User's Guide to NASDA's SAR products Ver.3

Masanobu Shimada, Ph.D.

July 12, 2002

Senior Scientist
Earth Observation Research Center
National Space Development Agency of Japan
Harumi 1-8-10, Harumi island triton square office tower X 22
Chuo-Ku, Tokyo-To, Japan, 104-6023

Shimada@eorc.nasda.go.jp

NDX-000291

Voice 81-3-6221-9071
Fax: 81-3-6221-9192

Revision Record

Revision	Date	Page	Description	Approved
0	March 10 '93	All	New issue	
1	Feb. 16, 1998	P. 11	Cal coefficients	
2	Aug. 9 2001	P.11	Cal factor	
3	July 12 2002	P.16	Source code	

Contents

1. Guideline

This document summarizes the algorithms to obtain geometric and radiometric information from NASDA's SAR products. A list of the physical information to be derived from the SAR products is given in the following section and its derivation method is introduced in this documentation is given in the following contents.

This document temporally covers level 2.1, NASDA's standard product.

2. Contents of the parameters

A list of the parameters is given in the following table.

2.1 Geometric conversion

a)	Slant range	R_{slant}
b)	Off nadir angle	ξ
c)	Incidence angle	θ
d)	Latitude and longitude of image pixel	φ,λ
e)	Averaged radius of the earth at satellite sub-track	R_{sub}
f)	Averaged radius of the earth at scene center	R_{scene}

2.2 Radiometric conversion

a) Radiometric conversion method

2.3 Antenna Pattern

a) Antenna elevation pattern

2.4 Sensor parameters

2.5 Computer algorithm to obtain above parameters.

3. Geophysical parameters

3.1 Slant range

NASDA's SAR correlated product experiences the range walk processing before azimuth correlation, then, slant range does not have the same geometric parameters, such as incidence angle, slant range, etc., over one azimuth line (See Figure 1). To help the users to obtain the slant rage information, NASDA's product, whose level is more than 1.1(3 looks), makes each azimuth line include three slant ranges' values, each of which is corresponding to the first, mid, and last range line, respectively, at the top of the each image record. Because these slant ranges do not change radically in range direction, the values are set to the same over N azimuth lines. Then, these values change every N azimuth lines.

Let R_{slant}(P, L) be the slant range at some certain point in the imagery with the address of P and L. On the azimuth line, which includes the meaningful slant range information, and hereafter also named i^{th} azimuth line with slant range information (ALSRI). Then, R_{slant}(P, L) at i^{th} ALSRI is given by the following square power expression.

At i^{th} ALSRI, following data set will be available, such as:

$$\begin{aligned}&(R_{slant,1}{}^{i},1)\\&(R_{slant,3000}{}^{i},3000)\ \text{ for } i^{th}\text{ ALSRI,}\\&(R_{slant,6000}{}^{i},6000)\end{aligned}\tag{1}$$

where 1, 3000, 6000 are the addresses of the range lines with slant range information for each ALSRI. Arranging these three values, slant range over i^{th} ALSRI is expressed by a function of second order power formula as:

$$R_{slant}{}^{i}(P) = a^{i}P^{2} + b^{i}P + c^{i} \qquad (2)$$

where, i is corresponding to Li th azimuth line.

Over an image, several sets of coefficients a^i, b^i, c^i, are obtained. Then, these parameters will be approximated by the second order power formula of azimuth line L as above equation by the least square method. Finally, we will have the following model.

$$R_{slant}(P,L) = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ b_{11} & b_{12} & b_{13} \\ c_{11} & c_{12} & c_{13} \end{pmatrix} \begin{pmatrix} L^{2} \\ L \\ 1 \end{pmatrix} \begin{pmatrix} P^{2} \\ P \\ 1 \end{pmatrix} \qquad (3)$$

where, a_{1i} is the coefficient of the second order power approximation model over all the slant range lines available. b_{1i}, c_{1i} take same information.

3.2 Off nadir angle x

Off nadir angle of the certain pixel (P, L) is given by the triangular relationship.

$$x = \cos^{-1}\left(\frac{R_{slant}(P,L)^{2} + R_{1}{}^{2} - R_{scene}{}^{2}}{2R_{slant}(P,L)R_{1}}\right) \qquad (4)$$

where, R_1 is the distance between earth center and satellite position, and is given by the following formula:

$$R_{1} = \sqrt{x^{2} + y^{2} + z^{2}} \qquad (5)$$

where, x, y, and z are the satellite positions, which are placed at the addresses of 45 - 60, 61 - 76, and 77 - 92 with the format of F16.7 in platform position record of the leader file.

R_{scene} : Earth radius at the scene center, which is given by the section

3.3 Incidence angle q

Incidence angle of the certain pixel (P, L) is given by the following relationship.

$$q = j + x \qquad (6)$$

where,

$$j = \cos^{-1}\left(\frac{R_{1}{}^{2} + R_{scene}{}^{2} - R_{slant}{}^{2}}{2R_{scene}R_{1}}\right) \qquad (7)$$

3.4 Latitude and longitude of image pixel

Latitude and longitude of each image pixel, whose address is given by (P, L), is modeled by the following power expression with ten coefficients for each component.

$$
\begin{aligned}
j &= f_1(P,L) = a_0 + a_1P + a_2L + a_3PL + a_4P^2 + a_5L^2 + a_6P^2L + a_7PL^2 + a_8P^3 + a_9L^3 \\
l &= f_2(P,L) = b_0 + b_1P + b_2L + b_3PL + b_4P^2 + b_5L^2 + b_6P^2L + b_7PL^2 + b_8P^3 + b_9L^3
\end{aligned}
\qquad (8)
$$

where, φ,λ are geodetic latitude and longitude, respectively. a_i, b_i are the coefficients stored in byte address from 947 to 1346 of data processing data record in the leader file, each of which has a format of E20.10.

