level 1 b Generator

This part of the programme takes the windvector in a given WVC and the geometry of the measurement of that WVC and calculates a set of σ^0 values using the appropriate LUT.

Then the noise characteristics are used to randomly add some noise to the values of σ^0.

For this the K_p values provided by the pseudo level 1b generator are used (see section 3.3) combined with the geophysical noise (as described in the task 2a report [RD4]) and a Gaussian noise distribution is taken. Note that the σ^0 per view are the result of averaging over many samples (spatial filtering). Hence central limit theorem justifies to assume Gaussian noise, even so the single sample measurement follows Raleigh distribution.

The Level 1 generator is able to use more than one Pseudo Level 1b file in order to simulated the multi beam or multi polarisation options as outlined in section 2.4, Figure 2-2 to 2-5. Each pseudo level 1b file will correspond to one beam and/or polarisation.

3.5 The Wind Retrieval Module

Wind retrieval or inversion of the scatterometry data will be done by the well known algorithm that is used for the current and past scatterometer instruments as well (e.g. seawinds on quickscat).

A maximum likelyhood estimator (MLE) is calculated for a given geometry and simulated measurement using:

$$MLE = \sum_{views} \frac{\left(\sigma^0_{tried} - \sigma^0_{measured}\right)^2}{f^{norm}}$$

with

$\sigma^0_{measured}$ the measured backscatter coefficient σ^0 of a view

$\sigma^0_{measured}$ the LUT value for the geometry of the current measurement, for the current wind speed and direction that is tried out.

f^{norm} a weighting factor, usually a proportional to the instrument noise in the system, in order to let the more accurate measurements have more impact then the very noisy ones.

The 2 dimensional wind-domain (speed and direction) is then scanned to find the local minima that are present in this plane. These should give the set of possible wind-vectors that give σ^0 values that best coincide with the measured values.

More details on the way the wind retrieval is implemented and the minimum search is done are given in the task 2a report [RD4].

The wind retrieval returns a set of one or more possible wind-vectors for each WVC. In order to concentrate on the RFSCAT system properties no ambiguity removal will be included in the wind retrieval module for two reasons:

- Operational ambiguity removal algorithms are highly specialised, and hence would introduce many new simulation parameters. They would complicate the search for a parameter optimum without adding to much information about the RFSCAT system itself.
- Operational ambiguity removal algorithms often make use of the spatial coherency of the wind-vectors and/or the difference with a windfield produced by a numerical weather prediction code is taken into account. Since it is planed to run the simulator on randomly produced wind-vectors this would make it impossible to use this.

Ambiguity removal is replaced by finding the solution vector that has the smallest difference in direction to the input wind vector will have the highest rank. This approach corresponds to an optimal ambiguity removal algorithm.

Additionally the wind retrieval module will calculate the case in which ambiguity removal is not applied at all. In this case only the solution with the best MLE score or "first rank" is chosen as result of the wind retrieval.

Ambiguities still are a concrete complication in scatterometer optimisation. The closest solution to truth will deviate less from truth on average, in case the number of ambiguities increases. However, the more wind ambiguities, the less skilful is the scatterometer in providing a unique wind solution. The NRMS (Normalised RMS, as described in section 3.1 of [RD10]) takes account of the wind direction interval represented by the closest, and as such compensates for the number of solutions. Therefore, the NRMS is an improved measure of the wind direction skill.

Ranking of solutions can be affected by noise. The skill in MLE, or probability, ranking can vary from one scatterometer concept to another. In principle these MLE variations can be independent from the wind solution pattern. Statistics of the first rank wind solution reveal information on the wind solution selection skill in the MLE.

Thus output of the wind retrieval module will be the closest and first rank solutions as well as the input wind field to the simulator.

3.6 The Wind Comparison Module

The wind comparison module provides the tools for comparing the retrieved wind fields (the closest and the rank-1 solution) with the reference wind field, which was used for computing the NRCS in the simulation step. The parameters to be compared are wind speed, wind direction, and the two wind components. This results in values for the deviation, the root-mean-square (RMS) difference and the bias of the two parameters to be compared. Due to the complex nature of the GMF the wind retrieval performance will vary with measurement geometry across the instrument swath as well as with wind speed and satellite flight track with respect to wind direction. Therefore, in-depth analysis can be performed by interactively specifying ranges of these parameters as well as by plotting deviation, RMS-difference or bias as a function of wind speed or direction and node number.

From a suite of comparisons a figure of merit (FoM) will be calculated by combining the statistical measures to a representative quantity, which characterises the system performance within the overall optimisation process. To obtain a good enough sampling of the wind domain for the random wind vector approach a large number of vectors has to be calculated. From experience it is known that a number of 10.000 to 100.000 is usually sufficient.

The wind comparison module contains a variety of graphic routines for displaying results of the computer screen as well as for creating respective PostScript files.

The statistical results of the comparisons (this suite has still to be defined) together with all simulator parameters, will be stored in a database for further evaluation and intercomparison of different instrument configurations.

4 Architectural Software Design

4.1 Overall System Architecture

The overall dataflow of the RFSCAT simulator system is shown in Figure 3-1. The Wind Field generator, Geophysical Model function, the Level 1 b generator and the Wind Retrieval module are combined to single Fortran 90 program, which will be addressed as the “Geophysical Module” in the following.

The Wind Field generator may have a file with ECMWF windfield data as input, or may create completely artificial wind-vectors.

The Geophysical Model function reads (a) binary file(s) containing the Look-Up-Tables(s) used.

The L1b-generator reads an ASCII file produced by the Pseudo L1b generator, containing the noise properties of the system.

The results from the Windfield retrieval module are written to an ASCII file at the end of the execution.

All communication between the 4 programme blocks forming the fortran90 programme is done by data-structures defined inside fortran90 style modules.

The ASCII file interface between the Geophysical Module, the Pseudo-Level 1b and the Windfield Comparison module will allow easy cross platform exchange of the data and easy visual inspection of intermediate results.

4.2 Windfield Generator Architecture

The windfield generator is written in the form of a fortran90 module. The generator returns a single windvector on every call, which is taken from a certain distribution, as is described in the task 2a report [RD4]. Additionally it will be possible to let the routine read a windvector from a supplied file containing for example ECMWF data, or artificial. windfields with a spatial consistency in a given form, such as laminar, rotational, etc.

Programming Language	FORTRAN 90
Compiler	MIPSpro version 7.2.1
Operating System	UNIX (IRIX Release 6.5 IP27)

As no operating system specific calls are used, the S/W is expected to be portable to all platforms, where a Fortran 90 Compiler is available.

4.3 Geophysical Modelling Function Architecture

The Geophysical Model Function is also handled by a fortran90 function to be found inside the inversion module. It is called in the form of a function and returns a σ^0 value for given windspeed, relative azimuth, incidence angle, and polarisation.

Inside the module the appropriate Look Up Table (LUT) needed to get the requested σ^0 value, is automatically read from file at the first call of the function. When desired interpolation can be done in between the table elements of the LUT to obtain more accurate results.

However, this will also take more time for computing.

Programming Language	FORTRAN 90
Compiler	MIPSpro version 7.2.1
Operating System	UNIX (IRIX Release 6.5 IP27)

As no operating system specific calls are used, the S/W is expected to be portable to all platforms, where a Fortran 90 Compiler is available.

4.4 Pseudo Level 1 Generator Architecture

The Pseudo Level 1b Generator accepts simulation parameters characterising

- the RFSCAT antenna system,
- the RFSCAT satellite orbit
- and Level 1 processing parameters.

Output is the Pseudo Level 1 b file to be used by as input to the Level 1 b Generator.

4.4.1 Programming Languages

The flow of control within the Pseudo Level 1 Generator is controlled by code written in interpreter language IDL (version 5.4) . This will allow

- easy modification of the code towards upcoming new ideas in course of the study
- easy visualisation of final and intermediate results
- easy unit level testing

Additionally the Pseudo Level 1b processor makes use of a C++ library called “STARS” developed by the Astrium department of “Simulation & Dynamics”. The advantages of the usage of the STARS library are:

- reliable and fully verified code
- better runtime performance than IDL

The functions of the STARS library can be called directly from IDL through the IDL Dynamical Loadable Module (DLM) mechanism. The interface functions for the DLM layer are coded in C++ and linked with the STARS library to a common shared object. All C++ code will be compiled using the GNU C++ compiler gcc version 2.95.3 or newer.

4.4.2 Operating System

The target and development platform is Linux (SuSE Distribution 7.3, kernel 2.4.10).

4.4.3 Files

The Pseudo Level 1b Generator consists of the following files:

genpl1doc.pro	Utility to generate on-line documentation and detailed design from the IDL headers.
libStarsIDL.dlm	ASCII file used by the IDL interpreter to locate the modules of libStarsIDL.so
libStarsIDL.so	Shared object containing the “STARS” library and its IDL interface.
node__define.pro	Definition of the “node data structure containing all node specific data.
nodepos.pro	All IDL routines used to compute the location of nodes on the earth surface.
orbit.pro	IDL routines interfacing the orbit propagator.
pl1.html	On-line documentation of the pseudo level 1b generator.
pl1.pro	Main routine.
pl1simpars.pro	Definition and initialisation of the common blocks containing the external settings for the Pseudo Level1b generator.
plotps.pro	Utility routines to plot to configure post-script files and display environment for coloured plotting
pulseloop.pro	Main control of functions to be called within the loop over all pulses (see Figure 3-6 Top Level Control Flow of the Pseudo Level 1 b Generator).
pulsepos.pro	Computation of the satellite position at the time of given pulse and computation of vector of the rotating antenna for a given pulse
samplepos.pro	Computation of the geometrical quantities (incidence angle, position) for all samples for a given pulse. This includes the computation of the range resolution depending on Doppler-bandwidth.
snrcalc.pro	Computation of SNR´ per sample.
starsdoc.pro	IDL headers for the routines used from the STARS (C++) library.
test_drivers.pro	Collection of routines used to for analysis of intermediate results or activation of subroutines with special test conditions setup.

viewcalc.pro	Routines to compute the view specific quantities.
writepl1.pro	Routines writing the Pseudo Level 1b file.

4.4.4 Control Flow Diagram and Data Flow Diagram

The Pseudo Level 1 Generator top level control flow is shown in Figure 3-6. The detailed flow of control in corresponding algorithm sections. The data flow is defined through input output tables of the algorithm sections.

4.4.5 Calling Hierarchy

This section gives an overview of the calling hierarchy in the Pseudo Level 1b Generator. All routines used from the STARS library have the prefix “STARS_” in the function names. All of them are designed to accept vector input, because loop processing in C/C++ is much faster then in the IDL language interpreter. The interfaces to test-drivers are not shown.

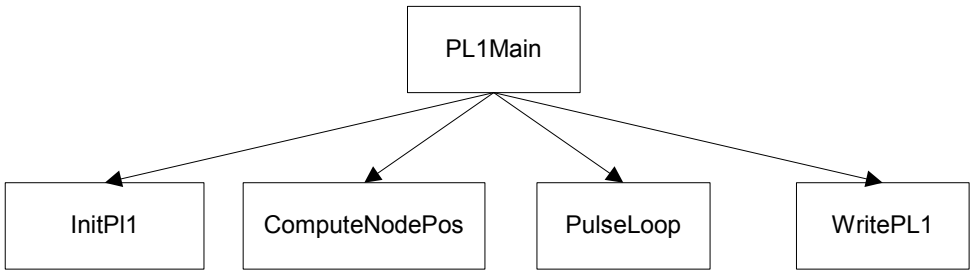


Figure 4-1 Top Level Calling Hierarchy of the Pseudo Level 1 b Generator

Figure 4-1 shows the top-level calling hierarchy of the pseudo Level 1 generator. The “PL1Main” routine calls “InitPI1” to initialise all simulation parameters in common blocks. Then it calls “ComputeNodePos” to compute all node positions in earth fixed frame. The “PulseLoop” function computes in a loop over all pulses the data for all views. Finally, the “WritePL1b” routine writes the Pseudo Level 1b product.

