



Laboratoire de Physique des Plasmas

www.lpp.polytechnique.fr



Développement de logiciels scientifiques

www.scientidev.fr

The ROPROC COMMANDS

A set of procedures
for spatial time serie data processing

by

Patrick ROBERT

Version 5.2

LPP-ScientiDev, December 2021

Contents

I	THE ROPROC COMMANDS	13
1	OVERVIEW OF RPC COMMANDS	15
1.1	Birth of Roproc	15
1.2	Function of RPC commands	16
1.3	A few words on RFF format	16
1.4	Generic RPC commands	17
1.4.1	Software information	17
1.4.2	RFF file handling	17
1.4.3	Data processing	19
1.4.4	RFF files visualization	22
1.4.5	Dates and calendar operations	23
1.4.6	PostScripts conversion	25
1.4.7	System and directories	25
1.4.8	File conversion to RFF	31
1.4.9	Help of commands	31
1.5	RPC_menu	32
2	SOFTWARE INSTALLATION	35
2.1	Download the RPC package	35
2.2	Installation on the target machine	35
2.2.1	Compatibility of executables	36
2.2.2	Update the Makefile	36
2.2.3	Source compilation	37
2.2.4	Setting the environment variables	37
2.2.5	Automatic execution of the RPC_config.bash	38
2.2.6	Make_bin_pack.bash	38
2.2.7	Package tree	39
2.2.8	Installation tests	40
3	DATA MANAGEMENT	41
3.1	Main types of RFF files	41
3.1.1	Example of VecTime file names :	41
3.1.2	Example of Spectrograms file names :	41
3.2	Convert time serie data to RFF	41
3.2.1	RPC_flat_to_rff command	41
3.2.2	Check the validity of the new file	42
3.3	Choose RFF file names	42
3.3.1	Example of GEOS RFF file names	42
3.3.2	Example of CLUSTER RFF file names	43
3.3.3	Choose your RFF file names	43
3.4	Setting up a local database	43
3.5	Interest of a local database	44
3.6	Commands relating to the database	44
3.6.1	Put files into the database	44

3.6.2	Extract files from the database	44
3.6.3	Files list for a given database	44
II	APPLICATION TO SIMULATED DATA	45
4	MAKE SIMULATED DATA	47
4.1	Create ULF simulated data	47
4.1.1	RPC_create_simulated_data ULF	47
4.1.2	Spectrogram plot of simulated ULF waveform	47
4.2	Create MAG simulated data	47
4.2.1	RPC_create_simulated_data MAG	47
4.2.2	Waveform plot of MAG data	47
4.3	Computation of Curl(B) on simulated data	49
4.3.1	Compute MAG and POS simulated data for a current tube	49
4.3.2	Computation of Curl(B) and div(B) on simulated data	49
4.3.3	Visualization of simulated data	49
4.3.3.1	Result with a tetrahedron of large size	50
4.3.3.2	Result with a tetrahedron of small size	50
4.4	Create trajectory simulated data	51
4.4.1	Trajectory plot	52
4.5	Using generic RPC commands	52
5	USING SCRIPTS FOR EFFICIENT PROCESSING	53
5.1	do_simulated_data_processing.sh	53
5.2	do_simulated_data_MVA_plot.sh	53
5.3	do_simulated_data_Polar_plot.sh	55
5.3.1	Produce simulated data in GSE	55
5.3.2	Compute spectrogram in MFA system	55
5.3.3	Compute and plot polarisation parameters	55
5.3.4	Interpretation of polarization parameters	56
5.4	do_simulated_orbit.sh	57
5.4.1	Produce simulated trajectory	57
5.4.2	Compute orbital parameters	57
5.4.3	Compute new theoretical trajectory	57
6	TESTS	59
6.1	Test compute spacecraft trajectory	59
6.1.1	RPC_compute_sat_trajectory	61
6.1.2	do_simulated_orbit_CLUPOS.sh	61
6.1.3	do_simulated_orbit_GEOPOS.sh	62
6.1.4	run_test_compute_curl_div_4sat.sh	62
6.2	Test generic scripts	63
6.3	Test RPC_dir	65
6.3.1	dir_pretty_tree	67
6.3.2	dir_size_pretty_tree	67
6.3.3	dir_propereties_pretty_tree	68
III	APPLICATION TO GEOS DATA	69
7	ACCESS TO GEOS DATA	71
7.1	Downloads RFF data files	71
7.2	Setting up a local database	71
7.3	Interest of a local database	72

7.4	Commands relating to the database	72
8	WORKING ON GEOS RFF FILES	73
8.1	Main types of RFF files	73
8.1.1	Example of VecTime file names :	73
8.1.2	Example of Spectrograms file names :	73
8.1.3	Checking the validity of an RFF file	73
8.2	Special commands for GEOS	74
8.3	Make an ULF spectrogram	75
8.3.1	From a VTL1 file	75
8.3.2	From a VTL2 file	75
8.3.3	Spectrogram visualization	76
8.4	Make a spectrogram of the Magnetometer	76
8.5	Visualize an average spectrum	76
8.6	Draw a waveform	77
8.7	Extract part of a VecTime file	77
8.8	Change coordinate system	77
8.9	Waves polarisation computation	78
8.9.1	Transformation in MFAV coordinate system	78
8.9.2	Calculation and plot of the polarization parameters	78
8.10	Data position visualization	78
8.10.1	2D-Visualization	78
8.10.2	3D-visualization	79
8.10.3	3D-visualization in GSE	79
8.10.4	Transform positions in another system	79
8.11	Compute orbital parameters	79
8.12	Compute an elliptic trajectory	79
9	USING GEOS SCRIPTS	81
9.1	Draw a waveform	81
9.1.1	do_VT_plot_GEOULF.sh	81
9.2	Make a spectrogramme	82
9.2.1	do_SP_plot_GEOULF.sh	82
9.2.2	do_SP_plot_GEOMAG.sh	83
9.3	Plotting satellite trajectory	83
9.3.1	do_2D_plot_GEOPOS.sh	83
9.3.2	do_3D_plot_GEOPOS.sh	83
9.4	Calculate and visualize waves polarization	86
9.4.1	do_Polar_plot_GEOULF.sh	86
9.4.2	Interpretation of polarization parameters	87
9.4.3	Two other examples of wave polarization	88
9.4.3.1	Low frequency linear polarization	88
9.4.3.2	Hight frequency linear polarization	90
9.5	ULF and Magnetometer comparison	92
9.5.1	do_compare_ULF_MAG_GEOS.sh	92
9.6	Calculate the orbital parameters of a satellite	93
9.6.1	do_simulated_orbit_CLUPOS.sh	93
10	TESTS OF GEOS SCRIPTS	95
10.1	test_do_VT_plot_GEOS.sh	95
10.2	test_do_SP_plot_GEOS.sh	95
10.3	test_do_POS_3D_plot_GEOS.sh	95
10.4	test_do_Polar_plot_GEOS.sh	95
10.5	test_do_ULF_MAG_plot_GEOS.sh	95
10.6	test_do_join_vectime.sh	96

10.7 Test of GEOS scripts	96
10.8 run_test_demo.sh	97
IV APPLICATION TO GEOS GROUND DATA	103
11 ACCESS TO GEOS GROUND DATA	105
11.1 Introduction	105
11.2 Downloads RFF data files	105
11.3 Setting up a local database	105
11.4 Interest of a local database	106
11.5 Commands relating to the database	106
12 WORKING ON GEOS GROUND RFF FILES	107
12.1 Main types of RFF files	107
12.1.1 Example of VecTime file names :	107
12.1.2 Example of Spectrograms file names :	107
12.1.3 Checking the validity of an RFF file	107
12.2 Special commands for GEOGRD	108
12.3 Make an ULF spectrogram	109
12.3.1 From a VTL1 file	109
12.3.2 From a VTL2 file	109
12.3.3 Spectrogram visualization	109
12.4 Visualize an average spectrum	110
12.5 Draw a waveform	110
12.6 Extract part of a VecTime file	110
12.7 Waves polarisation computation	111
12.7.1 Transformation in MFAV coordinate system	111
12.7.2 Calculation and plot of the polarization parameters	111
13 USING GEOGRD SCRIPTS	113
13.1 Draw a waveform	113
13.1.1 do_VT_plot_GEOGRD.sh	113
13.2 Make a spectrogramme	114
13.2.1 do_SP_plot_from_VTL1_GEOGRD.sh	114
13.2.2 do_SP_plot_from_VTL2_GEOGRD.sh	115
13.3 Calculate and visualize waves polarization	116
13.3.1 do_Polar_plot_GEOGRD.sh	116
13.3.2 Interpretation of polarization parameters	117
13.3.3 Other examples of wave polarization	118
14 TESTS OF GEOGRD SCRIPTS	119
14.1 test_do_VT_plot_GEOGRD.sh	119
14.2 test_do_SP_plot_from_VTL1_GEOGRD.sh	119
14.3 test_do_SP_plot_from_VTL2_GEOGRD.sh	119
14.4 test_do_Polar_plot_GEOGRD.sh	119
14.5 Test of all GEOGRD scripts	119
14.6 run_test_demo.sh	120
V APPLICATION TO CLUSTER DATA	123
15 ACCESS TO CLUSTER DATA	125
15.1 Download CEF files	125
15.1.1 Download command examples	125

15.1.2	Example of downloaded files	126
15.2	CEF to RFF conversion	126
15.2.1	Examples of conversion commands	126
15.2.2	Example of RFF files obtained	126
15.3	Make a local database	127
15.3.1	The CEF database	127
15.3.2	The RFF database	128
15.3.3	Interest of a local database	129
15.4	Database commands	129
16	WORKING ON CLUSTER RFF FILES	131
16.1	Main types of CLUSTER RFF files	131
16.1.1	Example of VecTime file names	131
16.1.2	Example of Spectrograms file names	131
16.1.3	Checking the validity of an RFF file	132
16.2	Special commands for CLUSTER	132
16.2.1	Special commands for all CLUSTER experiments	132
16.2.2	Special commands for CLUSTER/STAFF	133
16.2.3	Special commands for CLUSTER/FGM	135
16.2.4	Special commands for CLUSTER/POS	137
16.3	Make a STAFF spectrogram	138
16.3.1	From a VTL1 file	138
16.3.2	From a VTL2 file	139
16.3.3	Spectrogram visualization	139
16.4	Make a FGM spectrogram	139
16.5	Visualize an average spectrum	140
16.6	Draw a waveform	140
16.7	Extract part of a VecTime file	140
16.8	Change coordinate system	141
16.9	Waves polarisation computation	141
16.9.1	Transformation in MFA coordinate system	141
16.9.2	Calculation and plot of the polarization parameters	141
16.9.3	Plot of polarisation parameters	142
16.10	Curl(B) and Div(B) computation	142
16.10.1	Time alignment	142
16.10.2	Computation	142
16.10.3	Visualization	142
16.11	Data position visualization	143
16.11.1	2D-Visualization	143
16.11.2	3D-visualization	143
16.11.3	Transform positions in another system	143
16.12	Compute orbital parameters	143
16.13	Compute an elliptic trajectory	144
17	USING CLUSTER SCRIPTS	145
17.1	Draw a waveform	145
17.1.1	do_VT_plot_CLUSTA.sh	145
17.1.2	do_VT_plot_CLUFGM.sh	145
17.2	Make a spectrogramme	147
17.2.1	do_SP_plot_CLUSTA.sh	147
17.2.2	do_SP_plot_CLUFGM.sh	147
17.3	Spectrogram of 4 S/C for Bz component	150
17.3.1	do_SP_plot_CLUSTA_4Bz	150
17.3.2	Result	150
17.3.3	do_SP_plot_CLUFGM_4Bz	151

17.3.4	Result	151
17.4	Plot the trajectory of the 4 satellites	152
17.4.1	do_2D_plot_CLUPOS.sh	152
17.4.2	do_3D_plot_CLUPOS.sh	152
17.5	Calculate and visualize waves polarization	155
17.5.1	do_Polar_plot_CLUSTER.sh	155
17.5.2	Example of result in HBR	155
17.5.3	Example of result in NBR	155
17.6	STAFF and FGM comparison	159
17.6.1	do_compare_STA_FGM_CLUSTER.sh	159
17.6.2	Examples of STAFF/FGM DC field comparison	159
17.7	Calculate the orbital parameters of a satellite	160
17.7.1	do_simulated_orbit_CLUPOS.sh	160
17.7.2	Results	160
17.8	CLUSTER tetrahedron geometry	161
17.8.1	Geometry parameters computation	161
17.8.2	Geometry parameters visualization	161
17.9	Curl(B) and Div(B) computation	163
17.9.1	Introduction to the method	163
17.9.2	do_compute_curl_div_CLUFGM.sh script	163
17.9.3	RPC_visu_curl_div_4sat command	164
18	TESTS OF CLUSTER SCRIPTS	169
18.1	test_do_VT_plot_CLUXXX.sh	169
18.2	test_do_SP_plot_CLUXXX.sh	169
18.3	test_do_SP_plot_CLUXXX_4Bz.sh	169
18.4	test_do_SP_plot_CLUSTER_from_VTL1.sh	170
18.5	test_do_Polar_plot_CLUSTER.sh	170
18.6	test_do_2D_plot_CLUPOS.sh	170
18.7	test_do_3D_plot_CLUPOS.sh	170
18.8	test_do_compare_STA_FGM_CLUSTER.sh	170
18.9	test_CLUSTER.sh	170
18.10	test_curl_div_CLUFGM.sh	170
18.11	Tests of all CLUSTER scripts	171
VI	ANNEXES	177
19	DEFINITION OF USED COORDINATE SYSTEMS	179
19.1	The Geographic System (GEO)	179
19.2	The Geocentric Equatorial Inertial system (GEI)	180
19.3	The Geocentric Solar Ecliptic system (GSE)	180
19.4	The Geocentric Solar Magnetospheric system (GSM)	181
19.5	The Vertical Dusk Horizontal system (VDH)	181
19.6	The Sensor Coordinate System (SCS)	182
19.7	The Orthogonal Sensor System (OSS)	182
19.8	The Data Sensor System (DSS)	182
19.9	The Body Build System (BBS)	183
19.10	The Spin Reference System (SRS or SR)	183
19.11	The spin reference2 system (SR2)	183
19.12	The Inverse SR2 system (ISR2)	184
19.13	The Spin Reference Vertical system (SRV)	184
19.14	The Magnetic Field Aligned system (MFA)	185
19.15	The Magnetic Field Aligned Vertical system (MFAV)	185
19.16	The Tangente Paraboloid Normal system (TPN)	186

19.17	The Euler's angles	187
19.18	Transformation matrix VDH-SRV	187
20	CALIBRATION OF SPECTRA AND WAVEFORMS	189
20.1	Introduction	189
20.2	Spectra calibration	189
20.2.1	Theory	189
20.2.2	The successive steps of the calibration	190
20.2.3	Calculation of the complex spectrum	197
20.3	Vectime_L1_to_spectro_L2 parameters	198
20.4	Continuous waveform calibration	202
20.4.1	Theory	202
20.4.2	The successive steps of calibration	203
21	USEFUL INFORMATIONS	205
21.1	Estimation of the DC field in the spin plane	205
21.1.1	Problem	205
21.1.2	Amplitude and phase of B_{\perp} in spinning system SR	206
21.1.3	Tests on $B_{\perp x}$ et $B_{\perp y}$	206
21.1.4	Amplitude and phase of B_{\perp} in fixed reference SR2	206
21.2	Estimation of "misalignment angle"	207
21.2.1	Definitions	207
21.2.2	Signal delivered by a rotating antenna into a DC field	207
21.2.3	Estimate of the misalignment angle θ_z for low value	208
21.2.4	Estimate of the misalignment angle θ_z for high value	208
21.3	Calculation of the depointing matrix	210
21.3.1	Passing from non-orthogonal coordinates to an orthogonal system	210
21.3.1.1	Position of the problem	210
21.3.1.2	Switching from a non-orthogonal system to an orthogonal system	211
21.3.1.3	Calculation of XYZ components from PQR	212
21.3.2	Definition of the depointing matrix	212
21.3.2.1	Misalignment correction matrix	212
21.3.2.2	Calculation of the inverse depointing matrix	213
21.3.3	Example of application: case of Interball electric antennas	213
21.3.4	Experimental estimation of certain terms	214
21.4	Small reminders on the Fourier transform	216
21.4.1	The Fourier transform in the mathematical sense	216
21.4.2	The discrete Fourier transform	216
21.4.3	The "Fast Fourier Transform"	217
21.4.3.1	Negative and positive frequencies	217
21.4.3.2	Some useful relationships	217
21.4.3.3	Normalization test of an FFT	217
21.4.3.4	Edge effects	219
21.5	Usefulness of circular components	220
21.5.1	Définition	220
21.5.2	Passage in left and right circular components	220
21.5.3	Calculation from the spectra	221
21.5.4	Consequences of the rotation on the spectrum	221
22	CURLMETER TECHNIQUE USED FOR CLUSTER	223
22.1	Introduction	223
22.2	Computation Method	223
23	USE OF RPC ON OTHER MISSIONS	225
23.1	ICI-3 and CUSP Rocket	225

23.2	CASSINI magnetometer	226
23.3	THEMIS / SCM	227
23.3.1	Spectrogram at large scale	227
23.3.2	Polarisation of the Whistler	228
23.3.3	Polarization parameters	229
23.3.4	THEMIS position position	230
23.4	MMS / SCM	231
23.4.1	Spectrogram at large scale	231
23.4.2	Polarisation of the event	232
23.4.3	Polarization parameters	233
23.4.4	MMS position	234
24	SUMMARY OF F90 PROGAMS AND LIBRARIES	235
24.1	Programs F90	235
24.1.1	Common programs	235
24.1.2	CLUSTER programs	238
24.1.3	GEOS programs	240
24.1.4	GEOS/GROUND data programs	241
24.2	Libraries	241
24.2.1	Common libraries	241
24.2.1.1	lib_deconvo.f90	241
24.2.1.2	lib_time.f90	242
24.2.1.3	lib_alitime.f90	242
24.2.1.4	lib_utility.f90	242
24.2.1.5	lib_rw_rff.f90	243
24.2.1.6	lib_signal.f90	243
24.2.1.7	rocotlib_V3p2.f90	244
24.2.1.8	lib_gainant.f90	245
24.2.1.9	lib_kepler.f90	246
24.2.1.10	lib_rwcopolar.f90	246
24.2.1.11	lib_igrf13.f90	246
24.2.2	CLUSTER libraries	246
24.2.2.1	lib_rw_cef.f90	246
24.2.2.2	lib_CLUORB.f90	247
24.2.2.3	lib_tools_CLUSTER.f90	247
24.3	Total code size	248
25	LIST OF RPC COMMANDS	249
25.1	Alphabetical list of all RPC commands	249
25.2	Fonctional list of RPC commands	251
25.2.1	Generic commands	251
25.2.2	Special GEOS commands	253
25.2.3	Special GEOS/GROUND commands	253
25.2.4	Special CLUSTER commands	254
26	BIBLIOGRAPHY	255

FOREWORD

RoProc commands (Robert's procedures), or RPC for short, are bash procedures that can be used on a Linux operating system in command mode. They can be used on a Linux machine or on a Windows PC using a subset of Msys, emulating a Linux terminal, downloadable along with the command pack.

Initially developed for the mass processing of CLUSTER data, these commands have been generalized to data from any mission or experiment which is in the form of a time series, i.e. time dependent vector data. Type includes data such as waveform, magnetometer, trajectory, etc.

For the CLUSTER and GEOS data, this set of commands has been supplemented by a certain number of commands specific to this data. For example, for the four magnetometers of the four CLUSTER satellites, a special RPC command was developed to calculate the rotational and divergence of the magnetic field.

This document summarizes the generic commands applicable to any data of the time series form. The only condition for using RPC commands on time series data is to put this data in RFF format [3]. A conversion filter must therefore be developed for each type of new data. After that, the user has the possibility of developing specific commands, as it was done for the GEOS and CLUSTER missions.

This document is presented in 6 parts :

- 1- Overview of Roproc commands
- 2- Application to simulated data
- 3- Application to GEOS data
- 4- Application to GEOS/Ground based data
- 5- Application to CLUSTER data
- 6- Annexes

According you interest, you can go to the part of you choice.

Part I

THE ROPROC COMMANDS

Chapter 1

OVERVIEW OF RPC COMMANDS

1.1 Birth of Roproc

Roproc commands (Robert's Procedures) are a set of procedures, or commands, issued from programs in F90, and bash, allowing the processing and visualization of data delivered in the form of time series. These procedures are primarily for scientific use, but some relate to data acquisition and calibration. Before describing them, it may be useful to recall their sources.

The RoProcs procedures were initially developed before the 2000s for CLUSTER data from STAFF-SC and FGM, where the author is CoI (Co-Investigator) of these two experiments. They also process trajectory data as time series of vectors. But they were also used for other missions, such as Double-Star / STAFF, CUSP and ICI-3 rockets. They can be used for other past missions (THEMIS / SCM and FGM, ISEE, etc. .), present (MMS) or futures providing time series.