For the derivation of P and L corresponding to latitude and longitude, it is given by the iteration method based on the Newtonian method. Suppose that φ and λ are given, and P and L are the first guesses for the corresponding answers, ΔP and ΔL are the contributions to the next step. Then, above equation (8) is given by the following form.

$$
\begin{aligned}
j &= f_1(P+\Delta P, L+\Delta L) = f_1(P,L) + \frac{\P f_1(P,L)}{\P P}\Delta P + \frac{\P f_1(P,L)}{\P L}\Delta L \\
l &= f_2(P+\Delta P, L+\Delta L) = f_2(P,L) + \frac{\P f_2(P,L)}{\P P}\Delta P + \frac{\P f_2(P,L)}{\P L}\Delta L
\end{aligned}
\qquad (9)
$$

Rearranging equation (9) in matrix form, we have the following linear equation.

$$\mathbf{M} \bullet \mathbf{DR} = \mathbf{B} \qquad (10)$$

where

$$\mathbf{M} = \begin{pmatrix} \dfrac{\P f_1(P,L)}{\P P} & \dfrac{\P f_1(P,L)}{\P L} \\ \dfrac{\P f_2(P,L)}{\P P} & \dfrac{\P f_2(P,L)}{\P L} \end{pmatrix} \qquad (10\text{-}1)$$

$$\mathbf{DR} = \begin{pmatrix} \Delta P \\ \Delta L \end{pmatrix} \qquad (10\text{-}2)$$

$$\mathbf{B} = \begin{pmatrix} j - f_1(P,L) \\ l - f_2(P,L) \end{pmatrix} \qquad (10\text{-}3)$$

Then, the following gives the contribution to the next iteration.

$$\mathbf{R} = \mathbf{R} + \mathbf{DR} \qquad (11)$$

$$\mathbf{DR} = \mathbf{M}^{-1} \bullet \mathbf{B} \qquad (11\text{-}1)$$

$$\begin{pmatrix} \Delta P \\ \Delta L \end{pmatrix} = \frac{1}{\dfrac{\P f_1}{\P P}\dfrac{\P f_2}{\P L} - \dfrac{\P f_1}{\P L}\dfrac{\P f_2}{\P P}} \begin{pmatrix} (j - f_1)\dfrac{\P f_2}{\P L} - (l - f_2)\dfrac{\P f_1}{\P L} \\ (l - f_2)\dfrac{\P f_1}{\P P} - (j - f_1)\dfrac{\P f_2}{\P P} \end{pmatrix} \qquad (11\text{-}2)$$

Iteration will be continued until the following condition is cleared.

$$\left|\frac{\mathbf{DR}}{\mathbf{R}}\right| \le 1.0e^{-5} \qquad (12)$$

where,

$$\begin{aligned} \frac{\P f_1}{\P P} &= a_1 + a_3 L + 2a_4 P + 2a_6 PL + a_7 L^2 + 3a_8 P^2 \\ \frac{\P f_1}{\P L} &= a_2 + a_3 P + 2a_5 L + a_6 P^2 + 2a_7 PL + 3a_9 L^2 \\ \frac{\P f_2}{\P P} &= b_1 + b_3 L + 2b_4 P + 2b_6 PL + b_7 L^2 + 3b_8 P^2 \\ \frac{\P f_2}{\P L} &= b_2 + b_3 P + 2b_5 L + b_6 P^2 + 2b_7 PL + 3b_9 L^2 \end{aligned} \tag{13}$$

3.5 Averaged radius of the earth at satellite sub-track

This radius R_{sub} is given by the following equation.

$$R_{sub} = \frac{b}{\sqrt{1 - e^2 \cos(f)^2}} \tag{14}$$

where, $$e = \frac{\sqrt{a^2 - b^2}}{a}$$

(15)

f : Geocentric latitude of the sub satellite point

j : Geodetic latitude of the sub satellite point. This information is placed at the address of 453 - 460 in the Data set summary of the Leader File with the character form of F8.3.

a : Equatorial radius of the earth model, which is given at the address of 181 - 196 in the Data Set summary of leader file with the character form of F16.7.

b : Polar Radius the earth model, which is given at the address of 197 - 212 in the Data Set summary of leader file with the character form of F16.7.

A conversion of geodetic value to geocentric value is given by:

$$f = \tan^{-1}\left(\frac{b^2}{a^2} \tan j\right) \tag{16}$$

3.6 Averaged radius of the earth at scene center

Averaged radius of the scene center (*R_{scene}*) is given by the above method with inputting the scene center geodetic latitude given by the address of 117 - 132 of the Data Set Summary Record of the Leader File with the character form of F16.7.

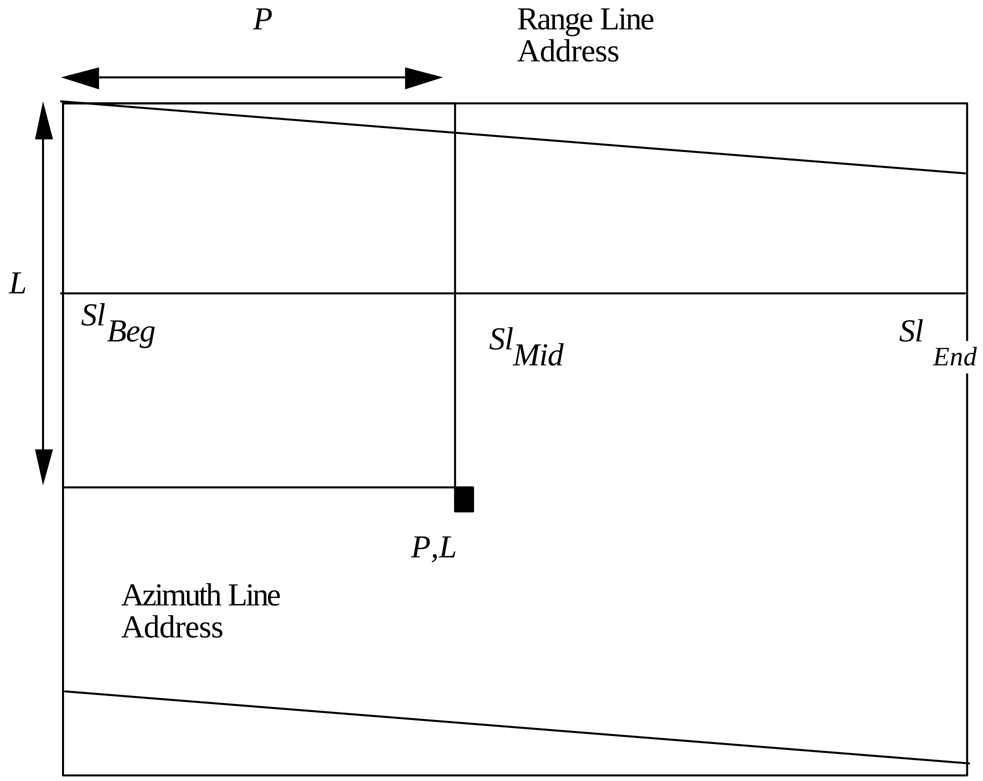

Figure 1 Image coordinate of standard SAR product (Level 2.1)