Figure 4-2 Shows which functions are called by the “ComputeNodePos” subroutine. “ComputeTimeVector4Nodes” computes the appropriate times where the nadir is to be computed. “ProgateOrbit” computes the satellite position and the corresponding nadir. “ComputeVelocities” computes numerically the velocities of the nadir in earth fixed co-ordinates

by calling the “PropagateOrbit”. The routine STARS_IntersectionPlane_Ellipsoid is used to compute the across track line where the nodes corresponding to one nadir a located. “ComputeNodeVectors” computes the position of the nodes along these across track lines, finally "ComputeNodeShere" computes the latitude and longitude for each node.

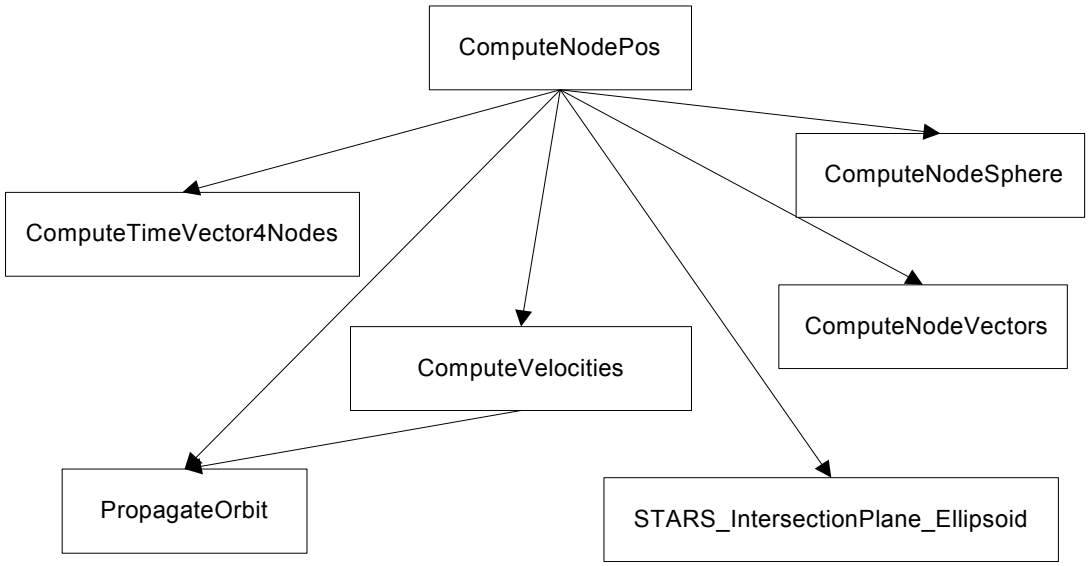


Figure 4-2 Calling Hierarchy for the “ComputeNodePos” Subroutine.

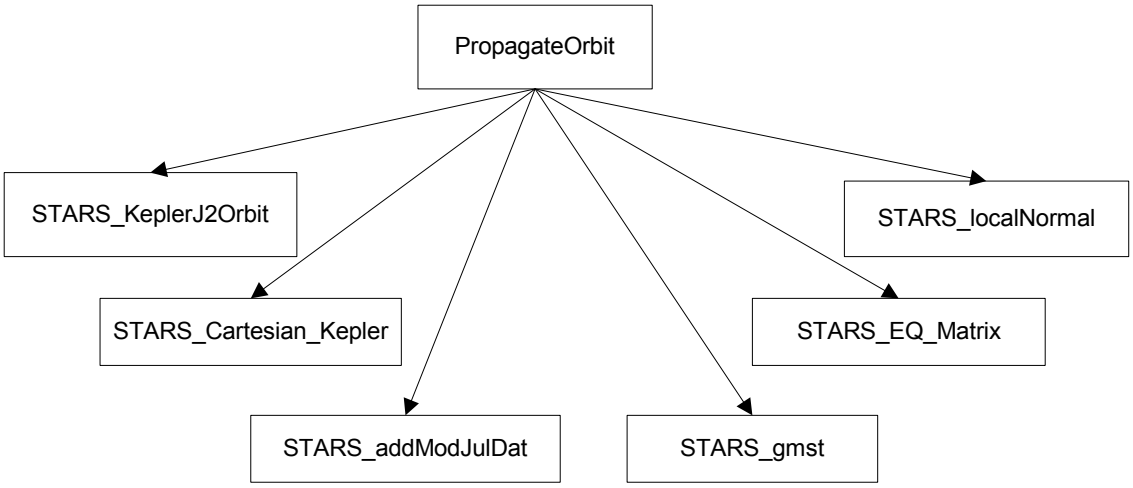


Figure 4-3 Calling Hierarchy for the “PropagateOrbit” Subroutine.

Figure 4-3 shows the calling hierarchy of the “PropagateOrbit” subroutine. This IDL routine interface several routines from the STARS library:

- STARS_KeplerJ2Orbit computes analytically the osculating Kepler elements, given the configuration at t=0 for any time-point t.
- STARS_Cartesian_Kepler performs the co-ordinate transformation for Kepler elements to Cartesian co-ordinates in the true inertia system
- STARS_addModJulDat is utility routine to perform high precision addition of modified Julian date elements in vectorised fashion.
- STARS_gmst computes the Greenwich Mean Sidereal Time for a given modified Julian date.
- STARS_EQMatrix computes the transformation matrix from true inertia system to earth fixed system.
- STARS_localNormal computes the vector from satellite to the nadir. This vector is perpendicular to the local tangent on the earth ellipsoid.

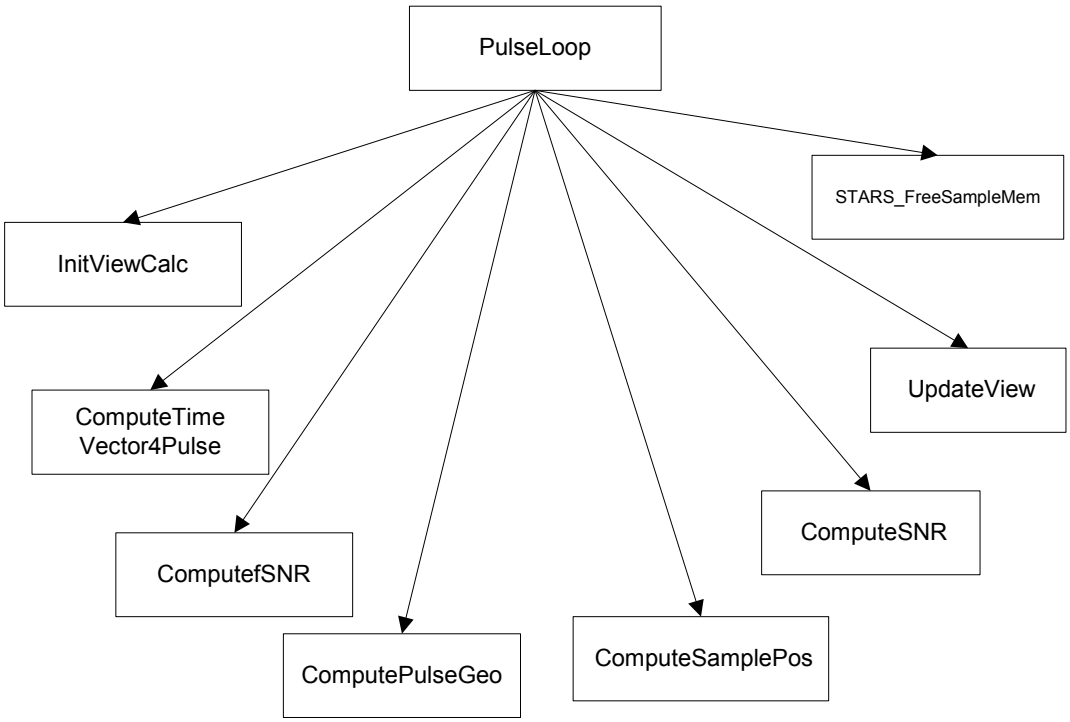


Figure 4-4 Calling Hierarchy for the “PulseLoop” Subroutine

Figure 4-4 shows the calling hierarchy for “PulseLoop” routine. Before the actual loop is entered The routines "InitViewCalc", "ComputeTimeVector4Pulse" and "ComputefSNR" are called performing all computations, memory allocation and software initialisation that maybe performed independent of the pulse specific operations. The purpose of these routines is runtime optimisation. The routines "ComputePulseGeo" , "ComputeSamplePos", "ComputeSNR" and "UpdateView" are called inside the loop over all pulse, the correspond directly to the control flow diagram shown in Figure 3-6. After the pulse loop is left "STARS_FreeSampleMem" is called the free dynamically allocated memory in the STARS library.

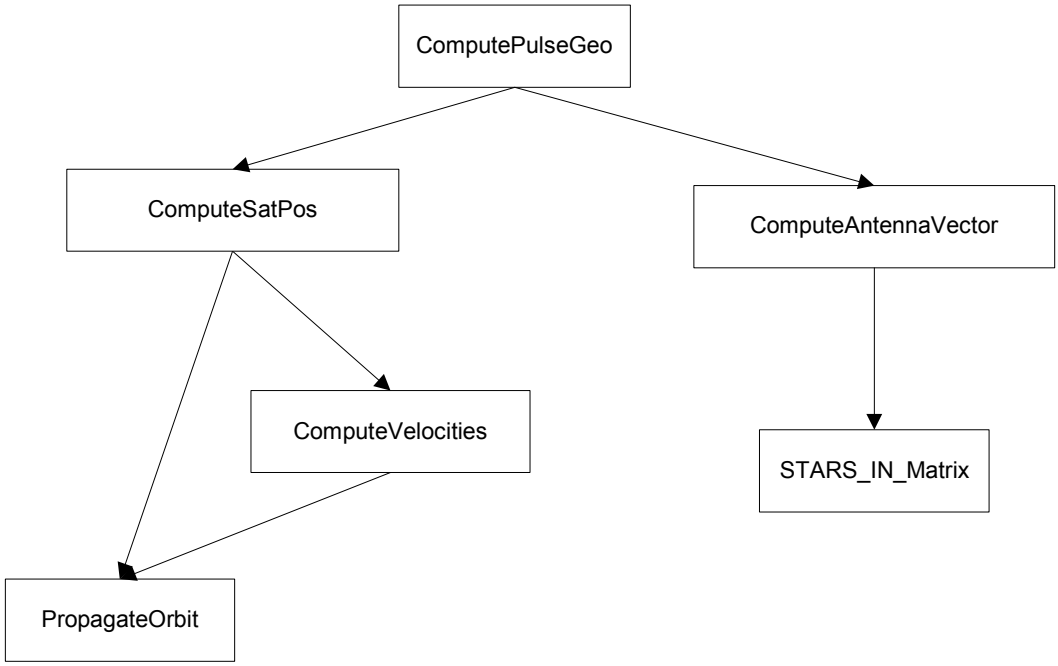


Figure 4-5 Calling Hierarchy for the “ComputePulseGeo” Subroutine

The “ComputePulseGeo” routine performs the algorithm defined in section 3.3.7. Its work can be split into two logical blocks. The computation of the satellite position at each pulse time, performed by the routine “ComputeSatPos” and the computation of the antenna position, performed by the routine “ComputeAntennaVector”. “ComputeSatPos” calls the orbit propagator and the “ComputeVelocities” routines similar, as done for the node position calculations. “ComputeAntennaPos” makes use of the STARS-IN-Matrix for transformation from satellite fixed co-ordinates to earth-fixed co-ordinates.

