Developing and validation tests for the Resolution Enhancement program

Aline Hermans, Luis Gonzalez

September 15, 2000

Contents

1 Scope of this document **3**

2 Introduction **3**

3 Tests of the development using simulated artificial data **3**
3.1 ResEnhFilesExistenceTest 4
3.2 InitialiseCSbyInterpolCG 5
3.3 CalculSmoothingFactor 5
3.4 SeviriCoordOfGerbGeoTable 6
3.5 CalculGeoconvolConstraintErr 6
3.6 CalculLagrangeGradient 7
3.7 CalculCoefForSmoothingFactor 8
3.8 Check stop criteria of iterative scheme 8
3.9 Test iterative process 8
3.10 Error propagation tests 9
3.11 Test of output files storage on disk 12

4 Improvement tests on images derived from Meteosat 7 **12**
4.1 Generation of testing images 12
4.2 Test methods of α selection 13
4.3 Test of influence of correction area 13

5 Tests on images derived from Meteosat 7 , rescaled. **14**
5.1 Tests on r factor 14
5.2 Tests algorithm with conjugated gradient. 15
5.3 Tests on images perturbed to 10% , 20% or 40%. 15
5.4 Test $\epsilon_s^{required}$influence 16
5.5 Summary of our optimal choice of parameters and options 17
5.6 Required RAM and disk space. 17
5.7 Examples of input and output testing images. 18
CHANGE RECORD

Issue	Date	Approved by	Reason for change
Version 1	25/08/00		new document
Version 2	06/09/00		DELTA review RIDs

1 Scope of this document

The scope of this document is to describe tests performed for development and validation of the Resolution Enhancement program. This document should be read as an annex of the main RE document (MSG-RMIB-GE-TN-03[3]) that explain development strategy and give and describes the final algorithm implementation.

2 Introduction

Tests performed for development and validation of the Resolution Enhancement program were divided into 3 stages :

During the first stage, focused on development of algorithm sub-functions, we used very simple modeled images, representing a sphere recovered by big moving rectangles, as those used for Gerb Processing tests [1]. We call this test phase, described in section 2, the tests of the development using simulated artificial data.

Then when each function and subfunction has been checked and when a first algorithm version was running properly, we started the second stage, focused on testing and improving algorithm performance. For this purpose, we tried to use more realistic testing images. We simulated "what should be" the GERB and SEVIRI view from Meteosat 7 images. We take into account of calibration differences between both instruments by introducing artificial perturbation factors.

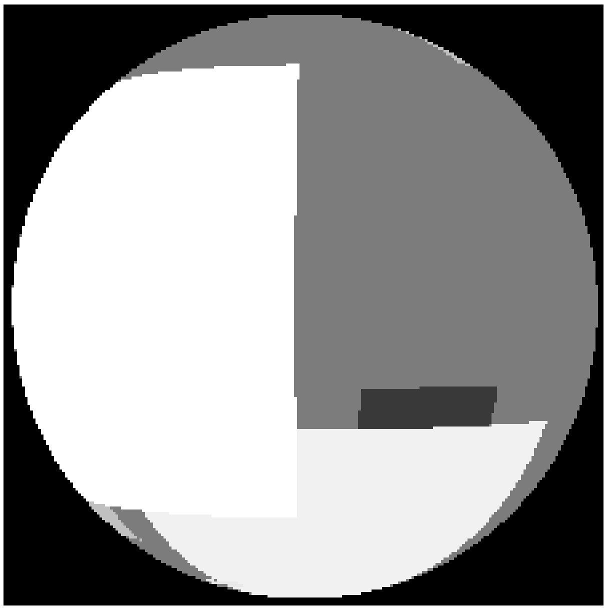
First, we simulated GERB and SEVIRI images at a reduced scale, i.e. SEVIRI size is matching Meteosat 7 image size (833*853 pixels) and, according to this scale, GERB size is 64*64. These testing stage is described in section 3.

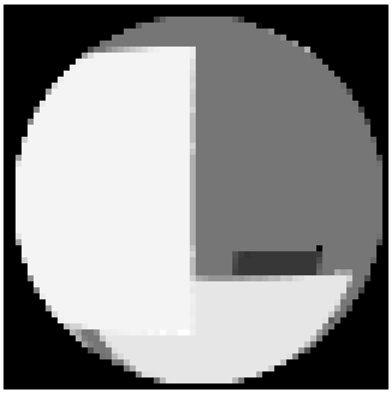
Finally, to go closer and closer to the real configuration, we scaled up the simulated images to their real sizes, and we used to generate the GERB images, PSF computed randomly, slightly different according to different GERB detectors. This last testing stage is described in section 4.

3 Tests of the development using simulated artificial data

The Resolution Enhancement (RE) algorithm was checked, as development goes along, using the same sets of artificially simulated data, as for Gerb Processing tests (see [1] and [2]). SEVIRI images of size 1200x1200 are generated as if the instrument was looking to a sphere covered by big moving rectangles. Then, Simulated GERB images of size 256x256 are derived from simulated SEVIRI images. Each GERB pixel results from the convolution of a 15x15 SEVIRI pixels area with a function that models the GERB detector PSF (Point Spread Function).

The following figures show an example of a simulated SEVIRI image (left side) and a derived GERB image (right side):





These sets of simply modeled images allowed us to check whether each function and subfunction of the RE algorithm is working correctly and to get a first version of the program running properly.

We will describe the performed tests according to each subfunction following the order of the algorithm structure description (see RE schemes in annex or in document [4]) and using sets of artificial data issued from Gerb Processing tests. The table describing these artificial data sets according to a reference date is printed in annex (p.23) or in document [1].

3.1 ResEnhFilesExistenceTest

Purpose of the test:

Check the "source files existence test", i.e check function "ResEnhFilesExistenceTest".

Methods:

We launch the main Resolution Enhancement program according to one specific L20 file, after having deleted a source file.

Pass/fail criterion:

Test passes if the program stops and exits properly.

Inputs:

We use data set according to reference time 19800101120237, by typing the command lines :

ResEnh SF -v -p -i L20_SF_19800101120237_01.rma
ResEnh TF -v -p -i L20_TF_19800101120237_01.rma

Outputs:

- SF case :

Missing source file	result	status of the test
	program exits properly	PASS
L15_GEO_19800101120237_01.ims	“	“
L15_MaD_19800101120237_01	“	“
DetectorIndex_used	“	“
PSF_SW_045.rma	“	“
C_SF_19800101001737_01.rma	“	“
GSFE_19800101120000_01.rma	“	“
SevB_SF_19800101120237_121.rma	“	“

- TF case :

Missing source file	result	status of the test
L20_TF_19800101120237_01.rma	program exits properly	PASS
L15_GEO_19800101120000_01.ims	“	“
L15_MaD_19800101120000_01	“	“
DetectorIndex_used	“	“
PSF_LW_045.rma	“	“
C_TF_19800101001737_01.rma	“	“
GTFE_19800101120000_01.rma	“	“
SevB_TF_19800101120237_121.rma	“	“

Date of the test: March 2000

Status of the test: PASS

3.2 InitialiseCSbyInterpolCG

Purpose of the test:

Check the first guess of high resolution correction factors “cS” that consists of interpolated CG (low resolution or Gerb Correction factors)values , assuming perfect GERB geolocation, i.e check function “InitialiseCSbyInterpolCG”.