The Roproc process two categories of data:

- Waveform data, where the sample rate is constant. Operations such as spectrum calculation, change of reference, polarization calculation, etc. are available on this type of data, such as CLUSTER / STAFF or FGM.
- Vectime data is a sequence of temporally dated vectors, where the sampling rate may be constant or variable.
- Image data, or spectrograms, where each image or spectrum is dated.

For the CLUSTER mission, the calculation of quantities including the spatial dimension provided by the 4 satellites such as the rotational and the divergence of the magnetic field has been added as RPC commands.

A part of the RoProc commands was used in the early 2000s to form the RCL pack dedicated to the mass production of CLUSTER data, see [1]. The RoProc have therefore evolved into two branches since 2015: Part of them was used to form the RCL software (Roproc Command Language), which initially aimed to take all the RoProc commands in a large frame of use. Due to lack of time and availability, the objective was reduced, and only a set of essential commands was retained. Thus it was developed all the commands of mass production for CLUSTER, for the archiving center of the ESA (CAA then CSA) by omitting most of the commands of scientific processing. The whole has been documented in a fairly exhaustive way (see ref. [1], 230 pages) but only concerns CLUSTER.

Along with RCL commands, Roproc have evolved since 2015 towards scientific use, without mass production commands, but based on the RFF exchange format (Roproc File Format, see [3]), which allows to form a coherent frame for all the treatments. In addition this exchange format allows them to easily adapt to any other experiment where a vector quantity depends on time (or a vector). This is how this set of commands now jointly process CLUSTER and GEOS data, and could easily be applied to other time series experiment.

1.2 Function of RPC commands

RPC commands can be used as a script, much like a shell or bash script, and can be viewed as a true processing language. The introduction of the RFF format has also made it possible to facilitate the exchange and transformation of files from one procedure to another.

It should be noted that in its current state, RPCs can easily process data from any other “wave” type experiment, or time series, as long as the input data is put in RFF format. For the RPC calibration procedure `RPC_vectime_calibration_CLUSTER`, just take inspiration from this command and the corresponding program and introduce into it the new transfer function to obtain continuously calibrated waveforms for any past experience or future.

This is what was done for the UBF data from GEOS. The transfer function being known, the calibration program, derived from that of `CLUSTER / STAFF`, was easy to develop. In addition, all generic commands like plotting waveforms, positions, calculating and plotting spectrograms, wave polarization, and others was immediately applicable to GEOS data, via the RFF exchange format.

1.3 A few words on RFF format

The RFF format (Roproc File Format) was originally designed to process vector-type data (scalar, vector, matrix, tensor), which may vary depending of another parameter, itself could be a scalar or a vector [3]. A simple and often encountered application is a vector or a matrix, depending on a scalar which is time, like a waveform or a spectrogram. But this format can also describe a set of field lines, or a vector field in a volume, whose index is the x, y, z position and the data is the vector at this point V_x , V_y , V_z . It can also describe images, where the index can be time, and the data a matrix corresponding to the values of the image (i.e. the color of each pixel).

Thus, the level 1 data at the output of a decommutation program are generally blocks of vectors coded as integers. They correspond to the telemetry values, each dated in time. In the RFF frame, these blocks are therefore time-dependent matrices, the class of which is called `MatTime`, but more commonly called `WaveForm`, because most of the time series from alternative magnetometers originate in this format. The corresponding files are therefore of type `WFL1` (Wave Forms Level 1).

Procedure `RPC_waveform_to_vectime` performs the time interpolation of these data blocks, and now produces a sequence of all time dated vectors, whose class is called `VecTime`. When the corresponding data is level 1, the files are of class `VTL1.rff`. When it is calibrated data, it is `VTL2.rff`.

As for a spectrogram, it is made up of a series of values that depend both on time and on frequency. We must then consider the spectrogram as a series of dated spectra, therefore of dated vectors (whose components are frequency). The RFF format therefore recognizes a `VecTime` class there. But if we want to include 3 components, we are then dealing with a `MatTime` class (one dimension for the frequencies, one dimension for the components). This is the case for the files called `SPL2` (level 2 spectrograms, therefore calibrated), which contain a succession of complex spectra dated with 6 components. These spectra are coded as real matrices of dimension (6, N_f) where N_f is the number of frequency steps. Each frequency spectrum is represented by a vector (B_xR , B_xI , B_yR , B_yI , B_zR , B_zI). For convenience of syntax, this particular class of `MatTime` file is called “Spectrogram”.

So we can see that the different classes of RFF files describe how the data is organized. For more details, we can refer to [3], Robert, 2004-2011.

1.4 Generic RPC commands

1.4.1 Software information

RPC_menu

Return list of RPC commands sorted by categories, as in this section, with a very short description on only one line.

Usage: `RPC_menu`

RPC_doc

Display this document.

Usage: `RPC_doc`

RPC_list

Return alphabetic list of all RPC commands; create file `RPC_list.txt`

Usage: `RPC_list`

RPC_version

Return current RPC version used, ex: `RPC_V5.0`

Usage: `RPC_version`

RPC_compare_versions

Compare two versions of RPC pack. The given directory is another version of RPC, which is compared to the current version.

Usage: `RPC_compare_versions` `RPC_previous_dir`

ex : `RPC_compare_versions /blabla/RPC_V3p0`

RPC_compare_versions_long

Compare two versions of RPC pack. Same as `RPC_compare_versions`, but comparison is deeper.

Usage: `RPC_compare_versions_long` `_previous_dir`

ex : `RPC_compare_versions_long /blabla/RPC_V3p0`

RPC_make_minidoc

Creates a `Mini_doc.txt` file containing the list of RPC commands, and a brief description and list of parameters for each command. Served notably for writing this chapter.

Usage: `RPC_make_minidoc`

1.4.2 RFF file handling

RPC_check_rff

Used to check the consistency of RFF file by reading and some meta data consistency check.

Usage: `RPC_check_rff` `toto.rff` `toto.rff` : name of a RFF file

RPC_check_rff_dir

Check of all RFF files of a given directory

Usage: `RPC_check_rff_dir` `toto` `toto` : name of a directory containing RFF files

RPC_info_rff

Return main info of given RFF file

Usage: `RPC_info_rff` `toto.rff` `option` `toto.rff` : name of a RFF file, `option` : l (long) s (short)

RPC_info_rff_dir

Return main info of all RFF files present into a given directory

Usage: `RPC_info_rff_dir` `toto` `toto` : name of a directory containing RFF files

This command uses `RPC_info_rff`

RPC_clean_rff

RFF cleans a file by removing useless comments and reformatting the data itself, for a read operation and re-write.

Usage: RPC_clean_rff toto.rff

toto.rff : name of a RFF file

RPC_copy_rff

To copy a file by updating the field FILE_NAME.

Usage: RPC_copy_rff toto1.rff toto2.rff

toto1.rff : RFF source file

toto2.rff : RFF target file

RPC_diff_rff

Compute difference between 2 RFF files of same class and same block number.

Usage: RPC_diff_rff toto1.rff toto2.rff

toto1.rff : name of a first RFF file

toto2.rff : name of a second RFF file

RPC_rename_rff

Rename a RFF file with update of file name inside

Usage: RPC_move_rff toto1.rff toto2.rff

toto1.rff : RFF file

toto2.rff : RFF new name

This command uses RPC_copy_rff

RPC_give_rff_param

give header a RFF file

Usage : RPC_give_rff_param toto.rff

toto.rff : RFF file

RPC_put_rff_database

put RFF file in the right place in data base (use RPC_info_rff)

Usage : RPC_put_rff_database toto.rff

toto.rff : RFF file

RPC_copy_rff_database

put RFF file in the right place in data base (use RPC_info_rff)

Usage : RPC_copy_rff_database toto.rff

toto.rff : RFF file

RPC_list_rff_database

list of all rff files in rff data base

Usage : RPC_list_rff_database mission experiment

Ex:

```
RPC_list_rff_database CLUTER STA
RPC_list_rff_database CLUTER FGM
RPC_list_rff_database GEOS ULF
RPC_list_rff_database GEOS POS
RPC_list_rff_database GEOS ALL
RPC_list_rff_database GEOGRD ULF
```

RPC_join_vectime

Join 2 rff vectime files of same experiment in a single file

Usage : RPC_join_vectime file_1 file_2 file_3

with:

```
file_1 : input VTL2 file
file_2 : input VTL2 file to add to the first
file_3 : output VTL2 file containing file_1 and file_2
```


1.4.3 Data processing

These commands were developed for the processing of CLUSTER / SATFF-SC, *but can be used for any type of data of type Waveform or Vectime*, for any experiment of type “Wave”.

RPC_add_DxDy_to_BxBy

Add Dx and Dy to Bx and By on a 5 dimension VT file (Bx, By, Bz, Dx, Dy)

Usage : RPC_add_DxDy_to_BxBy file1 file2

with:

file1 : input VTL2 file with 5 comp., Bx,By,Bz,Dx,Dy

file2 : output VTL2 file with 3 comp., Bx+Dx, By+Dy,Bz

Example : RPC_add_DxDy_to_BxBy CLU2_STASC_VTL2_NBR_ISR2_20010923.rff CLU2_STASC_VTL2_NBR_ISR2_20010923_DxDy.rff

RPC_alitime_4_vectime

Time alignment of 4 rff vectime files (made for CLUSTER, but Generic)

Usage : RPC_alitime_4_vectime VT1.rff VT2.rff VT3.rff VT4.rff

RPC_compute_curl_div_4sat

Compute curl(B) and div(B) from MAG and POS data (made for CLUSTER, but Generic)

Usage :

RPC_compute_curl_div_4sat MAG1 MAG2 MAG3 MAG4 POS1 POS2 POS3 POS4 (RFF files)
Create file compute_curl_div_4sat.resu

RPC_change_coordinate_system

Change coordinate system on a Vectime RFF file in Cartesian coordinates.

Usage: RPC_change_coordinate_system toto.rff gse toto_gse.rff

with:

toto.rff : name of an input RFF VT file

gse : name of new desired coordinate system
(geo, gei, gsq, gse, gsm, sma, mag)

toto_gse.rff : name of an output RFF VT file

RPC_compute_sat_orbit_param

Compute elliptical orbital parameters of a Earth satellite from a data position file

Usage : RPC_compute_sat_orbit_param position_file.rff

with:

position_file.rff : VT file of position over one orbit

RPC_compute_sat_trajectory

Compute elliptical trajectory of a Earth satellite

Usage : RPC_compute_sat_trajectory Apo Per PerTime a1,a2,a3 b1 b2 b3 c1 c2 c3 dt rff_out datiso_per scale

with:

Apo : Apogee in km, real*8

Per : Parigee in km, real*8

PerTime : Perigee Time, ISO format

a1,a2,a3 : direction of semi-major axis, in GEI system
(from ellipse center to perigee, unit vector)

b1,b2,b3 : direction of semi-minor axis, in GEI system
(a,b vector in the sens of the motion, unit vector)

c1,c2,c3 : direction of semi-minor axis, in GEI system

dt : time resolution for output positions

rff_out : output rff_file

datiso1 : starting date from computed trajectory

scale : factor to apply to x,y,z

RPC_create_simulated_data

Create simulated data of ULD, MAG and POS data

Usage: RPC_create_simulated_data

work without arguments; create files:

 VTL2_simulated_MAG_data.rff

 VTL2_simulated_POS_data.rff

 VTL2_simulated_ULF_data.rff

RPC_current_tube

create 8 simulated data files containing position and mag. field values

require 7 argument(s)

 R tube ? (ex: 2000. km)

 Density in A/km2 ? (ex: ., 0. 0.25)

 characteristic length fo the tetrahedron . (ex: 1. or 10., or 100.)

 impact parameter ? (ex: -0.7, in fraction or R tube)

 Velocity in X ? (ex: 10 km/s)

Usage: RPC_current_tube R Dx Dy Dz CL para Vx

Example :

RPC_current_tube 2000. 0. 0. 0.25 10. -0.7 10.

RPC_reduce_time_rff

Reduce the time period of a RFF file (works only on WF and VT RFF files)

Usage: RPC_reduce_time_rff toto1_VT.rff toto2_VT.rff datiso1 datiso2

with:

 toto1_VT.rff : name of an input vectime RFF file

 toto2_VT.rff : name of an output vectime RFF file

 datiso1 datiso2 : date in ISO format

Example :

RPC_reduce_time_rff toto1_VT.rff toto2_VT.rff 2001-09-23T09:10:00.000Z 2001-09-23T09:30:00.000Z

RPC_reduce_time_vectime

Reduce the time period of a RFF file (works only on WF and VT RFF files)

Usage: RPC_reduce_time_vectime toto_VT.rff toto_VT.rff datiso1 datiso2

with:

 toto_VT.rff : name of an input vectime RFF file

 toto_VT.rff : name of an output vectime RFF file

 datiso1 datiso2 : date in ISO format

Example :

RPC_reduce_time_vectime GEOS1_ULF_VTL2_19770713.rff GEOS1_ULF_VTL2_19770713_red.rff 1977-07-13T09:12:00Z 1977-07-13T09:30:00Z

RPC_spectro_to_polar

Produce a file containing all polarization parameters from a SP file.

Usage: RPC_spectro_to_polar toto_SP.rff polar.resu

with:

 toto_SP.rff : name of an input RFF SPL2 file

 polar.resu : name of an output txt file containing results

RPC_spectro_xyz_to_lrz

Transform a RFF spectrogram file with XYZ Cartesian coordinates in circular Left Right and Z (unchanged) components..

Usage: RPC_spectro_xyz_to_lrz toto_SP.rff

with:

 toto_SP.rff : name of an input RFF SP file

RPC_vectime_gse_to_isr2

Change coordinate system from GSE to ISR2

Usage: RPC_vectime_gse_to_isr2 toto_gse.rff rasc dec toto_isr2.rff

with:

toto_gse.rff : name of a RFF file
 rasc, dec : right Ascension and Declination in GEI (degrees)
 toto_isr2.rff : name of a new RFF file

RPC_vectime_to_mfa

Transform a vectime file in GSE into the MFA coordinates

Usage : RPC_vectime_to_mfa toto1.rff toto2.rff toto3.rff

with:

toto1.rff : name of input ULF VTL2 RFF file in GSE
 toto2.rff : name of output ULF VTL2 RFF file in MFA
 toto3.rff : name of input MAG VTL2 RFF file in GSE

RPC_vectime_to_mva

Transform a vectime file in any system into MVA coordinates (Minimum Variance Analysis)

Usage : RPC_vectime_to_mva toto1.rff toto2.rff

with:

toto1.rff : name of input ULF VTL2 RFF file
 toto2.rff : name of output ULF VTL2 RFF file in MVA

RPC_vectime_to_spectro

Produce a RFF spectrogram class from a VecTime file, whatever the data inside (calibrated or not, and any origin). The spectrograms have the same units as the VT.

Usage: RPC_vectime_to_spectro VT.rff SP.rff N_Kern N_shift Apod

with:

VT.rff : name of an input vectime RFF file
 SP.rff : name of an output spectrogram RFF file
 N_Kern : Kernel size
 N_shift : for sliding window
 Apod : trapezium (t) or Gaussian (g)

Example :

RPC_vectime_to_spectro CLU1_STASC_NBR_VT_20091213.rff CLU1_STASC_NBR_SP_20091213.rff 512 512 t

RPC_vectime_to_indexed_data

create indexed_data file from a rff VT file

Usage : RPC_vectime_to_indexed_data file_1.rff file_2.ind

with:

file_1 : input VT file
 file_3 : output file with only indexed data

RPC_waveform_to_vectime

Convert a Waveform RFF class to a RFF Vectime class.

All information of RFF WaveForm file is kept in the file RFF VecTime file. All vectors are now dated to the time of the block thank to a precise sampling frequency.

Usage: RPC_waveform_to_vectime toto_WF.rff toto_VT.rff

with:

toto_WF.rff : input file name of class WaveForm RFF.
 toto_VT.rff : output file name of class VecTime RFF.

RPC_test_keplerlib

Compute Earth orbit around Sun, to check routines

Usage: RPC_test_keplerlib

work without arguments

RPC_list_block_WF

Return the list of all blocks in a WF file. Print the block header and skip the data.

Usage: RPC_list_block_WFL1.rff WFL1.rff

Example :

RPC_list_block_WF CLU4_STASC_NBR_WFL1_20050323.rff
will give :

```
=====
Run of RPC\_list_block_WF.exe :
Please wait...
2005-03-23T00:00:00.965544Z 0000000000001 102.70
2005-03-23T00:00:01.965538Z 0000000000001 189.71
2005-03-23T00:00:02.965531Z 0000000000001 276.72
2005-03-23T00:00:03.965524Z 0000000000001 3.74
2005-03-23T00:00:04.965517Z 0000000000001 90.75
etc...
```

1.4.4 RFF files visualization**RPC_visu_vectime**

Visualization of a VecTime RFF file

Usage: RPC_visu_vectime toto_VT.rff datiso1 datiso2
toto_VT.rff : name of a vectime RFF file
datiso1 datiso2 : iso date/time first and end

RPC_visu_vectime_3D

Visualization of a VecTime RFF file on 3 planes + 3D view

Usage: RPC_visu_vectime_3D toto_VT.rff datiso1 datiso2
toto_VT.rff : name of a vectime RFF file
datiso1 datiso2 : iso date/time first and end

RPC_visu_2_vectime

Visualization of 2 VecTime RFF file on the same plot

Usage: RPC_visu_2_vectime toto1_VT.rff toto2_VT.rff datiso1 datiso2
toto1_VT.rff : name of a 1st vectime RFF file
toto2_VT.rff : name of a 2nd vectime RFF file
datiso1 datiso2 : iso date/time first and end

RPC_visu_2_vectime_3D

visualization 3D of 2 Vectime RFF file

Usage: RPC_visu_2_vectime_3D toto1_VT.rff toto2_VT.rff datiso1 datiso2
toto1_VT.rff : name of a 1st vectime RFF file
toto2_VT.rff : name of a 2nd vectime RFF file
datiso1 datiso2 : iso date/time first and end

RPC_visu_vectime_4sat

Visualization of 4 VecTime RFF file on the same plot

Usage: RPC_visu_vectime_4sat toto1_VT.rff toto2_VT.rff toto3_VT.rff toto4_VT.rff datiso1 datiso2
toto*_VT.rff : 4 names of a vectime RFF file
datiso1 datiso2 : iso date/time first and end

RPC_visu_vectime_3D_4sat

Visualization of 4 VecTime RFF file on the same plot, on 3 planes + 3D view

Usage: RPC_visu_vectime_3D_4sat toto1_VT.rff toto2_VT.rff toto3_VT.rff toto4_VT.rff datiso1 datiso2
toto*_VT.rff : 4 names of a vectime RFF file
datiso1 datiso2 : iso date/time first and end

RPC_visu_spectro

Visualization of a Spectrogram RFF file

Usage: `RPC_visu_spectro SP.rff datiso1 datiso2 f1 f2 pmin pmax XY fi1 fi2`
 `SP.rff` : name of a spectro RFF file
 `datiso1 datiso2` : iso date/time first and end
 `f1 f2` : frequency bounds to plot
 `pmin pmax` : min max power for dynamic colors
 `XY or LR` : for XYZ or Left Right Z
 `fi1, fi2` : frequency bounds for integrated power
 `ex1 : RPC_visu_spectro toto_SP.rff 2001-09-23T09:00:00.000Z 2001-09-23T11:00:00.000Z 0. 5. 0. 0. XY 0.2 10.`
 `ex2 : RPC_visu_spectro toto_SP.rff 0 0 0. 200. -9.6 -3. LR 0. 200.`

RPC_visu_spectro_4Bz

Visualization of 4 Spectrogram RFF file Bz only

Usage: `RPC_visu_spectro_4Bz SP1.rff SP2.rff SP3.rff SP4.rff datiso1 datiso2 f1 f2 pmin pmax fi1 fi2`
 `SP1-4.rff` : names of 4 spectro RFF files
 `datiso1 datiso2` : iso date/time first and end
 `f1 f2` : frequency bounds to plot
 `pmin pmax` : min max power for dynamic colors
 `fi1, fi2` : frequency bounds for integrated power
 `ex1 : RPC_visu_spectro_4Bz SP1.rff SP2.rff SP3.rff SP4.rff 2001-09-23T09:00:00.000Z 2001-09-23T11:00:00.000Z 0. 5. 0. 0. 0.2 10.`
 `ex1 : RPC_visu_spectro_4BzSP1.rff SP2.rff SP3.rff SP4.rff 0 0 0. 0. 0. 0. 0.2 0.`

RPC_visu_ave_spectrum

Visualization of an average spectrum from Spectrogram RFF file

Usage: `RPC_visu_ave_spectrum SP.rff datiso1 datiso2 f1 f2 pmin pmax XY y y`
 `SP.rff` : name of a spectro RFF file
 `datiso1 datiso2` : iso date/time first and end
 `f1 f2` : frequency bounds to plot
 `pmin pmax` : min max power for dynamic colors
 `XY or LR` : for XYZ or Left Right Z
 `info_orb` : y/n
 `info_FGM` : y/n
 `ex1 : RPC_visu_ave_spectrum toto_SP.rff 2001-09-23T09:00:00.000Z 2001-09-23T11:00:00.000Z 0. 5. 0. 0. XY y n`
 `ex2 : RPC_visu_ave_spectrum toto_SP.rff 0 0 0. 0. 0. 0. XY y n`

RPC_visu_polar

Visualization of a copolar.resu file produced by `RPC_spectro_to_polar`.