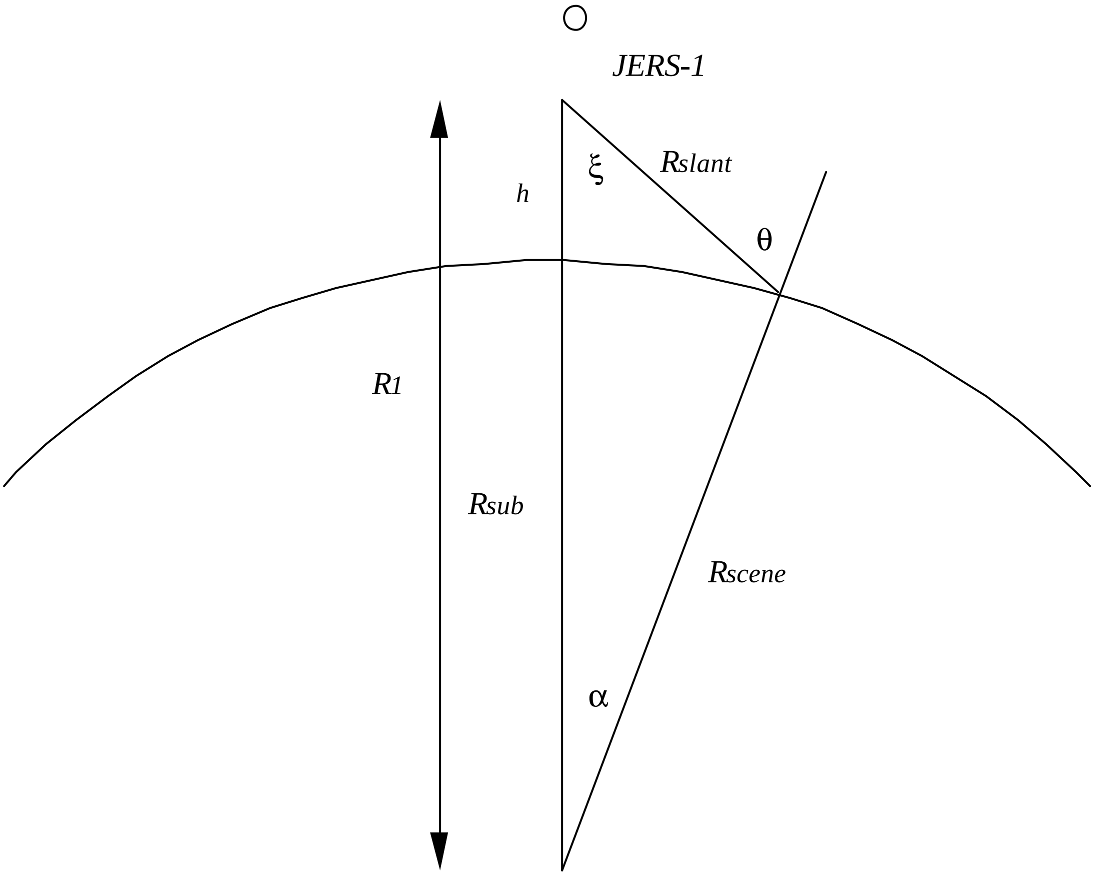

Figure 2. Coordinate system of the SAR imaging, where JERS-1 is moving from the above of this paper to the behind.

4. Radiometric conversion

a) NRCS conversion of the image count

Normalized Radar Cross Section (NRCS) of the target area in NASDA's standard product, whose product level is 2.1, is calculated from the following equation. This simple equation is obtained by the analysis of the point target responses (ref 1) which are included in the NASDA's standard test area having several 2.4 and 2.0 m trihedral corner reflectors, and active radar calibrators.

As shown in Figure 1, the dependency of the conversion factor on off nadir angle seems to a linear relationship with a small inclination. The less confidence due to the smaller experiments points, we have obtained the constant over all the off nadir angles.

The processing parameters of the SAR processor has experience several improvements, such as the antenna off nadir angle, AGC correction parameters. Then, the conversion parameters were listed in Table 1 as the function of the processed date.

$$NRCS = 20\log_{10}(I) + CF\,[dB] \tag{17}$$

where, I stands for pixel value of each pixel, which expresses from 0 to 2^{15}-1.

Table 1 processing parameters of NASDA JSAR products

Processed date	CF[dB] for JSAR
By Feb. 14, 1993	-70.0
After Feb. 15, 1993	-68.5
After Nov. 1, 1996	-68.2
After April 1 2000	**-85.34**

Table 2 processing parameters of NASDA AMI products

Processed date	CF[dB] for JAMI
By Feb. 14, 1993	-65.3
After Feb. 15, 1993	-65.3

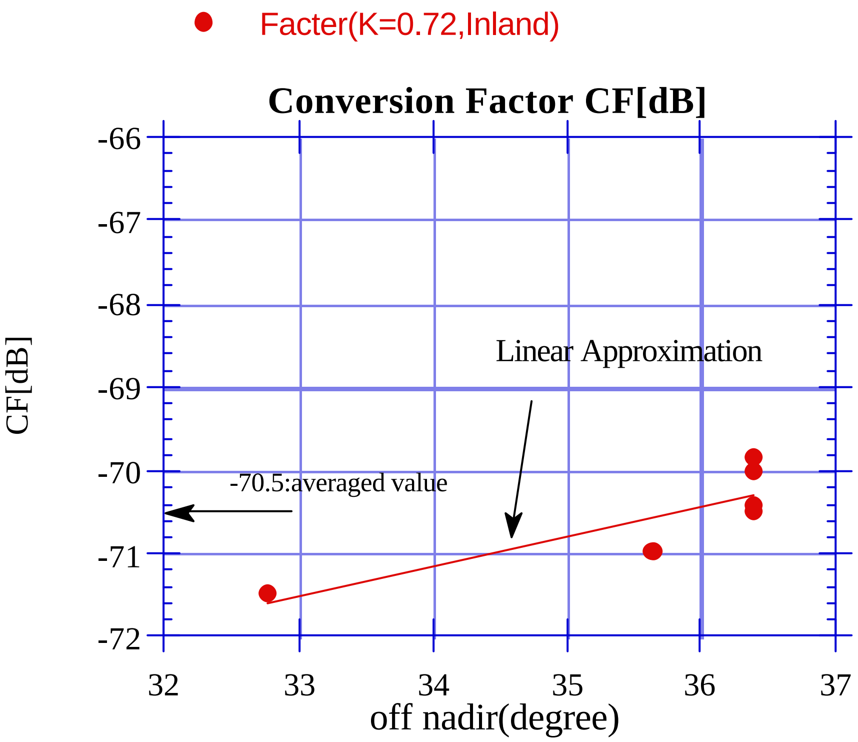

Figure 3 Off nadir angle dependency of Conversion factor (CF) from SAR standard product to NRCS

[Remarks]; Above mentioned conversion relation between pixel intensity and normalized radar cross section (NRCS) has the following limitation on its application to the SAR images. If the SAR product includes week intensity targets, such as ocean, river, lake, and dry desert area, in the near edge region (0 - 12 km from the near edge) and higher intensity targets, such as urban area, forested area, and wet land in the middle and far edge, the product **should not** be applied for the NRCS conversion because of the following reasons.