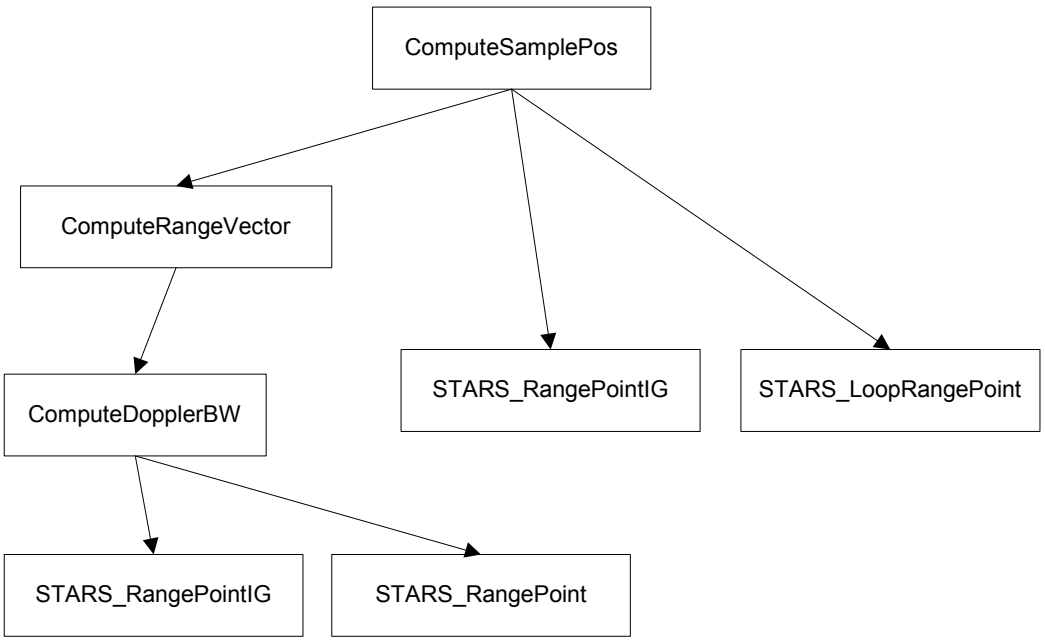


Figure 4-6 Calling Hierarchy for the “ComputeSamplePos” Subroutine

As shown The “ComputeSamplePos” subroutine computes the positions of all samples of a pulse (see Figure 4-6) as defined in section 3.3.10. The first step is to compute a vector ranges where the samples are (“ComputeRangeVector”), which is a function of the local Doppler bandwidth (“ComputeDopplerBW”, algorithm see section 3.3.9). The subroutine STARS_RangePointIG and STARS_RangePoint finds the position of one sample on the earth surface in a certain range and in the beam direction (see section 3.3.8, STARS_RangePointIG is used to compute the initial guess for the Newton iteration). Then position of all samples are computed using in runtime optimized fashion using the STARS_RangePointIG and STARS_LoopRangePoint routines.

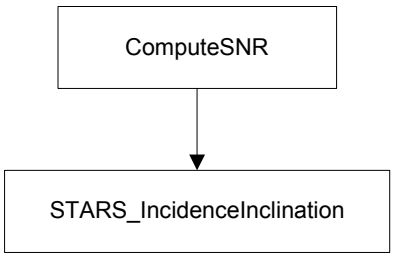


Figure 4-7 Calling Hierachie for the "ComputeSNR" subroutine

The "ComputeSNR subroutine, computes SNR´ for each sample of a pulse. The operations which are "expensive" wrt. to runtime (e.g. computation of inclination angle and incidence angle) are performed in the external C++ routine "STARS_InclidenceInclination".

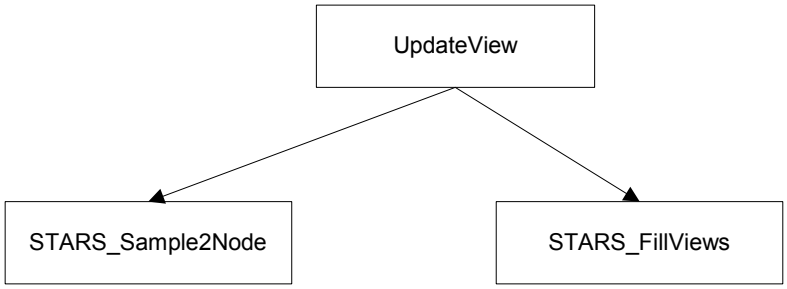


Figure 4-8 Calling Hierachie for the "UpdateView" subroutine

The "UpdateView" subroutine (see Figure 4-8) makes use of the STARS_Sample2Node routine to find all nodes affected by each sample and the "STARS_FillView" routines to update the corresponding views of each affected node. The STARS_Sample2Node operation is the most time consuming part of the pseudo level 1b generator, which justifies the implementation in C++.

Figure 4-9 shows the calling hierarchy for the “WritePL1” subroutine. It calls the “FinaliseViews” function in order to compute the final view specific data from the intermediate accumulated view data. One of the tasks to do be performed in the frame of the view finalisation is to compute the look angle. This is done by the “ComputeLookAngle” subroutine. After writing the header of the pseudo-level 1 product calling "WritePL1header" a loop over all nodes is entered and all views of the nodes are written by "WriteView". "WriteView" calls "ProgateOrbit" to determine if the view was taken from the ascending or descending part of the satellites orbit.

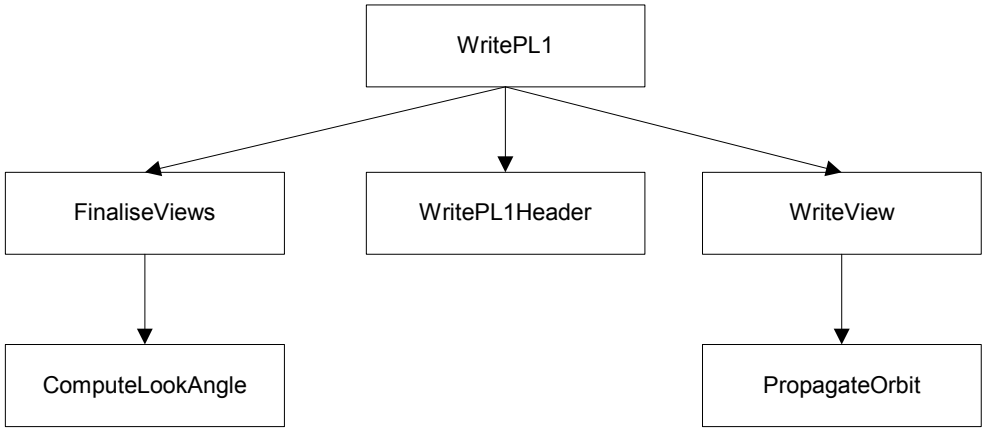


Figure 4-9 Calling Hierarchy for the “WritePL1” Subroutine

4.5 Level 1 b Generator Architecture

The implementation of this part of the programme is very simple. It only consists of some fortran90 function calls to be done in the main fortran programme. First a subroutine named calc_nrcs_values() is called to calculate the σ^0 values for all views of the current WVC. Then the k_p values are calculated including geophysical noise and, using these kp values, noise is added to the already calculated σ^0 values by calling the subroutine add_instr_and_geo_noise_to_nrcs().

4.6 Wind Retrieval Module Architecture

This is a single fortran90 subroutine, that performs one inversion at a time for one WVC.

The internal actions of this routine are described in the task 2a report [RD4].

4.7 Wind Comparison Module Architecture

The wind comparison module is at the bottom end of the simulator. It presents the main results of the simulation process to the user and allows an in depth analysis of the retrieval performance. This functionality is achieve by five components:

i. Input Routine - for reading the files resulting from the wind retrieval module. This covers all header information provided by the simulator, the reference wind field which has been used in the simulation, the closed solution wind field, and the rank-1 solution wind field. A menu allows to choose from all simulation runs which have been performed and which data are stored in the RFSCAT database.

ii. Statistics Routine - A suite of comparisons will be performed in order to derive a figure-of-merit (FoM). The components entering this FoM will be biases and RMS-deviations computed from comparisons between the reference wind field and closest solution as well as rank-1 solution. The combination will be done with a weighting function for wind speed and direction dependence of the retrieval and for the node location across the swath. The later is important since a retrieval error at the far end of the swath with only one NRCS estimate entering the wind retrieval must not have the same weight as an error in a region of the swath were best performance can be expected. The components of the FoM as well as the weighting function are not defined yet. All results of the statistical analysis will be stored in the database.

iii. Analysis Routine - An interactive and menu-driven interface allows the user to perform in-depth analysis of the wind retrieval. At least the following components will be available :

   (a) Scatterplots and the respective PDF isoline plots for comparison of wind speed, wind direction, and wind components. Parameters not being plotted, e.g. node number, can set to selectable ranges.

   (b) Regression analysis for the comparisons under a)

   (c) Plots of deviation, bias, and RMS as a function of wind speed, wind direction, and node number. Parameters not being plotted can set to selectable ranges.

iv. Graphical output is provided on screen as well as in PostScript files.

v. Statistical output is stored in the database of the RFSCAT simulator.

This module will be written entirely in IDL using the actual Version 5.4, however it will be tested to run also under Versions 5

5 Software Verification Approach

Pseudo L1b Generator

The subroutines used from the STARS library are reused from fully verified Astrium S/W projects. No dedicated verification will take place.

The interface from IDL to the STARS library is a new development in the frame of RFSCAT project, but an identical layer from the STARS library to MatLab exists. The IDL-STARS layer will be verified by systematic comparison of results achieved with the MatLab–STARS S/W combination and results achieved with IDL-STARS S/W combination on subroutine level.

The IDL routines used in the Pseudo-Level 1b S/W will be verified on unit and integration level, by analysis of intermediate results. This will usually be achieved by a call to dedicated test-routines that performs crosscheck computations and visualisation of intermediate results. All test routines will be collected in the module test_plots.pro. The calls to test-routines will be left at the appropriate places of the main S/W, but deactivated through the comment token ";".

Additionally test-routines that explicitly activate the subroutine under test may be necessary. These routines will then replace the "PL1Main routine" and initialise the used common blocks and calling parameters as needed. They will as well be collected in the file test_drivers.pro

Fortan 90 modules

The produced windfields are output together with the inverted windvectors in the produced result file. So these can be easily plotted and verified (e.g. for zero noize).

The Windfield Retrieval (inversion) routine has been extensively tested using real data.

This was done using measurements from seawinds on quikscat and comparing the result by the Windfield Retrieval routine to the routine, which is used in the operational use of this data.

The Level 1b Generator has been tested by simulating measurements without noise. This gave perfect results (for the retrieved closest windvector) without any bias, and only minimal rms (caused by the finite steps in the Look Up Table of the GMF).

6 Simulator Parameters

6.1 Settings of the Geophysical Module

The following choices are the most important ones that can be made at compile time:

- radar frequency: C-band or Ku-band.
- interpolation type in the LUT (1,2 or 3 dimensional)
- apply a parabolic fit around the minimum to get a more precise position of the MLE minimum in the wind-domain.
- choose how to step through the LUT, what steps to take.

Further settings are:

- switch to save all results or save binned results.
- limit the windspeed to a certain interval (for example exclude very low windspeeds below 1 m/s)
- selection of windfield scenario

Note: The settings described in this section and are not in a data-file, but are defined in two header-files that are included in the program at compile time. This was necessary for reasons of calculating efficiency.

The files are inversion.h for the settings in the wind retrieval module and simulate.h for the additional FORTRAN settings.

