# GERB Geolocation Rectification : Algorithms and tests description.

Aline Hermans

September 15, 2000

## Contents



CHANGE RECORD

| Issue | Date | Approved by | Reason for change |
|---|---|---|---|
| Version 1 | 25/07/00 | | new document |
| | | | |

# 1 Introduction

## 1.1 Algorithm Requirement.

The purpose of the GERB Geolocation Rectification algorithm is to interpolate GERB data (radiances or fluxes, or... ), from a Non Averaged, Non Rectified Geolocation (NANRG) grid, into a nominal, regular and centered geolocation grid. This nominal grid is defined by constant sampling viewing angles in both directions (East-West and South-North), from satellite nominal position.

## 1.2 Algorithm implementation.

The GERB Geolocation Rectification algorithm has been implemented as a function to be included in the RMIB GERB internal library. Subsequently, the function is using structures and sub-functions of this internal library. The main function called "GP_GerbGeolocRectification" and its new sub-functions are described in the following section.

## 1.3 GERB Geolocation Rectification in the scope of the RMIB GERB ground segment.

The GERB Geolocation Rectification is used in an auxiliary program of the RMIB GERB ground segment, called "ToARG" program.

This auxiliary program is aiming to convert the L2 radiances and fluxes images, produced by the GERB processing at NANRG (Non-Averaged and Non-Rectified GERB Geolocation) resolution and at GERB SW time, into ARG products (i.e. nominal time Averaged and Rectified Geolocated data -cf. L1.5 ARG RAL products-[1]).

# 2 Functions description

## 2.1 GP_GerbGeolocRectification

**Purpose:**

–> interpolates data value from a NANRGeolocation grid
to a regular and centered grid.

**Syntax:**

```
IP_image *GP_GerbGeolocRectification
        ( IP_image      *data,
          IP_images     *geoloc,
          double        input_D_azim,
          double        input_D_elev,
          double        grid_D_azim,
          double        grid_D_elev,
          int           grid_width,
          int           grid_height,
          IP_images     **rectified_geoloc )
```

**Subordinates:**

–> GP_ConvertToCoordinatesInRegularGrid
MP_NoRegularBiLinearInterpolation
GP_FindNeighbours

**Dependencies:**

used in program “ToARG” (see doc : ToARG,ToBARG,CorrApply [2])

**Description:**

This function is aiming to interpolate data values ”data”,
from a Non Regular Geolocation grid ”geoloc” (cf. NANRG files),
to a regular grid (cf ARG files).
The input geolocation is defined by its geolocation files ”geoloc”,
and by its mean sampling angles : ”input_D_azim”, ”input_D_elev”
( For usual NANRG files :
input_D_azim = GERB_D_AZIM = 0.0703...(defined in gerbproc.h)
input_D_elev = GERB_D_ELEV = 0.0703...(defined in gerbproc.h) )
The regular grid is defined as a centered, equiangular sampling grid
(of size : grid_width * grid_height)
The constant angular sampling following West-East direction is
defined by argument ”grid_D_azim”, while the constant angular sampling
following South-North direction is defined by argument ”grid_D_elev”.
The algorithm can be divided in 2 steps :

A/ The coordinates of the input non-regular geolocation grid,
(expressed in latitude and longitude)
are converted :
First, into viewing angular coordinates (azimuth, elevation)
from the nominal satellite position.
Then, into float coordinates in the regular grid

B/ For each regular grid point,
1. A sub-function ”GP_FindNeighbours”,
identifies the 4 nearest valid data positions
surrounding the grid point.
2. Then data value interpolation is performed :
- If all 4 neighbours are valid,
a non-regular bilinear interpolation is calculated
- If only 3 or 2 neighbours are valid,
the 3 or 2 data values are weightened according to distance
from the grid point.
- If only one neighbour is valid and its distance < preset limit
this data value is used.
- If no valid neighbour point is found,
the grid point is declared invalid (ERR_VALUE)

**Outputs**

The main output is an IP_image structure representing data values
interpolated into the nominal, regular and centered geolocation grid.
Moreover, if the argument ”IP_images **rectified_geoloc” !=NULL,
the function outputs the new rectified geolocation tables
and an interpolation flag :

rectified_geoloc->image[0] is expressing latitudes in degrees
rectified_geoloc->image[1] is expressing longitudes in degrees
rectified_geoloc->image[2] is a flag image representing the number
of data points used for interpolation of
the data value at each grid point.