In order for the SAR to work, even in the smaller dynamic range of 3 bit AD converter in signal processing unit of SAR system, for the various types of the target signal, SAR controls the receiver gain by using the AGC, Automatic Gain Control technique. The 7 μsec sampling window through of which data are evaluated for setting the AGC gain is placed at the very beginning of the 360 μsec SAR observation window, whose leading edge is at 22 μsec from the beginning of observation window. This region is corresponding to 0.675 km to 6.975 Km in slant range, and it also corresponds to from 1 km to 12 km on the ground. If there are very low intensity targets at the near edge of the SAR product as shown in Fig 4, some areas out side of this near edge region are supposed to be saturated because of AGC procedure.

Then, this conversion relation should not be used for these regions (dotted region in Fig. 4). Instead, other regions, which are white in Fig 4 are valid for the NRCS conversion relation.

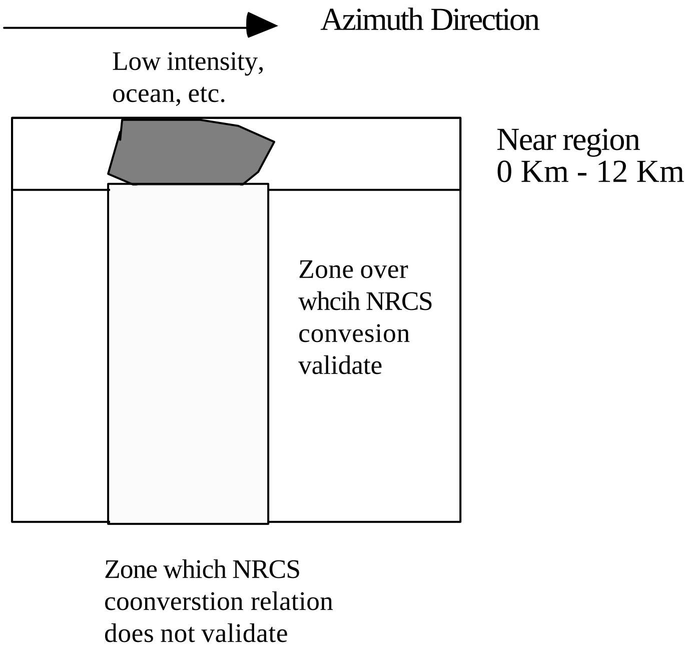
Azimuth Direction
Low intensity, ocean, etc.
Near region 0 Km - 12 Km
Zone over whcih NRCS convesion validate
Zone which NRCS coonverstion relation does not validate

Figure 4 Validation criteria of Eq. (17) on the image intensity distribution in a scene

5 Antenna Pattern

a) Antenna elevation pattern

Antenna Elevation pattern was obtained by the statistical evaluation of the SAR radar signals received from the Amazon rain forest which is supposed to be the uniform scatterer from the near range to the far range. An average of the twelve screened images which were acquired by the constant SAR parameters such as the constant STC start time and constant AGC level in the image offered the following approximation for the antenna elevation pattern.

$$G_{ele}(f) = a(f - f_0)^2 + c(f - f_0)^4 \quad \text{[dB]} \qquad (18)$$

where, a , c, and ϕ_0 are unknown parameters and antenna off nadir angle (degree). These parameters were determined as follows;

a	=	-0.38392
c	=	-0.0039542
ϕ_0	=	35.023 degree

Standard deviation of model error is 0.1 dB

The modeled antenna pattern is given in the figure 1.

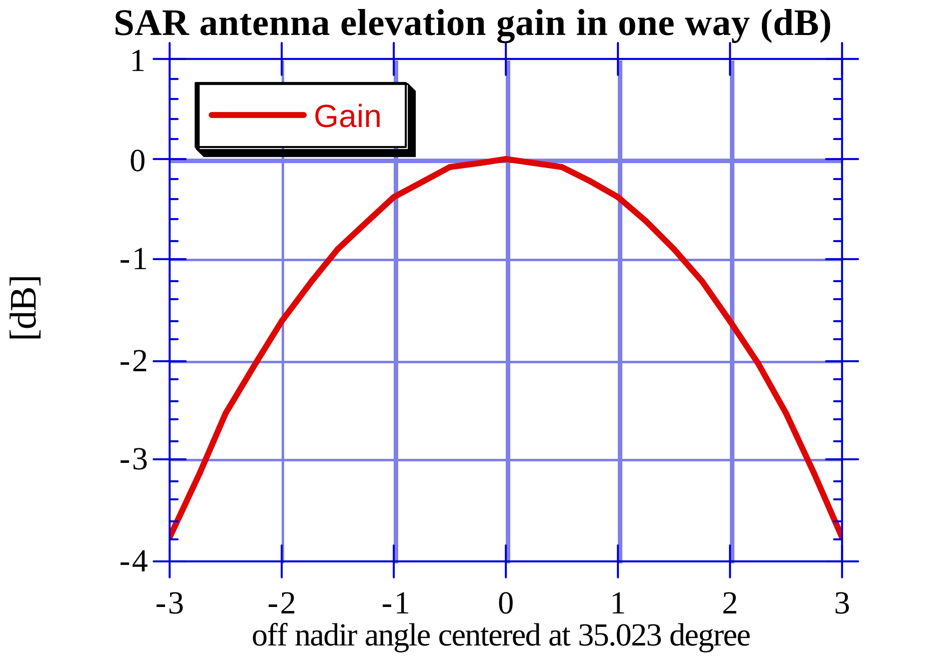

Figure 5. SAR antenna elevation pattern (One way pattern)

6 Sensor parameters

If the data product is in level 0, 1.0, and 1.1 (1 look complex product), sensor operation parameters of total 69 bits are included at the beginning of each image record. These parameters represented by STC start time, AGC gain, STC change pattern, STC offset time, combination of the high power amplifier, etc., are the ones change rapidly. These parameters locate from 301th byte to 323 th byte of each image record in such a way that 69 bits data in total is divided into 23 in the unit of 3 bits, then these 3 bits are put at the lower positions of each byte. The more retailed information is given in the following table.