Methods:

For different cases of the “C_SF” input file :

1. We compare, by image visualization, the global shape of the “cS” first guess image with the input CG image.
2. We calculate by hand the interpolated value at one pixel and compare with the computed value.

Date of the test: November 1999

Status of the test: PASS

3.3 CalculSmoothingFactor

Purpose of the test:

Check function “CalculSmoothingFactor”, that should compute, at each SEVIRI pixel, the smoothing factor of the high resolution correction ratio ”cS”, and should estimate a mean smoothing error .

Methods:

1. We check function outputs for several characteric input "cS" fields.

Input cS field characterization	Output smoothing factor	Status of the test
uniform (cS=1 everywhere)	uniform (=0 everywhere)	PASS
Computed from C_SF_19800101120237_01.rma	= 0 everywhere except at Earth limits	PASS
Computed from C_SF_19900702151737_01.rma	= 0 everywhere except at Earth and rectangles limits	PASS

2. We calculate by hand the smoothing factor value at one pixel (where smoothing factor ><0) and compare with the computed value.

Date of the test: November 1999

Status of the test: PASS

3.4 SeviriCoordOfGerbGeoTable

Purpose of the test:

Check the function "SeviriCoordOfGerbGeoTable" that computes SEVIRI coordinates according to the geolocation of each Gerb pixel (x,y) and returns this float coordinates values in an IP_Images structure (= 3 images, 1 for line index, 1 for column index, and 1 for flag "valid/no_valid" pixel).

Methods:

Visualization of the global shape of the output images for one perfect GERB geolocation case and then one perturbed GERB geolocation case.

Input GERB geolocation table	Output PASS criterion	Status of the test
Gerb geo nb 0 (*)	linear variation of index	PASS
Gerb geo nb 2	small perturbation in index variation	PASS
Gerb geo nb 1	global deviation	PASS

see table with description of artificially simulated GERB geolocation tables in GERB Processing tests document [1]

Date of the test: November 1999

Status of the test: PASS

3.5 CalculGeoconvolConstraintErr

Purpose of the test:

Check function "CalculGeoconvolConstraintErr" which main purpose is to compute, at each GERB pixel (x,y), the "geoconvolution" constraint error E :

E(x,y) = CONVOLUTION [PSF * (cS x SEVIRI_flux)] - L2_Flux(x,y)

Methods:

Step-by-step, we checked :

1. The convolution algorithm :

- INPUT 1: We give as L2_flux input the SEVIRI based flux convolved at GERB resolution, saved from Gerb Processing, and as cS input, an uniform field cS = 1 everywhere.
- OUTPUT 1: We expect E = 0 everywhere.
- TEST 1 STATUS : PASS
- INPUT 2: We give as L2_flux input the SEVIRI based flux convolved at GERB resolution, saved from Gerb Processing, and as correction factor cS input, an uniform field cS = 0.5 everywhere.
- OUTPUT 2: We expect E<0 everywhere.
- PASS/FAIL CRITERION 2: We check, for some pixels, by "manual" calculation, that the computed E value is right.
- TEST 2 STATUS : PASS

2. The exception rules for E calculation :

- if pt(x,y) is geolocated out of the earth or on thedark part of the Earth => E = 0 ; STATUS OF THE TEST = PASS.
- if pt(x,y) is geolocated on the Earth, but the detector(y) is out of work, or the Gerb geolocation(x,y) is wrong, or the L2flux(x,y) is not valid (=ERR_VALUE), or CG(x,y) is out of [c_min,C_max] limits or the convolution fails => E= ERR_VALUE ; STATUS OF THE TEST = PASS.

Date of the test: March 2000

Status of the test: PASS

3.6 CalculLagrangeGradient

Purpose of the test:

Check function "CalculLagrangeGradient" that computes, at each high resolution pixel (i,j), the gradient of the Lagrange function ($\delta\Lambda(i,j)$), relative to the correction factor $cS(i,j)$.

As described in the main RE document [3], the Lagrangian gradient is a function of $Cmean(i,j)$ and $E(x,y)$, representing respectively, the smoothing function of the correction, and the "geo-convolution" constraint function :

$$dL(i,j) = \delta\Lambda(i,j) = \left(\delta_1(Cmean^{(k)}) + \frac{r}{nb}\delta_2(E^{(k)})\right)$$

where

$$\delta_1 = Cmean(i,j) - \frac{1}{8}\sum_{m,n=-1,1}(Cmean(i+m,j+n))$$

$$\delta_2 = 2\sum_{x,y}\left(E(x,y).\widetilde{F}_{HR}(i,j).P^{d=x}(i-xS,j-yS\right)$$

with x,y such as the corresponding xS is included in {i-N/2,i+N/2} and yS is included in {j-N/2,j+N/2}

Methods:

- We check the global shape of the gradient dL for different input cases.
- We calculate "by hand" the dL value at one pixel and compare with the computed value.

Date of the test: March 2000

Status of the test: PASS

3.7 CalculCoefForSmoothingFactor

Purpose of the test:

Check function "CalculCoefForSmoothingFactor", i.e. check the calculation of the coefficients c1 and c2 that allow to estimate the smoothing error factor $\epsilon_S^{(k+1)}$from its value at the last iteration (see section 4.7.2 from doc[3]):

$$\epsilon_S^{(k+1)} = \sqrt{\epsilon_S^{(k)} - 2c_1\,\alpha + c_2\,\alpha^2}$$

Methods:

Check that the smoothing factor $\epsilon_S^{(k+1)}$computed in this way, is equal to the value returned by function "CalculSmoothingFactor".

Date of the test: March 2000

Status of the test: PASS

3.8 Check stop criteria of iterative scheme

Purpose of the test:

Check the stop criteria of the iterative scheme.

More precisely, we check that the iterative process stops if :

1. The smoothing factor $\epsilon_S \leq \epsilon_S^{required}$
2. And the geoconvolution constraint error factor $\epsilon_E \leq \epsilon_E^{required}$

Or if the number of allowed iterations is exceeded (k > 100).

Methods:

We change the value of parameters : "max_it, err_smoothing_required and err_constraint_required" and check when the program is ending.

Date of the test: March 2000.

Status of the test: PASS

3.9 Test iterative process

Purpose of the test:

Check that the iterative scheme works properly :

- check that all the data needed at step k-1 are well computed at step k.
- check the minimization of ϵ_S and ϵ_E, respectively the global smoothing factor and the geoconvolution constraint error factor.

Methods:

The RE is launched for a 100 iterations run.

- We check that no error occurs, that the program stops when the 2 stop criteria are satisfied or after 100 iterations.
- We check, at each iteration, by image visualization, the shape of the following computed fields:

1. dL, the Lagrange gradient
2. cS, the correction factor at high resolution
3. Cmean, the smoothing factor
4. E, the geoconvolution constraint error

- We also check evolution of $\epsilon_S$and ϵ_E, respectively the global smoothing factor and the geo-convolution constraint error factor.

Input:

This test was performed for all the sets of artificial images presented in table 1

Output:

For all the tested sets of artificial images the program exits properly out of the iterative scheme, either the 2 stop criteria are satisfied, or after 100 iterations. Moreover, ϵ_S and ϵ_E decrease with iteration until they reach their respective required value.