6.2 Pseudo Level 1b Generator

6.2.1 Orbit Parameters

ID	Name	Symbol	Unit	Dim.	Type	Description
PL1-10	k_year0	-	-	0	int	Year at t=0 (start of simulation)
PL1-20	k_month0	-	-	0	int	Month at t=0
PL1-30	k_day0	-	-	0	int	Day at t=0
PL1-40	k_hour0	-	-	0	int	Hour at t=0

PL1-50	k_min0	-	-	0	int	Minute at t=0
PL1-60	k_sec0	-	-	0	int	Second at t=0
PL1-70	k_duration	T^{Dur}	s	0	int	Duration of simulated period
PL1-80	k_a0	a_0	m	0	double	Semimajor axis of orbit
PL1-90	k_e0	e_0	-	0	double	Eccentricity of orbit
PL1-100	k_ta0	ν_0	rad	0	double	True anomaly of orbit at t=0
PL1-110	k_raan0	Ω_0	rad	0	double	Right ascension of ascending node of orbit at t=0
PL1-120	k_i0	i_0	rad	0	double	Inclination of orbit
PL1-130	k_aop0	ω_0	rad	0	double	Argument of perigee of orbit at t=0
PL1-140	k_IsSunSync	-	0/1	0	bool	If set to TRUE (=1), k_i0 will be overwritten, such that the resulting orbit is sun synchronous

Table 6-1 Orbit Parameters

6.2.2 Instrument Parameters

ID	Name	Symbol	Unit	Dim.	Type	Description
PL1-150	i_phi0	$\varphi^{ant}{}_0$	rad	0	double	Angle of antenna beam wrt. satellite velocity at t=0
PL1-160	i_scanRate	ω^{scan}	rad/s	0	double	Antenna scan rate
PL1-170	i_prf	f^{pulse}	Hz	0	double	Pulse Repetition Frequency
PL1-180	i_NoiseRatio	f_{noise}	-	0	int	Frequency of Noise measurement relative to the pulse repetition frequency. Every f_{noise}-th, pulse is skipped and a noise measurement takes place instead.

PL1-190	i_ptx	P_{TX}	W	0	double	Transmit power at the HPA output
PL1-200	i_nGain	N^{Gain}	-	0	int	Number of elements of vector PL1-180
PL1-210	i_gain2	G^2_{Ant}	-	1	double	Vector containing the tabulated product of transmit and receive antenna gain. The vector index corresponds to an inclination angle (angle between z-axis of body system (nadir) and antenna beam line).
PL1-220	i_incGain	i^{Gain}	rad	1	double	Vector containing the inclination angles for PL1-210.
PL1-230	i_lamba	λ	m	0	double	Free space wavelength. (c/<RFSCAT centre freq.>)
PL1-240	i_LTxRx	L_{TxRx}		0	double	Losses within instrument
PL1-250	i_tauD	τ_D	s	0	double	Duration of Transmit Pulse
PL1-260	i_chirpRate	$\frac{\partial f_{Chirp}}{\partial t}$	$1/s^2$	0	double	Chirp Rate
PL1-270	i_aziWidth	$\theta^{azi}_{-1.5dB}$	rad	0	double	-1.5 dB width in azimuth of antenna pattern
PL1-280	i_frot	$f_{rotation}$	-	0	double	Correction factor for azimuth width of antenna resolution, to consider signal damping caused by antenna rotation
PL1-290	i_Latm	$L_{atm/km}$	1/km	0	double	Atmospheric loss (db) per km
PL1-300	i_Lrain	L_{rain}	1/km	0	double	Rain loss (db) per km
PL1-310	i_Hatm	$H_{atm/km}$	km	0	double	Atmospheric height in km

PL1-320	i_Hrain	Hrain	km	0	double	Rain height in km
PL1-330	i_NoiseFig	n_f	Hz	0	double	Noise figure, used to compute the noise power
PL1-340	i_Tref	T_{ref}	K	0	double	Reference temperature used to compute the noise power
PL1-350	i_Pol	-	-	0	string	“VV” or “VH” or “HH”

Table 6-2 Instrument Parameters

6.2.3 Processing Parameters

ID	Name	Symbol	Unit	Dim.	Type	Description
PL1-360	p_Toffset	T^{offset}	s	0	double	Time offset from sending of pulse to begin of receive window
PL1-370	p_TauRec	τ^{rec}	s	0	double	Length of receive window
PL1-380	p_Dnode	d^{Node}	m	0	double	Distance of Nodes
PL1-390	p_TotalSwath	d^{Swath}	m	0	double	Total width of swath. The number of nodes will be computed such that this width fully covered.
PL1-400	p_Nfilter	N^{filter}	-	0	double	Number of elements used for the spatial filter vector
PL1-410	p_dist	r^{filter}	m	1	double	x axis for tabulated spatial filter. The values correspond to the distance form the centre of node.
PL1-420	p_FLUT	w^{filter}	-	1	double	Tabulated spatial filter to be used along axis defined by PL1-390
PL1-430	p_dtview	Δt^{View}	s	0	double	Maximum allowed time difference for samples to be collected to one view
PL1-440	p_BNoise	B^{Noise}	Hz	0	double	Bandwidth for noise measurement
PL1-450	p_nNoiseInt	$N^{Integration}$	-	0	double	Number of receive windows used for noise integration

Table 6-3 Processing parameters

6.2.4 Software Settings

The software settings serve the purpose to optimize runtime and memory allocation of the pseudo level 1b processor.

ID	Name	Unit	Type	Description
PL1-460	s_MaxViews	-	integer	Maximum number of view per node. Used to allocate memory
PL1-470	s_RangeSkipFactor	-	integer	If this parameter is > 1 only every "s_RangeSkipFactor" sample will be computed. The number of effective samples is then finally multiplied with this factor. This setting allows a rough estimation of simulation results with a significant speed up of computation time.
PL1-480	s_PulseSkipFactor	-	integer	If this parameter is > 1 only every "s_PulseSkipFactor" pulse will be computed. The number of effective samples is then finally multiplied with this factor. This setting allows a rough estimation of simulation results with a significant speed up of computation time.

6.2.5 Constants

ID	Name	Symbol	Unit	Type	Description
CPL1-10	A_EARTH	A^{Earth}	m	double	Semi-major axis of earth ellipsoid 6378137
CPL1-20	B_EARTH	B^{Earth}	m	double	Semi minor axis of earth : $B^{Earth} = A^{Earth} \cdot \sqrt{1 - \left(e^{Earth}\right)^2}$
CPL1-30	E_EARTH	e^{Earth}	-	double	Eccentricity of Earth : 0.08181919
CPL1-30	F_EARTH	f^{Earth}	-	double	Flattening of Earth : $f^{Earth} = 1 - \sqrt{1 - \left(e^{Earth}\right)^2}$
CPL1-40	R_EARTH	R^{Earth}	m	double	Mean Earth Radius : 6378000
CPL1-50	C_LIGHT	c	m/s	double	Vacuum Speed of Light: $2.99792458 \cdot 10^8$
CPL1-60	K_BOLTZ	k_B	J/K	double	Boltzmann Constant : $1.380662 \cdot 10^{-23}$
CPL1-70	!DPI	π	-	double	π = 3.1415927

Table 6-4 Constants used by the Pseudo Level 1 Processor

7 File Formats

7.1 Pseudo-L1b File

The Pseudo L1b File is in ASCII format, all field are separated by at least one space character, the field length may vary.

The File consists of three types of data “The General Data Block” at the beginning of the file, followed by the “Node Data Blocks”. Each Node Data Block contains several “View Data Blocks”.

7.1.1 Detailed Description of the General Data Block

Each line of the general data block ends with a “#” symbol followed by a description of the corresponding line, except for vectors Those are preceeded by one line comment starting with a “#”, then the values are listed with five values per line.

Line Number	Type	Description
1	int	File version
2	int	Number of lines of the General Data Block
3	int	Number of lines of a Node Data Block excluding its View Data Blocks
4	int	Number of Node Data Blocks in this file
6	int	Number of lines of a View Data Block
7	int	Number of View Data Blocks in this file
8	long	Time of creation of this file in seconds since 1.Jan.1970
9	int	Number of Noise Samples: $= B^{Noise} \cdot \tau^{rec} \cdot N^{Integration}$
10	string	Start date in format yyyymmdd
11	string	Start time in format hhmmss
12	int	Duration of simulated period in seconds
13-XXX	-	Parameters PL1-80 to PL1-480 (in that order).
XXX+1	-	“#End of General Data Block#”“ - line

Table 7-1 Pseudo Level 1b File: Definition of the General Data Block

7.1.2 Detailed Description of the Node Data Blocks

Each Node Data Block is preceded by one blank line, simplifying visual inspection of the file. This blank line is considered part of the Node Data Block:

The Node Data itself is written in one line with the fields separated by space characters. Each Node data block is followed directly by the corresponding View Data Blocks. In general case the Number of View Data Block of a Node Data Block might be 0.

Field Number within Node Data Block Line	Type	Description
1	int	Row number of this node. The row counts the nodes in across track direction starting with row index 0.
2	int	Column number of this node. The column counts the nodes in across-track direction starting with number 0 for the left most (looking in flight direction) node in each row.
3	double	Geodetical latitude of this node in degrees [–90.0 to 90.0]
4	double	Longitude of this node in degrees [-180.0 to 180.0]
5,6,7	double	x,y,z -component of along track unit vector of center node
8,9,10	double	x,y,z -component of across track unit vector of center node
11	int	Number of Views of this Node

Table 7-2 Pseudo Level 1b File: Definition of the Node Data Block

7.1.3 Detailed Description of View Data Blocks

Each View Data Block is written into one line with the fields separated by space characters.

Field Number within View Data Block Line	Type	Description
1	int	View counter, starting with 0 for each node.
2	double	Time of pulse of sample nearest to center of node [sec]
3	char	ascending/descending flag a : ascending d : descending
4	double	Azimuth angle of antenna beam wrt geographic north in degrees [-180,180]. Positive look angles correspond to a transmit pulse traveling eastwards. The angle is compute using the sample nearest to center of node.
5	double	Incidence angle in degrees. The incidence angle is computed using the sample nearest to center of node.
6	double	Number of effective Samples N^{eff}
7	double	Average SNR´. Averaging is performed using the spatial filter.
8	string	Polarisation “VV” or “VH” or “HH”

Table 7-3 Pseudo Level 1b File: Definition of the View Data Block

7.2 Retrieved Windfield in All Data Format

This file is in ASCII format. It contains

-4 text lines describing the layout of the file

-2 integers giving the number of windvectors and the number of windvector cells in this file.

Then for each windvector cell:

-2 reals, 1 integer, 1 real, 2 integers

giving:

- -the two components of the input windvector of the simulation
- -the WVC number
- -the across track position of the wind vector cell
- -the number of wind-solutions
- -a wind quality code, specifying if the wind retrieval was
- successful or not, and if not, what was the problem.

Then for each found solution:

(the solutions are sorted on MLE, in such a way, that the

first rank comes first, etc.) 1 integer, 3 reals and 3 dummy zeros integers

- -the first integer contains a count of the solutions
- -then 2 reals giving the found windspeed and found wind-direction
- for this solution the 3rd real gives the MLE
- for this solution. the 3 dummy zeros are for future extensions and are not used here.

7.3 Retrieved Wind Field in Binned Format

This is an ASCII file containing:

-4 text lines describing the layout of the file

-1 integer describing the number of bin arrays that are written to this file. Usually that will be 8. Wind-direction, Windspeed, Wind u-component and wind v-component, each for closest and first rank.

A bin array is a 2D array. One dimension is defined by the input windvector, the other dimension is defined by the output wind-vector. Each array element contains a count describing how often this output-wind-vector was found for this input-wind-vector.

Then for each array:

-3 integers, dimension of the arrays, start-value and step-value of the bins

-dim^2 integers being the contents of the array

7.4 GMF Look Up Table File

This is a binary file containing 4 byte reals in a 3 dimensional array.

The first index runs over the windspeed, starting at 0.2 m/s for the first element and stepping with 0.2 m/s per step. The size of this dimension is 250 steps, so the largest value in the table is 50.0 m/s.

The second index runs over the relative azimuth (wind-direction), starting at 0 degrees and stepping with 2.5 degrees per step. The size of this dimension is 73 steps, so the largest value in the table is 180 degrees. Using the geometry of the problem also the range 180-360 degrees is described by the same table.