Usage: `RPC_visu_polar copolar_file datiso1 datiso2 threshold f1 f2`
 `datiso1 datiso2` : iso date/time first and end of desired time section (0 0 for all)
 `threshold` : for visualization ex: -6.1
 `f1 f2` : frequency bounds for visualizations
 `ex : RPC_visu_polar copolar.resu 0 0 -6.1 0. 2.`

1.4.5 Dates and calendar operations**RPC_current_date**

Retourne la date courante sous la forme : 2012-08-28 14:08:16 day 241 Julsec 1346155696

Usage: `RPC_current_date`
 `ex: RPC_current_date`
 `return 2014-11-20 15:32:33 day 324 Julsec 1416493953`

RPC_next_day

Return date of next day, as '20070925'

Usage: RPC_next_day date

ex: RPC_next_day 20070925

return 20070926. Valid since 1972 to 2058.

RPC_previous_day

Return date one day before, as '20070925'

Usage: RPC_previous_day date

ex: RPC_previous_day 20070925

return 20070924. Valid since 1972 to 2058.

RPC_nday_of_month

Return number of day in a month

Usage: RPC_nday_of_month year month

ex: RPC_nday_of_month 1997 11

return 30. Valid since 1972 to 2058.

RPC_list_days_of_month

Return list of days in a month as 01 02.... 29 30 according month and year (1971 < year < 2059)

Usage: RPC_list_days_of_month year month

ex: RPC_list_days_of_month 1997 11

return 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30

RPC_date_time_to_datiso

Return ISO_date from yymmdd hhmmssc

Usage : RPC_date_time_to_datiso date time

ex: RPC_date_time_to_datiso 20010923 090000

return : 2001-09-23T09:00:00.000Z

RPC_datiso_to_date_time

Return yymmdd hhmmss from ISO_date

Usage: RPC_datiso_to_date_time ISO_date

ex: RPC_datiso_to_date_time 2001-09-23T09:00:00.000Z

return : 20010923 090000

RPC_decode_datim

Return elements of a date_time variable.

Usage: RPC_decode_datim datim

ex: RPC_decode_datim 20010923_091703

return 2001 09 23 09 17 03

RPC_decode_datiso

Return elements of an ISO date .

Usage: RPC_decode_datiso datiso

ex: RPC_decode_datiso 2001-09-23T09:17:03.000Z

return 2001 09 23 09 17 03

RPC_encode_datiso

Return ISO_date from year month day hour min sec.

Usage: RPC_encode_datiso month day hour min sec

ex: RPC_encode_datiso 2001 09 23 09 17 03

return 2001-09-23T09:17:03.000Z

1.4.6 PostScripts conversion

RPC_ps_to_pdf

Create PDF file with/without image compression ;

Produce .pdf file from .ps one

Usage: RPC_ps_to_pdf.sh toto.ps

RPC_ps_to_pdf.sh toto.ps compress (active image compression)

RPC_ps_to_png

Create PNG file for a given resolution (dpi, 300=good) and given number of colors ;

Produce .png file from .ps one

Usage: RPC_ps_to_png.sh toto.ps 80 256 (80 ppi or dpi, and 250 colors)

RPC_ps_to_png.sh toto.ps 300 16m (300 ppi or dpi, and 16 Millions colors)

RPC_ps_to_png_256

Create PNG file for a given resolution (dpi, 300=good) and 256 colors ;

Produce .png file from .ps one

Usage: RPC_ps_to_png.sh toto.ps 300 (300 ppi or dpi)

RPC_ps_to_png_16m

Create PNG file for a given resolution (dpi, 300=good) and 16 Millions colors ;

Produce .png file from .ps one

Usage: RPC_ps_to_png.sh toto.ps 300 (300 ppi or dpi)

1.4.7 System and directories

RPC_clean_dir

Remove remove *.in *.out *.old *.tmp of the current directory

Usage : RPC_clean_dir

with arg -a remove also *.ps *.png *.pdf

RPC_chmod_tree

Set files modes on all the tree.

all dir : chmod go-w chmod ugo+x

all .exe : chmod ugo+x

all .bash : chmod ugo+x

all .sh : chmod ugo+x

all RPC : chmod ugo+x RPC*

Usage: RPC_chmod_tree

require no argument

RPC_convert_names

Convert recursively files name having blank or special character in the name

blank or special characters are replaced by "_"

Usage : RPC_convert_names path_dir

path_dir : absolute path of a directory

RPC_check_dirname_tree

Search in a tree all directories having a character blank in the name, and propose to replace it by "_"

Usage: RPC_check_dirname_tree

RPC_dir_pretty_tree

Give directories for all the tree (pretty form)

Usage: RPC_dir_pretty_tree Usage: RPC_dir_pretty_tree dir
dir : name of a directory

ex : RPC_dir_pretty_tree \$RPC_V5p0
will give:

```
-----
nb. of directories considered:      23
Maximum number of levels          :    3

|_bash
|_bin
|_doc-----
|          |_src_doc
|_gainantset
|_mod
|_obj
|_pro
|_scripts___
|          |_scripts_generic
|          |_scripts_CLUSTER
|          |_scripts_GEOS
|_src
|_tests-----
|          |_tests_generic_scripts
|          |_test_compute_sat_trajectory
|          |_tests_RPC_dir-----
|                                     |_dummy_dir_1
|                                     |_dummy_dir_2
|          |_tests_CLUSTER
|          |_tests_GEOS
|_work
```

RPC_dir_save

Save recursively a directory in another one

Usage: RPC_dir_save toto1 toto2

toto1 : name of source directory (absolute path)

toto2 : name of target directory (absolute path)

- If some files of toto1 are existing in toto2, and are identical, there will be not recopied;
- If some files of toto1 are existing in toto2, but are different, there will recopied in toto2;
- files of toto2 not present in toto1 are deleted.

RPC_dir_size

Give directory size (octets or Mo) for all the tree *gives exact value, whatever the 0.S.*

Usage: RPC_dir_size

RPC_dir_size Mo

RPC_dir_size DIR

RPC_dir_size DIR Mo

ex : RPC_dir_size tests

will give :

5491371429 tests

RPC_dir_size tests Mo

will give :

5491 tests

RPC_dir_size_tree

Give directory size (octets or Mo) for all the tree

Usage: RPC_dir_size_tree

RPC_dir_size_tree Mo

RPC_dir_size_tree DIR

RPC_dir_size_tree DIR Mo

ex1 : RPC_dir_size_tree \$RPC_DIR

will give :

system/host: Linux lx-robert

directory: .

size in octets

```

196425315 .
  536876 ./bash
  77037312 ./bin
108692864 ./doc
  65786307 ./doc/src_doc
  1379869 ./gainantset
    7009 ./mod
  6625992 ./obj
  535904 ./pro
    61188 ./scripts
    34738 ./scripts/scripts_CLUSTER
      0 ./scripts/scripts_generic
    26450 ./scripts/scripts_GEOS
1376584 ./src
  35269 ./tests
    1244 ./tests/test_compute_sat_trajectory
    6282 ./tests/tests_CLUSTER
      0 ./tests/tests_generic_scripts
    6123 ./tests/tests_GEOS
  21620 ./tests/tests_RPC_dir
    0 ./tests/tests_RPC_dir/dummy_dir_1
    0 ./tests/tests_RPC_dir/dummy_dir_2
  1219 ./work

```

RPC_dir_size_pretty_tree

Same as above, but with more pretty appearance. Uses RPC_dir_size_tree

Usage: RPC_dir_size_pretty_tree

ex : RPC_dir_size_pretty_tree \$RPC_DIR

will give :

Procedure dir_size_pretty_tree.exe - P. Robert, CETP, 1990 - Rev. Dec. 1995/ Feb 2014

Starting date : 2021-02-05 00:01:07

host name : lx-robert

kernel name/release : Linux 4.4.0-200-generic

Starting directory : .

Total number of directories : 23

Total number of files : 443

nb. of directories considered: 23

Maximum number of levels : 3

size in octets

```

196425561. _
  536876  |_bash

```

```

77037312 | _bin
108692864 | _doc_____
65786307 | | _src_doc
1379869 | _gainantset
7009 | _mod
6625992 | _obj
535904 | _pro
61188 | _scripts____
34738 | | _scripts_CLUSTER
0 | | _scripts_generic
26450 | | _scripts_GEOS
1376584 | _src
35269 | _tests_____
1244 | | _test_compute_sat_trajectory
6282 | | _tests_CLUSTER
0 | | _tests_generic_scripts
6123 | | _tests_GEOS
21620 | | _tests_RPC_dir_____
0 | | | _dummy_dir_1
0 | | | _dummy_dir_2
1219 | _work

```

RPC_dir_properties

Give directory size (octets or Mo), number of files & directories

Usage: RPC_dir_properties

RPC_dir_properties Mo

RPC_dir_properties DIR

RPC_dir_properties DIR Mo

ex1 : RPC_dir_properties tests

will give : 5491371429 616 23 | tests

ex2 : _dir_properties tests Mo

will give : 5491 616 23 | tests

RPC_dir_properties_tree

Give directory properties for all the tree

Usage: RPC_dir_properties_tree

RPC_dir_properties_tree Mo

ex : RPC_dir_properties_tree \$RPC_DIR

will give :

size in octets, Nb files, Nb Dir.

```

-----
196428715      441      22 | .
536876         138      0 | ./bash
77037312        42      0 | ./bin
108692864       19      1 | ./doc
65786307        5      0 | ./doc/src_doc
1379869        34      0 | ./gainantset
7009           4      0 | ./mod
6625992        52      0 | ./obj
535904         30      0 | ./pro
61188         22      3 | ./scripts
34738         12      0 | ./scripts/scripts_CLUSTER
0              0      0 | ./scripts/scripts_generic
26450         10      0 | ./scripts/scripts_GEOS
1376584        53      0 | ./src
35269         33      7 | ./tests
1244           3      0 | ./tests/test_compute_sat_trajectory
6282         12      0 | ./tests/tests_CLUSTER
0             0      0 | ./tests/tests_generic_scripts
6123         10      0 | ./tests/tests_GEOS
21620          8      2 | ./tests/tests_RPC_dir
0             1      0 | ./tests/tests_RPC_dir/dummy_dir_1
0             2      0 | ./tests/tests_RPC_dir/dummy_dir_2
1219          1      0 | ./work

```

RPC_dir_properties_pretty_tree

Same as above, but with more pretty appearance....

This command uses `RPC_dir_properties_tree`

ex1 : `RPC_dir_properties_pretty_tree tests Mo`
will give :

size in octets, Nb files, Nb Dir.

```
-----
196427119 441 22 ._
 536876 138 0 | _bash
77037312 42 0 | _bin
108692864 19 1 | _doc
65786307 5 0 | | _src_doc
1379869 34 0 | _gainantset
 7009 4 0 | _mod
6625992 52 0 | _obj
535904 30 0 | _pro
 61188 22 3 | _scripts
 34738 12 0 | | _scripts_CLUSTER
 0 0 0 | | _scripts_generic
26450 10 0 | | _scripts_GEOS
1376584 53 0 | _src
35269 33 7 | _tests
1244 3 0 | | _test_compute_sat_trajectory
6282 12 0 | | _tests_CLUSTER
 0 0 0 | | _tests_generic_scripts
6123 10 0 | | _tests_GEOS
21620 8 2 | | _tests_RPC_dir
 0 1 0 | | | _dummy_dir_1
 0 2 0 | | | _dummy_dir_2
1219 1 0 | _work
```

RPC_dir_diff

Compare recursively two directories

Usage: `RPC_dir_diff toto1 toto2 dir_depth`

toto1 : name of first directory (absolute path)

toto2 : name of second directory (absolute path)

dir_depth : depth to explore, 1=short comparison, > 1=long

This command uses `RPC_dir_properties_tree`

ex1 : `RPC_dir_diff /data/RPC/RPC_V5p2 /data/RPC/RPC_V5p1 1`
will give :

```
-----
déc. 7 19:04 /data/RPC/RPC_V5p2                                     mars 15 2021 /data/RPC/RPC_V5p1
-----
270918479 518 24 | /RPC_V5p2
 536157 148 0 | /RPC_V5p2/bash
115338984 63 0 | /RPC_V5p2/bin
143718175 13 1 | /RPC_V5p2/doc
1379869 34 0 | /RPC_V5p2/gainantset
 7009 4 0 | /RPC_V5p2/mod
7219184 77 0 | /RPC_V5p2/obj
 83704 30 4 | /RPC_V5p2/scripts
2378564 80 0 | /RPC_V5p2/src
102040 55 9 | /RPC_V5p2/tests
 0 0 0 | /RPC_V5p2/work

261748651 483 22 | /RPC_V5p1
 517689 142 0 | /RPC_V5p1/bash
106904672 59 0 | /RPC_V5p1/bin
143718175 13 1 | /RPC_V5p1/doc
1379869 34 0 | /RPC_V5p1/gainantset
 7011 4 0 | /RPC_V5p1/mod
6759208 72 0 | /RPC_V5p1/obj
 75123 27 3 | /RPC_V5p1/scripts
2163094 74 0 | /RPC_V5p1/src
 90181 48 8 | /RPC_V5p1/tests
 0 0 0 | /RPC_V5p1/work
-----
```

ex1 : `RPC_dir_diff /data/RPC/RPC_V5p2 /data/RPC/RPC_V5p1 2`
will give :

 déc. 7 19:04 /data/RPC/RPC_V5p2

270918493	518	24		/RPC_V5p2
536171	148	0		/RPC_V5p2/bash
115338984	63	0		/RPC_V5p2/bin
143718175	13	1		/RPC_V5p2/doc
96419793	4	0		/RPC_V5p2/doc/src_doc
1379869	34	0		/RPC_V5p2/gainantset
7009	4	0		/RPC_V5p2/mod
7219184	77	0		/RPC_V5p2/obj
83704	30	4		/RPC_V5p2/scripts
39667	13	0		/RPC_V5p2/scripts/scripts_CLUSTER
9024	4	0		/RPC_V5p2/scripts/scripts_generic
8571	3	0		/RPC_V5p2/scripts/scripts_GEOGRD
26442	10	0		/RPC_V5p2/scripts/scripts_GEOS
2378564	80	0		/RPC_V5p2/src
102040	55	9		/RPC_V5p2/tests
4199	4	0		/RPC_V5p2/tests/test_compute_curl_div
10213	7	0		/RPC_V5p2/tests/test_compute_sat_trajectory
38250	14	0		/RPC_V5p2/tests/tests_CLUSTER
7475	4	0		/RPC_V5p2/tests/tests_generic_scripts
11108	7	0		/RPC_V5p2/tests/tests_GEOGRD
25573	12	0		/RPC_V5p2/tests/tests_GEOS
5222	7	2		/RPC_V5p2/tests/tests_RPC_dir
0	0	0		/RPC_V5p2/work

 mars 15 2021 /data/RPC/RPC_V5p1

261748651	483	22		/RPC_V5p1
517689	142	0		/RPC_V5p1/bash
106904672	59	0		/RPC_V5p1/bin
143718175	13	1		/RPC_V5p1/doc
96419793	4	0		/RPC_V5p1/doc/src_doc
1379869	34	0		/RPC_V5p1/gainantset
7011	4	0		/RPC_V5p1/mod
6759208	72	0		/RPC_V5p1/obj
75123	27	3		/RPC_V5p1/scripts
39667	13	0		/RPC_V5p1/scripts/scripts_CLUSTER
9024	4	0		/RPC_V5p1/scripts/scripts_generic
26432	10	0		/RPC_V5p1/scripts/scripts_GEOS
2163094	74	0		/RPC_V5p1/src
90181	48	8		/RPC_V5p1/tests
4160	4	0		/RPC_V5p1/tests/test_compute_curl_div
10210	7	0		/RPC_V5p1/tests/test_compute_sat_trajectory
38227	14	0		/RPC_V5p1/tests/tests_CLUSTER
7462	4	0		/RPC_V5p1/tests/tests_generic_scripts
25568	12	0		/RPC_V5p1/tests/tests_GEOS
4554	7	2		/RPC_V5p1/tests/tests_RPC_dir
0	0	0		/RPC_V5p1/work

RPC_search_duplicates

Search and list duplicate files in a tree .

Usage: RPC_search_duplicates mode (I or B)

I : interactive run (stop at each duplicate found)

B : batch use (a duplicates.txt file will be created)

RPC_system_info

Return system information (host, system, machine, OS etc.)

Usage: RPC_system_info

will give something like :

 network node hostname : lx-robert
 kernel name : Linux
 kernel release : 4.4.0-197-generic
 kernel version : #229-Ubuntu SMP Wed Nov 25 11:05:42 UTC 2020
 operating system : GNU/Linux
 machine hardware name : x86_64
 processor type : x86_64
 hardware platform : x86_64

RPC_dos_to_unix_all_files

convert all files in tree from DOS to UNIX format

Usage: RPC_dos_to_unix_all_files

require no argument

RPC_unix_to_dos_all_files

convert all files in tree from UNIX to DOS format

Usage: RPC_unix_to_dos_all_files

require no argument

1.4.8 File conversion to RFF

RPC_flat_to_rff

convert a flat file to RFF

Usage : `RPC_flat_to_rff toto.flat`

`toto.flat` : ascii flat file ; `toto.rff` will be created.

1.4.9 Help of commands

Each RPC command includes a minimum online help explaining the arguments to be supplied. Just follow the command with the '-h' argument. For example:

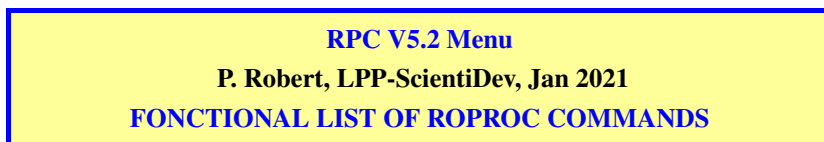
RPC_visu_vectime -h

will return:

```
RPC_visu_vectime : visualization of a Vectime RFF file
RPC_visu_vectime  require 3 argument(s), ex:
RPC_visu_vectime  toto_VT.rff datiso1 datiso2
                  toto_VT.rff : name of a vectime RFF file
                  datiso1 datiso2 : iso date/time first and end
```

1.5 RPC_menu

This command display on the screen a list of RPC commands sorted by categories, with a very short description on only one line. A PDF viewer windows appears, showing:



• Software Informations

RPC_menu	: display the fonctionnal list of RPC commands in html.
RPC_doc	: display the documentation in HTML
RPC_list	: return list of all RPC commands; create RPC_list.txt
RPC_version	: return current RPC version used, ex: RPC_V5p2
RPC_compare_versions	: compare two different versions of RPC package
RPC_compare_versions_long	: same as previous but more deeper
RPC_make_minidoc	: create file mini_doc.txt

• RFF file handling

RPC_check_rff	: check content of a RFF file
RPC_check_rff_dir	: check of all RFF files of a given directory
RPC_info_rff	: return main info of given RFF file
RPC_info_rff_dir	: return main info of all RFF files of a given directory
RPC_clean_rff	: clean content of a RFF file
RPC_copy_rff	: copy a RRF file into another, name updated
RPC_diff_rff	: compute difference between 2 RFF files
RPC_rename_rff	: rename a RFF file with update of file name inside
RPC_give_rff_param	: give header a RFF file
RPC_put_rff_database	: move given file in the right data base
RPC_copy_rff_database	: copy given file in the right data base
RPC_list_rff_database	: according mission / manip
RPC_join_vectime	: join 2 rff vectime files of same experiment in a single file

• RFF file processing

RPC_add_DxDy_to_BxBy	: add Dx and Dy to Bx and By
RPC_alitime_4_vectime	: time alignment of 4 rff vectime files
RPC_compute_curl_div_4sat	: compute curl & div from MAG & POS data
RPC_change_coordinate_system	: on VecTime files only
RPC_compute_sat_orbit_param	: compute orbital parameters of a Earth satellite
RPC_compute_sat_trajectory	: compute elliptical trajectory of a Earth satellite
RPC_create_simulated_data	: create ULF, MAG or POS simulated vectime files
RPC_current_tube	: create a simulated file compute_curl_div_4sat.resu
RPC_reduce_time_rff	: reduce the time period of a RFF file
RPC_reduce_time_vectime	: reduce the time period of a VT RFF file (more fast)
RPC_spectro_to_polar	: produce a polar file from a Spectrogram file
RPC_spectro_xyz_to_lrz	: convert a Spectrogram in XYZ into a Left-Right-Z one
RPC_vectime_gse_to_isr2	: change coordinate system from GSE to ISR2
RPC_vectime_to_mfa	: change coordinate system to MFA
RPC_vectime_to_mva	: change coordinate system to MVA
RPC_vectime_to_spectro	: produce a Spectrogram file from a VecTime file
RPC_vectime_to_indexed_data	: create indexed data file from a rff VT file
RPC_waveform_to_vectime	: convert a RFF ofWaveform class to one of Vectime class
RPC_test_kepler_lib	: test components of the lib_Kepler
RPC_list_block_WF	: return the list of all blocks in a WF file