We present in annex the evolution of ϵ_S and ϵ_E according to iteration for the reference date : 19950702151737.

Date of the test: March 2000.

Status of the test: PASS

3.10 Error propagation tests

Purpose of the test:

Test the propagation of an error value in the input data through all the processing.

Methods:

We introduce an error value (i.e. error value = -10000) at one (or more) pixel(s) of one input image, and we check its propagation through the process by visualization of each intermediate image.

Pass criterion:

The areas in the intermediate and output images that are influenced by this value, are set to ERR_VALUE (= -10000)

Test table for SF 19950702151737 case :

Acronym description :

Acronym	Description
cS(0)	First guess of correction factor at high resolution
xS,yS	SEVIRI coordinates of GERB pixels
C_mean(0)	Initial smoothing factor
E(0)	Initial geoconvolution constraint Error
dL(k)	Lagrangian gradient at the kth iteration
C_mean(k)	Smoothing factor at the kth iteration
E(k)	Geoconvolution constraint error at the kth iter.
cS(k)	Estimation of correction factor at high resolution at the kth iter.

Name of the file where an error is introduced	Acronym name of computed, checked images	Expected contamination	Observed contamination	Status of the test
L20_SF	cS(0)	none		PASS
with 1 error value	xS,yS	none		“
at pixel (100,100)	E(0)	E(100,100)= Error Value	E(100,100)= Error Value	“
	dL(k)	none		“
	C_mean(k)	none		“
	E(k)	E(100,100)=EV	E(100,100)=EV	“
	cS(k)	none		“
C_SF	cS(0)	none		PASS
with 1 error value	xS,yS	none		“
at pixel (100,100)	E(0)	E(100,100)=EV	E(100,100)=EV	“
	dL(k)	none		“
	C_mean(k)	none		“
	E(k)	E(100,100)=EV	E(100,100)=EV	“
		no calculated		
	cS(k)	none		“
L20_GEO	cS(0)	none		PASS
with 1 error value	xS,yS	error at pt (100,100)	error at pt.(100,100)	“
at pixel (100,100)	E(0)	E(100,100)=EV	E(100,100)=EV	“
		no calculated		
	dL(k)	none		“
	C_mean(k)	none		“
	E(k)	E(100,100)=0	E(100,100)=0	“
		no calculated		
	cS(k)	none		“
L20_SF	cS(0)	none		PASS
with 50*50 error values	xS,yS	none		“
around pixel (100,100)	E(0)	E around pt(100,100)	E=EV around pt(100,100)	
		no calculated		PASS
	dL(k)	none		“
	C_mean(k)	none		“
	E(k)	E(100,100)=EV	E(100,100)=EV	
		no calculated		PASS
	cS(k)	no updated around pt(100,100)		“
C_SF	cS(0)	cS in GERB pt(100,100) area set to1		PASS
with 50*50 error values	xS,yS	none		“
around pixel (100,100)	E(0)	E around pt(100,100)	E=EV around pt(100,100)	
		no calculated		PASS
	dL(k)	none		“
	C_mean(k)	none		“
	E(k)	E around pt(100,100)	E=EV around pt(100,100)	
		no calculated		PASS
	cS(k)	no updated around pt(100,100)		“

3.11 Test of output files storage on disk

Purpose of the test:

Test final files storage on disk.

Methods:

We check, after program running, that all the output images are saved on disk and can be re-read with the "viewrma" application.

Test table:

Input file	Expected output data -> Name of the saved file	Status
L20_SF_19901002151737_01.rma	Correction factor for SF -> cS_SF_19901002151737_01.rma	PASS
L20_TF_19901002151737_01.rma	Correction factor for TF -> cS_TF_19901002151737_01.rma	PASS

Status of the test : PASS

4 Improvement tests on images derived from Meteosat 7

In order to test RE algorithm performance, the program needs to be tested on more realistic images. Since neither real SEVIRI data, nor real GERB data exist already, we simulated them from Meteosat 7 images.

4.1 Generation of testing images

The initial sources files are Meteosat images 833*853, dated the 11th of January 2000. These files are saved on directory /gerbdata/nic/FluxesMeteosat/LastA/:

- rad_sol_1300.rma.gz
- flux_sol_1300.rma.gz
- adm_1300.rma.gz

From this reference or ideal solar configuration, we generate GERB and SEVIRI images. GERB image of size 64x64 are computed by down sampling on 41x41 PSF area. SEVIRI images (of size 833*853) are generated by applying small perturbation factors to the reference Meteosat images. These perturbations introduce small difference between GERB and SEVIRI data, and are aiming to simulate the calibration differences between both instruments and hence, spectral modelling errors. Three types of perturbation were generated :

- For "im 0", SEVIRI images are perturbed by a constant factor according to the scene identification.
- For "im 1", SEVIRI images are perturbed by a factor that varies linearly with the radiance value.
- For "im 2", SEVIRI images are perturbed by a factor that is a sinusoidal function of the radiance value.

	reference date	ref.number	SEVIRI image 833*853 with perturbation	GERB image 64*64 with PSF 41*41
im 0	19900101130237	13	according to ADM	“
im 1	19900111130237	13_1	= linear function of radiance	“
im 2	19900121130237	13_2	=sinusoidal function of radiance	“

Remark : These 3 sets of test images are stored in directory /raindata/meteosat/.

4.2 Test methods of α selection

Four methods to select the α factor were implemented in the RE program (see description of RE implementation on real problem ([3] section 4.7)) :

1. Method of constant step : $\alpha = \frac{0.015}{maxcorr}$.
2. Method of the N regularly spaced values in the range [0,α_{MAX}], with $\alpha^{(k)}_{MAX} = \frac{0.05}{maxcorr(k)}$.
3. Golden Section Search.
4. Recursive bracketing method, starting from the same interval [0,α_{MAX}].

Tests on the 3 sets of images lead us to choose the fourth method. Computing time of runs with set of images “im 0” are shown in the next table [1]. :

test 0			
$\epsilon_E(0)$	17.01	$\epsilon_E^{required}$	3.31
α options	nb.iter	time	$\epsilon_E(end)$
1	48	7m19s	3.29
2	45	6m53s	2.81
3	45	7m	2.95
4	25	4m26s	3.21

4.3 Test of influence of correction area

Up to now, corrections were applied on the full image, all the correction factors were updated at each iteration. Aiming to decrease computing time, we tried to reduce this correction area.

Several ways of applying the corrections were tested (see next table):

- 4 a : correction on all the cS.
- 4 b : correction is applied only on the cS located in the PSF area (-PSF/2,+PSF/2) of a bad E(x,y) point
- 4 c : correction is applied only on the cS located in the area (-PSF,+PSF)of a bad E(x,y) point.
- 4d : correction is applied only on the cS located in the area (-3/2PSF,+3/2PSF)of a bad E(x,y) point.
- 4bb : the correction is applied only on the cS located in the PSF area (-PSF/2,+PSF/2) of a bad E(x,y) point and max_corr is evaluated only among the points where a correction was done at the last iteration.

[1]For test computed on a PII 450 Mhz under Linux with gcc as compiler.