Bit No.	Item	Bit definition			Remarks;
1	PRF ON/OFF 1	ON			
		0	OFF		
2 - 4	PRF	000	1505.8Hz		
		001	1530.1Hz		
		010	1555.2Hz		
		011	1581.1Hz		
		100	1606.0 Hz		
5	Calibration mode	1	ON		"1" is given in both of
CAL					
	ON/OFF	0	OFF		and OBS mode
6	Observation mode	1	ON		"1" is given in OBS mode
	ON/OFF	0	OFF		"0" is given in CAL mode
7 - 16	Initial Position of	00000		Pattern 1	The first 5 bits represents
	STC Time variation	00001		Pattern 2	STC START TIME
		-------			Change pattern
		10111		Pattern 24	
			00101	60 µsec	The last 5 bits shows
			00110	70 µsec	Initial STC start time

			11101	300 µsec	
17 - 21	STC START TIME	00101	60 µsec		See Fig 6
		00110	70 µsec		Validate if mode is OBS
		-------			CAL mode shows "00000"
		11101	300 µsec		
22 - 24	STC Offset TIME 000	0 - 360 µsec			See Fig 6
		001	10 - 370 µsec		

		111	70 - 430 µsec		
25	AGC/MGC	1	AGC		valid in OBS mode
26	AGC time cons.	1	128/PRF		
		0	64/PRF		
27 - 31	AGC data	00000	0 dB		valid at both of CAL and
		00001	1 dB		OBS mode in AGC mode

		11111	31 dB		
32 - 36	gain control status	00000	0 dB		valid at OBS in MGC
	(GC STA)	00001	1 dB		"00000" is given in OBS
of		-------			AGC
		11111	31 dB		

37 - 39STP	ATT	000	0 dB		valid in CAL
	(GC STA)	001	3 dB		"000" is given in OBS

		111	21 dB		
40	AUTO/MANUAL	1	AUTO		
		0	MANUAL		
41	Observation start/	1	start		"1" is given in the CAL of
	end	0	end		OBS mode
42 - 44	Combiner of	Sw1	Sw2	Sw3	
	SW1, SW2, SW3	001	*	*	B
		010	A	B	*
		011	A	B	B
		100	B	*	*
		101	B	*	B
		110	B	A	A
		111	*	A	A
45	Receiver SW	1	A or B		
		0	unfixed		
46	STC CAL	1	CAL mode		
		0	OBS mode		
47	CAL SW	1	CAL mode		
		0	OBS mode		
48 - 49	Standby mode	00	emergency off		"00" is given at the first selection
		01	Standby (1)		CAL in OBS mode
		10	Standby (2)		
		11	Standby (3)		
50	Change of status	1	MGC		
		0	AGC		
51	Frequency Synth.	1	ON		
		0	OFF		
52	Control Unit	1	ON		
		0	OFF		
53	Transmitter Unit	1	ON		
		0	OFF		
54	Receiver Unit	1	ON		
		0	OFF		
55	Signal Proc Unit	1	ON		
		0	OFF		
56	High Power Amp	1	ON		
	Unit A	0	OFF		
57	High Power Amp	1	ON		
	Unit B	0	OFF		
58	High Power Amp	1	ON		
	Unit R	0	OFF		
59	STC initial start	1	OBS. mode		
	Time	0	CAL. mode		
60	STR	1	ON		
		0	OFF		
61 - 68STR status					

69 spare

Acronym

PRF : Pulse repetition frequency

Initial position of STC start time: SAR has 24 patterns for the STC start time changes.

AGC/MGC : Normally SAR is activated in AGC mode only. MGC is no longer expected in the standard operation.

AGC data : The bigger the AGC value is, the smaller the receiver gain is. This means that AGC represents the attenuator.

STP ATT : This value stands for the attenuation value applied for the calibration signal generator.

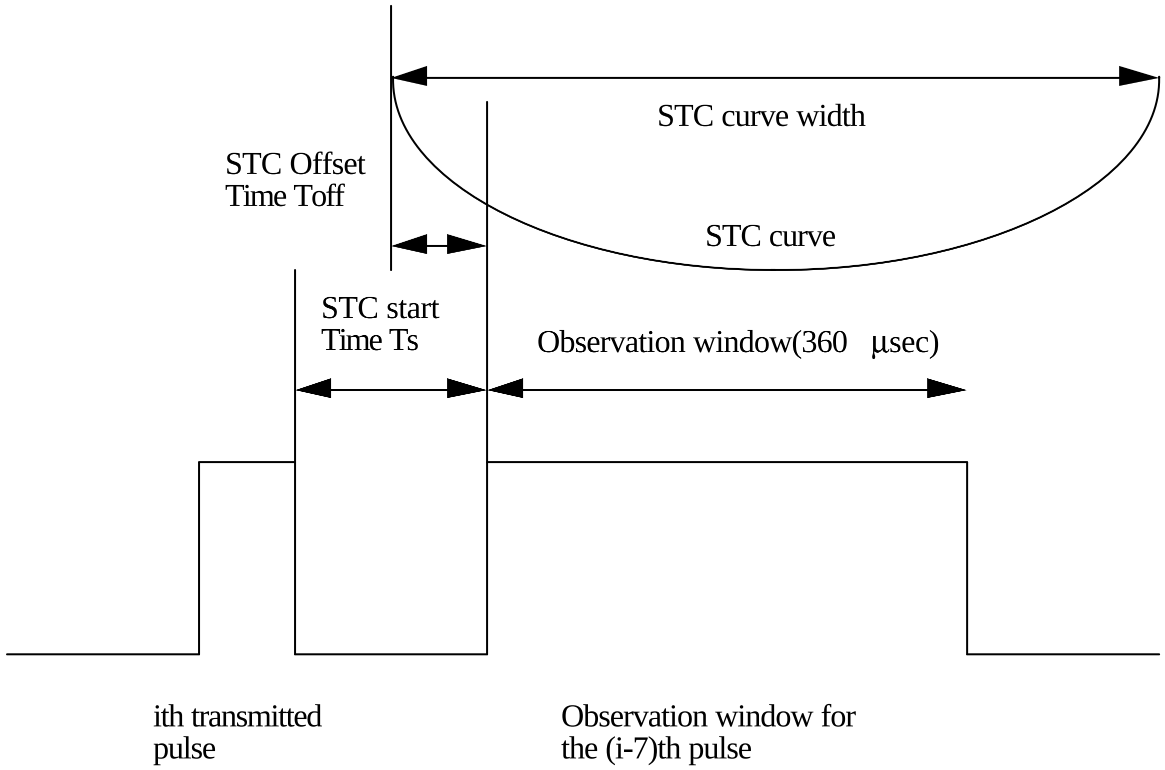

Figure 6. A relationship among transmission pulse window and receiving window, STC start time, offset time, etc.