## 2.2 GP_ConvertToCoordinatesInRegularGrid

Purpose:
–> convert a GERB geolocation table into table of coordinates (i,j)
with respect to a regular and centered grid.

**Syntax:**

```
IP_images *GP_ConvertToCoordinatesInRegularGrid
                        ( IP_images *geoloc,
                          double D_azim,  double D_elev,
                          int grid_width, int grid_height,
                          double *i_deviation, double *j_deviation )
```

**Subordinates:**

GP_GeodeticGeolocToSatAzimElev (RMIB GERB internal library)

**Dependencies:**

called by GP_GerbGeolocRectification in program “ToARG”

**Description:**

This function calculates,for each geolocation
(defined by its geodetic latitude and longitude in IP_images ”geoloc”),
the coordinates (i,j), expressed in double, with respect to a regular
and centered grid.
This regular and centered grid is defined by its size (grid_width* grid_height),
and its constant sampling viewing angles :
”D_azim” in the West-East direction,
”D_elev” in the South-North direction.
This function is also computing mean (i,j) deviations.
(i.e. difference between real pixel coordinates (i,j)
and ideal pixel coordinates if GERB geolocation grid
was regular and centered )

## 2.3 GP_FindNeighbours

Purpose:
–> Look for the 4 closest data points surrounding one regular grid pixel (i,j).

**Syntax:**

```
int GP_FindNeighbours ( int i, int j, int box_size,
                        double input_D_azim, double input_D_elev,
                        double grid_D_azim, double grid_D_elev,
                        double i_meandev, double j_meandev,
                        IP_images *grid_position, IP_image *data
                        double *neighbours_coord[4][3],
                        double *neighbours_data[4]          )
```

**Subordinates:**

**Dependencies:**

sub-function of function "GP_GerbGeolocRectification"used in program "ToARG"

**Description:**

This function look for the 4 closest data points to the regular grid pixel (i,j)
and store the coordinates of these 4 closest neighbours
(in table "neighbours_coord")
and their data values (in table "neighbours_data").

Step-by-step, this function :

* Looks for the data pixel (x_center,y_center) assuming to be the nearest from
the considered grid point (i,j), taking into account the global mean
deviations (i_mean_deviation, j_mean_deviation)

* Does an exhaustive search among data pixels located in a box
(of size "box_size") centered at the supposed nearest pixel (x_center,
y_center) in order to find :
- the closest top-left neighbour data point -> index 0
- the closest top-right neighbour data point -> index1
- the closest bottom-right neighbour data point -> index 2
- the closest bottom-left neighbour data point -> index 3
Note no assignement is done if :
- either the data point is geolocated outside Earth ,
- or the data value is not valid (=ERR_VALUE).

* Store in array "neighbours_coord" :
- The position (in the regular grid) of the 4 closest neighbours
and their distance with respect to the considered regular grid
point (i,j).

* Store in array "neighbours_data" the according data values.

* The function returns the number of valid neighbours
that have been stored.

## 2.4 MP_NoRegularBiLinearInterpolation

Purpose:
–> Performs bilinear interpolation between 4 data points
that are not located on a perfect square.

**Syntax:**

```
double MP_NoRegularBiLinearInterpolation
                              ( double data[4],
                                double data_coord[4][2],
                                double output_coord[2] ) ;
```

**Subordinates:**

**MP_LinearInterpolation (RMIB GERB internal library)**

**Dependencies:**

called by GP_GerbGeolocRectification in program “ToARG”

**Description:**

Performs the bilinear interpolation between the 4 values ”data”,
located at points of coordinates ”data_coord”,
to the location ”output_coord”.
The interpolation weighting coefficients are computed with respect
to the 4 input points coordinates.
The coordinate [0] is assumed to be the horizontal coordinate.
The coordinate [1] is assumed to be the vertical coordinate.
The 4 input points (*) are assumed to surround the ”output_coord”
point (+), as represented in the following figure.

```
                 pt.1
     pt.0         *
         *


            +


      pt.3 *
                  *  pt.2
```