The third index runs over the incidence angle theta, starting at 16.0 degrees and stepping with 1.0 degree per step. The size of this dimension is 51 steps, so the largest value is 66 degrees.

If the simulation would require it, these settings may be adapted.

7.5 Windfield Map

This is an ASCII file containing:

-1 integer describing the number of wind vectors

-1 text line giving the start date and time of the windfield in the format yyyymmdd:hh:mm:ss

Then for each wind-vector 1 line containing:

-2 reals, giving the u,v component of the wind-vector

-2 reals, giving the lat, lon position of this wind

-1 integer, giving the time offset in seconds compared to the start date and time in the fileheader, for which this wind-vector is valid

Note: Not all fields need to be used, for example for an artificial constant rotational windfield the time tag is of no importance.

8 Acronymns

ADEOS	Advanced Earth Observation Satellite
ASCAT	Advanced ESA C-band scatterometer
C-band	Radar wavelength at about 5 cm
ECMWF	European Centre for Medium-range Weather Forecasts
ERS	European Remote-sensing Satellite
ESA	European Space Agency
FoM	Figure of Merit
GMF	Geophysical Model Function
GOS	Global Observing System
GOOS	Global Ocean Observing System
GUI	Graphical User Interface
GPP	Ground Processing Prototype
HPA	High Power Amplifier
HR	High Resolution
IDL	Interactive Development Language
IFARS	Institut für angewandte Fernerkundung
KNMI	Koninklijk Nederlands Meteorologisch Instituut (Royal Netherlands Meteorological Institute)
Ku-band	Radar wavelength at about 2 cm
L1b	Level 1 b product (geo-located, calibrated data in physical units)
LNA	Low Noise Amplifier
LOS	Line Of Sight
METOP	Future European meteorological polar orbiting satellite
NSCAT	NASA fan-beam Ku-band scatterometer on ADEOS-I
NWP	Numerical Weather Prediction

8 Acronymns

QC	Quality Control
QuikSCAT	NASA dedicated SeaWinds scatterometer mission
RF	Radio Frequency
RFSCAT	Rotating Fan Beam Scatterometer
RMS	Root Mean Square
SCAT	ESA C-band scatterometer on ERS
SeaWinds	NASA conical pencil-beam Ku-band scatterometer
SNR	Signal to Noise Ratio
SOW	Statement of Work
S/W	Software
WVC	Wind Vector Cell, used as a synonym to node in this study
WMO	World Meteorological Organisation
4D-var	Four dimensional variational assimilation

9 Detailed Design of the Pseudo Level 1b Generator

9.1 List of Routines

- CLOSE_PS
- COMPUTEANTENNAVECTOR
- COMPUTEDEPENDENTPARAMETERS
- COMPUTEDOPPLERBW
- COMPUTEFSNR
- COMPUTELOOKANGLE
- COMPUTENODEPOS
- COMPUTENODESPHERE
- COMPUTENODEVECTORS
- COMPUTEPULSEGEO
- COMPUTERANGEVECTOR
- COMPUTESAMPLEPOS
- COMPUTESATPOS
- COMPUTESNR
- COMPUTETIMEVECTOR4NODES
- COMPUTETIMEVECTOR4PULSE
- COMPUTEVELOCITIES
- DEFINECOMMONINPUT
- DEFINENODEVIEWCOMMONBLOCKS
- FINALISEVIEW
- GENERATEPL1DOC
- GETCOLOR
- INITANTENNAGAIN
- INITCOLORINDEX

- INITCONSTANTS
- INITDISPLAY
- INITINSTRUMENTPARS
- INITORBITPARS
- INITPARAMETERS
- INITPL1
- INITPROCESSINGPARS
- INITSIMULATIONPARS
- INITVIEWCALC
- NODE
- OPEN_PS
- PL1MAIN
- PROPAGATEORBIT
- PULSELOOP
- STARS_ADDMODJULDAT
- STARS_ANOMALISTICPERIOD
- STARS_CARTESIAN_KEPLER
- STARS_EQ_MATRIX
- STARS_FILLVIEWS
- STARS_FREESAMPLEMEM
- STARS_GEODETIC_GEOCENTRIC
- STARS_GMST
- STARS_INCIDENCEINCLINATION
- STARS_INITFILLVIEWS
- STARS_INTERSECTIONPLANE_ELLIPSOID
- STARS_IN_MATRIX
- STARS_KEPLERJ2ORBIT

- STARS_LOCALNORMAL
- STARS_LOOPRANGEPOINT
- STARS_MODJULDAT_CALDAT
- STARS_RANGEPOINT
- STARS_RANGEPOINTIG
- STARS_SAMPLE2NODE
- STARS_SUNSYNCINCLINATION
- TEST_COMPUTEPULSEGEO
- TEST_COMPUTERANGEVECTOR
- TEST_COMPUTESNR
- TEST_PLOTLN
- TEST_PLOTNODEMAP
- TEST_PLOTNODES
- TEST_PLOTSEARTH
- TEST_PLOTSELLIPSE
- TEST_PULSELOOPSAMPLES
- TEST_RANGEIG
- TEST_SAMPLE2NODES
- TEST_SETUPEARTHBOX
- UPDATEVIEW
- WAITKBRD
- WRITEPL1
- WRITEPL1HEADER
- WRITEPL1VECTOR
- WRITEVIEW

9.2 Routine Descriptions

9.2.1 CLOSE_PS

[Next Routine] [List of Routines]

```
NAME:
     close_ps

PURPOSE:
     Set the device from postscript back to monitor
     and load corresponding color table


INPUT:
     common block display


OUTPUT:
     Modified display environment
```

(See plotps.pro)

9.2.2 COMPUTEANTENNAVECTOR

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     ComputeAntennaVector

PURPOSE:
     Compute the vector in direction of the antenna and the pulse
     for a given time and satellite position/velocity


INPUT:
     common block InstrumentPars
     t time where the quantities shall be computed
     rSat_E    : position of the satellite
     rG_E      : nadir
     vnSat     : unit vector in direction of satellite velocity
     vnG_E     : unit vector in direction of ground track vel.

OUTPUT:
    rAnt_E    : unit vector in antenna direction
    rBeam_E   : unit vector in Beam direction projected into
                the plane of antenna rotation


    all input/output is in earth fixed coordinates
```

(See pulsepos.pro)

9.2.3 COMPUTEDEPENDENTPARAMETERS

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     ComputeDependentParameters

PURPOSE:
     Compute the parameters, which depend on the input
     settings


INPUT:
    common block constants, OrbitPars, InstrumentPars,SimulationPars

OUTPUT:
```

(See pl1simpars.pro)

9.2.4 COMPUTEDOPPLERBW

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     ComputeDopplerBW

PURPOSE:
     Compute the local Doppler Bandwidth


INPUT:
     rSample_E   : where the samples on earth are located [m]
     rSat_E      : position of satellite [m]
     vSat_E      : vector of satellite velocity [m]

OUTPUT
     Doppler bandwidth  [Hz]

    all input is in earth fixed coordinates
```

(See samplepos.pro)

9.2.5 COMPUTEFSNR

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     ComputefSNR

PURPOSE:
     Compute the quantities in the SNR-prime formula, which depend
     only on the settings
```

```
 INPUT:

      common block constants
      common block InstrumentPars
      common block  ProcessingPars
 OUTPUT
      fSNR , fAtm  factors such that
      SNR-prime = fSNR* G^2
                /( Beff sin(incidence) *range^3 *
10^(fAtm/(cos(incidence))) )
```

(See snrcalc.pro)

9.2.6 COMPUTELOOKANGLE

[Previous Routine] [Next Routine] [List of Routines]

```
 NAME:
      ComputeLookAngle

 PURPOSE:
      Compute the look angle to view


 INPUT:
     XS,YS,ZS   components of the sample position on earth surface
     Beam_X,Y,Z components of the vector from  sample to satellite
rSat-rSample

 OUTPUT
    look angle
```

(See viewcalc.pro)

9.2.7 COMPUTENODEPOS

[Previous Routine] [Next Routine] [List of Routines]

```
 NAME:
       ComputeNodePos

 PURPOSE:
       Compute the Positions of Nodes


 INPUT:
```

```
        None,indirectly the settings are passed through the common
blocks
        to the called subroutines

 OUTPUT:
      r_nodes  vectors to the position of the nodes
                 (dim = 3 x N_Along_Track x N_Across_Track )
      nodes    Two dimensional array of nodes structures. First index
is along
               track (row) second index across track (column)
      dtNodes  Time increment use for of center node position
```

(See nodepos.pro)

9.2.8 COMPUTENODESPHERE

[Previous Routine] [Next Routine] [List of Routines]

```
 NAME:
        ComputeNodeSphere

 PURPOSE:
        Transform the vectors to nodes into spherical coordinates



 INPUT:
        r_nodes    Vectors to the nodes as compute by
ComputeNodeVectors


 OUTPUT:

        nodes.lon   Longitude       [rad]
        nodes.latc  geocentrical Latidude [rad]
        node.r      length of node vector  [m]
        node.latd   geodetical Latitude  [rad]
```

(See nodepos.pro)

9.2.9 COMPUTENODEVECTORS

[Previous Routine] [Next Routine] [List of Routines]

```
 NAME:
        ComputeNodeVectors

 PURPOSE:
        Compute the vectors to the nodes in earth fixed co-ordinates
```

```
        such that they are on input ellipse, equidistant around the
        Nadir (which is the center node)


INPUT:
        ISEllipse:      vector containing AE,BE and CE of an ellipse in
                        3D space
                - AE vector of semi-major axis
                        - BE vector of semi-minor axis
                - CE vector to center of ellipse
                        - N  vector to the point on the ellipse

OUTPUT:
       r_node[3][N_alongTrack][N_across_Track]
```

(See nodepos.pro)

9.2.10 COMPUTEPULSEGEO

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     ComputePulseGeo

PURPOSE:
     Compute the pulse specific geometrical quantities

INPUT:
     t  time where the geometrie shall be computed

OUTPUT:
     rSat_E    : position of the satellite
     rG_E      : nadir
     vSat      : vector of satellite velocity
     rAnt_E    : unit vector in antenna direction
     rBeam_E   : unit vector in Beam direction projected into
                 the plane of antenna rotation

     all output is in earth fixed coordinates
```

(See pulsepos.pro)

9.2.11 COMPUTERANGEVECTOR

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     ComputeRangeVector
```

```
 PURPOSE:
      Compute the vector of ranges where the samples are to be
computed


 INPUT:
      rSat_E    : position of the satellite
      rG_E      : nadir
      vSat_E    : vector of satellite velocity
      rAnt_E    : unit vector in antenna direction
      rBeam_E   : unit vector in Beam direction projected into
                  the plane of antenna rotation
      localRadius : length of rG_E = lokal radius of the earth
      common block ProcessingPars to find out the start and end of
                   receive window

 OUTPUT
     range     :   vector with range positions [m]
     Beff      :   effective Bandwidth of the pulse

     all input is in earth fixed coordinates
```

(See samplepos.pro)

9.2.12 COMPUTESAMPLEPOS

[Previous Routine] [Next Routine] [List of Routines]

```
 NAME:
      ComputeSamplePos

 PURPOSE:
      Compute the sample positions on earth surface


 INPUT:
      rSat_E    : position of the satellite
      rG_E      : nadir
      vSat_E    : vector of satellite velocity
      rAnt_E    : unit vector in antenna direction
      rBeam_E   : unit vector in Beam direction projected into
                  the plane of antenna rotation


 OUTPUT
     rSample_E : vector of position of all samples on the earth
                 surface
     Beff      : effective Bandwidth of this pulse [Hz]

     all input/output is in earth fixed coordinates
```

(See samplepos.pro)