7 Computer algorithm to obtain above parameters.

```
#include    <stdio.h>
#include    <math.h>
#include    <stdlib.h>
#include    <string.h>
#define RA1    6378.140  /* radius at equator(C-7) */
#define RB1    6356.755  /* radius at polar  (C-7) */
#define P18    0.017453292519943З  /* PAI/180.0   */

extern      char  *cvector(int nl, int nh);
extern      void  image_inf_21(FILE *fp);
extern      void  free_cvector(char *v,int nl);
extern      void  covert_latlon_pl(float lat, float lon, int *l, int *p, FILE *fp);
extern      void  nrerror(char *error_text);
extern      void  slant_pre(FILE *fp, float **para, int imax, int imaxyn, int nop);
extern      unsigned char *ucvector(int nl,int nh);
extern      int   *ivector(int nl,int nh);
extern      void free_ucvector(unsigned char *v,int nl);
extern      void free_ivector(int *v,int nl);
extern      float **matrix(int nrl,int nrh,int ncl,int nch);
extern      void free_matrix(float **m,int nrl,int nrh,int ncl);
extern      void  geo_parameter(int l, int p, float *parameter, float **para, FILE
*fp, FILE *fp1);
extern      int         lstsq(int n,double *x,double *t,double *c,int m);



/***********************************************************/
/*    General of scene.c                  Jan. 22 '92       */
/*    Masanobu Shimada                                      */
/***********************************************************/
/*    Four corners' and scene center latitude and longitude are stored in the processor
addition information record. This subroutine shows how to obtain these information
"*/
/* This routine is assigned by the file pointer "fp" for the data summary record,
and this print out the scene center and four corner latitude and longitude */
/*    cvector  : see Mathematic recipe in C     */
void  image_inf_21(FILE *fp)
{
      static char *datac1, *datac2;
      float dataf[10];
      int   j, ii1, ii;
      long  n;

      datac1      = cvector(0, 2048);
      datac2      = cvector(0, 8);

      rewind(fp);
      n = fread(datac1, sizeof(char), 2048, fp);
      for(j=0 ; j<10;j++){
            ii1   = j*8;
            for(ii = 0 ; ii<8 ; ii++)
            datac2[ii]  =     datac1[1746 + ii1 + ii];

            dataf[j] = (float) atof(datac2);
      }

      printf(" center (%8.3f,%8.3f)¥n", dataf[0], dataf[1]);
      printf(" up       (%8.3f,%8.3f)  -  (%8.3f,%8.3f)¥n",  dataf[2],  dataf[3],
      dataf[4], dataf[5]);
      printf(" bottom (%8.3f,%8.3f) - (%8.3f,%8.3f)¥n", dataf[6], dataf[7],
      dataf[8], dataf[9]);

      free_cvector(datac1, 0);
      free_cvector(datac2, 0);
}
/*void      covert_latlon_pl(lat, lon, l, p, fp)     */
/****************************************************************/
/*    covert_latlon_pl.c                      Dec. 26 '92       */
/*    Jan. 22 '93                                               */
/*    Masanobu Shimada                                          */
/****************************************************************/
/*    This routine determines the appropriate pixel and line address of the point
with specified latitude and longitude with the processor additional information record,
which is specified by the file pointer of "fp" */
/*    The determination of these image address is based on the Newtonian equation
given in the section 3.4     */
void  covert_latlon_pl(float lat, float lon, int *l, int *p, FILE *fp)
{
      static      char  *datac1, *datac2;
      double      dataf[20];
      double      f1, f2, df1dp, df2dp, df1dl, df2dl, l1, p1, dl1, dp1, rat,
            delta;
      int         j, ii1, ii;
      long        n;

      datac1      = cvector(0, 2048);
      datac2      = cvector(0, 22);

      rewind(fp);
      n = fread(datac1, sizeof(char), 2048, fp);
      for(j=0 ; j<20;j++){
            ii1   = j*20;
            for(ii = 0 ; ii<20 ; ii++)
                  datac2[ii]  =     datac1[946 + ii1 + ii];

            dataf[j] = ((double) atof(datac2));
      }

      l1    =     3000.0;
      p1    =     3000.0;
      rat   =     1.0;

      while(rat >= 1.0e-3){

            f1    = dataf[0] + dataf[1]*p1 + dataf[2]*l1 + dataf[3]*p1*l1
                  + dataf[4]*p1*p1 + dataf[5]*l1*l1 + dataf[6]*p1*p1*l1 +
                  dataf[7]*p1*l1*l1 + dataf[8]*p1*p1*p1 + dataf[9]*l1*l1*l1;
            f2    = dataf[10] + dataf[11]*p1 + dataf[12]*l1 + dataf[13]*p1*l1
                  + dataf[14]*p1*p1 + dataf[15]*l1*l1 + dataf[16]*p1*p1*l1
                  + dataf[17]*p1*l1*l1 + dataf[18]*p1*p1*p1
                  + dataf[19]*l1*l1*l1;
            df1dp = dataf[1] + dataf[3]*l1 + 2*dataf[4]*p1 + 2*dataf[6]*p1*l1
                  + dataf[7]*l1*l1 + 3*dataf[8]*p1*p1;
            df1dl = dataf[2] + dataf[3]*p1 + 2*dataf[5]*l1 + dataf[6]*p1*p1
                  + 2*dataf[7]*p1*l1 + 3*dataf[9]*l1*l1;
            df2dp = dataf[11] + dataf[13]*l1 + 2*dataf[14]*p1
                  + 2*dataf[16]*p1*l1 + dataf[17]*l1*l1 + 3*dataf[18]*p1*p1;
            df2dl = dataf[12] + dataf[13]*p1 + 2*dataf[15]*l1 + dataf[16]*p1*p1
                  + 2*dataf[17]*p1*l1 + 3*dataf[19]*l1*l1;
            delta = df1dp*df2dl - df1dl*df2dp;

            dp1   = ((lat - f1)*df2dl - (lon-f2)*df1dl)/delta;
            dl1   = ((lon - f2)*df1dp - (lat-f1)*df2dp)/delta;

            p1    += dp1;
            l1    += dl1;

            rat   = fabs(dp1/p1);
      }

      *p    = (int)p1;
      *l    = (int)l1;
      free_cvector(datac1 , 0);
      free_cvector(datac2 , 0);
}
/*****************************************************/
/*    Geophysical parameters                         */
/*    slant range - preparation                      */
/*    Masanobu Shimada                               */
/*****************************************************/
/*    Slant range of each pixel is taken out from the information given at the beginning
of each azimuth line. Then, the two dimension al modeling of these data in azimuth
and range is given      */
void  slant_pre(FILE *fp, float **para, int imax, int imaxyn, int nop)
{
      static      float       **data;
      static      unsigned    char  *datauc;
      static      int               *datai;
      int         i, line, j, n;
      long        n1;
      float       sl[3];
      double      xdata[10], tdata[10], cdata[10];

      data        = matrix(0, 100, 0, 3);
      datauc      = ucvector(0, imax);
      datai       = ivector(0, 100);

      rewind(fp);
      n     =     0;
      for(line = 0 ; line<imaxyn ; line++){

            n1 = fread(datauc, sizeof(unsigned char), imax, fp);

            for(i=0 ; i<3;i++){
                  sl[i] = 0.0;
                  for(j=0 ; j<3;j++)
                        sl[i]+= (float)datauc[64+j+4*i]*pow(256.0,3.0-j)/1000.0;
            }

            if(line == 0 || sl[0]   !=    data[n-1][0]){
                  printf("s1 =%e s2 = %e s3 = %e n = %5d
                        line = %5d¥n", sl[0],sl[1],sl[2],n, line);
                  tdata[0]    =     0;
                  tdata[1]    =     (nop-1)/2.0;
                  tdata[2]    =     nop-1.0;
                  for(i=0 ; i<3;i++)
                        xdata[i]    =     sl[i];
                  lstsq( 3, xdata, tdata, cdata, 2);
                  for(i=0 ; i<3;i++)
                        data[n][i]  =     cdata[i];
                  datai[n]    = line;
                  n++;
            }

      }

      for(j=0 ; j<3;j++){
            for(i=0 ; i<n ; i++){
                  tdata[i]    = (double)datai[i];
                  xdata[i]    = (double)data[i][j];
            }
            lstsq(n, xdata, tdata, cdata, 2);
            for(i=0 ; i<3;i++)
                  para[j][i]  = cdata[i];
      }

      free_matrix(data, 0, 100, 0);
      free_ucvector(datauc, 0);
      free_ivector(datai, 0);

}
/*****************************************************/
/*    geo-parameter.c              Dec. 26 '92       */
/*    latitude, longitude                            */
/*    off nadir angle, incidence angle               */
/*    slant range                                    */
/*    this is available for level 2.1                */