Some results are presented in the following table :

	test 0			test 1		test 2	
$\epsilon_E(0)$	17.01			23.39		16.4	
$\epsilon_E^{required}$	3.31			3.33		3.303	
α option	nb.iter	time	(-03)*	nb.iter	time	nb.iter	time
4 a	25	7m49	4m26s	>50(12.6)	16m12s	14	4m37
4 b	41	13m36s	7m43s				
4c	30	9m52s	5m36s				
4d	25	8m32s	4m33s	>50(12.6)	16m23	16	5m24
4bb	38	13m3s	7m24s	>50 (12.6)	16m11s	>50(3.37)	17m7s
4d_50	30		2m26s				

- (-03) means that an option of compilation optimisation (of level 3) has been used

These tests lead us to chose a combination between methods 4a and 4d. We denote this new method : 4d_50. During the first iterations, corrections are applied at all the SEVIRI pixel (i,j). Then, when the number of points where the correlation criterion is not satisfied is inferior to a specified value (50), corrections are applied only at the pixels (i,j) located in the neighborhood (-3/2PSF,+3/2PSF)of a bad E(x,y) point (i.e. where $\epsilon_E(x,y) < \epsilon_E^{required}$). In this way, during the last iterations, we save a big part of computing time.

5 Tests on images derived from Meteosat 7 , rescaled.

To go closer to the real configuration, further tests are executed at real scale. The reference flux and radiance images are scaled up to the operational high resolution sizes (i.e. 1233x1233 pixels). Then, GERB images of size 256x256 are computed by down sampling of 15x15 PSF area and SEVIRI images are derived as in the preceding tests.

We apply here 4 kinds of perturbation on SEVIRI images :

- For "im 0", SEVIRI images are perturbed by a constant factor according to scene identification.
- For "im 1", SEVIRI images are perturbed by a factor that varies linearly with the radiance value.
- For "im 2", SEVIRI images are perturbed by a factor that is a sinusoidal function of the radiance value.
- For "im 3", SEVIRI images are perturbed by a factor that is another sinusoidal function (higher frequency) of the radiance value.

These new sets of testing images are also stored in directory /raindata/meteosat/.

5.1 Tests on r factor

Several slightly different values of r, the weight factor between the 2 terms of the Generalized Lagrange function, were tested. All these tested values are issued from the idea that the factor r should roughly verify the relation (see Description of implementation on real problem[3]) :

$$r = \frac{\delta_1}{\delta_2} \frac{\epsilon_E^{rel}}{\epsilon_S^{rel}}$$

For instance, we tested , $r = \frac{\frac{\epsilon_E}{\epsilon_E^{required}}}{\frac{\epsilon_S}{\epsilon_S^{required}}} \frac{\|\delta_1\|^2}{|\delta_2|}$,$r = \frac{\frac{\epsilon_E}{\epsilon_E^{required}}}{\frac{\epsilon_S}{\epsilon_S^{required}}} \frac{\|\delta_1\|^2}{|\delta_2|_{atmaxE}}$,...
The relation which leads the to the fastest convergence, is :

$$r = \frac{\frac{\overline{|E*|}}{\epsilon_E^{required}}}{\frac{\epsilon_S}{\epsilon_S^{required}}} \frac{\|\delta_1\|^2}{\overline{|\delta_2|}}$$

where

- $\|\delta_1\|^2 = \sqrt{\frac{\sum_{i,j}(\delta_1(i,j)^2)}{nb(\delta_1)}}$
- $\overline{|\delta_2|} = \frac{\sum_{i,j}|\delta_2|}{nb(\delta_2)}$
- $\overline{|E^*|} = \frac{\sum_{x,y}|E^*|}{nb(E^*)}$ with x, y such as $|E(x,y)| > \epsilon_E^{required}$

5.2 Tests algorithm with conjugated gradient.

For all the test sets of images, the algorithm based on the conjugated gradient runs slower than the one based on the simple gradient. In most of the cases the conjugated gradient method requires more iterations. And in the few test cases where convergence is reached after less iterations with a combination of gradients than with a simple gradient, the computing time increase is higher than the time savings with less iteration. Indeed, the conjugated gradient calculation requires a bit more storage and a little bit more computing time at each iteration, than simple gradient calculation.

Considering these disappointing results, we decided not to retain the conjugated gradient, and then to use the simple gradient caculation.

options	nb.iter	time	$\epsilon_E(0)$	$\epsilon_E(end)$	$\epsilon_S(0)$	$\epsilon_S(end)$
im 0 :						
dL simple	16	5m14s	39.7	4.37	0.0006	0.0007
dL conjugated	21	6m27s	39.7	4.4	0.0006	0.0008
im 1 :						
dL simple	20	5m53s	44.9	4.40	0.0010	0.0009
dL conjugated	37	10m45s	44.9	3.36	0.0010	0.0006

5.3 Tests on images perturbed to 10% , 20% or 40%.

The following table compares RE performance between 3 ranges of perturbation :

- images perturbed in a range of 10% (i.e. cS={ 0.9 ; 1.1 }),
- images perturbed in a range of 20% (i.e. cS={ 0.8 ; 1. 2 })
- and images perturbed in a range of 40% (i.e. cS={ 0.6 ; 1.4 }).

These 3 perturbation ranges are applied on the 4 sets of perturbed images “im 0”,“im 1”,“im 2”,“im 3”.

	im 0			im 1		
	$\epsilon_E(0)$	nb.iter	time	$\epsilon_E(0)$	nb.iter	time
c={ 0.8 ; 1. 2 }	34	26	5m47s	44,9	26	5m29s
c={ 0.9 ; 1. 1 }	9.5	2	48s	23	6	1m52s
c={ 0.6 ; 1. 4 }	87.5	>100	34m15s	85.8	>100	20m37s
		(ϵ_E=7.2	ϵ_s=0.0014)		(ϵ_E=5.56	ϵ_s=0.0016)

	im 2			im3		
	$\epsilon_E(0)$	nb.iter	time	$\epsilon_E(0)$	nb.iter	time
c={ 0.9 ; 1. 1 }	37.5	19	3m16s	32.4	29	6m
c={ 0.8 ; 1. 2 }	74	>100	20m15s	65.4	>100	34m48s
		(ϵ_E=5.37	ϵ_s=0.00104)		(ϵ_E=8.72	ϵ_s=0)
c={ 0.6 ; 1. 4 }				138	>100	34m58s
					(ϵ_E=23.1	ϵ_s=0.0021)

At this satge of the development, the RE iterative process is too long (mor than 100 iterations) if images are perturbed in a range exceeded 20 %.