• Visualization tools

RPC_visu_vectime	: visualization of a VecTime RFF file (2D)
RPC_visu_vectime_3D	: visualization of a VecTime RFF file (3D, for orbit data)
RPC_visu_2_vectime	: visualization of 2 Vectime RFF files
RPC_visu_2_vectime_3D	: 3-D visualization of 2 Vectime RFF files
RPC_visu_vectime_4sat	: visualization of 4 VT files for 4 S/C
RPC_visu_vectime_3D_4sat	: 3D visualization of 4 VT files for 4 S/C
RPC_visu_spectro	: visualization of a Spectrogram RFF file
RPC_visu_spectro_4Bz	: visualization of 4 spectrogrammes Bz of 4 S/S
RPC_visu_ave_spectrum	: visualization of an average spectrum from Spectrogram file
RPC_visu_polar	: visualization of a copolar.resu file

• Date conversion

RPC_current_date	: return 2012-08-28 14:08:16 day 241 Julsec 1346155696
RPC_next_day	: return date of next day, as '20070925'
RPC_previous_day	: return date one day before, as '20070925'
RPC_nday_of_month	: return number of day in a month
RPC_list_days_of_month	: return a list of day for a given year/month as 01 02 03 30
RPC_date_time_to_datiso	: return ISO_date from yymmdd hhmmssc
RPC_datiso_to_date_time	: return yymmdd hhmmssc from ISO_date
RPC_decode_datim	: for 20010923_091703 return 2001 09 23 09 17 03
RPC_decode_datiso	: for 2001-09-23T09:17:03.000Z return 2001 09 23 09 17 03
RPC_encode_datiso	: return ISO_date from year month day hour min sec

• PostScript conversion

RPC_ps_to_pdf	: create PDF file with or without image compression
RPC_ps_to_png	: create PNG file for a given resolution and color number
RPC_ps_to_png_256	: create PNG file for given dpi and 256 colors
RPC_ps_to_png_16m	: create PNG file for given dpi and 16M colors

• System and Directories

RPC_clean_dir	: remove *.in, *.out, *.old, *.resu from current dir.
RPC_chmod_tree	: set mode to all files in the tree, from current directory
RPC_convert_names	: convert recursively files name having blank or special character
RPC_check_dirname_tree	: check dir name in the tree having blank in the name
RPC_dir_pretty_tree	: give directories for all the tree
RPC_dir_save	: save recursively a directory in another one
RPC_dir_size	: give directory size (octets or Mo)
RPC_dir_size_tree	: give directory size (octets or Mo) for all the tree
RPC_dir_size_pretty_tree	: same with pretty presentation
RPC_dir_properties	: give directory size (octets or Mo), files # /directories #
RPC_dir_properties_tree	: give directory properties for all the tree
RPC_dir_properties_pretty_tree	: same with pretty presentation
RPC_dir_diff	: compare recursively two directories
RPC_search_duplicates	: search and list duplicate files in a tree
RPC_system_info	: return system information (host, system, OS etc.)
RPC_dos_to_unix_all_files	: convert all files in tree from DOS to UNIX format
RPC_unix_to_dos_all_files	: convert all files in tree from UNIX to DOS format

• Files conversion

RPC_flat_to_rff	: conversion of a flat file to a rff file
-----------------	---

see also: RPC_menu_CLUSTER RPC_menu_GEOS RPC_menu_GEOGRD

Chapter 2

SOFTWARE INSTALLATION

2.1 Download the RPC package

The pack is available at CDPP, Software Libraries:

<https://cdpp-archive.cnes.fr/>

The pack is available in Linux version, or Windows with the Linux Msys emulator. It is also available in Mac Intosh version, but has not been tested on this platform. All graphical applications making plots are done by a dedicated library written in F90. The pack is also delivered with a minimal database, including data files in RFF format used for the tests.

The Windows version includes a Linux emulator, Msys, whose basic configuration has been completed in order to have all the necessary commands. It also contains the gfortran and gcc compilers.

So the use of RPC software does not require any other software or external library. It can be used on any Linux sytem having a F90 compiler, such as gfortran (free GNU product).

2.2 Installation on the target machine

After downloading, the RPC software is installed by simply copying it onto the target machine, Linux or Windows, and wherever you want.

• For windows

There is hardly anything to do. After copying the pack, go to the directory `RPC_V5p2_Msys_i686` and copy / paste the `msys_for_RPC` shortcut on the desktop. A single double-click on this icon will launch a Linux console, where you can work and run RPC commands. A good way to verify that the installation is correct is to run the command:

RPC_menu

There is no interaction with the registry or anything else. To uninstall the software, simply destroy the directory. The software can also be installed on an external drive.

• For Linux

After copying the pack, simply add at the end of the `.bashrc` file the following lines :

```
rc1_pack=/data/Robert_HOME/RPC/RPC_V5p2_Linux_x86_64
export RPC_DIR=$rc1_pack/RPC_V5p2
. $RPC_DIR/RPC_config.bash
```

Of course, the path indicated corresponds to the directory where the pack was installed. After this initialization, all RPC commands are accessible from any directory, for example « `RPC_menu` » that you can type to verify that the path mentioned above is correct. For Ubuntu, the `RPC.desktop` file can be copied to the desktop, double-clicking on it will open a terminal.

2.2.1 Compatibility of executables

The F90 sources were compiled under gfortran on a windows I86-64 platform, and a linux x86-64 platform. To test the compatibility of the binaries, you can run the command:

RPC_dir_rff_pretty_tree

If this command fails, you have to recompile the sources with an F90 compiler, like gfortran, or another one you would prefer, and therefore edit the makefile

2.2.2 Update the Makefile

For a general recompilation of source codes, or when modifying programs, or to use a more efficient compiler, you must update the Makefile:

• Choosing compiler

The user compilers and their options are already predefined in the makefile . If you want to use another compiler, you must indicate it here.

```
# =====
# Choice of compiler
# =====

FC = gfortran
```

• Option debug or optimized

For the used compilers, you can choose a compilation with the option « debug », which will output a more complete diagnosis in the event of a crash, thus making it possible to quickly find the error. This option is recommended during development or when modifying codes. When we go into production, we can choose the option « optimize », which is much faster at runtime :

```
# =====
# optimize or debug mode
# =====

MODE=O
#MODE=D
```

• Compilation Options

The default compiler is GNU gfortran. The compilation options are important, and if you decide to change of compiler, you must find the equivalent options.

```
# =====
# Choice of compilation options
# =====

# gfortran linux and Windows/MinGW compiler

ifeq ($(FC),gfortran)
  ifeq ($(MODE),D)
    FCOMP = -c -g -static -Wall -fbacktrace -finit-local-zero -ffpe-trap=invalid,
            zero,overflow,underflow,denormal -fstack-arrays -fpack-arrays -Jmod

    FLINK = -g -static -Wall -fbacktrace -finit-local-zero -ffpe-trap=invalid,
            zero,overflow,underflow,denormal -fstack-arrays -fpack-arrays -Jmod

  else
    FCOMP = -c -O2 -static -Wall -fbacktrace -finit-local-zero -ffpe-trap=invalid,
            zero,overflow,underflow,denormal -fstack-arrays -fpack-arrays -Jmod

    FLINK = -O2 -static -Wall -fbacktrace -finit-local-zero -ffpe-trap=invalid,
            zero,overflow,underflow,denormal -fstack-arrays -fpack-arrays -Jmod

  endif
endif
```

• Make clean options

For the make clean, the actions are classic. You must run this command when changing machines and all binaries, objects and executables are erased. We can then do the make all, which recompiles everything. If you want make clean to erase other files, you can indicate it in this section.

```
# =====
# make clean
# =====

# Delete all .exe , .o and .mod
# Delete all.tmp, *.core, *~

clean :
@echo "*** Removing *.exe *.O, .mod files..."

@find ./bin -name "*.exe" -exec rm -f {} \;
@find ./obj -name "*.o" -exec rm -f {} \;
@find ./mod -name "*.mod" -exec rm -f {} \;

@find . -name ".tmp" -exec rm -f {} \;
@find . -name "*core" -exec rm -f {} \;
@find . -name "*~" -exec rm -f {} \;

@echo "clean done."
```

2.2.3 Source compilation

Just run the command « make clean » which will delete all the executables in the bin directory, then « make all » which will produce all the executables for the processor (s) of the target machine, i.e. :

```
make clean
make all
```

2.2.4 Setting the environment variables

The machine must be configured in sh or in bash. To find out which shell is used, just type echo \$SHELL.

The setting of the environment variables is done according to the preferences you want. You have to edit the RPC file RPC_config.bash, whose part to be modified is as follows:

```
#!/bin/bash
# =====
#
# Object: set env. variables to use RPC commands everywhere.
# usage : Linux sh or bash : . RPC_config.bash
# Update variables if you change data directories.
#
# Window: nothing to do
# Linux:
# file $HOME/.bashrc file must contains the location of the RPC directory as:
# rpc_pack=/data/Robert_HOME/Copy_TOSH_EXT/Appli_bibi/RPC/RPC_V5p2_Linux_x86_64
# export RPC_DIR=$rpc_pack/RPC_V5p2
# . $RPC_DIR/RPC_config.bash
#
# Patrick ROBERT, CNRS/LPP-ScientiDev, Jan 2021.
#
# =====

# xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
# These options must be defined by the user
# xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
# Visualization option
R_VISU_PDF=yes
R_VISU_PNG=yes_300_256 ; # could be yes_300_16m
R_VISU_SPLASH=yes

# PATH OF CLUSTER DATA (CEF) :
R_CEF_data_CLU=$rpc_pack/CEF_database_CLUSTER

# PATH OF CLUSTER DATA (RFF) :
R_RFF_data_CLU=$rpc_pack/RFF_database_CLUSTER

# PATH OF GEOS DATA (RFF) :
R_RFF_data_GEO=$rpc_pack/RFF_database_GEOS

# xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
# do not change following commands
# xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

R_RFF_data_clu: path of the local data directory CLUSTER/RFF (to be created)

R_RFF_data_geo: path of the local data directory GEOS/RFF (to be created)

R_RFF_data_example: path of the local data directory Example/RFF (to be created)

The databases are located by default at the package directory level, just above the RPC directory `RPC_V5p2`. If you want to move these databases elsewhere, or to another disk, you have to update these variables.

R_VISU_PDF=yes to output PDF from PostScripts after running a `RPC_visu`.

R_VISU_PNG= yes_300_256 to output 300 dpi resolution PNG, in 256 colors, along with PostScripts after command execution `RPC_visu`.

R_VISU_SPLASH=yes if you want the pdf to be displayed automatically on the screen after executing a command `RPC_visu`. This variable must be set to "no" for an evaluation. The display on the screen is done by the software "acroread" or "evince". Under Linux, and by the command "start" under Windows.

2.2.5 Automatic execution of the `RPC_config.bash`

As we saw during the installation under unix, this file is executed each time a terminal is opened. So, under Linux, don't forget to declare its location in the file `$HOME/.bashrc` by adding the following lines:

```
rcl_pack=/data/Robert_HOME/RPC/RPC_V5p2_Linux_x86_64
export RPC_DIR=$rcl_pack/RPC_V5p2
. $RPC_DIR/RPC_config.bash
```

Of course, the path indicated corresponds to the directory where the pack was installed. Under windows, it is executed automatically by the Msys launcher, which executes the run file `run_RPC_config.bash`, who himself launches the `RPC_config.bash`.

2.2.6 `Make_bin_pack.bash`

This will create a directory in the current directory **`RPC_V5p2_Linux_x86_64_bin`** if the machine is a `Linux_x86_64` that can be transported on a machine of the same type. This procedure uses the Unix command `uname -sm` to identify the platform.

Before creating the binary pack, this procedure cleans the directory by performing a `make clean`, and a `make all` which recompiles everything. It will then execute the command in the tests directories **`clean_all_tests.sh`** which will have the effect of erasing any results in the test directory. Before starting this procedure, you must check that the Makefile is configured in mode « optimized » and not in mode « debug » so that the binary pack is optimized for production.

2.2.7 Package tree

After installation the RPC directory RPC_V5p2 looks like this :

```

|_bash
|_bin
|_doc_____
|           |_src_doc
|_gainantset
|_mod
|_obj
|_scripts___
|           |_scripts_CLUSTER
|           |_scripts_generic
|           |_scripts_GEOGRD
|           |_scripts_GEOS
|_src
|_tests_____
|           |_test_compute_curl_div
|           |_test_compute_sat_trajectory
|           |_tests_CLUSTER
|           |_tests_generic_scripts
|           |_tests_GEOGRD
|           |_tests_GEOS
|           |_tests_RPC_dir_____
|                                           |_dummy_dir_1
|                                           |_dummy_dir_2
|_work

```

bash : contains RPC commands
 bin : contains the executable *.exe and the .sav
 doc : contains the documentation, including this one, in pdf, and the sources in word or Latex.
 gainantset : contains the transfer functions of the instruments
 mod : contains F90 *.mod modules
 obj : contains *.obj objects
 scripts : contains scripts for generic commands, CLUSTER and GEOS
 src : contains the sources F90 *.f90
 tests : contains tests of the RPC scripts for generic commands, CLUSTER and GEOS
 work : work directory. you can do what you want inside and put it elsewhere.

In this directory we also find the files:

Makefile, Make_bin_pack.sh, RPC_config.bash, csa_cookie.txt, RPC.desktop, RPC.ico, RPC_V5.2, README.txt, and term_colors.bash

csa_cookie.txt allows downloading CLUSTER data from the CSA. Useless for another mission. ¹

The RPC.desktop file is a shortcut for the launcher on Linux / Ubuntu. It is to be copied to the desktop where it will appear as an RPC icon. Double clicking on it will open a Linux terminal (gnome-terminal). Under Ubuntu it may be necessary to *Right Click - Permissions - Allow the file to run as a program*.

The fileterm_color.bash explains the colors chosen for the terminal.

¹ In the distributed pack, it is in the name of patrick.robert. If you use the download of CLUSTER data extensively, it is preferable to request your own cookie from the CSA (CLUSTER Science Archive).

2.2.8 Installation tests

When the installation is complete, and once a terminal is opened by double-clicking on the RPC icon, the command `RPC_menu` should display the menu of generic RPC commands.

Chapter 3

DATA MANAGEMENT

3.1 Main types of RFF files

We will mainly handle files of type VecTime [3] which are vectors indexed by time, and files of type Spectrogram [3] which are series of dated spectra.

3.1.1 Example of VecTime file names :

```
GEOS1_ULF_VTL2_19770713.rff
GEOS1_MAG_VTL2_19770713.rff
GEOS1_POS_VTL2_19771207.rff
```

And for a new mission/experiment, one can suggest:

```
New-mission_New-Wave-experiment_VTL2_20010321.rff
New-mission_New-Mag-experiment_VTL2_20010321.rff
New-mission_New-Pos-experiment_VTL2_20010321.rff
```

VTL1 for vectime Level 1 (uncalibrated, or T.M. data)

VTL2 for vectime Level 2 (calibrated data).

3.1.2 Example of Spectrograms file names :

The nomenclature is like :

```
GEOS1_ULF_SPL2_19770713.rff
GEOS1_MAG_SPL2_19770713.rff
GEOS1_POS_SPL2_19771207.rff
```

and could be:

```
New-mission_New-Wave-experiment_SPL2_20010321.rff
New-mission_New-Mag-experiment_SPL2_20010321.rff
New-mission_New-Pos-experiment_SPL2_20010321.rff
```

For a file named toto.rff, Note that the viewing commands will create files like :

```
toto.ps      toto.pdf      toto.png
```

3.2 Convert time serie data to RFF

3.2.1 RPC_flat_to_rff command

To use RPC commands, you have to convert your data to the RFF format [3]. To make easy this work, you can use the generic command:

```
RPC_flat_to_rff file.flat
```

An example of flat_file is given in R_RFF_data_example database directory.

The command **RPC_flat_to_rff CAS_FGM_VTL2_20040628.flat**

will create the file **CAS_FGM_VTL2_20040628.rff**

You can apply this command to your data provided that the header of your file must be in the form:

```
# -----
# Comments ZONE
# KRTP : Kronocentric Radial-Theta-Phi Spherical Polar Saturn centered coordinates
# Time: decimal second of the day
# data: Time, Bx, By, Bz
# -----
CASSINI                ! MISSION_NAME
Cassini                ! OBSERVATORY_NAME
1                      ! OBSERVATORY_NUMBER
FGM                    ! EXPERIMENT_NAME
Normal                 ! EXPERIMENT_MODE
Magnetometer           ! INSTRUMENT_TYPE
Magnetic field         ! MEASUREMENT_TYPE
KRTP                   ! DATA_COORDINATE_SYSTEM
Bx ; By ; Bz           ! DATA_LABEL
nT ; nT ; nT           ! DATA_UNITS
3                      ! DATA_DIMENSION
2749035                ! BLOCK_NUMBER
0.0320000              ! TIME_RESOLUTION
2004-06-28T00:00:01.000Z ! BLOCK_FIRST_INDEX
# -----
1.00000002e-001  1.16000000e-001  -1.52000000e-001  1.60000001e-002
1.31000004e-001  1.39000000e-001  -1.23000000e-001  1.89999999e-002
1.61999994e-001  1.07000000e-001  -1.86000000e-001  2.00000000e-002
```

The following data being: Time (second of the day), Bx, By, Bz

The comment zone can have any number lines, provided that the first character be '#'.

The example is given for the magnetometer of CASSINI mission (thank to Lina Hadid for provided the data) and you just have to put the right fields for your data to create your RFF file.

After that, check the validity of your file as in next section, and you can use after that any RPC commands.

In section 23.2 an example of spectrogram plot is given on figure 23.3 and 23.4.

3.2.2 Check the validity of the new file

Put your code in the src directory, add the instructions into the Makefile, compile and run your program. The file 'VTL2_your_data.rff' will be created, you just have now to check it by:

```
RPC_check_rff VTL2_your_data.rff
```

and for instance plot it by:

```
RPC_visu_vectime VTL2_your_data.rff 0 0
```

3.3 Choose RFF file names

3.3.1 Example of GEOS RFF file names

For instance, the downloaded GEOS files are of the following type:

```
GEOS/ULF VTL2 : GEOS1_ULF_VTL2_19770713.rff
GEOS/MAG      : GEOS1_MAG_VTL2_19770713.rff
GEOS/POS      : GEOS1_POS_VTL2_19771207.rff
```


3.3.2 Example of CLUSTER RFF file names

For Cluster, the downloaded GEOS files are of the following type:

CLUSTER/STAFF VTL1 et VTL2:

```
CLU3_STASC_VTL1_NBR_20010110_000000_20010110_020000.rff
CLU3_STASC_VTL2_NBR_ISR2_20010110_000000_20010110_020000.rff
CLU3_STASC_VTL2_NBR_GSE_20010110_000000_20010110_020000.rff
```

CLUSTER/FGM 5VPS:

```
CLU3_FGM_VTL2_5VPS_20010110_000000_20010110_020000.rff
```

CLUSTER/POS in GSE:

```
CLU3_AUX_VTL2_POS_20010110_000000_20010110_235959.rff
```

3.3.3 Choose your RFF file names

It is recommended to carefully choose a nomenclature for the file names of your RFF data

Indeed, this convention will be useful for the management of a possible database.

3.4 Setting up a local database

It is recommended to make a database of the files in RFF format used by the RPC commands.

The RPC package comes with a minimal database, just containing samples for testing. These directories can be anywhere on your machine, even on a different disk or on the network as long as its path is correctly indicated in the RPC `RPC_config.bash`.

You just have to copy them into the report, and then the sub-directories will be automatically created when filling with the command **RPC_put_rff_database**.

A database such as GEOS database is organized as follows :