/*    Masanobu Shimada                               */
/*****************************************************/
/*    This routine obtains geometric parameters for nay address of line and pixel,
such as latitude, longitude, earth radius at the pixel, earth radius at satellite
nadir, height of the satellite nadir, slant range, off nadir, and incidence angle.
To obtain these parameters, two ancillary files should be allocated    */
void  geo_parameter(int l, int p, float *parameter, float **para, FILE *fp, FILE
*fp1)
                  /*    fp    : proc_21              */
                  /*    fp1   : obt_21               */
/*float     parameter[];*/    /* 0  : latitude                          */
                        /* 1  : longitude                       */
                        /* 2  : earth radius at the pixel       */
                        /* 3  : earth radius at satellite nadir */
                        /* 4  : height of the satellite         */
                        /* 5  : slant range                     */
                        /* 6  : off nadir                       */
                        /* 7  : incidence                       */
{
      static      char  *datac1, *datac2, *datac3;
      float dataf[20], a0, a1, a2, slant;
      float lat, lon, a_sub, r_sub, h_sub, r_scene, phai, inci, off, px, eps2,
      phai1;
      int         ii, ii1, j;
      long  n;

      datac1      = cvector(0, 4680);
      datac2      = cvector(0, 20);
      datac3      = cvector(0, 16);

      rewind(fp);
      rewind(fp1);
      n = fread(datac1, sizeof(char), 2048, fp);
      for(j=0 ; j<20;j++){
```

```
      ii1   = j*20;
      for(ii = 0 ; ii<20 ; ii++)
            datac2[ii]  =     datac1[946 + ii1 + ii];

      dataf[j] = ((float) atof(datac2));
}

lat   =     dataf[0]
      +     dataf[1]*p
      +     dataf[2]*l
      +     dataf[3]*p*l
      +     dataf[4]*p*p
      +     dataf[5]*l*l
      +     dataf[6]*p*p*l
      +     dataf[7]*p*l*l
      +     dataf[8]*p*p*p
      +     dataf[9]*l*l*l;

lon   =     dataf[10]
      +     dataf[11]*p
      +     dataf[12]*l
      +     dataf[13]*p*l
      +     dataf[14]*p*p
      +     dataf[15]*l*l
      +     dataf[16]*p*p*l
      +     dataf[17]*p*l*l
      +     dataf[18]*p*p*p
      +     dataf[19]*l*l*l;

parameter[0]      = lat;
parameter[1]      = lon;

eps2              = (RA1*RA1-RB1*RB1)/RA1/RA1;
phai              = atan(RB1*RB1/RA1/RA1*tan(lat*P18));
r_scene           = RB1/sqrt(1-eps2*cos(phai)*cos(phai));
parameter[2]      = r_scene;

n = fread(datac1, sizeof(char), 4680, fp1);
for(j=0 ; j<3;j++){
      ii1   = j*16;
      for(ii = 0 ; ii<16 ; ii++)
            datac3[ii]  = datac1[44 + ii1 + ii];

      dataf[j] = ((float) atof(datac3))/1000.0;
}
a_sub = sqrt(pow(dataf[0],2.0) + pow(dataf[1],2.0) + pow(dataf[2],2.0));
phai1 = asin(dataf[2]/a_sub);
r_sub = RB1/sqrt(1-eps2*cos(phai1)*cos(phai1));
h_sub = a_sub - r_sub;
parameter[3]      = r_sub;
parameter[4]      = h_sub;

a0    = para[0][0] + para[0][1]*l + para[0][2]*l*l;
a1    = para[1][0] + para[1][1]*l + para[1][2]*l*l;
a2    = para[2][0] + para[2][1]*l + para[2][2]*l*l;

slant = a0 + a1*p + a2*p*p;

off   = acos((slant*slant + a_sub*a_sub - r_scene*r_scene)/2/slant/a_sub);
px    = acos((r_scene*r_scene - slant*slant + a_sub*a_sub )
/2/a_sub/r_scene);
```

```
      inci  = px + off;

      parameter[5]      = slant;
      parameter[6]      = off;
      parameter[7]      = inci;

      free_cvector(datac1 , 0);
      free_cvector(datac2 , 0);
      free_cvector(datac3 , 0);
}


/************************/
/*    cvector.c               */
/************************/
char *cvector(int nl,int nh)
{
      char *v;
      v     =(char *)malloc((unsigned) (nh-nl+1)*sizeof(char));
      if (!v) nrerror("allocation failure in cvector()");
      return v-nl;
}
/************************/
/*    free_cvector.c          */
/************************/
void free_cvector(char *v,int nl)
{
      free((char*) (v+nl));
}


/************************/
/*    nrerror.c               */
/************************/
void nrerror(char *error_text)
{
      fprintf(stderr,"Numerical Recipes run-time error...¥n");
      fprintf(stderr,"%s¥n",error_text);
      fprintf(stderr,"...now exiting to system...¥n");
      exit(1);
}
/************************/
/*    ucvector.c              */
/************************/
unsigned char *ucvector(int nl,int nh)
{
      unsigned char *v;
      v     =(unsigned char *)malloc((unsigned) (nh-nl+1)*sizeof(unsigned char));
      if (!v) nrerror("allocation failure in ucvector()");
      return v-nl;
}

/************************/
/*    ivector.c               */
/************************/
int *ivector(int nl,int nh)
{
      int *v;
      v     =(int *)malloc((unsigned) (nh-nl+1)*sizeof(int));
      if (!v) nrerror("allocation failure in ivector()");
      return v-nl;
```

```
}
/************************/
/*    free_ucvector.c         */
/************************/
void free_ucvector(unsigned char *v,int nl)
{
      free((char*) (v+nl));
}

/************************/
/*    free_ivector.c          */
/************************/
void free_ivector(int *v,int nl)
{
      free((char*) (v+nl));
}

/************************/
/*    matrix.c                */
/************************/
float **matrix(int nrl,int nrh,int ncl,int nch)
{
      int i;
      float **m;

      m      =(float **) malloc((unsigned) (nrh-nrl+1)*sizeof(float*));
      if (!m) nrerror("allocation failure 1 in matrix()");
      m -= nrl;

      for(i=nrl;i<=nrh;i++){
            m[i]  =(float *) malloc((unsigned) (nch-ncl+1)*sizeof(float));
            if(!m[i]) nrerror("allocation failure 2 in matrix()");
            m[i] -= ncl;
      }

      return m;
}

/************************/
/*    free_matrix.c           */
/************************/
void free_matrix(float **m,int nrl,int nrh,int ncl)
{
      int i;
      for(i=nrh;i>=nrl;i--) free((char*) (m[i]+ncl));
      free((char*) (m+nrl));
}

/*****************************************/
/*    lstsq.c               1988/11/06*/
/*****************************************/
/* obtain the most suitable function(power formula) for the input data;

   x(t) = c[0] + c[1]*t + c[2]*t**2 + ........... + c[m]*t**m

1) input parameter
                    m        :   order number( 1 <= m <= 20)
                    n        :   no of data ( n > m )
                    x[]      :   input data(observed data)
                    t[]      :   input data
2) output parameters
```

```
                    c[]       :   obtained coefficients
3) returned parameter
                    code      :   error code  0     normal end
                                             999   error
*/


int         lstsq(int n,double *x,double *t,double *c,int m)
{
    int     mp1, mp2, m2, i, j, k, l;
    double  w1, w2, w3, pivot, aik, a[21][22], w[42];

    if(m >= n || m < 1 || m > 20 ) return (999 ) ;
    mp1     =     m+1;
    mp2 =   m+2;
    m2      =     m*2;

    for( i=0; i<m2; i++){
        w1 = 0.0;
        for(j=0; j<n; j++){
            w2 = w3 = t[j];
            for(k=0; k<i; k++){
                w2 *= w3;
            }
            w1 += w2;
        }
        w[i] = w1;
    }

    for(i=0; i<mp1; i++){
        for(j=0; j<mp1; j++){
            l = i+j-1;
            a[i][j] = w[l];
        }
    }

    a[0][0] = n;
    w1      = 0.0;
    for(i=0; i<n; i++){
        w1 += x[i];
    }
    a[0][mp1] = w1;
    for(i=0; i<m; i++){
        w1 = 0.0;
        for(j=0; j<n; j++){
            w2 = w3 = t[j];
            for(k=0; k<i; k++){
                w2 *= w3;
            }
            w1 += x[j] * w2;
        }
        a[i+1][mp1] = w1;
    }
    for(k=0; k<mp1; k++){
        pivot = a[k][k];
        for(j=k; j<mp2; j++){
             a[k][j] /= pivot;
        }
        for(i=0; i<mp1; i++){
            if(i != k){
                aik = a[i][k];
                for(j=k; j<mp2; j++){
                    a[i][j] -= aik*a[k][j];
                }
            }
        }
    }
    for(i=0; i<mp1; i++){
        c[i] = a[i][mp1];
    }
    return(0);
}
```