9.2.13 COMPUTESATPOS

[Previous Routine] [Next Routine] [List of Routines]

```
 NAME:
      ComputeSatPos

 PURPOSE:
      Compute the Satellite Positions at pulse transmit


 INPUT:
      t time where the quantities shall be computed

 OUTPUT:

     rSat_E    : position of the satellite
     rG_E      : nadir
     vSat     : vector of satellite velocity
     vG_E     : vector of ground track vel.

     all output is in earth fixed coordinates
```

(See pulsepos.pro)

9.2.14 COMPUTESNR

[Previous Routine] [Next Routine] [List of Routines]

```
 NAME:
      ComputeSNR

 PURPOSE:
      Compute the SNR per sample


 INPUT:
      Beff        : effiective Bandwidth of this pulse
      rSat_E      : position of satellite
      rBeam_E     : unit vector in Beam direction projected into
                    the plane of antenna rotation
      rSample_E   : located of samples on earth
      fSNR        : precomputed factor for SNR computation
      fAtm        : precomputed factor for Atmospheric losses
compuation

      common block constants
 OUTPUT
     SNR         :   SNR-prime per sample
```

```
    all input is in earth fixed coordinates
```

(See snrcalc.pro)

9.2.15 COMPUTETIMEVECTOR4NODES

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
       ComputeTimeVector4Nodes

PURPOSE:
       Compute the time vector for the node positions


INPUT:
       common block constants
       common block OrbitPars
       common block ProcessingPars

OUTPUT:
       Array with times where the orbit propagator shall
       compute the subsatellite location
```

(See nodepos.pro)

9.2.16 COMPUTETIMEVECTOR4PULSE

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     ComputeTimeVector4Pulse

PURPOSE:
     Compute the TimeVector for the Satellite positions at pulse-
transmit


INPUT:

     common block OrbitPars
     common block InstrumentPars

OUTPUT:
     Array with times where the orbit propagator shall
     compute the subsatellite location
```

(See pulseloop.pro)

9.2.17 COMPUTEVELOCITIES

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     ComputeVelocities

PURPOSE:
     Compute the velocities of satellite and ground track
     numerically



INPUT:
     t       Vector of time points corresponding to rSat_E , rG_E
     rSat_E  Satellite position for each time step in earth
             fixed co-ordinates
     rG_E    position of Nadir point (local normal) in earth
             fixed co-ordinates



OUTPUT:
     vSat_E  Vector of satellite velocities in earth fixed
             co-ordinates

     vG_E    Vector of Nadir  velocities in earth fixed  co-ordinates


KEYWORDS :
    NORMALIZE   if set the output vectors are normalized
```

(See orbit.pro)

9.2.18 DEFINECOMMONINPUT

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
    DefineCommonInput


PURPOSE:
     Define the common blocks which hold simulation settings for the
     pseudo l1 generator.


INPUT:
     None

OUTPUT:
     None
```

(See pl1simpars.pro)

9.2.19 DEFINENODEVIEWCOMMONBLOCKS

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     DefineNodeViewCommonBlocks

PURPOSE:
    Define the common blocks ViewInfo and NodeInfo


INPUT:
     None

OUTPUT:
     common block
```

(See nodepos.pro)

9.2.20 FINALISEVIEW

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     FinaliseView

PURPOSE:
     Compute derived view specific quantities


INPUT:
    common Block ViewInfo

OUTPUT
   common Block ViewInfo
```

(See viewcalc.pro)

9.2.21 GENERATEPL1DOC

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
       GeneratePL1Doc

PURPOSE:
```

```
       Generate the online html documentation using the routine
headers
      Usage : compile with IDL and call GeneratePL1Doc


 INPUT:


 OUTPUT:
        file pl1.html
```

(See genpl1doc.pro)

9.2.22 GETCOLOR

[Previous Routine] [Next Routine] [List of Routines]

```
 NAME:
      GetColor

 PURPOSE:
     Return the color index for a color,
     depending on TRUE color (Monitor ) or not  (e.g. Postscript)

 INPUT:
      common block colors


 OUTPUT:
      Modified display environment
```

(See plotps.pro)

9.2.23 INITANTENNAGAIN

[Previous Routine] [Next Routine] [List of Routines]

```
 NAME:
      InitAntennaGain

 PURPOSE:
     Initialise the two way antenna gain parameters


 INPUT:
      None

 OUTPUT:
      common block
```

(See pl1simpars.pro)

9.2.24 INITCOLORINDEX

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     InitColorIndex

PURPOSE:
   Initialise common color Index correctly such that
   True-color and Non-True-Color is treated correctly


INPUT:
     none


OUTPUT:
     common block ColorIndex
```

(See plotps.pro)

9.2.25 INITCONSTANTS

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     InitConstants

PURPOSE:
     Define and initialise common block constants


INPUT:
     None

OUTPUT:

     R_EARTH    [m] Mean Earth Radius
     A_EARTH    [m] Semi-Major Axis of Earth
     E_EARTH    [-] Eccentricity of Earth
     C_LIGHT    [m/s] Vacuum Speed of Light
     K_BOLTZ    [J/K] Boltzmann constant

     derived constants
     B_EARTH    [m] Geocentric Pole Distance (computed from A and E)
     F_EARTH    [-] Flattening of Earth (MP) (computed from E )
     P_EARTH    [-] b^2/a
```

(See pl1simpars.pro)

9.2.26 INITDISPLAY

[Previous Routine] [Next Routine] [List of Routines]

```
 NAME:
      InitDisplay

 PURPOSE:
    Initialise common block display correctly for
    True-color and Non-True-color monitors
    activate backing store


 INPUT:
      common block display


 OUTPUT:
      modified display environment
```

(See plotps.pro)

9.2.27 INITINSTRUMENTPARS

[Previous Routine] [Next Routine] [List of Routines]

```
 NAME:
      InitInstrumentPars

 PURPOSE:
      Initialise common block InstrumentPars


 INPUT:
      None

 OUTPUT:

      i_phi0          [rad]   Angle of antenna beam wrt satellite v
at t=0
      i_scanRate      [rad/s] Rotation speed of antenna
      i_prf           [Hz]    Pulse repitition frequency
      i_NoiseRatio    [-]     Every i_NoiseRations pulse is used for
                              Noise measurements
      i_ptx           [W]     Transmit power at the HPA
      i_nGain         [-]     # of elements of the Gain2 vector
      i_gain2         [-]     two way antenna gain vector
      i_incGain       [rad]   vector of inclination angles for
                              i_gain2
      i_lambda        [m]     Wavelength of instruments
      i_LTxRx         [-]     Losses within the instrument
```

```
i_tauD          [s]     Duration of transmit pulse
i_chirpRate     [-]     Chirp Rate
i_aziWidth      [rad]   -1.5 db width of beam in azimuth
i_frot          [-]     Correction factor for azimuth width of
                        antenna cause by rotation of the antenna
i_Latm          [dB/km] Atmospheric loss per km
i_Lrain         [dB/km] Rain loss per km
i_Hatm          [km]    Atmospheric height in km
i_Hrain         [km]    Rain height in km
i_NoiseFig      [Hz]    Noise figure for Noise Power
i_Tref          [K]     Reference Temperature for Noise Power
i_Pol           [-]     String to flag Polarisation
```

(See pl1simpars.pro)

9.2.28 INITORBITPARS

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     InitOrbitPars

PURPOSE:
     Initialise common block kepler with defaults
     for the starting time t=0

INPUT:
     IsSunSync  Boolean TRUE for Sun synchronous orbits
                if set to TRUE the inclination is computed
                to force a sun-synchronous orbit

OUTPUT:


     k_year0     year of t=0
     k_month0    month of t=0
     k_day0      day of t=0
     k_hour0     hour of t=0
     k_min0      minute of t=0
     k_sec0      second of t=0
     k_duration  duration of simulation in seconds
     k_a0       Semimajor axis of orbit            [m]
     k_e0       Eccentricity of orbit              [-]
     k_ta0      True anomaly                       [rad]
     k_raan0    Right ascension of ascending node  [rad]
     k_i0       Inclination                        [rad]
     k_aop0     Argument of perigee                [rad]

     derived parameters
     k_mjd0     Modified julian date computed from k_year0 ...k_sec0
     k_k0       Array containing the elements k_a0 .. k_aop0
```

(See pl1simpars.pro)

9.2.29 INITPARAMETERS

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
      InitParameters

PURPOSE:
      Call Routines to Initialise Simulation Parameters


INPUT:
      None

OUTPUT:
      Defined and with consistent defaults initialized common blocks
      constants , OrbitPars, InstrumentPars,ProcessingPars
```

(See pl1simpars.pro)

9.2.30 INITPL1

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
      InitPl1

PURPOSE:
      Routine to initialise the pseudo level 1 generator


INPUT:
     None

OUTPUT:
     Defined and initialise common blocks
```

(See pl1.pro)

9.2.31 INITPROCESSINGPARS

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
      InitProcessingPars

PURPOSE:
      Initialise common block ProcessingPars
```

```
INPUT:
     None

OUTPUT:
     p_Toffset         [s]  Time offset of receive window
     p_TauRec          [s]  Duration of receive window
     p_Dnode           [m]  Spatial resolution
     p_TotalSwath      [m]  Width of total swath
     p_Nfilter         [-]  Number of elements of spatial filter LUT
     p_dist            [m]  Distance-axis of spatial filter LUT
     p_FLUT            [-]  LUT of spatial filter
     p_dtview          [s]  Delta time threshold for a view
     p_BNoise          [Hz] Noise measurement bandwidth
     p_nNoiseInt       [-]  Number of receive Window used for
                            noise measurement integration

     derived parameters :
     p_NumberOfNodesPerRow    per row computed from TotalSwath
```

(See pl1simpars.pro)

9.2.32 INITSIMULATIONPARS

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     InitSimulationPars

PURPOSE:
    Initialise Simulation Parameters


INPUT:
     None

OUTPUT:
     common block
```

(See pl1simpars.pro)

9.2.33 INITVIEWCALC

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     InitViewCalc

PURPOSE:
     Compute quantities used for view calculation,
     which need not to be computed within the pulse loop
```

```
INPUT:


OUTPUT
   common Block ViewInfo
```

(See viewcalc.pro)

9.2.34 NODE

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     Node

PURPOSE:
     Define the structure containing node specific data


ELEMENTS:
          xE  x-element of the Nodes in earth fixed co-ordinates
          yE
          zE
          lon   Longitude       [rad]
          latC  geocentrical Latitude [rad]
          r      length of node vector  [m]
          latD   geodetical Latitude  [rad]
          iAlongTrack
          iAcrossTrack
          number of assoziated views
```

(See node__define.pro)

9.2.35 OPEN_PS

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     open_ps

PURPOSE:
     Set the device to postscript and load
     corresponding color table


INPUT:
```

```
     name   Name of the postscript file
     common block display


OUTPUT:
     modified display environment
```

(See plotps.pro)

9.2.36 PL1MAIN

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     Pl1Main

PURPOSE:
     Main Routine for RFSCAT Pseudo Level 1b Generator


INPUT:
    filename  name of pseudo level 1b file
    Simulation Settings in common block

OUTPUT:
    Pseudo L1b File
```

(See pl1.pro)

9.2.37 PROPAGATEORBIT

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     PropagateOrbit

PURPOSE:
     Compute the Orbit Quantities for a given time vector



INPUT:
     common block constants
     common block OrbitPars
     Vector of times where orbit shall be computed

OUTPUT:
     rSat_E  Satellite position for each time step in earth
             fixed co-ordinates
     rG_E    position of Nadir point (local normal) in earth
              fixed co-ordinates
```

(See orbit.pro)

9.2.38 PULSELOOP

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     PulseLoop

PURPOSE:
     Control Loop over all pulses


INPUT:


OUTPUT:
```

(See pulseloop.pro)

9.2.39 STARS_ADDMODJULDAT

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     STARS_addModJulDat

PURPOSE:
     Adds a time interval to the Modified Julian Date.