```
_MAG
|_1977
|   |_1977_05
|   |_1977_07
|   |_1977_12
|_1978
|   |_1978_01
|   |_1978_05
|   |_1978_08
_POS
|_1977
|   |_1977_07
|   |_1977_12
|_1978
|   |_1978_01
|   |_1978_08
_ULF
|_VTL1
|   |_1977_____|_1977_12
|_VTL2
|   |_1977_____|_1977_05
|   |           |_1977_07
|   |           |_1977_12
|   |_1978_____|_1978_01
|   |           |_1978_05
|   |           |_1978_08
|   |_1980_____|_1980_06
```

3.5 Interest of a local database

RPC processing commands request one or more RFF files as arguments. These files can be in the current directory or elsewhere, if the path is correctly entered there is no problem. But if you are working on many files, it is more pleasant to store them directly in the local database. First, they can be easily found with the commands described in the next chapter, and then this database can be in another directory, a disk« data » for example.

The local database is also used for access to two simultaneous experiments, for example when we want to plot the gyrofrequency of protons (from MAG data) or the position of satellites (from POS) on a spectrogram of ULF data for example .

It is essential for all production orders (which are not detailed here, see [1]).

3.6 Commands relating to the database

3.6.1 Put files into the database

When you have a RFF file, obtained by download or made by yourself, it is convenient to put it in a local database. There 3 commands to manage database:

RPC_put_rff_database name.rff

This command *move* the RFF data files in the local data base

RPC_copy_rff_database name.rff

This command *copy* the RFF files in the local data base, and not move it.

These two commands work on GEOS and CLUSTER files. They were written according to the nomenclature of the names of the corresponding files, which is known and fixed. If you want these commands to work with files from other missions/experiments, *you will need to modify these procedures by adding the necessary information.*

3.6.2 Extract files from the database

As before, *you must write a procedure for extracting your data* according to the criteria of RFF file names. For example, for GEOS, we have the following commands, which *copy* a file from the GEOS database into the current directory:

RPC_get_data_GEOULF Level SatNum year month day

Level : VTL1 or VTL2

RPC_get_data_GEOMAG SatNum year month day

RPC_get_data_GEOPOS SatNum year month day

You can copy this procedures and modify it to build your own procedure for your data.

3.6.3 Files list for a given database

If we want to know what is in the RFF database, *the list of all RFF files* by experience is obtained by the commands :

RPC_list_rff_database CLUSTER STA

RPC_list_rff_database CLUSTER FGM

RPC_list_rff_database GEOS ULF

RPC_list_rff_database GEOS MAG

or

RPC_list_rff_database GEOS ALL

So, to list your files, *you must modify this procedure* according to the criteria of new RFF and new database.

Part II

APPLICATION TO SIMULATED DATA

Chapter 4

MAKE SIMULATED DATA

4.1 Create ULF simulated data

4.1.1 RPC_create_simulated_data ULF

This command create the VecTime file **VTL2_simulated_ULF_data.rff** for a mission 'IMAG-INE' and an experiment 'SIM_ULF'. This file contain a simulated waveform of a monochromatic wave in GSE system with following characteristics:

- $\vec{K} = (0.5, 0.5, 1)$
- $a = 10nT, b = 9nT$
- Frequency is linearly variing from 3. to 8 Hz.
- Sample frequency is 25 Hz ($\delta t = 0.04s$).
- The time duration of the signal is 3 hours, and the intensity is modulated
- A randow noise between 0. and 0.02 nT is added.

4.1.2 Spectrogram plot of simulated ULF waveform

This is simply done by the generic command:

RPC_vectime_to_spectro VTL2_simulated_ULF_data.rff SPL2_simulated_ULF_data.rff 512 512 t

and the plot is produced by:

RPC_visu_spectro SPL2_simulated_ULF_data.rff 0 0 0. 0. 0. 0. XY 0.5 12. n

Figure 4.1 show the produced plot.

4.2 Create MAG simulated data

4.2.1 RPC_create_simulated_data MAG

This command create the VecTime file **VTL2_simulated_MAG_data.rff** for the 'IMAGINE' mission and experiment 'SIM_MAG'. This file contain a linear variation of a DC magnetic field as:

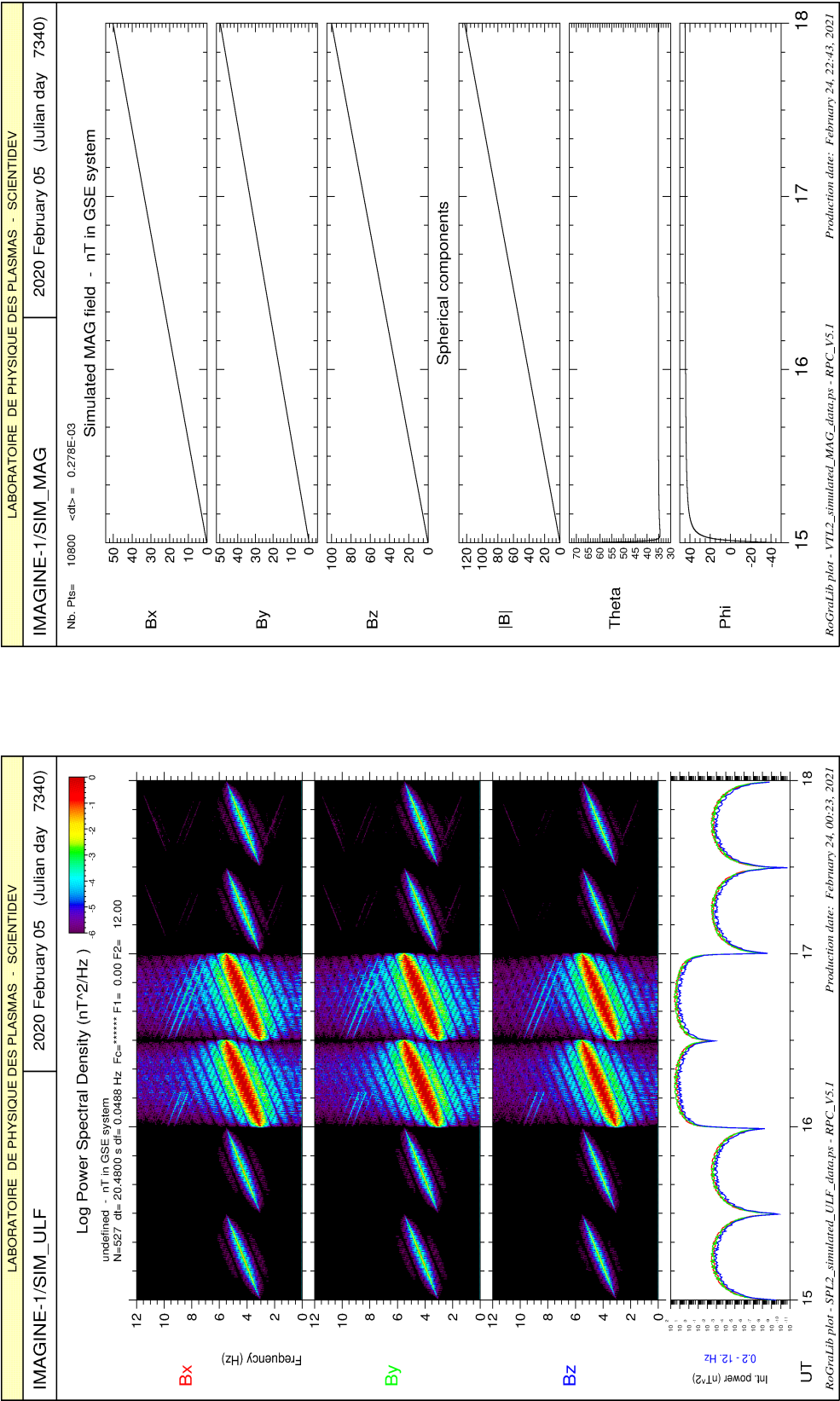
- $x = 50.(t - tz)$
- $y = 50.(t - tz)$
- $z = 100.(t - tz)$
- Sample frequency is 1 Hz ($\delta t = 1s$).
- A randow noise between 0. and 1 nT is added.

4.2.2 Waveform plot of MAG data

This is simply done by the generic command:

RPC_visu_vectime VTL2_simulated_ULF_data.rff VTL2_simulated_MAG_data.rff 0 0

Figure 4.2 show the produced plot.



LABORATOIRE DE PHYSIQUE DES PLASMAS - SCIENTIDEV

2020 February 05 (Julian day 7340)

IMAGINE-1/SIM_MAG

Nb. Pls= 10800 <dt>= 0.278E-03

Simulated MAG field - nT in GSE system

50

40

30

20

10

0

Bx

50

40

30

20

10

0

By

100

80

60

40

20

0

Bz

120

100

80

60

40

20

0

|B|

70

65

60

55

50

45

40

35

30

Theta

40

20

0

-20

-40

Phi

15

16

17

18

RoGridLib plot - VTL2_simulated_MAG_data.ps - RPC_V5.1

Production date: February 24, 22:43, 2021

Fig. 4.1: Spectrogram of simulated ULF waves

Fig. 4.2: Simulated DC magnetic field

4.3 Computation of Curl(B) on simulated data

4.3.1 Compute MAG and POS simulated data for a current tube

To do this, we use the command **RPC_current_tube**, which create a set of 4 positions files, and 4 magnetic data files, containing simulated data corresponding to the crossing of a current tube by a cluster of 4 spacecrafts.

Use of this command is above:

```
RPC_current_tube Rtube Jx Jy Jz Scale Impactparam Vx
```

Example:

```
RPC_current_tube 2000. 0. 0. 0.25 300. 0.7 10.
```

This command create the following files:

```
SC1_MAG.rff    SC1_POS.rff
SC2_MAG.rff    SC2_POS.rff
SC3_MAG.rff    SC3_POS.rff
SC4_MAG.rff    SC4_POS.rff
```

4.3.2 Computation of Curl(B) and div(B) on simulated data

The command **RPC_compute_curl_div_4sat** has been developed for CLUSTER/FGM data but can be used for any experiment with 4 spacecraft. In order to check the computation done by this procedure, it s necessary to apply this command to simulated data, where we know what we are going to found.

This is done by the following command where we put the simulated files:

```
RPC_compute_curl_div_4sat SC1_MAG.rff SC2_MAG.rff SC3_MAG.rff SC4_MAG.rff
                           SC1_POS.rff SC2_POS.rff SC3_POS.rff SC4_POS.rff
```

This command use the Ampere's law to compute Curl(B) as:

$$\oint \vec{B}(M) \cdot d\vec{l} = \mu_0 \cdot I$$

And to compute div(B) we use the divergence law, or Green-Ostrogradski law, as:

$$\iiint_V \vec{\nabla} \cdot \vec{B} \, dV = \oint_{\partial V} \vec{B} \cdot d\vec{S}$$

with $\mu_0 = 4\pi \times 10^{-7} T \cdot m/A$

This command create a file **compute_curl_div_4sat.resu** which contains all data about the fields, curl, div etc.

Computation is detailed in chapter [22](#)

4.3.3 Visualization of simulated data

We use the following commands:

```
RPC_visu_curl_div_4sat datiso1 datiso2
```

This command read the file **compute_curl_div_4sat.resu** and produce the plots shown in following figures.

If we set datiso1 and datiso2 to 0 0, all data in the file will be displayed.

4.3.3.1 Result with a tetrahedron of large size

On figure 4.3 we clearly see the effect of the size of the tetrahedron on magnetic waveforms (300km), while we can see on figure 4.4 that the effect of no linearity of \vec{B} is strongly visible on the estimate of $\text{curl}(\vec{B})$.

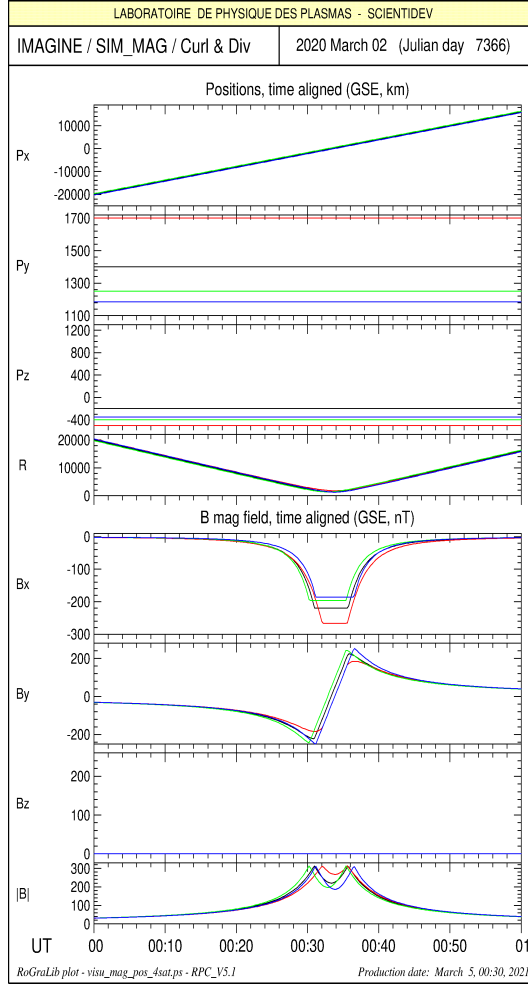


FIG. 4.3: Simulated positions and DC magnetic field for a tetrahedron of large size crossing a current tube

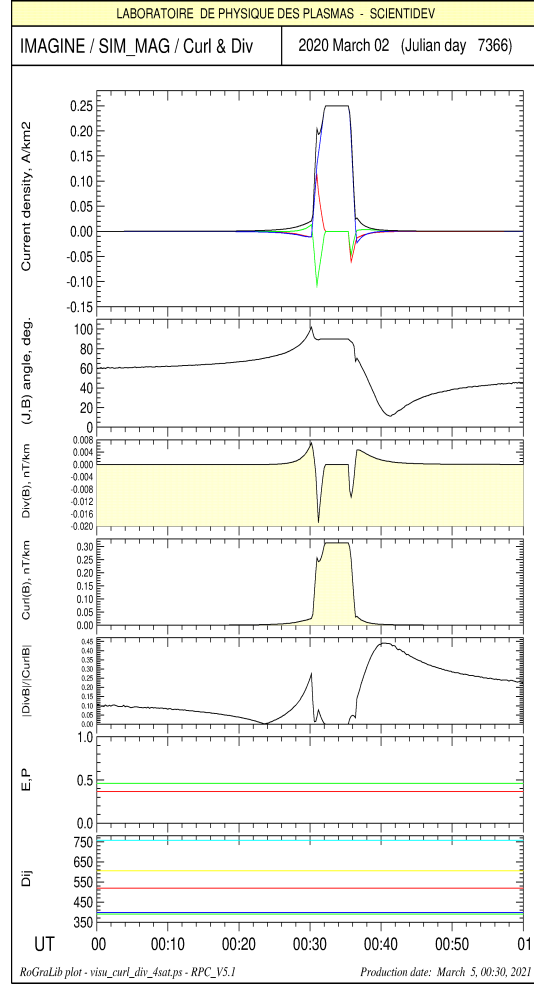


FIG. 4.4: $\text{Curl}(\vec{B})$, $\text{Div}(\vec{B})$ and others parameters

4.3.3.2 Result with a tetrahedron of small size

On the contrary, we can see on figure 4.5 that for a very small size of the tetrahedron (3 km, very low by respect to the 200 km od radius tube) the estimate of \vec{J} is perfect, equal to the value of the model, as well as the divergence (close to zero).

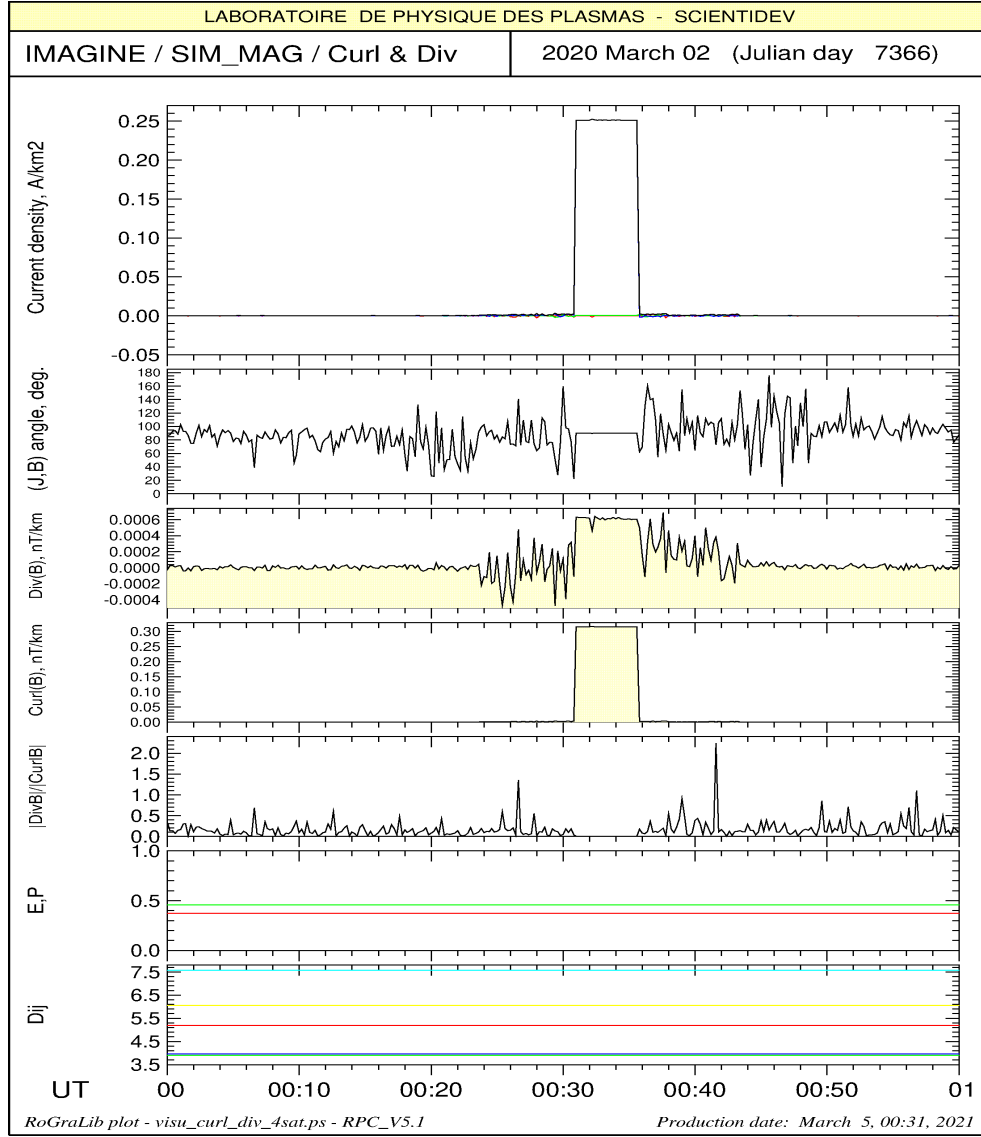


FIG. 4.5: $\text{Curl}(\mathbf{B})$ and $\text{Div}(\mathbf{B})$ computation for a tetrahedron of small size crossing a current tube. Note that for a very small scale value of the simulated cluster tetrahedron (3km) the computations are perfectly right.

4.4 Create trajectory simulated data

The command **RPC_create_simulated_data POS** create the VecTime file `VTL2_simulated_POS_data.rff` for a mission 'IMAGINE' and an experiment 'SIM_POS'. This file contain a simulated elliptic trajectory in GSE system with following characteristics:

- Exccentricity: 0.7
- semi major axis of the elliptical trajectory: 80000 km
- direction of the semi-major axis in GEI: (1, 0, 0)
- direction of the semi-minor axis in GEI: (0, 1, 0)
- Earth mass: 5.9736×10^{24} kg

The trajectory is computed at the starting date of 2020-02-05T15:00:00.000000Z taken as the perigee time and according the Kepler's law, which leads to a time revolution of 62.5 hours.

4.4.1 Trajectory plot

This is simply done by : `RPC_visu_vectime_3D VTL2_simulated_POS_data.rff 0 0`
to get the plot of figure 4.6

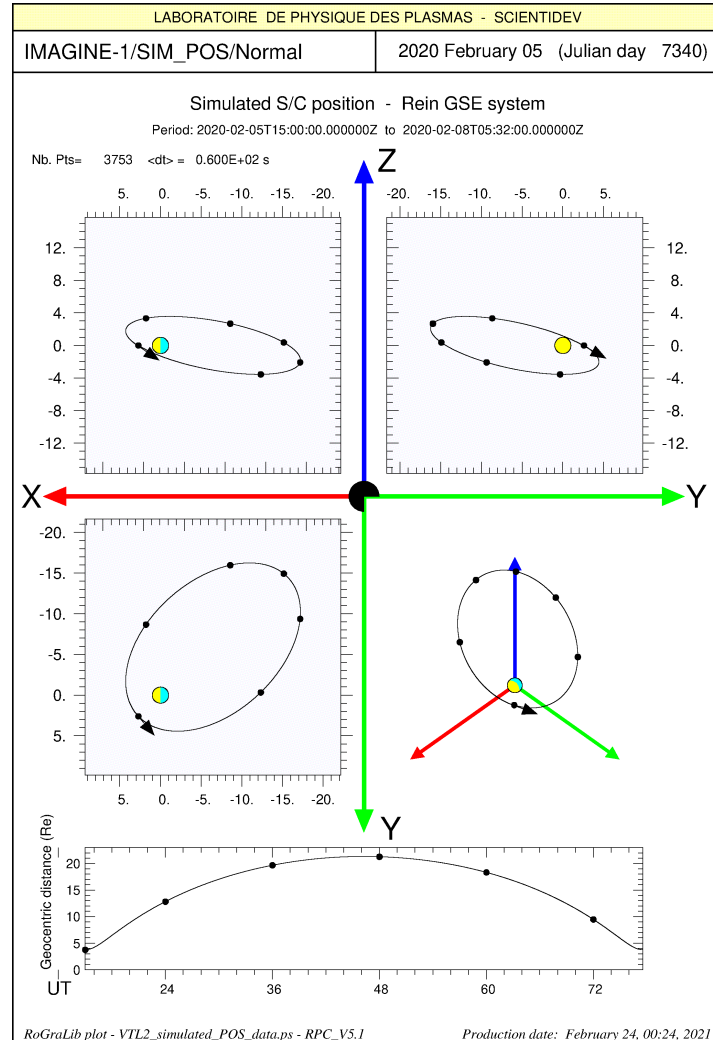


Fig. 4.6: Simulated positions in GSE system

4.5 Using generic RPC commands

All generic RPC commands can now be used on simulated data. The most used are:

<code>RPC_visu_vectime</code>	→ for VT ULF and VT MAG waveform files, or VT POS files.
<code>RPC_vectime_to_spectro</code>	→ for VT ULF, providing SP files.
<code>RPC_visu_spectro</code>	→ for SP ULF
<code>RPC_visu_ave_spectrum</code>	→ for SP ULF
<code>RPC_spectro_to_polar</code>	→ for SP ULF files, providing copolar.resu file.
<code>RPC_visu_polar</code>	→ for copolar.resu file.
<code>RPC_reduce_time_vecvtime</code>	→ for VT ULF, MAG ans POS.
<code>RPC_change_coordinate_system</code>	→ for VT ULF, MAG ans POS.
<code>RPC_visu_vectime_3D</code>	→ for VT POS file

Of course, all other generic RPC commands are available.

Chapter 5

USING SCRIPTS FOR EFFICIENT PROCESSING

Scripts are bash files that string together multiple RPC commands, to avoid the tedious task of typing them all one after the other. The RPC package ships with a number of scripts that perform the most common processing. The user can use this as inspiration to write their own treatments. Each script has its test program, explaining its use.

The directory 'scripts' contains scripts that work on simulated data. They can also be used as an example; their execution allows on the one hand to test the commands used, and on the other hand to provide new information on the commands of data production.

The script directory contains also a lot of scripts for the GEOS and CLUSTER mission. You can read them and use them to write your own scripts.

5.1 do_simulated_data_processing.sh

This script run the command **RPC_create_simulated_data**, wich produce the following files:

```
VTL2_simulated_MAG_data.rff
VTL2_simulated_POS_data.rff
VTL2_simulated_ULF_data.rff
```

Then, the scrip use the following commands to plot vectime data, compute and plot spectrograms and average spectra, as shown in chapter 4.