5.4 Test $\epsilon_s^{required}$ influence

In order to test the influence of the smoothing error requirement ($\epsilon_s^{required}$) on the convergence speed, the RE program was launched on the same set of images, with several values of $\epsilon_s^{required}$. In the following results table, we can see that larger smoothing requirements (i.e. $\epsilon_s^{required}$=0.002 or 0.003 instead of 0.001) allow us to reach faster the constraint stop criterion.

im	$\epsilon_s^{required}$	nb.iter	time	$\epsilon_E(0)$	$\epsilon_E(end)$	$\epsilon_s(0)$	$\epsilon_s(end)$
im 1	0.001	>100	20m37s	85.8	5.56	0.0022	0.0016
c={ 0.6 ; 1. 4 }	0.002	35	8m44s	85.8	4.38	0.0022	0.0018
	0.001++(15)	45	11m12s	85.8	4.41	0.0022	0.0018
	0.001++(10)	90	12m30s	85.8	4.39	0.0022	0.0019
im2	0.001	>100	20m15s	74	5.37	0.0013	0.00104
c={ 0.8 ; 1. 2 }	0.002	>100	15m52s	74	4.45	0.0013	0.0012
	0.003	39	13m16s	74	4.38	0.0013	0.0014
	0.001++(15)	35	11m30s	74	4.37	0.0013	0.0013
	0.001++(10)	34	11m03s	74	4.38	0.0013	0.0013
im 3	0.001	>100	34m48s	65.5	8.72	0.0015	0.0021
c={ 0.8 ; 1. 2 }	0.001++(15)	91	28m26s	65.5	4.40	0.0015	0.002
	0.001++(10)	77	23m30s	65.5	4.39	0.0015	0.0023
c={ 0.6 ; 1. 4 }	0.001	>100	34m58s	138	23.1	0.003	0.0021
	0.001++(15)	>100	33m17s	138	22	0.003	0.0042
	0.001++(10)	>100	34m5s	138	13.53	0.003	0.0048

To better understand these results, it is relevant to remember the link between $\epsilon_s^{required}$ and the factor r (i.e. the balance factor between the 2 terms of the gradient) :

$$r = \frac{\frac{\overline{|E^*|}}{\epsilon_E^{required}}}{\frac{\epsilon_S}{\epsilon_S^{required}}} \frac{\|\delta_1\|^2}{\overline{|\delta_2|}} \Rightarrow r \div \epsilon_S^{required}$$

When increasing the smoothing error requirement $\epsilon_s^{required}$, the r factor increase .

We know the choice of a well fitted balance factor value (r) is of major importance for the convergence of our iterative process. On one hand, if r is chosen too big the problem becomes ill conditionned and is numericaly unstable. On the other hand, r must be chosen big enough to reach an optimal solution (i.e. to reach low enough value of E).

From these considerations, it seems that the best way to avoid numerical difficulties, and to get a result as accurate as possible, with a reasonable computing time, is to start with a small $\epsilon_s^{required}$ value and then to adapt this value as iteration number goes up.

In the test case, here above mentionned, "0.001++(10)" (respectively "0.001++(15)"), the program starts with an smoothing error requirement value of 0.001, then this value is incremented by 0.001 each 10 (respectively 15) iterations.

This periodic incrementation allow us to reach the constraint requirement, with an acceptable computing time, and without too high smoothing factor increase, for nearly all the tested cases.

Only the last case," im 3 " perturbed to 40% can not be tamed.

5.5 Summary of our optimal choice of parameters and options

For all the tested data, the optimal combination of parameters and options seems to be :

Parameter value in the RE program	Option description
alpha_option = 4	Method of α selection = Recursive bracketing method
limit_nb_badE = 50	Correction area = full image during the first iteration, then only bad E PSF areas.
alpha_max_init = 0.03	α is chosen in the interval $\left\{0\,,\ \alpha_{max} = \frac{0.03}{max(corr)}\right\}$
r	$r = \frac{\frac{\overline{\lvert E^* \rvert}}{\epsilon_E^{required}}}{\frac{\epsilon_S}{\epsilon_S^{required}}} \frac{\lVert\delta_1\rVert^2}{\overline{\lvert\delta_2\rvert}}$
err_smoothing_required = 0.001 nb_iter_beforchange = 15	Start with $\epsilon_s^{required}$= 0.001 ,then, at 15 iteration interval, increase $\epsilon_s^{required}$ by 0.001
option_combined_dL = 0	Use simple gradient
NB_maxE 5	The number of point (x,y) selected to compute $\epsilon_E^*(\alpha) = 5$

5.6 Required RAM and disk space.

For the optimized combination of parameters and options, the required memory is : 27 M. And the disk space needed to store output data, i.e. the Correction ration at high resolution, is 6081 Kb.

5.7 Examples of input and output testing images.

Image 1 : Meteosat solar flux image at SEVIRI resolution before perturbation (flux_sol_13_12.rma)

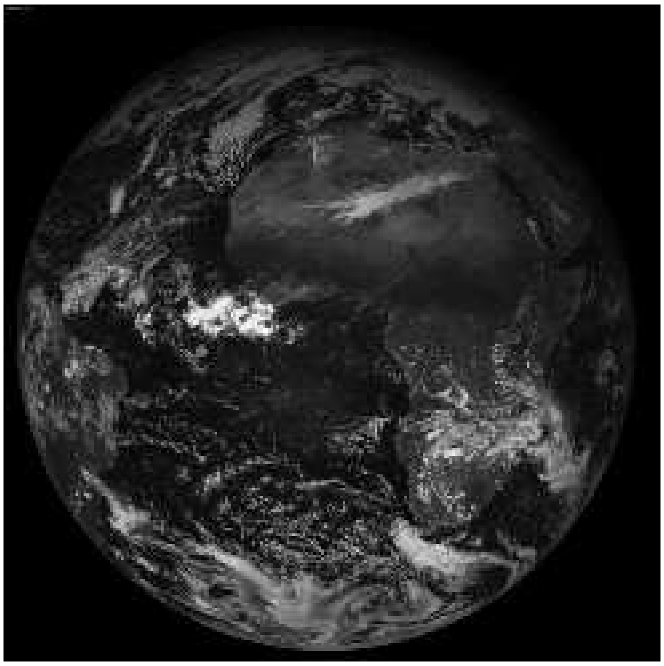


Image 2 : Derived GERB solar flux image (gerb_SF_13_256.rma)

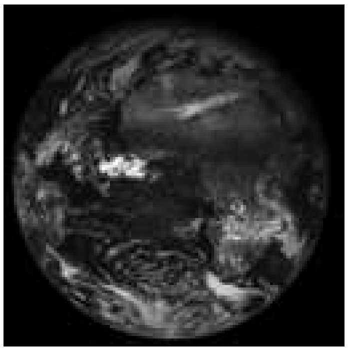


Image 3 : Perturbation factor constant according to scene identification (cS13_12_01.rma)

Image 4 : Resulting perturbed SEVIRI solar flux image (flux_sol_pertu_13_12_01.rma)

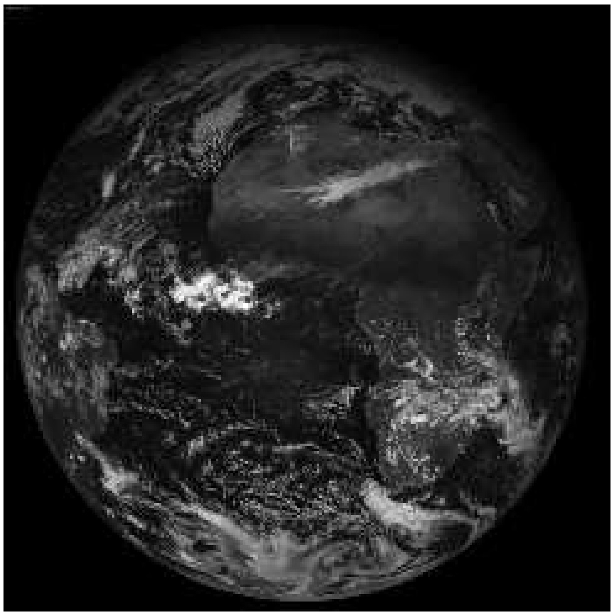


Image 5 : Perturbation factor that varies linearly with the radiance value (cS13_12_1.rma)