INPUT:
     mjd       Modified Julian Date       [days, frac of day]
     dt        Time interval to be added  [s]


OUTPUT:
     mjdAdd    New Modified Julian Date   [days, frac of day]
```

(See starsdoc.pro)

9.2.40 STARS_ANOMALISTICPERIOD

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     STARS_anomalisticPeriod

PURPOSE:
```

```
      Compute the anomalistic period considering the flattening of the
body.

      Anomalistic means from one anomaly to the next, e.g. from
perifocus to
      perifocus. The flattening is considered by the J2-Term and the
semimajor
      axis of the body. This implementation uses the gravitation, J2
      Term and radius of the earth

      Reference:  Wertz, Spacecraft Attitude Determination and
      Control

 INPUT:
    a          Semimajor axis of orbit                    [m]
    e          Eccentricity of orbit                      [-]
    i          Inclination                                [rad]

 OUTPUT:
    Ta         Anomalistic Period                         [s]
```

(See starsdoc.pro)

9.2.41 STARS_CARTESIAN_KEPLER

[Previous Routine] [Next Routine] [List of Routines]

```
 NAME:
      STARS_Cartesian_Kepler

 PURPOSE:
      Converts osculating keplerian to cartesian state elements.

      Keplerian elements consider the acting gravitation. Osculating
elements are
      valid only for the actual moment as the flattening of the body
is not
      considered. The classical cartesian elements are the position
      and velocity.

 INPUT:
      Array(s) containing the osculating Kepler elements

 OUTPUT:
      rv_I  Array(s)  containing the
         -  Position vector given in inertial coordinates  [m] at
index rv_I[0:2,*]
         -  Velocity vector given in inertial coordinates  [m/s] at
             index rv_I[3:5,*]
```

(See starsdoc.pro)

9.2.42 STARS_EQ_MATRIX

[Previous Routine] [Next Routine] [List of Routines]

```
 NAME:
     STARS_EQ_Matrix

 PURPOSE:
    Transformation matrix from True Inertial (Q) to Earth-Fixed system
(E).
    Best approximation for thetaG is Greenwich Apparent Sidereal Time
(GAST).

    Both systems are Earth centered.

    Coordinate system definitions:

   E  (Earth fixed)
       x: from Earth center to intersection between equator and
Greenwich meridian
       y: complete to right system
       z: to celestial pole

   Q  (true Earth centered - inertial fixed)
       x: to true Vernal Equinox
       y: complete to right system
       z: to celestial pole

 INPUT:
      thetaG  Rotation angle of Earth (Greenwich meridian to  [rad]
Vernal Equinox)

 OUTPUT:
      EQ      Transformation matrices (DCM) from Q- to E-System  [-]
              EQ is an array (last index)  of 3 x 3 matrices (first
              and second index)
```

(See starsdoc.pro)

9.2.43 STARS_FILLVIEWS

[Previous Routine] [Next Routine] [List of Routines]

```
 NAME:
     STARS_FillViews

 PURPOSE:
     Fill the view arrays using the computed quantities for each
sample and the mapping
     samples to nodes as provided by STARS_Sample2Node
```

```
 INPUT:
      t              time of pulse transmit
      r_nodes        vectors to the centers of the nodes grid  [m]
                     first index 0=x 1=y 2=z
                     second index along track index of node (called
row)
                     third  index across track index of node (called
column)
      rSat_E   position of the Satellite in earth fixed co-ordiantes
[m]
      rSample_E      vectors to sample points in earth fixed co-
ordiantes [m]
                     first index sample
                     second index 0=x 1=y 2=z
      rc             output of the STARS_Sample2Node routine
      SNRprime       SNR´ for each sample
 OUTPUT:
     The arrays to the common block ViewInfo are filled/updated
     Following arrays are affected :

ViewtFirst,ViewdMin,ViewrSat_X,_Y,_Z,ViewSample_X,_Y,_Z,ViewWSum
          ViewNSum,ViewSNRsumm Viewtmin,ViewtLast
```

(See starsdoc.pro)

9.2.44 STARS_FREESAMPLEMEM

[Previous Routine] [Next Routine] [List of Routines]

```
 NAME:
      STARS_FreeSampleMem

 PURPOSE:
      Free the dynamical memory allocated by STARS_Sample2Node

 INPUT:
      None
 OUTPUT:
      None
```

(See starsdoc.pro)

9.2.45 STARS_GEODETIC_GEOCENTRIC

[Previous Routine] [Next Routine] [List of Routines]

```
 NAME:
     STARS_geodetic_geocentric

 PURPOSE:
     Transform geocentric latitude into geodetic latitude
```

```
INPUT:
    geocentric latitude

OUTPUT:
    geodetic latitude
```

(See starsdoc.pro)

9.2.46 STARS_GMST

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
    STARS_gmst

PURPOSE:
    Greenwich Mean Sidereal Time [rad], conversion between Universal
Time and
    Sidereal Time (Mean Equinox of Date) expressed in Modified Julian
Date with
    UT1. Rigorous formula. Compared to Astronomical Almanac B8-B15.

    Note, that the returned angle is not converted in the range
between 0 and 2 Pi!

    Reference:  IAU 1982, Astronomical Almanac 1984, B6

  INPUT:
    mjdUT1    UT1 as Modified Julian Date    [day, fod]

OUTPUT:
    gMST      Greenwich Mean Sidereal Time   [rad]
```

(See starsdoc.pro)

9.2.47 STARS_INCIDENCEINCLINATION

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     STARS_IncidenceInclination

PURPOSE:
     Compute in a run time optimized fashion the quantities needed to
     compute SNR per sample
INPUT:
    rSat_E    position of the S/C when pulse is transmitted [m]
    rSample_E vectors to the sample positions on earth surface [m]
              first index corresponds the range
              second index has values 0,1,2 for the x,y,z coordinate
```

```
                Note the is the transpose of the output of the
Stars_LoopRangePoint routine
                to optimize runtime
     rBeam_E    unit vector in the plane perpendicular of the nadir
vector
                and the plane of the beam at time of transmit
     range      vector containing the distances between the S/C and the
samples

 OUTPUT:
     rc         matrix of dimensions NumberOfSamplesOnGrid x 4 and type
double
                rc[*,0] holds the incidence angles for each sample
[rad]
                rc[*,1] holds the inclination angle for each sample
[rad]
                rc[*,2] holds range^3 * sin(incidence angle) for each
sample [m^3]
                rc[*,3] holds cos(inclination angle)
```

(See starsdoc.pro)

9.2.48 STARS_INITFILLVIEWS

[Previous Routine] [Next Routine] [List of Routines]

```
 NAME:
     STARS_InitFillViews

 PURPOSE:
     Initialisation of internal static variables to the STARS_FillView
routine
     It sets the pointers to the View variables for fast access
     in the STARS_FillView routine. Therefore the idl routine
"InitViewCalc"
     needs to have been called before STARS_InitFillViews is called
 INPUT:
    common block ViewInfo initialize by "InitViewCalc"

 OUTPUT:
    Internal pointer need for STARS_FillView routine
```

(See starsdoc.pro)

9.2.49 STARS_INTERSECTIONPLANE_ELLIPSOID

[Previous Routine] [Next Routine] [List of Routines]

```
 NAME:
      STARS_IntersectionPlane_Ellipsoid

 PURPOSE:
```

```
    Compute the intersection of plane with the earth ellipsoid
    This is used to compute the location of the nodes in RFSCAT


 INPUT:
    vnG_E     Unit vector normal the intersecting plane in
              earth fixed coordinates  ( in RFSCAT
              the vector in direction of the ground track velocity)
    rG_E      Location on earth surface, which is in the intersecting
              plane [m] (In RFSCAT this the local normal point )

 OUTPUT:
   ISEllipse   vector containing AE,BE and CE of an ellipse in
                      3D space
               - AE vector of semi-major axis
                       - BE vector of semi-minor axis
                - CE vector to center of ellipse
                     ISEllipse[0:2,*] contains the three AE
                        components (first index 0-2) for each
computed
                        Ellipse (second index)
                        analogous ISEllipse[3:5] contains BE and
                        ISEllipse [6:8,*] CE
```

(See starsdoc.pro)

9.2.50 STARS_IN_MATRIX

[Previous Routine] [Next Routine] [List of Routines]

```
 NAME:
     STARS_IN_Matrix

 PURPOSE:
  Transformation matrix from Nadir (N) to inertial system (I) or
  Earth fixed (depending on input)

  Note, that Nadir is here defined to the planet center, not
perpendicular
  to the planet surface.

  Coordinate system definitions:

  N
   x: complete to right system (flight direction for circular orbits)
   y: perpendicular to orbit plane (opposite normal direction)
   z: nadir pointed (opposite direction of position vector, to Planet
Center)

  I
   x: Inertial system, where position and velocity are defined
   y:    -"-
```

```
  z:    -"-

INPUT:
  r_I       Inertial position of the object  [m]   (or earth fixed)
  v_I       Inertial velocity of the object  [m/s] (or earth fixed)


OUTPUT:
  IN       Transformation matrix (DCM) from N- to I-System  [-]
```

(See starsdoc.pro)

9.2.51 STARS_KEPLERJ2ORBIT

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     STARS_KeplerJ2Orbit

PURPOSE:
      Compute S/C orbit underlying homogeneous gravity of a planet
considering its oblateness.

      It uses a semi-analytical algorithm (size of the timestep does
not effect the
      accuracy). Default planet is the Earth. Osculating Kepler
elements are
      required.

INPUT:
     t   time(s) where the orbit shall be computed
     k_k0=[a0,e0,ta0,raan0,i0,aop0 ] array with osculation Kepler
elements at t=0
     where
               a0       Semimajor axis of orbit            [m]
               e0       Eccentricity of orbit              [-]
               ta0      True anomaly                       [rad]
               raan0    Right ascension of ascending node  [rad]
               i0       Inclination                        [rad]
               aop0     Argument of perigee                [rad]

OUTPUT:
      Array(s) of osculating Kepler elements for each time t
```

(See starsdoc.pro)

9.2.52 STARS_LOCALNORMAL

[Previous Routine] [Next Routine] [List of Routines]

```
 NAME:
     STARS_localNormal

 PURPOSE:
    Compute vector from satellite to local normal point.

    The local normal is defined as the direction to the point of the
ellipsoid
    surface, which tangential plane is perpendicular to the direction.