If the configuration of the 4 input points (*) does not follow these figure
the function returns the defined value : ERR_VALUE = -10000

# 3 Test :

## 3.1 test of sub-functions

### 3.1.1 GP_ConvertToCoordinatesInRegularGrid

**Purpose of test:**

Test validity of values returned by function
(i.e. the returned IP_images, and hence the double values “i_deviation” and “j_deviation”)

**Methods:**

1. We generate different cases of geolocation tables (see following table), using our program “mainGerbGeo”. This program generates images representing latitude and longitude according to pixels of an GERB (256*256) image, and saves them in a IP_images structure.

2. We give these tables as inputs to the function and check outputs :

- sign of “i_deviation’ and “j_deviation”

- plot of outputs images ->image[0] = lines coordinate ,->image[1] = columns coordinate ,->image[2] = flag on/out Earth + errors.

Input geolocation tables :

| name | d_pixel | d_global | comments |
|---|---|---|---|
| geo0 | 0 | 0 | perfect centered grid |
| geo1 | 0 | 0.05 | grid de-centered in East direction |
| geo5 | 0.05 | 0 | randomly slightly perturbed grid |
| geo10 | 1 | 0 | randomly highly perturbed grid |
| geo11 | 0.1 | 0 | randomly quite perturbed grid |

**Pass/fail criterions:**

- the field values output for line coordinate (column coordinate) are included in grid_height (grid_width) and present a good shape.
- "i_deviation' and "j_deviation" are signed according to parameter "d_global"

**Test inputs and results :**

| geoloc | grid_width | D_azim | => | i_dev | j_dev | plot |
|---|---|---|---|---|---|---|
| | =grid_height | =D_elev | | | | |
| geo0 | 256 | GERB | | 0 | 0 | OK |
| geo1 | " | " | | 0 | 40.7 | OK |
| geo5 | " | " | | -0.005 | -0.01 | OK |
| geo11 | " | " | | -0.005 | -0.02 | OK |
| geo10 | " | " | | -0.005 | -0.19 | OK |
| | | | | | | |
| geo0 | 150 | GERB | | 0 | 0 | OK |
| geo0 | 400 | GERB | | 0 | 0 | OK |
| geo0 | 256 | 0.05 < GERB | | 0 | 0 | OK |
| geo0 | 256 | 0.1>GERB | | 0 | 0 | OK |
| geo0 | width>height | GERB | | 0 | 0 | OK |
| geo0 | 256 | D_azim>D_elev | | 0 | 0 | OK |

**Date of test: 29 june 2000**

**Status of test: PASS**

### 3.1.2 GP_FindNeighbours

**Purpose of test :**

–> Check for several chosen grid points, existence and positions of the returned neighbours.

**Methods:**

Write on screen for several chosen grid points :

1. Coordinates of the searching box center (x_center, y_center).
2. Searching Box limits (x: start->stop , y: start->stop).
3. Coordinates of the found neighbours according to their position indice.

**Pass/Fail criterion:**

Check - cases where 4 , 3, 2, 1 or no neighbours are found.
- neighbours position according to their index.
- neighbours distance calculation "by-hand"
- neighbours are surrounding with choosen grid point.

**Outputs examples:**

From geolocation table : geo 5 :

**4** neighbours case :

grid pt.(14,141)
box center = (14.005157,141.009828)
box limits : x=13->16 y=140->143
neighbour[0] : 13.877930 140.314713 dist: 0.484520 data: 31.229717
neighbour[1] : 13.877929 141.314713 dist: 0.113945 data: 29.948435
neighbour[2] : 14.771758 141.324493 dist: 0.700907 data: 38.595604
neighbour[3] : 14.771758 140.324493 dist: 1.051920 data: 39.104004

**3** neighbours case :

grid pt.(6,142)
box center = (6.005157,142.009828)
box limits : x=5->8 y=141->144
neighbour[0] : 5.400208 140.845001 dist: 1.693773 data: 0.000000
neighbour[2] : 6.123473 142.158768 dist: 0.040453 data: 0.000000
neighbour[3] : 6.123475 141.158768 dist: 0.722918 data: 0.000000