8. References

1. W.H.Press, B P. Flannery, S A. Teukolsky, B T Vetterling, Cambridge University Press, "Numeric recipes in C : The Art of Scientific Computing"
2. CCT format description
3. Japanese ERS-1 to ground station interface description, Rev-2, Oct. 1990, HE-88023

9. Q & A

(1) [Q] Is it possible to retrieve the necessary information like STC, AGC, gains control and ADC noise from a product level 1.1 or 2.1?

[A] This information should be referred to 69 bits telemetry data in each received signal. As mentioned in Section 6, these informations exist in the products of level 0, 1.0, and 1.1 (1-look complex). There are two level 1.1 products in NASDA. Level 1.1 1 look complex product has a data alignment in range direction, which is like the raw data. But, level 1.1 3 look product has a data alignment in azimuth as same as level 2.1.

STC changes frequently as every 30 sec. AGC does as every 64 pulses. Then, user who wants to know the status of AGC, STC, etc. in that corrected image, level 0, which is the smallest in terms of size, should be accessed.

(2) [Q] How to use the left and right fill counter

[A] Both of right and left fill numbers are stored at byte 17 th to 20 th and at 29 th to 32 th in each image record. They are constantly filled by 0. Single frequency SAR product does not have a good meaning for these right and left fill counter. But, when the image is multi band data, like Landsat, OPS, etc., these data are fixed to same values over the all bands so that there can not exist the color fringing due to the band to band registration.

(3) [Q] How to compute the incidence angle, slant range, etc.

[A] Please refer to section 3.

(4) [Q] How to convert the image intensity to NRCS

[A] Please refer to section 4.

Originally, we intended to store the conversion parameter in CCT format. But, currently, we are pessimistic to set one parameter for the conversion even we stated the conversion model with one parameter in this documentation. As we have more planning to acquire the calibration experiment, we expect to have the some relationship between correction parameter and off nadir angle. In this situation, one parameter is not good enough for the expression of conversion relation. Because of that, we abandoned to fix the CCT format for this purpose.

Whenever we change the correction parameters we will inform you of the coefficient by FAX, e-mail, etc.