```
RPC_visu_vectime          RPC_visu_spectro
RPC_visu_vectime_3D       RPC_visu_ave_spectrum
RPC_vectime_to_spectro
```

5.2 do_simulated_data_MVA_plot.sh

As above, this script first run the command **RPC_create_simulated_data**, wich produce among others the following file:

```
VTL2_simulated_ULF_data.rff
```

Then, the scrip use the following commands to plot vectime data, compute and plot data in the *Minimum Variance Analysis* system, as shown in following figures.

```
RPC_vectime_to_spectro    RPC_visu_spectro
RPC_visu_ave_spectrum
```

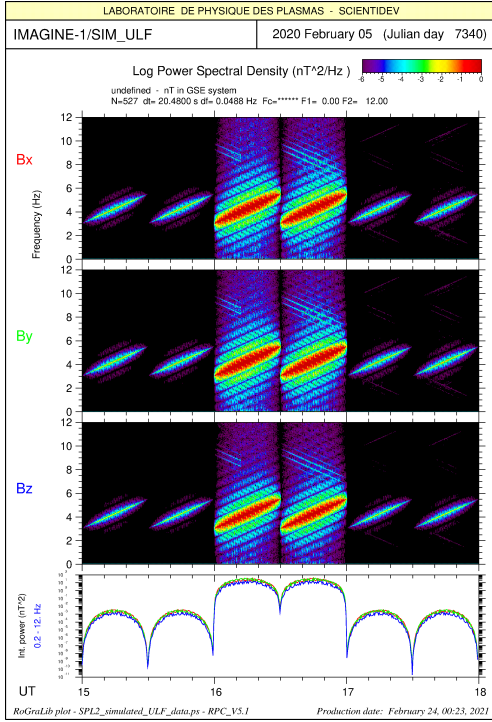


FIG. 5.1: Spectrogram of simulated ULF waves in GSE system

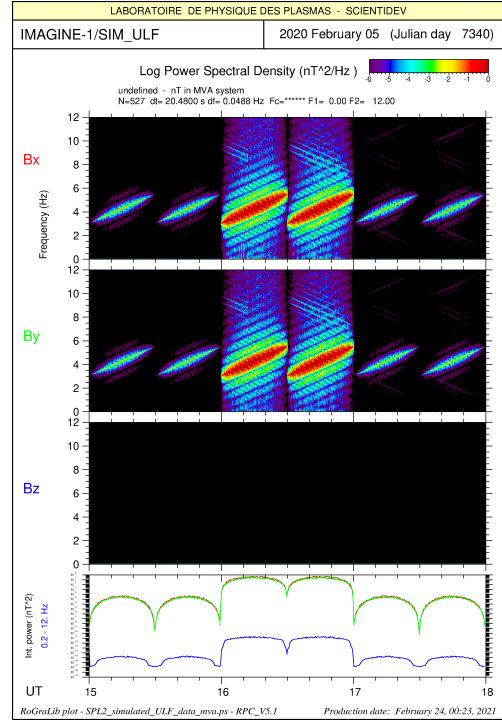


FIG. 5.3: Spectrogram of simulated ULF waves in MVA system

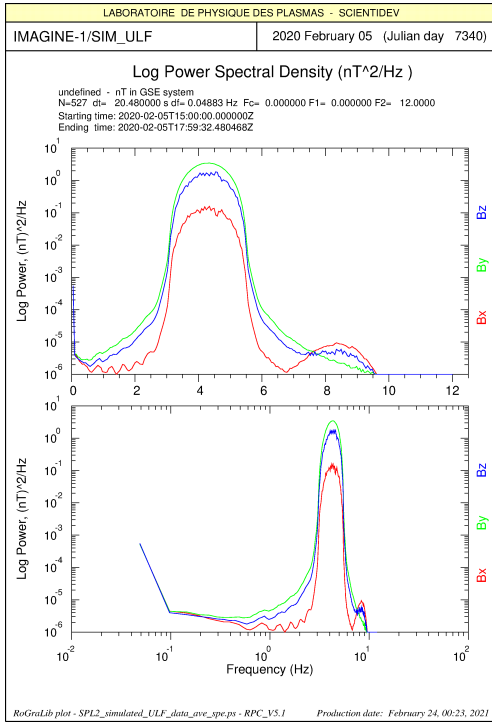


FIG. 5.2: Average spectrum of simulated ULF waves in GSE

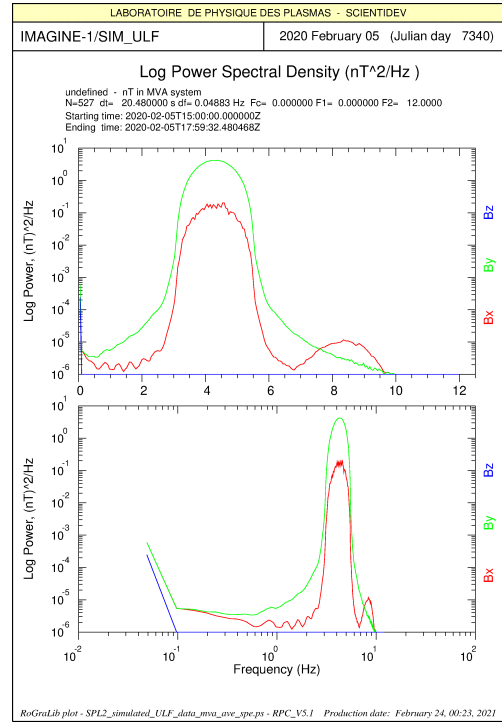


FIG. 5.4: Average spectrum of simulated ULF waves in MVA

5.3 do_simulated_data_Polar_plot.sh

5.3.1 Produce simulated data in GSE

As above, this script first run the command `RPC_create_simulated_data`, wich produce among others the following files:

`VTL2_simulated_ULF_data.rff`

5.3.2 Compute spectrogram in MFA system

Then, the scrip use the following commands to compute spectrogram of simulated ULF data:

```
RPC_vectime_to_mfa
RPC_vectime_to_spectro
RPC_visu_spectro
RPC_visu_ave_spectrum
```

The spectrogram plot in MFA system give already some useful information, as shown in chapter 7.

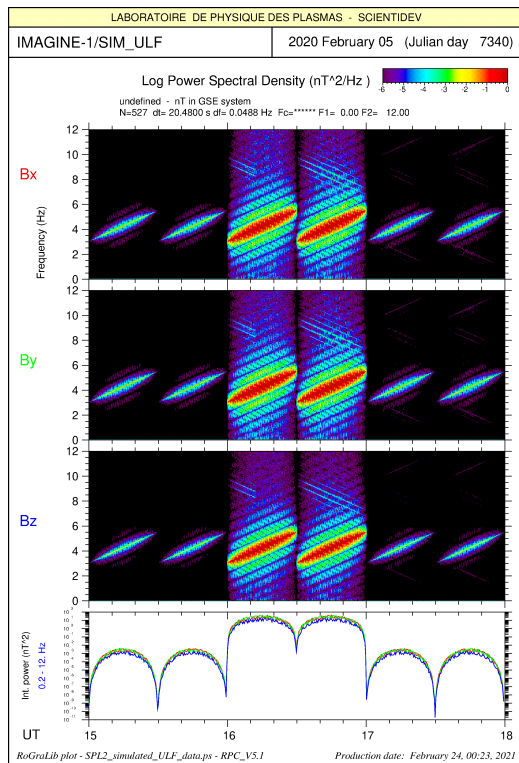


FIG. 5.5: Spectrogram of simulated ULF waves in GSE

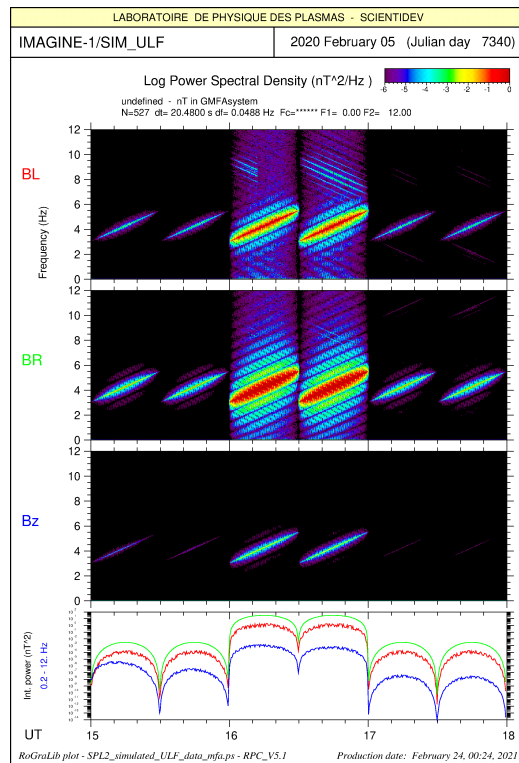


FIG. 5.6: Spectrogram of simulated ULF waves in MFA

5.3.3 Compute and plot polarisation parameters

At last, the script use the following command to compute and plot polarization parameters:

```
RPC_spectro_to_polar
RPC_visu_polar
```

All results are shown in figure 5.7.

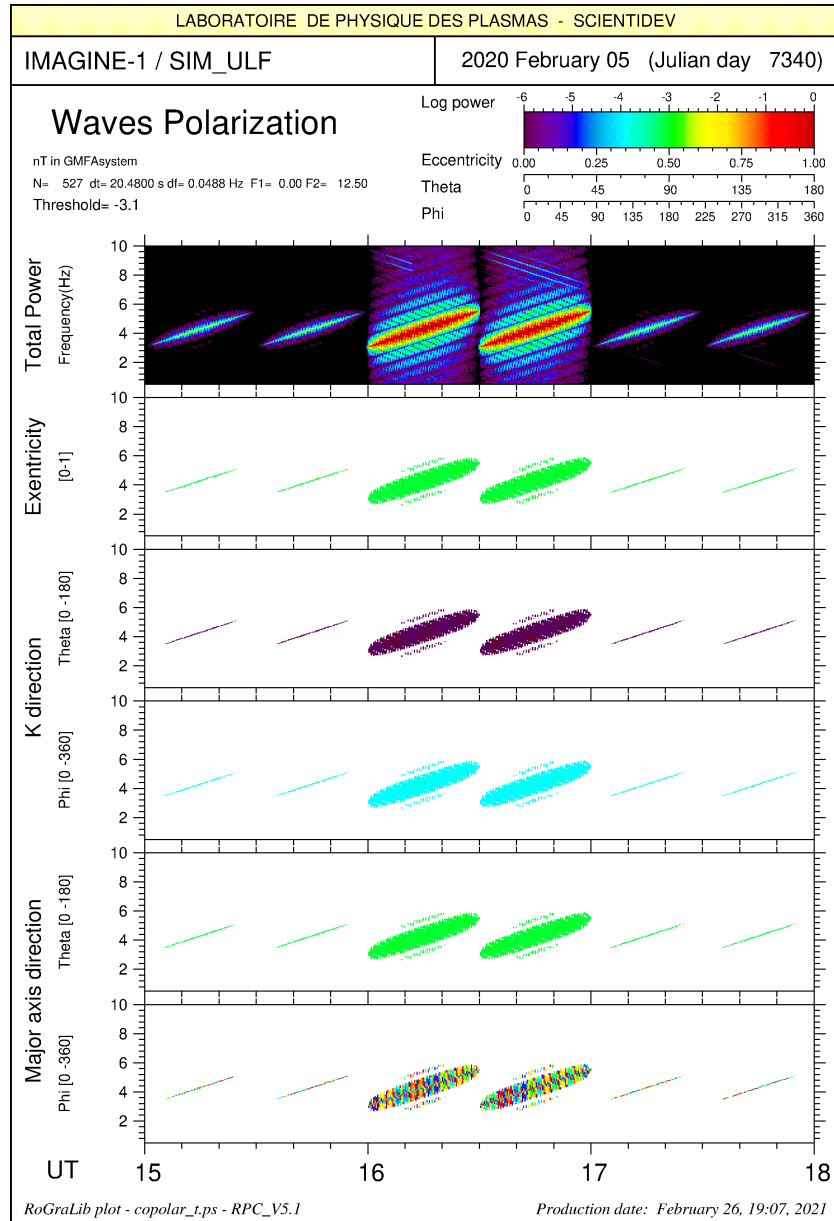


FIG. 5.7: Polarisation parameters of simulated ULF waves in MFA

5.3.4 Interpretation of polarization parameters

- eccentricity : green, so ~ 0.5 , \Rightarrow circular polarization.
- theta K : dark purple, so $\sim 0^\circ$, \Rightarrow **right circular polarization with K parallel to B_0 .**
- theta Major axis : green, so $\sim 90^\circ$, \Rightarrow perpendicular to B_0 , consistent with the direction of K .

Therefore one found again the initial parameters defining the wave model described in chapter 4 and reminded here:

- $\vec{K} = (0.5, 0.5, 1)$
- $\vec{B} = ([50.(t - t_z), 50.(t - t_z), 100.(t - t_z)]) \Rightarrow \vec{K} // \vec{B}$
- $a = 10nT, b = 9nT \Rightarrow e = \frac{\sqrt{a^2 + b^2}}{a} = 0.43$
- Frequency is linearly varying from 3. to 8 Hz.

5.4 do_simulated_orbit.sh

This rather complex script is not detailed here; it can be viewed in the scripts directory. Three steps are done, detailed below.

5.4.1 Produce simulated trajectory

First, the script run the command **RPC_create_simulated_data**, which produce the file:

VTL2_simulated_POS_data.rff

which is plotted via the command **RPC_visu_vectime_3D**

Result has been shown previously in figure 4.6

5.4.2 Compute orbital parameters

The file **VTL2_simulated_POS_data.rff** is passed in GEI system by the command:

RPC_change_coordinate_system VTL2_simulated_POS_data.rff gei VTL2_simulated_POS_data_gei.rff

Then it run:

RPC_compute_sat_orbit_param VTL2_simulated_POS_data_gei.rff

and the results are given in the file **compute_sat_orbit_param.out** as:

```
results, in this order:
-----
Apogee
Perigee

date where S/ is at perigee

direction of semi-major axis, in GEI system, MVA method
(from ellipse center to perigee, unit vector)

direction of semi-minor axis, in GEI system, MVA method
(a,b vector in the sens of the motion, unit vector)

a,b from minmax method
136000.078
23999.9668
2020-02-05T15:00:00.000000Z
1.000000000      5.36441860E-07      0.000000000
-6.55651093E-07      1.000000000      0.000000000
-0.999999583      9.09384398E-04      1.00014397E-07
-9.09384398E-04 -0.999999583      0.000000000

compute_sat_orbit_param.exe      : NORMAL TERMINATION
>
```

5.4.3 Compute new theoretical trajectory

From the estimate of orbital parameters, we can recompute the theoretical trajectory by:

RPC_compute_sat_trajectory with required arguments...

The trajectory is passed from GEI to GSE by:

RPC_change_coordinate_system VTL2_simulated_POS_data_recomputed_gei.rff gse VTL2_simulated_POS_data_recomputed_gse.rff

And finally we can compare the original trajectory with the recomputed trajectory by:

RPC_visu_vectime_3D_4sat file1 file2 file3 file4

results are compared with the original orbit in fig 5.8 above: in black, the original simulated orbit, in red the recomputed one.

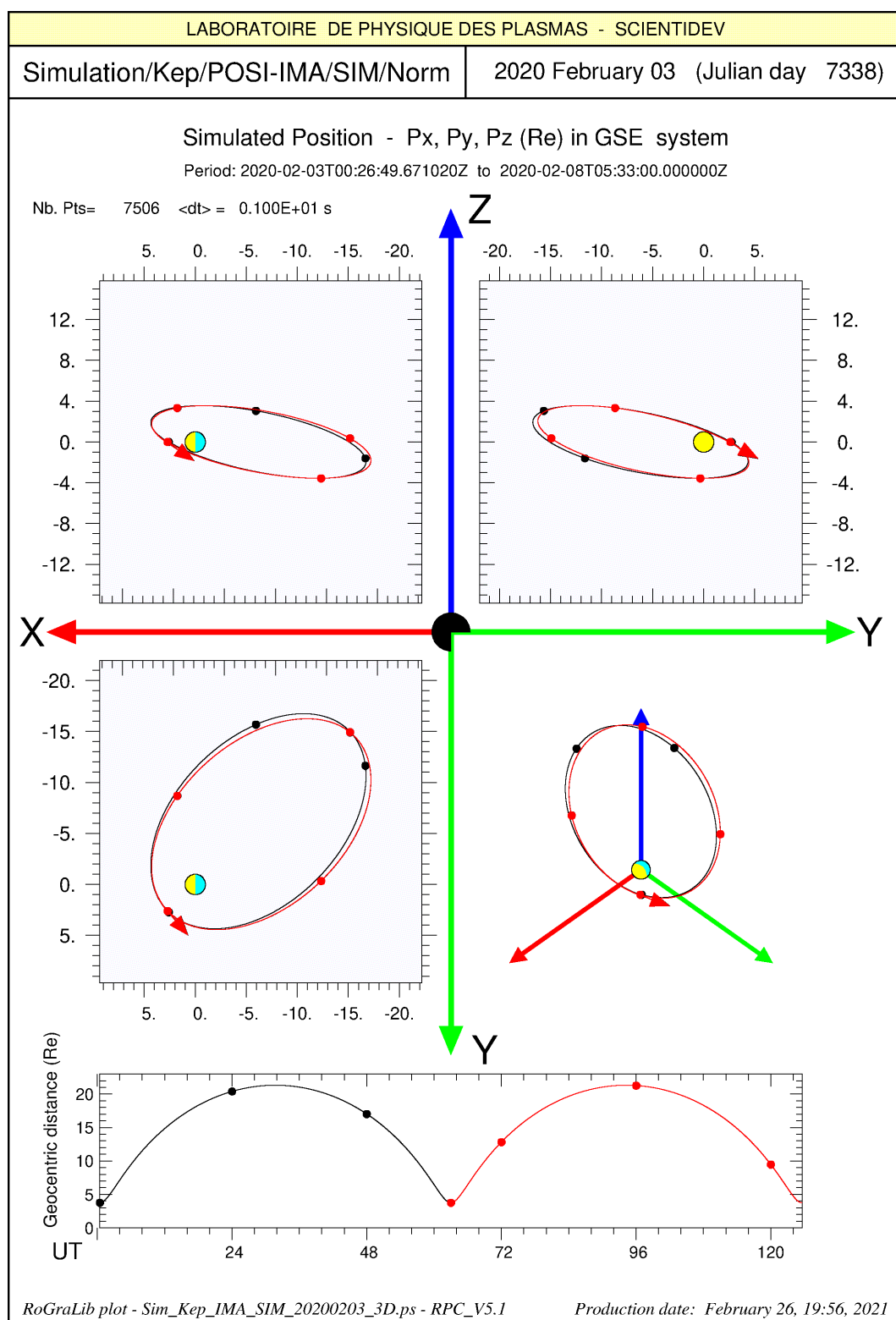


FIG. 5.8: Comparison with the original trajectory and the trajectory recomputed from the estimate orbital parameters of the first trajectory. During first orbit revolution, curves are perfectly superimposed; The red curve correspond to the computation of a second revolution, so is slightly moving in GSE system.

Chapter 6

TESTS

The tests of the RPC commands are made from the scripts, which, as we have seen, successively launch several RPC commands for a particular treatment. For each script it was made a test, which are summarized here.

Only tests on simulated data are shown in this section. In part III and IV of this document, various tests on real data as GEOS and CLUSTER mission will be viewed in details.

6.1 Test compute spacecraft trajectory

First, go to the directory `test/test_compute_sat_trajectory`. Then, run the command:

```
run_test_demo_with_log.sh
```

The results will be displayed on the screen. Only the *stderr* will be displayed on the screen, in the form given below, which allows you to verify that all the tests have been executed correctly.