**2** neighbours case :

grid pt.(5,119)
box center = (5.005157,119.009828)
box limits : x=4->7 y=118->121
neighbour[2] : 5.400207 119.845009 dist: 0.874206 data: 0.000000
neighbour[3] : 5.400207 118.845009 dist: 0.184188 data: 0.000000

**1** neighbour case :

grid pt.(13,84)
box center = (13.005157,84.009828)
box limits : x=12->15 y=83->86
neighbour[0] : 12.972995 84.073761 dist < 0.01 data: 5.481750
From geolocation table : geo1 :

**4** neighbours case :

grid pt.(10,140)
box center = (10.000000,99.256335)
box limits : x=9->12 y=98->101
neighbour[0] : 9.999999 139.743668 dist: 0.065706 data: 16.987831
neighbour[1] : 8.999998 140.743668 dist: 1.553045 data: 5.661047
neighbour[2] : 10.000001 140.743668 dist: 0.553042 data: 18.818810
neighbour[3] : 11.000001 139.743668 dist: 1.065708 data: 28.696436

**Date of test: 29 June 2000**

**Status of test: PASS**

#### 3.1.3 MP_NoRegularBiLinearInterpolation

**Purpose of test 1:**

Check output interpolated value if the 4 input points are really surrounding the unknown point.

**Methods:**

We compare output value with value calculated "by hand"

**Pass criterion:**

value computed by function = value caculated by hand.

**Date of test: March 2000**

**Status of test: Pass**

Purpose of test 2 :
Check output if the 4 input points do not surround the unknown point.

**Methods:**

We input different cases of bad configuration.

**Pass criterion:**

value computed by function = error value = -10000

**Date of test: March 2000**

**Status of test: Pass**

### 3.2 GP_GerbGeolocRectification

**Purpose of test:**

Test function ability to output correct data values interpolated into a regular and centered grid as defined by the inputs arguments.

**Methods:**

1. We generate different cases of geolocation tables (see input geolocation table used to test "Convert-ToCoordinatesInRegularGrid").
2. We combine the different cases of input geolocation tables with different sizes of regular grid ( For instance, regular grid of bigger (or smaller) size, regular grid with bigger (or smaller) sampling angles,...)
3. Then we check outputs :

- we plot image of output data values expressed in the regular and centered grid.
- we plot images of regular grid geolocation and flag : ->image[0] = latitudes ,->image[1] = longitude ,->image[2] = flag on/out Earth + numbers of valids neighbours.

**Pass/fail criterions:**

- We check if output image sizes are as required.
- We check if global Earth shape is according to “D_azim”, “D_elev”.
- We check if the global shape of output interpolated data is similar to the global shape of input data.
- We check the number of valid neighbours. We take care about pixels where no valid neighbour is found, and look for explanations.

**Test inputs and outputs:**

| geoloc | grid_width | D_azim | => | plot remarks | criterions |
|---|---|---|---|---|---|
| | =grid_height | =D_elev | | | |
| geo0 | 256 | GERB | | OK | PASS |
| geo1 | “ | “ | | OK | PASS |
| geo5 | “ | “ | | OK | PASS |
| geo11 | “ | “ | | OK | PASS |
| geo10 | “ | “ | | 165 error values | PASS |
| | | | | | |
| geo0 | 150 | GERB | | image smaller & Earth same size | PASS |
| geo0 | 400 | GERB | | image bigger & Earth same size | PASS |
| geo0 | 256 | 0.05 < GERB | | Earth bigger | PASS |
| geo0 | 256 | 0.1>GERB | | Earth smaller | PASS |
| geo0 | width>height | GERB | | rectangle image | PASS |
| geo0 | 256 | D_azim>D_elev | | Earth=ellipse | PASS |

**Date of test: 3 july 2000**

**Status of test: PASS**

# References

[1] S.Dewitte, “RGP: Data products description”, RMIB technical report, MSG-RMIB-GE-IF-2,2000.

[2] A.Hermans, “ToARG, ToBARG,CorrApply program descriptions”, RMIB technical report, MSG-RMIB-GE-TN_19 ,2000.