 INPUT:
    rSat_E  Position vector in equatorial system  [m]


 OUTPUT:
    rLN_E   Vector from satellite to local normal  [m]
```

(See starsdoc.pro)

9.2.53 STARS_LOOPRANGEPOINT

[Previous Routine] [Next Routine] [List of Routines]

```
 NAME:
     STARS_LoopRangePoint

 PURPOSE:
     Compute many RangePoints (see STARS_RangePoint) which are near
together
     using the same S/C craft position and the same antenna vector
defining the
     plane
     The algorithm avoids to compute the initial guess for each range
points
     using the "expensive" sin and cos function from STARS_RangeIG
     Instead it is only provided the initial guess for the first Range
point
     (index 0) and the RangePoint with index "nSubStep".
     The algorithm computes then the exact location of these Range
points.
     Next the initial guess for all RangePoint in this interval is
computed using
     linear interpolation from the exact range points. Is the interval
is
     sufficiently small ( few kilometers) the newtons method will
converge
     at the first iteration, avoiding the computation of the jacobian
     The algorithm moves then the interval to cover the next "SubStep"
points
     the outmost point be computed first, using linear extrapolation
as
```

```
     initial guess.
     Remark : The total number of Points need NOT to be a integer
multiple
     of "nSubStep"

 INPUT:
     rIG      initial guess vectors for RangePoint with index 0 and
RangePoint
              with index "nSubStep"
     rSat_E   position of the Satellite in earth fixed co-ordiantes
[m]
     rANt_E   Vector perpendicular to the plane of the beam (i.e. the
direction of
              (long axis of the antenna)
     range    distance (range) of the searched point(s) on earth
surface to the S/C
     nSubStep Width of interval (see PURPOSE)
 OUTPUT
     rSample_E position vectors of the RangePoints
               first index has values 0,1,2 for the x,y,z coordinate
               second index corresponds the range
```

(See starsdoc.pro)

9.2.54 STARS_MODJULDAT_CALDAT

[Previous Routine] [Next Routine] [List of Routines]

```
 NAME:
      STARS_ModJulDat_CalDat

 PURPOSE:
     Transformation from  Calenderian Date  [year, month, day, hour,
     min, sec] to Modified Julian Date

 INPUT:
    Calenderian Date as an array [year, month, day, hour,min, sec]

 OUTPUT:
      mjd       Modified Julian Date       [days, frac of day]
```

(See starsdoc.pro)

9.2.55 STARS_RANGEPOINT

[Previous Routine] [Next Routine] [List of Routines]

```
 NAME:
      STARS_RangePoint
```

```
 PURPOSE:
     The STARS_RangePoint, and STARS_LoopRangePoint routines computes
     the position of a point which is

                 (1) In certain distance of the satellite
                 (2) Is on the earth ellipsoid
                 (3) In a certain plane  defined through its normal
vector
     appling newtons method. The start point of the iteration
     (initial guess) is to be provided as input ( see
STARS_RangePointIG)

 INPUT:
     rIG      initial guess vector(s)
               first index has values 0,1,2 for the x,y,z coordinate
               second index corresponds the range
     rSat_E   position of the Satellite in earth fixed co-ordiantes
[m]
     rANt_E   Vector perpendicular to the plane of the beam (i.e. the
direction of
              (long axis of the antenna)
     range    distance (range) of the searched point(s) on earth
surface to the S/C        ; OUTPUT:
     rSample  vector(s) to the points on earth surface
               first index has values 0,1,2 for the x,y,z coordinate
               second index corresponds the range
```

(See starsdoc.pro)

9.2.56 STARS_RANGEPOINTIG

[Previous Routine] [Next Routine] [List of Routines]

```
 NAME:
      STARS_RangePointIG

 PURPOSE:
     The STARS_RangePoint, and STARS_LoopRangePoint routines computes
     the position of a point which is

                 (1) In certain distance of the satellite
                 (2) Is on the earth ellipsoid
                 (3) In a certain plane  defined through its normal
vector

     appling newtons method. The Stars_RangePointIG routine computes
     a good starting point for the newton iteration, approximating
     the earth as spherical with the local radius

 INPUT:
     rSat_E   position of the Satellite in earth fixed coordinates [m]
     rBeam_E  vector in the plane of the antenna beam (perpendicular
```

```
              to local normal vector
     localRadius local earth radius  [m]
     r        distance from satellite to point on earth surface
              (range) [m]

 OUTPUT:
        rSample     initial guess for the sample point
                   first index has values 0,1,2 for the x,y,z
coordinate
                   second index corresponds the range


         ^     rInline
          |--------->
          |.
          | .
    rSat  |  .
          |   . range
          |    .
          |     .
     --------------- Earth
          |     .
          |   .
          | .   rSample
```

(See starsdoc.pro)

9.2.57 STARS_SAMPLE2NODE

[Previous Routine] [Next Routine] [List of Routines]

```
 NAME:
      STARS_Sample2Node

 PURPOSE:
     Compute all nodes affected by all sample of a pulse

 INPUT:
      r_nodes       vectors to the centers of the nodes grid
                    first index 0=x 1=y 2=z
                    second index along track index of node (called
row)
                    third  index across track index of node (called
column)
      rSample_E     vectors to sample points
                    first index sample
                    second index 0=x 1=y 2=z
      CenterRow     along track index of the subsatellite node
      CentralIndex  across track index of the subsatellite node
      p_DNode       distance of nodes
      FilterWidth   width of the spatial filter to be applied
```

```
      phi           angle of the beam in [rad] 90.0 deg = forward
                    180 deg = left ...

 OUTPUT:
      matrix of dimensions : 3 x NumberOfSamplesOnGrid x MaxNumNodes
                and type double

      the matrix contains for each sample which is in the frame of
      the node grid (start and end of simulation !!) and each node
      that is affected by the sample ( distance < FilterWidth)
      the along track, and across track index as well as the distance.



      index 1 = indices of the sample. If no node within the radius
                sqrt(2) * p_DNode/2 is found no output will be
generate
                for the sample. Note that this will happen to the
outermost
                samples first. Therefore the index of the returned
                samples will alway correspond to the index of the
                input samples. However, the outermost samples might be
                cut off starting from a certain sample.
       index 2  runs with the affected nodes. The size is the maximum
                possible number of affected nodes. Unused elements
                are flagged through the values explained with index 3.
                The index2 =0
                will always point to the next neighbour node, order
                of the others have no meaning. However the array is
                filled from the starting index. That means a soon
                element is flagged unused the following elements
                of this sample need not to be considered any more


       index 3 =  0   :  along track indices of affected nodes
                         (-2 = unused)
                  1   :  across track indices of affected nodes

                  2   :  distance of affected nodes

                  3   :  weight of the sample for this node
```

(See starsdoc.pro)

9.2.58 STARS_SUNSYNCINCLINATION

[Previous Routine] [Next Routine] [List of Routines]

```
 NAME:
     STARS_sunSyncInclination
```

```
 PURPOSE:
     Compute inclination for sun synchronization.

     Sun synchronization is achieved, when the motion of the ascending
node (due to
     the flattening of the Earth) is equal to the motion of the Earth
about the
     Sun.

     The flattening is considered by the J2-Term and the semimajor
axis of the
     body. An approximation formula is used.

     Note, that only the Sun-Earth configuration is considered.

     Reference:  Renner, Satellitentechnik

 INPUT:
       a          Semimajor axis of orbit        [m]
       e          Eccentricity of orbit          [-]

 OUTPUT:
      i          Inclination for synchronicity [rad]
                 if no sun synchronous orbit exists, the routine stops
                 with an error message
```

(See starsdoc.pro)

9.2.59 TEST_COMPUTEPULSEGEO

[Previous Routine] [Next Routine] [List of Routines]

```
 NAME:
      test_ComputePulseGeo

 PURPOSE:
      Check of the PulseGeometry Computation Function
      the prf is lowered to speed up the test
      the scanRate is lower such that the single rotations are
      visible on the earth map



 INPUT
     None


 OUTPUT
    1. earth map with
         green subsatellite track
```

```
        magenta : outmost point of antenna beam
        yellow  : antenna postion magnified by TotalSwath/2.0
                  the antenna position shall be perpendicular to
                  the beam
  2.    3D plot with
        blue    : earth
        green   : orbit
        magenta : beam vector * TotalSwath + satellite vector
```

(See test_drivers.pro)

9.2.60 TEST_COMPUTERANGEVECTOR

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     test_computeRangeVector

PURPOSE:
     Check of the Range Vector routine with a simplified
     pulse loop

INPUT:
     None


OUTPUT:
```

(See test_drivers.pro)

9.2.61 TEST_COMPUTESNR

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     test_computeSNR

PURPOSE:
     Check the computation of SNR by performing all operations
     in IDL step by step with intermediate steps

INPUT:
     None


OUTPUT:
```

(See test_drivers.pro)

9.2.62 TEST_PLOTLN

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     test_plotLN

PURPOSE:
     Visualisation of local Normal and subsatellite track



INPUT
     r_E  Satellite position for each time step in earth fixed co-
ordinates
     rG_E    position of Nadir point (local normal) in earth fiex
     co-ordinates
OUTPUT
   some plots
```

(See test_drivers.pro)

9.2.63 TEST_PLOTNODEMAP

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     test_plotNodeMap

PURPOSE:
    plot all nodes computed so far in optimized map

INPUT:
     None


OUTPUT:
```

(See test_drivers.pro)

9.2.64 TEST_PLOTNODES

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     test_plotNodes

PURPOSE:
     Visualisation of the nodes
```

```
INPUT
    Nodes data-structure


OUTPUT
   some of the nodes on earth map , leftmost row red, middle row
   green, right most row blue , yellow line is the ellipse
   pependicular to filght direction
```

(See test_drivers.pro)

9.2.65 TEST_PLOTSEARTH

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     test_plotsEarth

PURPOSE:
     Visualisation of earth in a plot_3DBox



INPUT
   common block constants
OUTPUT
   plotted earth ellipsoid
```

(See test_drivers.pro)

9.2.66 TEST_PLOTSELLIPSE

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     test_plotsellipse

PURPOSE:
     Visualisation of intersection of plane with earth ellipsoid
     test_plotLN must be called first


INPUT
    vector containing AE,BE,CE of the ellipse to be drawn
    vector N with local normalized velocity
    vector G Nadir point
OUTPUT
   plotted ellipse in 3D and on earth map ( yellow line shows nadir
   and velocity)
```

(See test_drivers.pro)

9.2.67 TEST_PULSELOOPSAMPLES

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     test_pulseLoopSamples

PURPOSE:
     Check the Sample positions comupted in the pulseLoop


INPUT:
     None


OUTPUT:
    Plot of earth showing position of samples
```

(See test_drivers.pro)

9.2.68 TEST_RANGEIG

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     test_rangeIG

PURPOSE:
     Check of the Inital Guess routine for Range Point search
algorithm

INPUT:
     None


OUTPUT:
     plot with :
             white  lines from satellite to range points
             red    line  from centre of earth to range points
     print statement showing that the range point is in  the correct
     direction a has the correct length
```

(See test_drivers.pro)

9.2.69 TEST_SAMPLE2NODES

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     test_Sample2Nodes
```

```
PURPOSE:
     Verification of the STARS_SamplesNode routine

INPUT:


OUTPUT
```

(See test_drivers.pro)

9.2.70 TEST_SETUPEARTHBOX

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     test_SetupEarthBox

PURPOSE:
     Initialise a 3D Box plot such that the earth and orbits are
     displayed correctly



INPUT
    common block constants and ColorIndex


OUTPUT
```

(See test_drivers.pro)

9.2.71 UPDATEVIEW

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     UpdateView

PURPOSE:
     Accumulate the view specifiec data in view data structure


INPUT:
     t            time of current pulse
     rSat_E       position of satellite at time of pulse
     rSample_E    positions of the samples on earth related to this
                  pulse
```

```
      SNRprime     SNR/simga-naught for each sample

      common blocks NodeInfo, ProcessingPars,
                    InstrumentPars,ViewInfo,SimulationPars

OUTPUT
   the View data structures in the common block ViewInfo are filled
```

(See viewcalc.pro)

9.2.72 WAITKBRD

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
      WaitKbrd

PURPOSE:
      Stop execution until user hits keyboard if current device
      is not postscript
      Used for visual inspection

INPUT:
      None

OUTPUT:
      None
```

(See plotps.pro)

9.2.73 WRITEPL1

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
      WritePL1

PURPOSE:
      Write the pseudo level 1 b files


INPUT:


OUTPUT
```

(See writepl1.pro)

9.2.74 WRITEPL1HEADER

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     WritePL1Header

PURPOSE:
     Write the pseudo level 1b header


INPUT:
     lun    logical unit of the open file

OUTPUT
```

(See writepl1.pro)

9.2.75 WRITEPL1VECTOR

[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
     WritePl1Vector

PURPOSE:
     Write a vector of double/float values
     into the ASCII file


INPUT:
     lun  logical unit of the open file
     v    vector to be written to the file

OUTPUT
```

(See writepl1.pro)

9.2.76 WRITEVIEW

[Previous Routine] [List of Routines]

```
NAME:
     WriteView

PURPOSE:
```

```
     Write  the views on one node to the pseudo level 1 b file

INPUT:

OUTPUT
```

(See writepl1.pro)