```
=====
Tests of RPC_compute_sat_trajectory
=====

RPC_compute_sat_trajectory
-----
STOP COMPUTE_SAT_TRAJECTORY.exe      : NORMAL TERMINATION
RPC_compute_sat_trajectory           : NORMAL TERMINATION - time exe= 2 s.
-----

RPC_visu_vectime_3D
-----
STOP visu_vectime_3D.exe             : NORMAL TERMINATION
RPC_visu_vectime_3D                 : NORMAL TERMINATION - time exe= 0 s.
-----

RPC_change_coordinate_system
-----
STOP change_coordinate_system.exe    : NORMAL TERMINATION
RPC_change_coordinate_system         : NORMAL TERMINATION - time exe= 0 s.
-----

RPC_visu_vectime_3D
-----
STOP visu_vectime_3D.exe             : NORMAL TERMINATION
RPC_visu_vectime_3D                 : NORMAL TERMINATION - time exe= 0 s.
-----

=====
Tests of do_simulated_orbit_CLUPOS.sh
=====

do_simulated_orbit_CLUPOS.sh
-----
```

```

RPC_get_data_CLUPOS      : NORMAL TERMINATION - time exe= 0 s.
RPC_get_data_CLUPOS      : NORMAL TERMINATION - time exe= 0 s.
RPC_get_data_CLUPOS      : NORMAL TERMINATION - time exe= 0 s.
RPC_join_vectime         : NORMAL TERMINATION - time exe= 0 s.
RPC_join_vectime         : NORMAL TERMINATION - time exe= 0 s.
STOP visu_vectime.exe     : NORMAL TERMINATION
RPC_visu_vectime         : NORMAL TERMINATION - time exe= 0 s.
STOP change_coordinate_system.exe : NORMAL TERMINATION
RPC_change_coordinate_system : NORMAL TERMINATION - time exe= 0 s.
STOP visu_vectime_3D.exe  : NORMAL TERMINATION
RPC_visu_vectime_3D      : NORMAL TERMINATION - time exe= 1 s.
STOP visu_vectime_3D.exe  : NORMAL TERMINATION
RPC_visu_vectime_3D      : NORMAL TERMINATION - time exe= 0 s.
STOP compute_sat_orbit_param.exe : NORMAL TERMINATION
RPC_compute_sat_orbit_param : NORMAL TERMINATION - time exe= 0 s.
STOP COMPUTE_SAT_TRAJECTORY.exe : NORMAL TERMINATION
RPC_compute_sat_trajectory : NORMAL TERMINATION - time exe= 6 s.
STOP visu_vectime_3D.exe  : NORMAL TERMINATION
RPC_visu_vectime_3D      : NORMAL TERMINATION - time exe= 0 s.
STOP change_coordinate_system.exe : NORMAL TERMINATION
RPC_change_coordinate_system : NORMAL TERMINATION - time exe= 0 s.
STOP visu_vectime_3D.exe  : NORMAL TERMINATION
RPC_visu_vectime_3D      : NORMAL TERMINATION - time exe= 0 s.
STOP visu_2_vectime.exe   : NORMAL TERMINATION
RPC_visu_2_vectime       : NORMAL TERMINATION - time exe= 0 s.
STOP visu_vectime_3D_4sat.exe : NORMAL TERMINATION
RPC_visu_vectime_3D_4sat : NORMAL TERMINATION - time exe= 1 s.

```

```

=====
Tests of do_simulated_orbit_GEOPOS.sh
=====

```

```

-----
do_simulated_orbit_GEOPOS.sh
-----

```

```

RPC_get_data_GEOPOS      : NORMAL TERMINATION - time exe= 0 s.
STOP visu_vectime.exe     : NORMAL TERMINATION
RPC_visu_vectime         : NORMAL TERMINATION - time exe= 1 s.
STOP change_coordinate_system.exe : NORMAL TERMINATION
RPC_change_coordinate_system : NORMAL TERMINATION - time exe= 0 s.
STOP visu_vectime_3D.exe  : NORMAL TERMINATION
RPC_visu_vectime_3D      : NORMAL TERMINATION - time exe= 0 s.
STOP change_coordinate_system.exe : NORMAL TERMINATION
RPC_change_coordinate_system : NORMAL TERMINATION - time exe= 0 s.
STOP visu_vectime_3D.exe  : NORMAL TERMINATION
RPC_visu_vectime_3D      : NORMAL TERMINATION - time exe= 0 s.
STOP compute_sat_orbit_param.exe : NORMAL TERMINATION
RPC_compute_sat_orbit_param : NORMAL TERMINATION - time exe= 0 s.
STOP COMPUTE_SAT_TRAJECTORY.exe : NORMAL TERMINATION
RPC_compute_sat_trajectory : NORMAL TERMINATION - time exe= 3 s.
STOP visu_vectime_3D.exe  : NORMAL TERMINATION
RPC_visu_vectime_3D      : NORMAL TERMINATION - time exe= 0 s.
STOP change_coordinate_system.exe : NORMAL TERMINATION
RPC_change_coordinate_system : NORMAL TERMINATION - time exe= 0 s.
STOP visu_vectime_3D.exe  : NORMAL TERMINATION
RPC_visu_vectime_3D      : NORMAL TERMINATION - time exe= 0 s.
STOP visu_2_vectime.exe   : NORMAL TERMINATION
RPC_visu_2_vectime       : NORMAL TERMINATION - time exe= 1 s.
STOP visu_vectime_3D_4sat.exe : NORMAL TERMINATION
RPC_visu_vectime_3D_4sat : NORMAL TERMINATION - time exe= 0 s.

```

The command `run_test_demo_with_log.sh` run 3 main tests:

- Tests of `RPC_compute_sat_trajectory`
- Tests of `do_simulated_orbit_CLUPOS.sh`
- Tests of `do_simulated_orbit_GEOPOS.sh`

6.1.1 RPC_compute_sat_trajectory

The plots generated by these tests are given figures 6.1 and 6.2.

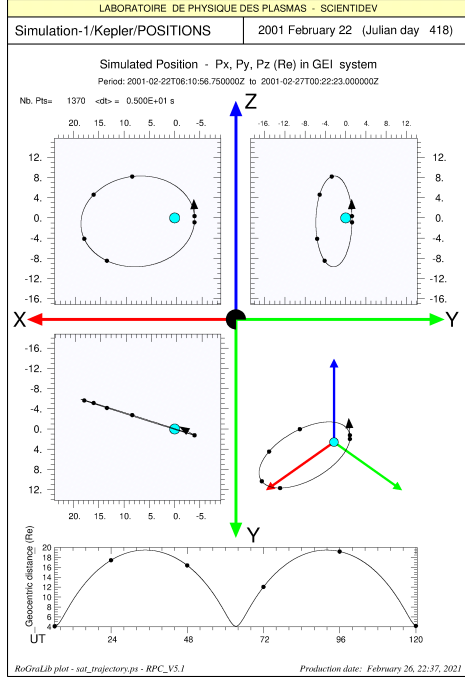


FIG. 6.1: Computed trajectory in GEI

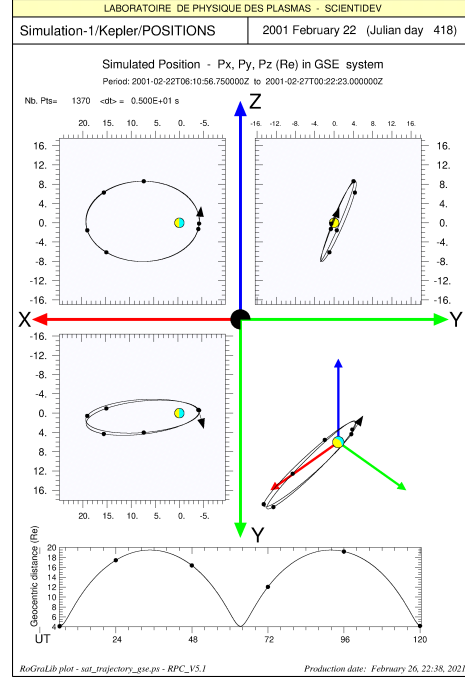


FIG. 6.2: Computed trajectory in GSE

6.1.2 do_simulated_orbit_CLUPOS.sh

The plots generated by these tests are given figures 6.3 and 6.4.

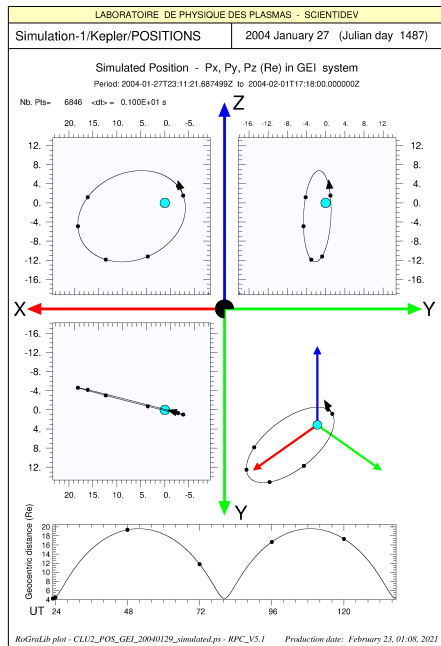


FIG. 6.3: Computed CLUSTER trajectory in GEI

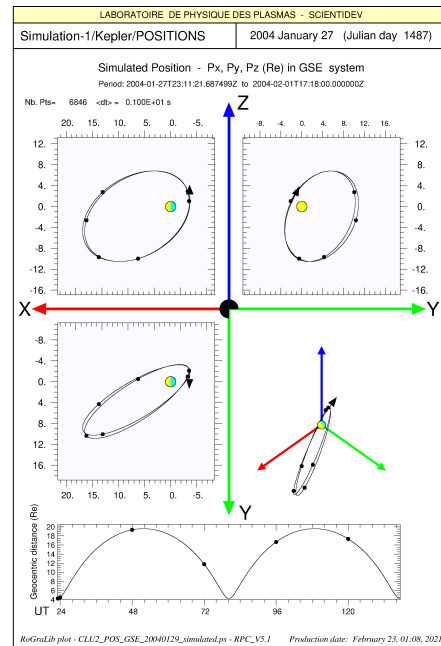


FIG. 6.4: Computed CLUSTER trajectory in GSE

6.1.3 do_simulated_orbit_GEOPOS.sh

The plots generated by these tests are given figures 6.3 and 6.4.

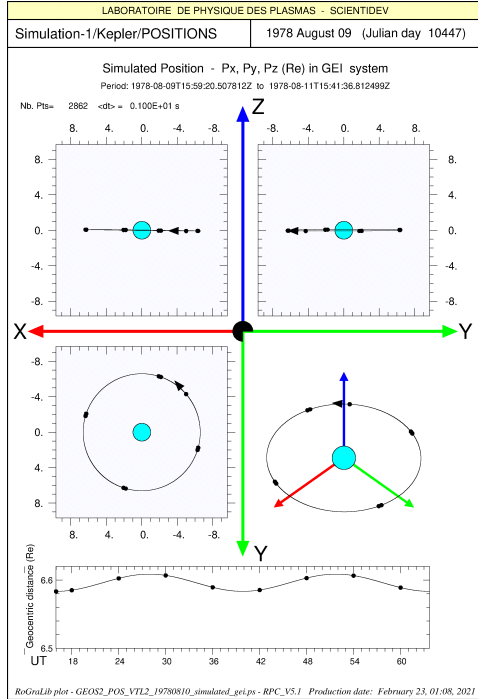


FIG. 6.5: Computed GEOS trajectory in GEI

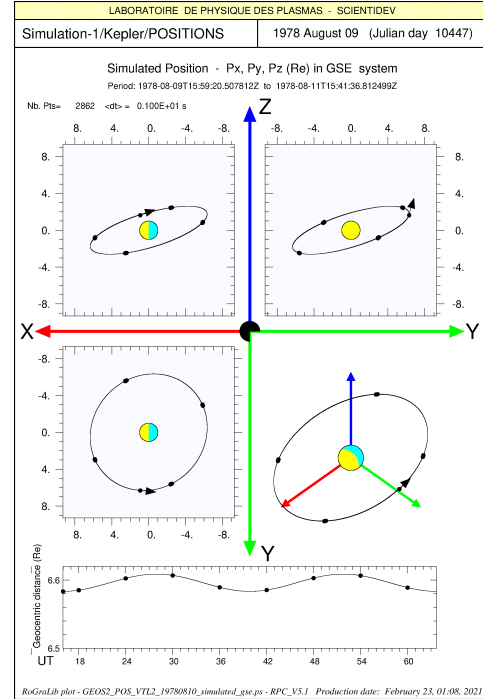


FIG. 6.6: Computed GEOS trajectory in GSE

6.1.4 run_test_compute_curl_div_4sat.sh

This script run the simulations with 3 values of the tetrahedron scale: 300 km, large value by respect to the radius of the tube (2000 km) 30 km, small value 3 km, very small values, where the linearization effect of \vec{B} can be neglected.

The script output is as following:

```
RPC_current_tube 2000. 0. 0. 0.25 300. 0.7 10.
RPC_compute_curl_div_4sat SC1_MAG.rff SC2_MAG.rff SC3_MAG.rff SC4_MAG.rff \
                          SC1_POS.rff SC2_POS.rff SC3_POS.rff SC4_POS.rff
RPC_visu_curl_div_4sat 0 0
echo '/////////////////////////////////////////////////////////////////'
RPC_current_tube 2000. 0. 0. 0.25 30. 0.7 10.
RPC_compute_curl_div_4sat SC1_MAG.rff SC2_MAG.rff SC3_MAG.rff SC4_MAG.rff \
                          SC1_POS.rff SC2_POS.rff SC3_POS.rff SC4_POS.rff
RPC_visu_curl_div_4sat 0 0
echo '/////////////////////////////////////////////////////////////////'
RPC_current_tube 2000. 0. 0. 0.25 3. 0.7 10.
RPC_compute_curl_div_4sat SC1_MAG.rff SC2_MAG.rff SC3_MAG.rff SC4_MAG.rff \
                          SC1_POS.rff SC2_POS.rff SC3_POS.rff SC4_POS.rff
RPC_visu_curl_div_4sat 0 0
echo '/////////////////////////////////////////////////////////////////'
```

6.2 Test generic scripts

First, you have to position yourself in the directory `test/test_generic_scripts`. Then, to erase the old tests, we can run the command:

clean_all_tests.sh

Only the following launch shells will remain:

clean_all_tests.sh

as well as the shell which launches all the tests:

run_test_demo.sh

and the one allowing to keep a copy of the screen in a file :

run_test_demo_with_log.sh

The **run_test_demo.sh** run the following scripts:

```
line='-----',
echo ' =====',
echo ' Tests of generic scripts ',
echo ' =====',
echo $line ; echo 'do_simulated_data_MVA_plot.sh' ; echo $line
           ; do_simulated_data_MVA_plot.sh      > /dev/null

echo $line ; echo 'do_simulated_data_Polar_plot.sh' ; echo $line
           ; do_simulated_data_Polar_plot.sh    > /dev/null

echo $line ; echo 'do_simulated_data_processing.sh' ; echo $line
           ; do_simulated_data_processing.sh    > /dev/null

echo $line ; echo 'do_simulated_orbit.sh' ; echo $line
           ; do_simulated_orbit.sh              > /dev/null
```

By running the command **run_test_demo_with_log.sh**, the successive scripts are executed and the corresponding plots displayed on the screen (many plots...). Only the *stderr* will be displayed on the screen, and saved in the file **run_test_demo.log**. Looking the screen, or reading the log file, allows you to verify that all the tests have been executed correctly.

log file is as below:

```
=====
Tests of generic scripts
=====

do_simulated_data_MVA_plot.sh
-----
STOP create_simulated_data.exe      : NORMAL TERMINATION
  RPC_create_simulated_data         : NORMAL TERMINATION - time exe= 4 s.
STOP vectime_to_spectro.exe         : NORMAL TERMINATION
  RPC_vectime_to_spectro            : NORMAL TERMINATION - time exe= 3 s.
STOP visu_spectro.exe              : NORMAL TERMINATION
  RPC_visu_spectro                  : NORMAL TERMINATION - time exe= 1 s.
STOP visu_ave_spectrum.exe         : NORMAL TERMINATION
  RPC_visu_ave_spectrum             : NORMAL TERMINATION - time exe= 0 s.
STOP vectime_to_mva.exe             : NORMAL TERMINATION
  RPC_vectime_to_mva                : NORMAL TERMINATION - time exe= 3 s.
STOP vectime_to_spectro.exe         : NORMAL TERMINATION
  RPC_vectime_to_spectro            : NORMAL TERMINATION - time exe= 3 s.
STOP visu_spectro.exe              : NORMAL TERMINATION
  RPC_visu_spectro                  : NORMAL TERMINATION - time exe= 1 s.
STOP visu_ave_spectrum.exe         : NORMAL TERMINATION
```

RPC_visu_ave_spectrum	: NORMAL TERMINATION - time exe= 0 s.
do_simulated_data_Polar_plot.sh	
STOP create_simulated_data.exe	: NORMAL TERMINATION
RPC_create_simulated_data	: NORMAL TERMINATION - time exe= 4 s.
STOP create_simulated_data.exe	: NORMAL TERMINATION
RPC_create_simulated_data	: NORMAL TERMINATION - time exe= 0 s.
STOP vectime_to_mfa.exe	: NORMAL TERMINATION
RPC_vectime_to_mfa	: NORMAL TERMINATION - time exe= 8 s.
STOP vectime_to_spectro.exe	: NORMAL TERMINATION
RPC_vectime_to_spectro	: NORMAL TERMINATION - time exe= 4 s.
STOP visu_spectro.exe	: NORMAL TERMINATION
RPC_visu_spectro	: NORMAL TERMINATION - time exe= 1 s.
STOP visu_ave_spectrum.exe	: NORMAL TERMINATION
RPC_visu_ave_spectrum	: NORMAL TERMINATION - time exe= 0 s.
STOP spectro_to_polar.exe	: NORMAL TERMINATION
RPC_spectro_to_polar	: NORMAL TERMINATION - time exe= 1 s.
STOP visu_polar.exe	: NORMAL TERMINATION
RPC_visu_polar	: NORMAL TERMINATION - time exe= 1 s.
do_simulated_data_processing.sh	
STOP create_simulated_data.exe	: NORMAL TERMINATION
RPC_create_simulated_data	: NORMAL TERMINATION - time exe= 4 s.
STOP create_simulated_data.exe	: NORMAL TERMINATION
RPC_create_simulated_data	: NORMAL TERMINATION - time exe= 0 s.
STOP create_simulated_data.exe	: NORMAL TERMINATION
RPC_create_simulated_data	: NORMAL TERMINATION - time exe= 3 s.
STOP visu_vectime.exe	: NORMAL TERMINATION
RPC_visu_vectime	: NORMAL TERMINATION - time exe= 3 s.
STOP visu_vectime.exe	: NORMAL TERMINATION
RPC_visu_vectime	: NORMAL TERMINATION - time exe= 0 s.
STOP visu_vectime.exe	: NORMAL TERMINATION
RPC_visu_vectime	: NORMAL TERMINATION - time exe= 0 s.
STOP visu_vectime_3D.exe	: NORMAL TERMINATION
RPC_visu_vectime_3D	: NORMAL TERMINATION - time exe= 1 s.
STOP vectime_to_spectro.exe	: NORMAL TERMINATION
RPC_vectime_to_spectro	: NORMAL TERMINATION - time exe= 4 s.
STOP visu_spectro.exe	: NORMAL TERMINATION
RPC_visu_spectro	: NORMAL TERMINATION - time exe= 0 s.
STOP visu_ave_spectrum.exe	: NORMAL TERMINATION
RPC_visu_ave_spectrum	: NORMAL TERMINATION - time exe= 1 s.
do_simulated_orbit.sh	
STOP create_simulated_data.exe	: NORMAL TERMINATION
RPC_create_simulated_data	: NORMAL TERMINATION - time exe= 4 s.
STOP visu_vectime_3D.exe	: NORMAL TERMINATION
RPC_visu_vectime_3D	: NORMAL TERMINATION - time exe= 0 s.
STOP change_coordinate_system.exe	: NORMAL TERMINATION
RPC_change_coordinate_system	: NORMAL TERMINATION - time exe= 0 s.
STOP compute_sat_orbit_param.exe	: NORMAL TERMINATION
RPC_compute_sat_orbit_param	: NORMAL TERMINATION - time exe= 0 s.
STOP COMPUTE_SAT_TRAJECTORY.exe	: NORMAL TERMINATION
RPC_compute_sat_trajectory	: NORMAL TERMINATION - time exe= 6 s.
STOP change_coordinate_system.exe	: NORMAL TERMINATION
RPC_change_coordinate_system	: NORMAL TERMINATION - time exe= 0 s.
STOP visu_vectime_3D.exe	: NORMAL TERMINATION
RPC_visu_vectime_3D	: NORMAL TERMINATION - time exe= 1 s.
STOP visu_2_vectime_3D.exe	: NORMAL TERMINATION
RPC_visu_2_vectime_3D	: NORMAL TERMINATION - time exe= 1 s.
Starting time : 2021-02-26 20:55:32	
Ending time : 2021-02-26 20:56:45	
Duration : 73 sec. (1.21 mn.)	

6.3 Test RPC_dir

First, you have to position yourself in the directory `test/tests_RPC_dir`. Then, run the command:

run_all_test_RPC_dir.sh

The content of this test is given above:

```
# !/ bin / bash

# directory de test
dir=$RPC_DIR

echo '-----',
    RPC_check_dirname_tree      > /dev/null
    RPC_check_dirname_tree $dir > /dev/null
echo '-----',
    RPC_dir_pretty_tree        > /dev/null
    RPC_dir_pretty_tree $dir    > /dev/null
echo '-----',
    RPC_dir_size                > /dev/null
    RPC_dir_size Mo             > /dev/null
    RPC_dir_size $dir           > /dev/null
    RPC_dir_size $dir Mo        > /dev/null
echo '-----',
    RPC_dir_size_tree           > /dev/null
    RPC_dir_size_tree Mo        > /dev/null
    RPC_dir_size_tree $dir      > /dev/null
    RPC_dir_size_tree $dir Mo    > /dev/null
echo '-----',
    RPC_dir_size_pretty_tree    > /dev/null
    RPC_dir_size_pretty_tree Mo > /dev/null
    RPC_dir_size_pretty_tree $dir > /dev/null
    RPC_dir_size_pretty_tree $dir Mo > /dev/null
echo '-----',
    RPC_dir_properties          > /dev/null
    RPC_dir_properties Mo       > /dev/null
    RPC_dir_properties $dir     > /dev/null
    RPC_dir_properties $dir Mo  > /dev/null
echo '-----',
    RPC_dir_properties_tree     > /dev/null
    RPC_dir_properties_tree Mo  > /dev/null
    RPC_dir_properties_tree $dir > /dev/null
    RPC_dir_properties_tree $dir Mo > /dev/null
echo '-----',
    RPC_dir_properties_pretty_tree > /dev/null
    RPC_dir_properties_pretty_tree Mo > /dev/null
    RPC_dir_properties_pretty_tree $dir > /dev/null
    RPC_dir_properties_pretty_tree $dir Mo > /dev/null
echo
echo '-----',
echo results :
echo '-----',
echo
ls -l dir_tre*.txt
echo
ls -l dir_size*.txt
echo
ls -l dir_prop*.txt
echo
echo '-----',
```

Only the *stderr* will be displayed on the screen, in the form given below, which allows you to verify that all the tests have been executed correctly.