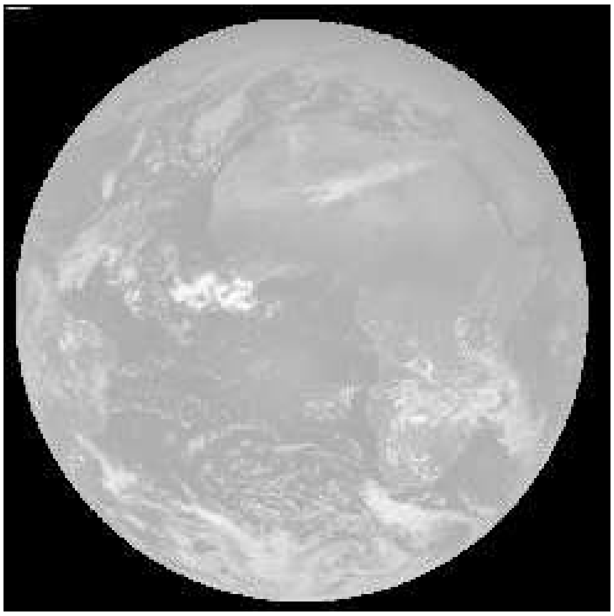


Image 6 : Resulting perturbed SEVIRI solar flux image (flux_sol_pertu_13_12_1.rma)

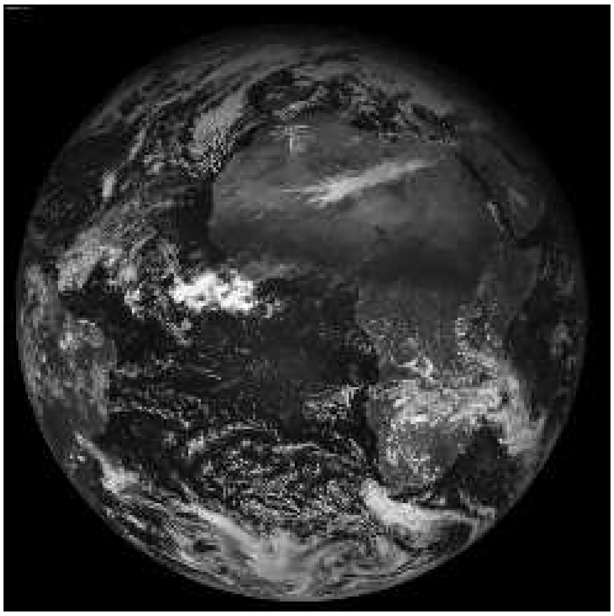


Image 7 : Perturbation factor that varies as a sinusoidal function of the radiance value (cS13_12_2.rma)

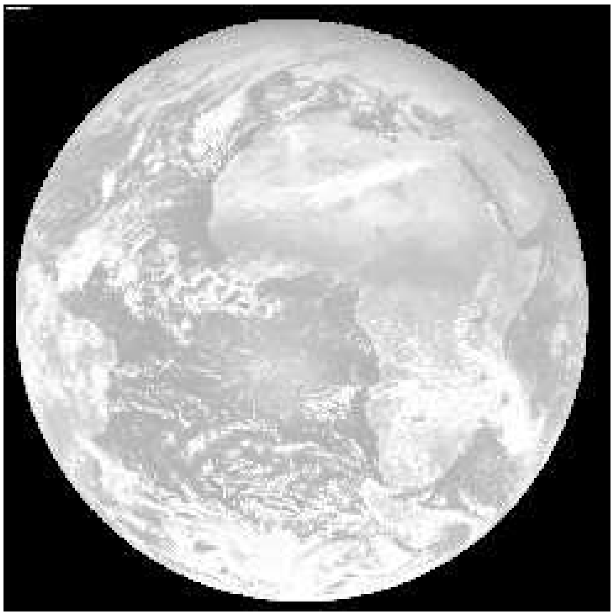
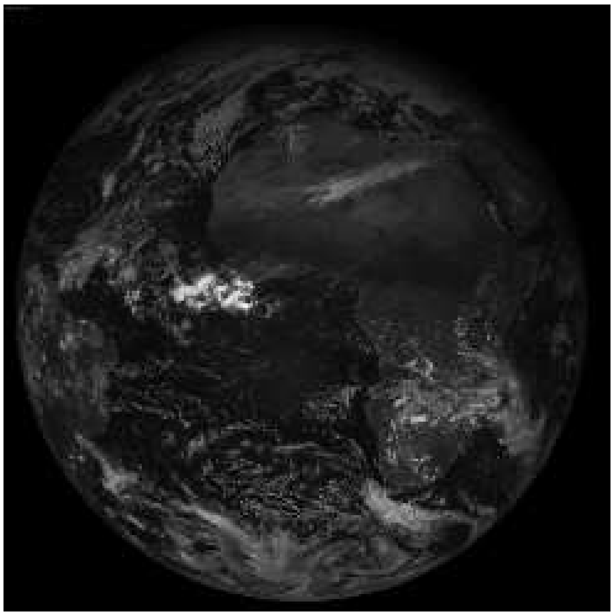


Image 8 : Resulting perturbed SEVIRI solar flux image (flux_sol_pertu_13_12_2.rma)



References

[1] A. Hermans:"Gerb Processing Annex A : tests description", RMIB technical notes, MSG-RMIB-GE-TN-14,2000.

[2] L. Gonzalez, A. Hermans:"Gerb Processing", RMIB technical notes, MSG-RMIB-GE-TN-04,2000.

[3] S. Dewitte, L. Gonzalez, A. Hermans:"Resolution Enhancement", RMIB technical notes, MSG-RMIB-GE-TN-03,2000.

[4] A. Hermans, L Gonzalez:"Resolution Enhancement Annex A : Program structure description", RMIB technical notes, MSG-RMIB-GE-TN-16,2000.

Table 1: Reference (GERB SW) dates of artificial data sets

Reference date	scene description	GERB	geoloc	PSF	A
		SW	TW		
19800101001737	uniform sphere	0	0	15*15	A_0
19800101031737	“	0	0	3*3	A_0
19800101120237	“	0	0	15*15	A_1
19800702151737	“	7	2	15*15	A_0
19900101150237	10 static rectangles	0	0	15*15	A_1
19900702031737	“	7	2	3*3	A_1
19900702151737	“	7	2	15*15	A_1
19901002151737	“	1	2	15*15	A_1
	10 static + 10 moving rectangles				
19950101031737	“	0	0	3*3	A_0
19950101151819	“	0	0	15*15	A_1
19950702031737	“	7	2	15*15	A_1
19950702151737	“	7	2	15*15	A_1

Figure 1: Scheme of the RE algorithm

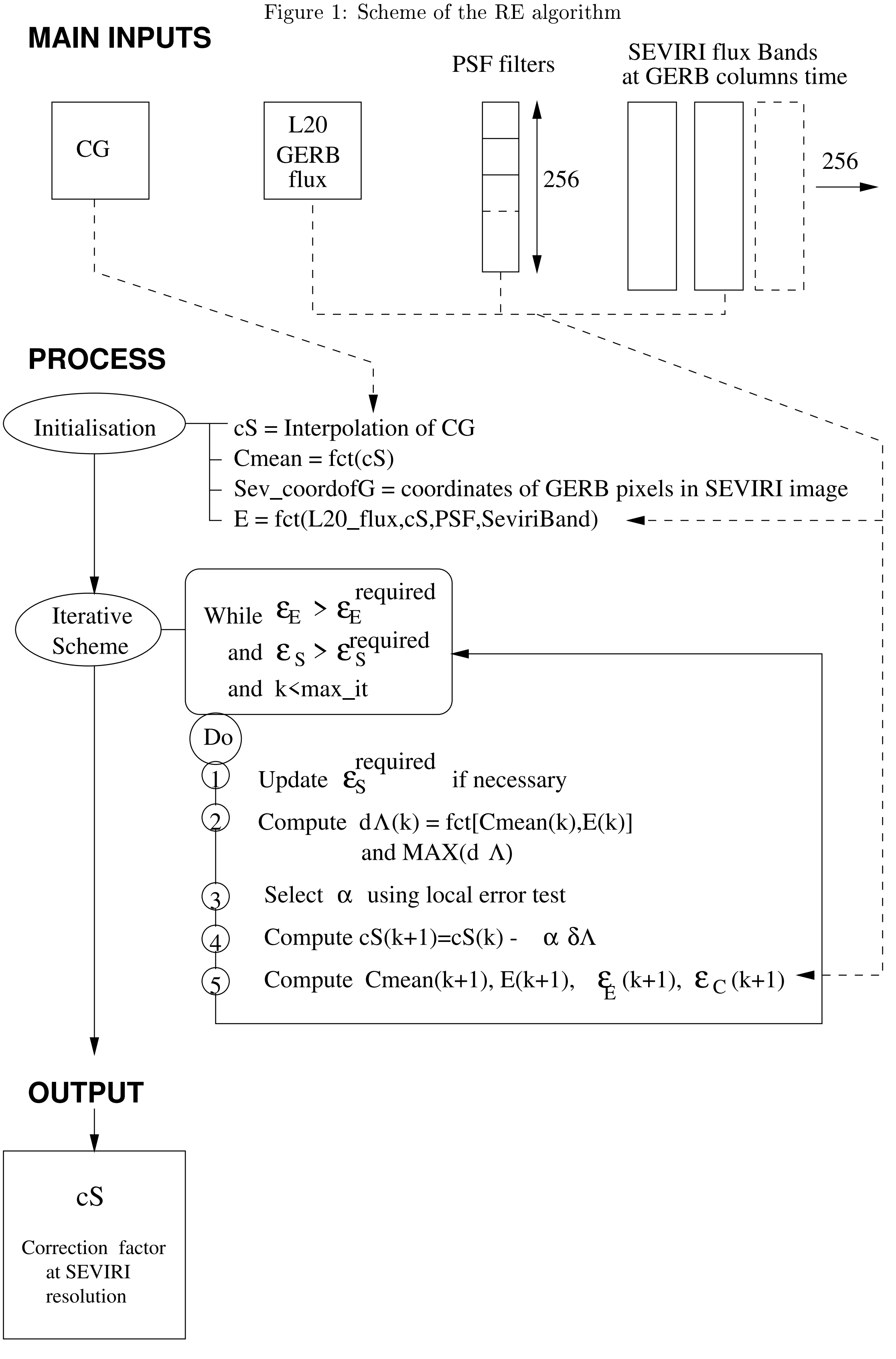
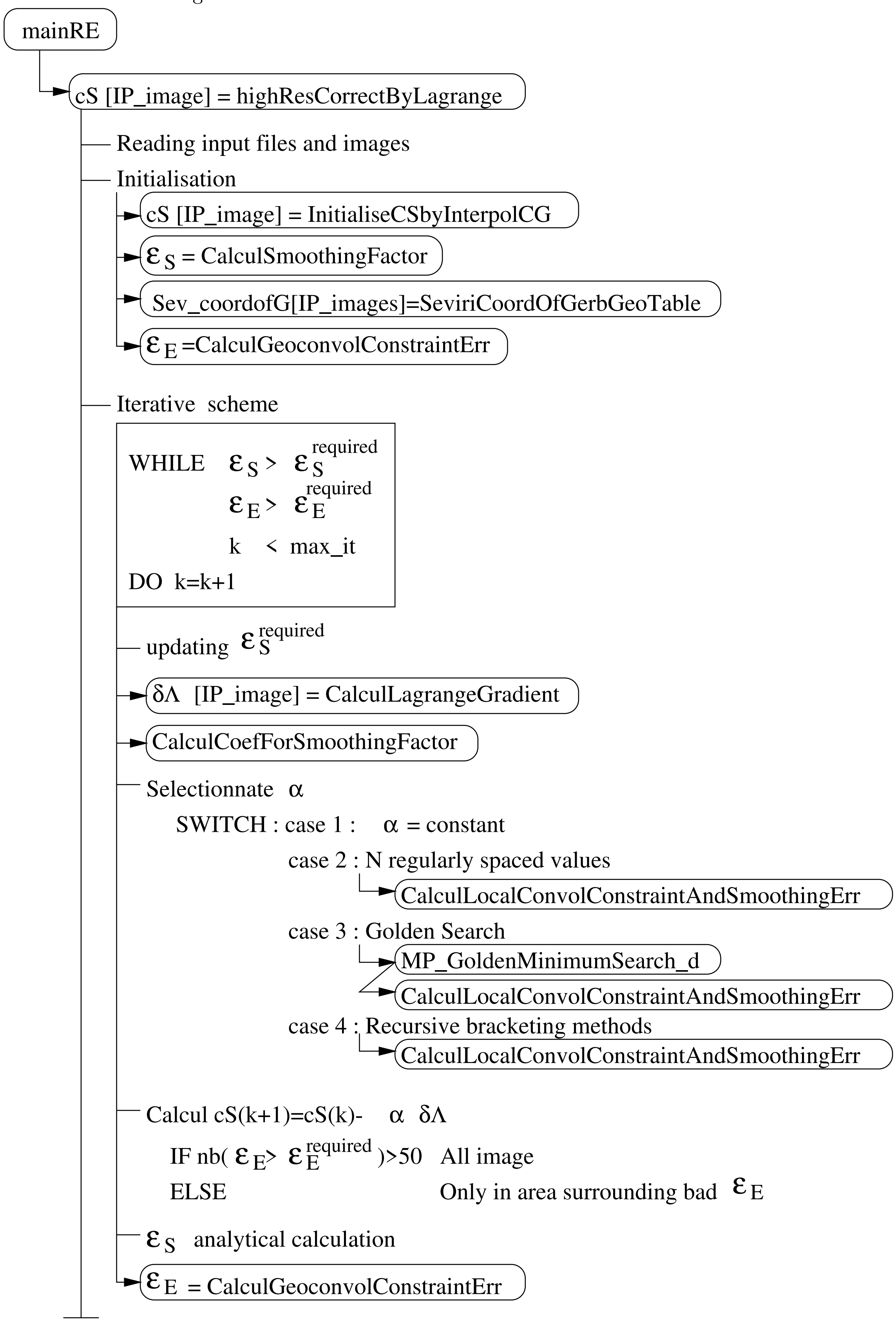
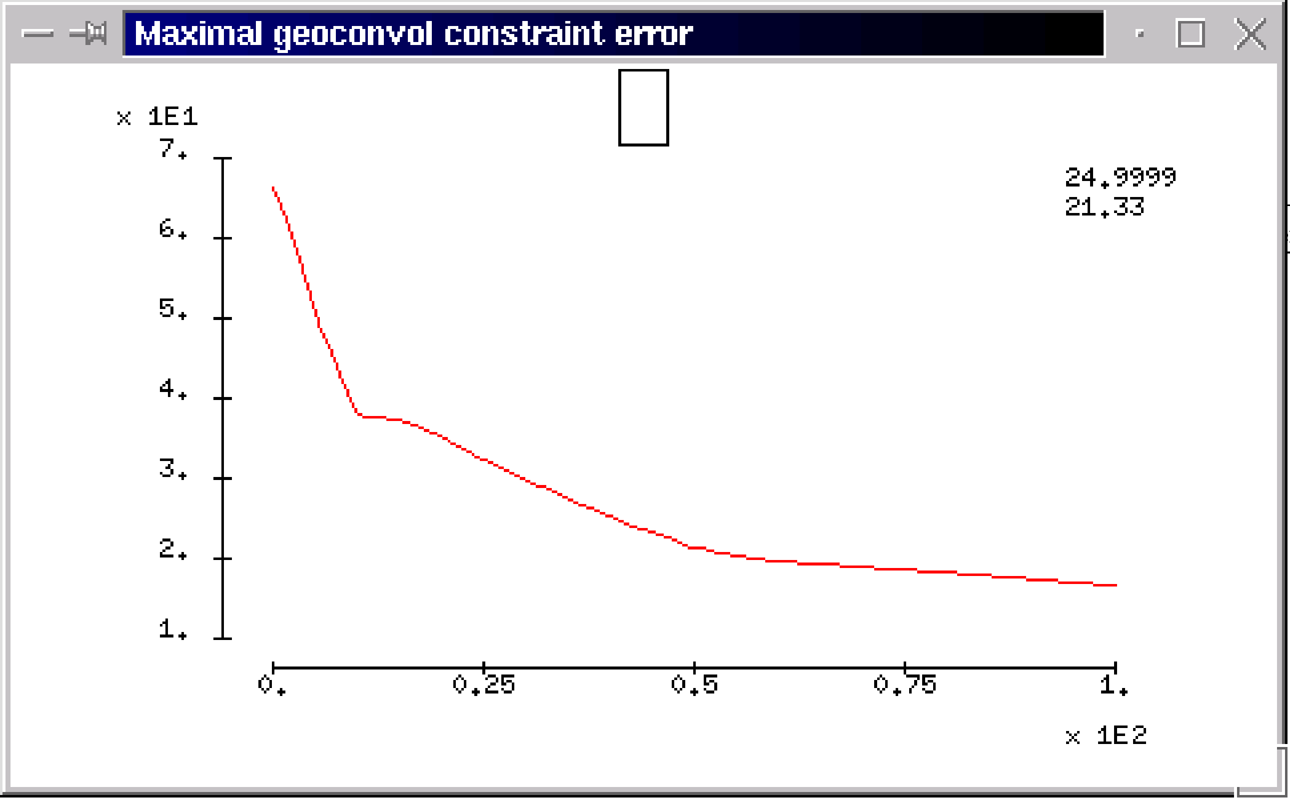
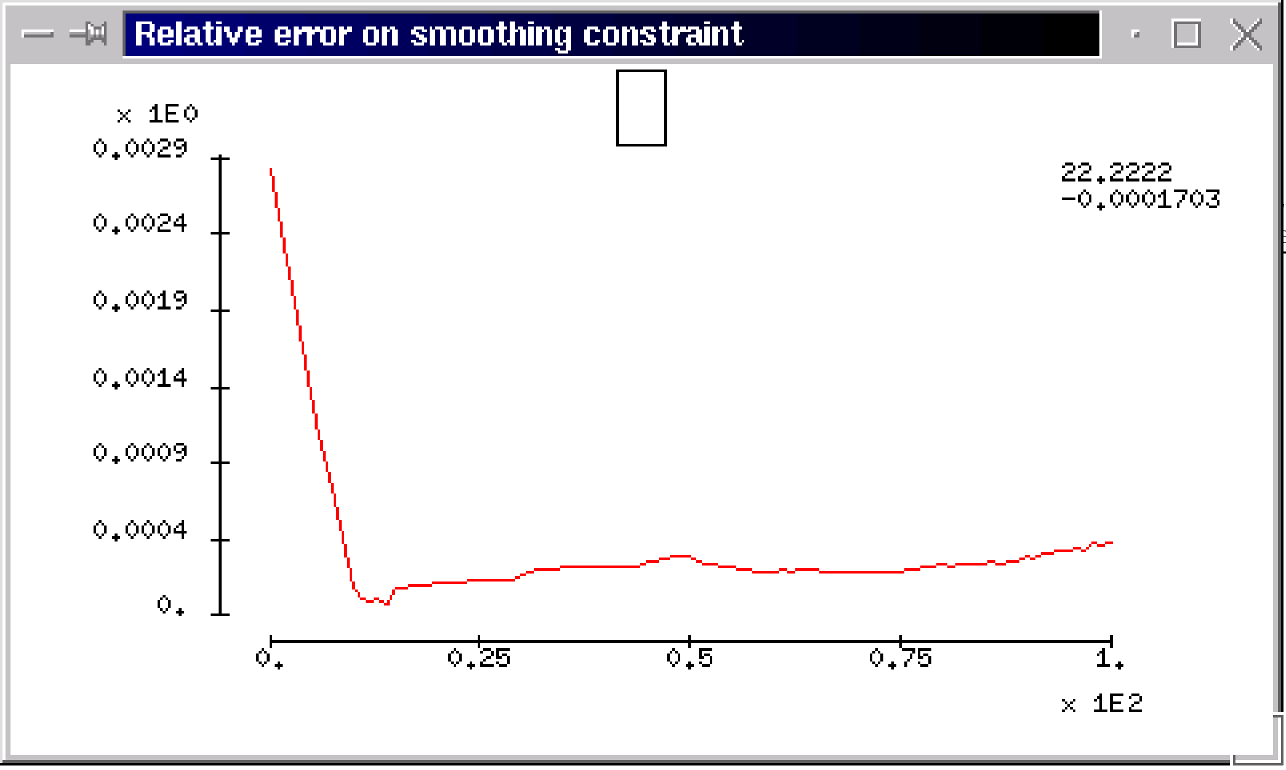


Figure 2: Scheme of the main RE function and subfunction




Maximal geoconvol constraint error
x 1E1
7.
6.
5.
4.
3.
2.
1.
24.9999
21.33
0.
0.25
0.5
0.75
1.
x 1E2


Relative error on smoothing constraint
x 1E0
0.0029
0.0024
0.0019
0.0014
0.0009
0.0004
0.
22.2222
-0.0001703
0.
0.25
0.5
0.75
1.
x 1E2