```

robert@lx-robert/work: ../tests/*dir/tes*sh
-----
There is 0 directories having blank in the name
There is 0 directories having blank in the name
-----
STOP dir_pretty_tree.exe           : NORMAL TERMINATION
STOP dir_pretty_tree.exe           : NORMAL TERMINATION
-----
STOP dir_size_pretty_tree.exe       : NORMAL TERMINATION
STOP dir_size_pretty_tree.exe       : NORMAL TERMINATION
STOP dir_size_pretty_tree.exe       : NORMAL TERMINATION
STOP dir_size_pretty_tree.exe       : NORMAL TERMINATION
-----
STOP dir_propereties_pretty_tree.exe : NORMAL TERMINATION
STOP dir_propereties_pretty_tree.exe : NORMAL TERMINATION
STOP dir_propereties_pretty_tree.exe : NORMAL TERMINATION
STOP dir_propereties_pretty_tree.exe : NORMAL TERMINATION
-----
results :
-----
dir_tree.txt
dir_size_pretty_tree.txt
dir_properties_pretty_tree.txt

```


6.3.1 dir_pretty_tree

```

nb. of directories considered:      21
Maximum number of levels          :    3

|_bash
|_bin
|_doc-----
|          |_src_doc
|_gainantset
|_mod
|_obj
|_scripts___
|          |_scripts_CLUSTER
|          |_scripts_generic
|          |_scripts_GEOS
|_src
|_tests-----
|          |_test_compute_sat_trajectory
|          |_tests_CLUSTER
|          |_tests_generic_scripts
|          |_tests_GEOS
|          |_tests_RPC_dir-----
|                                     |_dummy_dir_1
|                                     |_dummy_dir_2

-----
dir_pretty_tree.exe                : NORMAL TERMINATION
>

```

6.3.2 dir_size_pretty_tree

```

nb. of directories considered:      21
Maximum number of levels          :    3

size in Mo
-----
792._
 0 |_bash
103 |_bin
 81 |_doc-----
 65 |          |_src_doc
  1 |_gainantset
  0 |_mod
  6 |_obj
 14 |_scripts___
  0 |          |_scripts_CLUSTER
 14 |          |_scripts_generic
  0 |          |_scripts_GEOS
  2 |_src
582 |_tests-----
 17 |          |_test_compute_sat_trajectory
433 |          |_tests_CLUSTER
131 |          |_tests_generic_scripts
  0 |          |_tests_GEOS
  0 |          |_tests_RPC_dir-----
  0 |                                     |_dummy_dir_1
  0 |                                     |_dummy_dir_2

-----
dir_size_pretty_tree.exe          : NORMAL TERMINATION
>

```

6.3.3 dir_propereties_pretty_tree

Maximum number of levels : 3

size in Mo, Nb files , Nb Dir.

```
-----
792 1013 20. _
 0 143 0 | _bash
103 57 0 | _bin
81 15 1 | _doc-----
65 5 0 | | _src_doc
1 34 0 | _gainantset
0 4 0 | _mod
6 70 0 | _obj
14 88 3 | _scripts___
0 12 0 | | _scripts_CLUSTER
14 66 0 | | _scripts_generic
0 10 0 | | _scripts_GEOS
2 82 0 | _src
582 510 7 | _tests-----
17 109 0 | | _test_compute_sat_trajectory
433 270 0 | | _tests_CLUSTER
131 108 0 | | _tests_generic_scripts
0 12 0 | | _tests_GEOS
0 9 2 | | _tests_RPC_dir-----
0 1 0 | | _dummy_dir_1
0 2 0 | | _dummy_dir_2
```

Part III

APPLICATION TO GEOS DATA

Chapter 7

ACCESS TO GEOS DATA

7.1 Downloads RFF data files

GEOS data files are available from the CDPP : <https://cdpp-archive.cnes.fr/>

Rubrique: European GEOS mission

The downloaded files are of the following type:

```
GEOS/ULF VTL2 : GEOS1_ULF_VTL2_19770713.rff
GEOS/MAG      : GEOS1_MAG_VTL2_19770713.rff
GEOS/POS      : GEOS1_POS_VTL2_19771207.rff
```

7.2 Setting up a local database

It is recommended to make a database of the files in RFF format used by the RPC commands.

The RPC package comes with a minimal database, just containing samples for testing. These directories can be anywhere on your machine, even on a different disk or on the network as long as its path is correctly indicated in the RPC RPC_config.bash.

You just have to copy them into the report, and then the sub-directories will be automatically created when filling with the command **RPC_put_rff_database**.

This database is organized as follows :

```
_MAG
|
|_1977
|   |_1977_05
|   |_1977_07
|   |_1977_12
|_1978
|   |_1978_01
|   |_1978_05
|   |_1978_08
_POS
|
|_1977
|   |_1977_07
|   |_1977_12
|_1978
|   |_1978_01
|   |_1978_08
_ULF
|
|_VTL1
|   |_1977_____|_1977_12
|_VTL2
|   |_1977_____|_1977_05
|   |           |_1977_07
|   |           |_1977_12
|   |_1978_____|_1978_01
|   |           |_1978_05
|   |           |_1978_08
|   |_1980_____|_1980_06
```

7.3 Interest of a local database

RPC processing commands request one or more RFF files as arguments. These files can be in the current directory or elsewhere, if the path is correctly entered there is no problem. But if you are working on many files, it is more pleasant to store them directly in the local database. First, they can be easily found with the commands described in the next chapter, and then this database can be in another directory, a disk« data » for example.

The local database is also used for access to two simultaneous experiments, for example when we want to plot the gyrofrequency of protons (from MAG data) or the position of satellites (from POS) on a spectrogram of ULF data for example .

It is essential for all production orders (which are not detailed here, see [1]).

7.4 Commands relating to the database

After downloading an rff file, you can *move the RFF data files in the local data base* by :

RPC_put_rff_database name.rff

The same command is valid for VecTime from ULF, MAG and POS..

If one wants *copy the RFF files in the local data base* , and not move it, one use the command:

RPC_copy_rff_database name.rff

As before, the same command is valid for VecTime from ULF, MAG and POS.

Conversely, we can *copy the file in the working directory* by the commands:

RPC_get_data_GEOULF Level SatNum year month day
 Level : VTL1 or VTL2
RPC_get_data_GEOMAG SatNum year month day
RPC_get_data_GEOPOS SatNum year month day

If we want to know what is in the RFF database, *the list of all RFF files* by experience is obtained by the commands :

RPC_list_rff_database CLUSTER STA
RPC_list_rff_database CLUSTER FGM
RPC_list_rff_database GEOS ULF
RPC_list_rff_database GEOS MAG

or

RPC_list_rff_database GEOS ALL

Chapter 8

WORKING ON GEOS RFF FILES

8.1 Main types of RFF files

We will mainly handle files of type VecTime [3] which are vectors indexed by time, and files of type Spectrogram [3] which are series of dated spectra.

To go fast, you can go directly to chapter« USING SCRIPTS »

8.1.1 Example of VecTime file names :

```
GEOS1_ULF_VTL2_19770713.rff
GEOS1_MAG_VTL2_19770713.rff
GEOS1_POS_VTL2_19771207.rff
```

8.1.2 Example of Spectrograms file names :

The nomenclature is like :

```
GEOS1_ULF_SPL2_19770713.rff
GEOS1_MAG_SPL2_19770713.rff
GEOS1_POS_SPL2_19771207.rff
```

Note that the viewing commands will create files like :

```
GEOS1_ULF_SPL2_19770713.ps
GEOS1_ULF_SPL2_19770713.pdf
GEOS1_ULF_SPL2_19770713.png
```

and for the average spectra :

```
GEOS1_ULF_SPL2_19770713_ave_spe.ps
GEOS1_ULF_SPL2_19770713_ave_spe.pdf
GEOS1_ULF_SPL2_19770713_ave_spe.png
```

8.1.3 Checking the validity of an RFF file

The validity of an RFF file can be checked by the command:

```
RPC_check_rff toto.rff
```

In the event of non-compliance, it will be reported, and the reasons will be displayed in the check_rff.out file. Note that the validity of all the rff files found in a toto directory can be checked with the command **RPC_check_rff_dir** toto_dir

8.2 Special commands for GEOS

RPC_menu_GEOS

Displays on screen the general menu of commands, as well as special commands to GEOS.

RPC_vectime_calibration_GEOULF

Calibrate a VTL1 file and produce VTL2

Use:

RPC_vectime_calibration_GEOULF VTL1.rff VTL2.rff Fdet Fc F1 F2 Step N-Kern N-shift Apod

With :

VTL1.rff	: name of an input vectime RFF file
VTL2.rff	: name of an output vectime RFF file
Fdet	: detrend frequency (0. for classis despin)
Fc	: Frequency cut-off for calibration
Fmin	: frequency min for filtering (Min=0 => Fc)
Fmax	: frequency max for filtering (Max=0 => Nyquist)
Step	: Processing steps asked (1-7)
N-Kern	: Kernel size
N-shift	: for sliding window
Apod	: trapezium (t) or Gaussian (g)

Comments on processing steps:

- 1 : Volts, spinning system, with DC field
- 2 : Volts, spinning system, without DC field
- 3 : nTesla, spinning system, without DC field
- 4 : nTesla, fixed SR2 system, without DC field
- 5 : nTesla, fixed SR2 system, with DC field
- 6 : nTesla, fixed SR2 system, only DC-field
- 7 : nTesla, fix. ISR2 system + Dx,Dy in sperate fields

RPC_vectime_L1_to_spectro_L2_GEOULF

Produces a calibrated spectrogram from an uncalibrated VTL1 file

Use:

RPC_vectime_L1_to_spectro_L2_GEOULF VTL1.rff SPL2.rff Fdet Fc F1 F2 Step N_Kern N_shift Apod

With :

VTL1.rff	: name of an input vectime RFF file
SPL2.rff	: name of an output spectrogram RFF file
Fdet	: detrend frequency (0. for classis despin)
Fc	: Frequency cut-off for calibration
Fmin	: frequency min for filtering (Min=0 = Fc)
Fmax	: frequency max for filtering (Max=0 = Nyquist)
Step	: Processing steps asked (1-7)
N_Kern	: Kernel size
N_shift	: for sliding window
Apod	: trapezium (t) or nothing(n)

Example :

RPC_vectime_L1_to_spectro_L2_GEOULF GEOS1_ULF_VTL1_19770713.rff GEOS1_ULF_SPL2_19770713.rff 0. 0.1 0.5 0. 4 1024 2 g

RPC_vectime_to_mfav_GEOULF

Change coordinate of GEOS/ULF from SRV to MFAV

Usage :

RPC_vectime_to_mfav_GEOULF VTL2_ulf_srv VTL2_mag_srv VTL2_ulf_mfav

With :

VTL2_ulf_srv : name of a ULF S00 RFF file
 VTL2_mag_srv : name of a MAG S331 RFF file
 VTL2_ulf_mfav : name the ULF S00 RFF file in mfav

RPC_vectime_vdh_to_srv_GEOMAG

Change coordinate of GEOS / S331 from VDH to SRV

Usage :

RPC_vectime_vdh_to_srv_GEOMAG VTL2_mag_vdh VTL2_mag_srv

avec :

VTL2_mag_vdh : input file of a MAG S331 RFF file in VDH
 VTL2_mag_srv : output file of a MAG S331 RFF file in SRV

RPC_vectime_vdh_to_xyz_GEOMAG

Change coordinate of GEOS/S331 from VDH to XYZ (spinning frame)

Usage :

RPC_vectime_vdh_to_xyz_GEOMAG VTL2_mag_vdh VTL2_mag_xyz

avec :

VTL2_mag_vdh : input file of a MAG S331 RFF file in VDH
 VTL2_mag_xyz : output file of a MAG S331 RFF file in XYZ

8.3 Make an ULF spectrogram

8.3.1 From a VTL1 file

Launch the command (detailed in the previous chapter):

RPC_vectime_L1_to_spectro_L2_GEOULF VTL1.rff SPL2.rff Fdet Fc F1 F2 Step N_Kern N_shift Apod

Example :

RPC_vectime_L1_to_spectro_L2_GEOULF GEOS1_ULF_VTL1_19770713.rff GEOS1_ULF_SPL2_19770713.rff 0. 0.1 0.5 0. 4 1024 2 g

When the SPL2.rff file is created, it can be viewed with the command:

RPC_visu_spectro SP.rff datiso1 datiso2 f1 f2 pmin pmax XY fi1 fi2

With :

SP.rff : name of a spectro RFF file
 datiso1 datiso2 : iso date/time first and end
 f1 f2 : frequency bounds to plot
 pmin pmax : min max power for dynamic colors
 XY or LR : for XYZ or Left Right Z
 fi1, fi2 : frequency bounds for integrated power

8.3.2 From a VTL2 file

This is done with the generic procedure (valid whatever the VecTime file):

RPC_vectime_to_spectro VT.rff SP.rff N_Kern N_shift Apod

With :

VT.rff : name of an input vectime RFF file
 SP.rff : name of an output spectrogram RFF file
 N_Kern : Kernel size
 N_shift : for sliding window
 Apod : trapezium (t), Gaussian (g) or nothing (n)

Example :

```
RPC_vectime_to_spectro  GEOS1_ULF_VTL2_19770713.rff GEOS1_ULF_SPL2_19770713.rff 512 512 t
```

The visualization is done with the same command below :

```
RPC_visu_spectro  SP.rff datiso1 datiso2 f1 f2 pmin pmax XY fi1 fi2
```

8.3.3 Spectrogram visualization

Just run the following generic command:

```
RPC_visu_spectro  SP.rff datiso1 datiso2 f1 f2 pmin pmax XY fi1 fi2
```

With :

```
SP.rff           : name of a spectro RFF file
datiso1 datiso2  : iso date/time first and end
f1 f2           : frequency bounds to plot
pmin pmax       : min max power for dynamic colors
XY or LR        : for XYZ or Left Right Z
fi1, fi2        : frequency bounds for integrated power
```

Example :

```
RPC_visu_spectro  toto_SP.rff 1977-07-13T09:00:00.000Z 1977-07-13T11:00:00.000Z 0. 5. 0. 0. XY 0.2 10.
```

or again:

```
RPC_visu_spectro  toto_SP.rff 0 0 0. 200. -9.6 -3. LR 0. 200.
```

This command will add the plot of the gyrofrequencies and the satellite positions, *provided that the MAG and position files are present in the* database so that the procedure can fetch the corresponding data.

The frequencies are plotted according to the formulas:

$$F_{ci} = |B_o| * 1.52E-2 \text{ (Bo en nT)} \quad FLH = \sqrt{1836.} * 6.2832 * F_{ci}$$

8.4 Make a spectrogram of the Magnetometer

Since the data are already calibrated, it suffices to launch the generic procedure :

```
RPC_vectime_to_spectro  VT.rff SP.rff N_Kern N_shift Apod
```

Example :

```
RPC_vectime_to_spectro  GEOS1_MAG_19770713.rff GEOS1_MAG_SPL2_19770713.rff 64 64 t
```

We then visualize this spectrogram by the command :

```
RPC_visu_spectro  SP.rff datiso1 datiso2 f1 f2 pmin pmax XY fi1 fi2
```

With :

```
SP.rff           : name of a spectro RFF file
datiso1 datiso2  : iso date/time first and end
f1 f2           : frequency bounds to plot
pmin pmax       : min max power for dynamic colors
XY or LR        : for XYZ or Left Right Z
fi1, fi2        : frequency bounds for integrated power
```

Example :

```
RPC_visu_spectro  GEOS1_MAG_SPL2_19770713.rff 1977-07-13T09:00:00.000Z 1977-07-13T11:00:00.000Z 0. 2.1 0. 0. XY 0.2 2.
```

or, for the full duration of the file :

```
RPC_visu_spectro  GEOS1_MAG_SPL2_19770713.rff 0 0 0. 0. 0. XY 0.2 2.
```

8.5 Visualize an average spectrum

The file SPL2.rff being created, one can visualize the average spectrum of all the spectra of the SPL2 by the command :

RPC_visu_ave_spectrum SP.rff datiso1 datiso2 f1 f2 pmin pmax

With :

SP.rff : name of a spectro RFF file
 datiso1 datiso2 : iso date/time first and end
 f1 f2 : frequency bounds to plot
 pmin pmax : min max power for dynamic colors (log values)
 XY or LR : for XYZ or Left Right Z
 info_orb,info_FGM : y/n y/n

Example :

RPC_visu_ave_spectrum toto_SP.rff 1977-07-13T09:00:00.000Z 1977-07-13T11:00:00.000Z 0. 5. 0. 0. XY y n

RPC_visu_ave_spectrum toto_SP.rff 0 0 0. 0. 0. 0. XY y n

8.6 Draw a waveform

Just run the generic procedure :

RPC_visu_vectime name.rff

Example :

RPC_visu_vectime GEOS1_MAG_19770713.rff \$T1 \$T2

Avec \$T1 et \$T2 variables containing the ISO start and end date of the period to be displayed.

Example: T1=2002-09-23T10:30:00.000Z T2= 2002-09-23T10:33:00.000Z

for [T1 T2] = [0 0] the whole file will be displayed.

8.7 Extract part of a VecTime file

Note that we can also extract the interesting portion of the file before viewing or other. This is done with the command:

RPC_reduce_time_vectime toto_VT.rff toto_VT.rff datiso1 datiso2

With :

toto_VT.rff : name of an input vectime RFF file
 toto_VT.rff : name of an output vectime RFF file
 datiso1 datiso2 : date in ISO format

Example :

RPC_reduce_time_vectime GEOS1_ULF_VTL2_19770713.rff GEOS1_ULF_VTL2_19770713_red.rff 1977-07-13T09:12:00Z 1977-07-13T09:30:00Z

We can then do :

RPC_visu_vectime GEOS1_ULF_VTL2_19770713_red.rff 0 0

The start and end time set to 0 0 means that the entire file will be viewed.

8.8 Change coordinate system

You can change the coordinate system of any VecTime file via Rocotlib [4] with the command:

RPC_change_coordinate_system toto.rff gsm toto_gsm.rff

With :

toto.rff : name of a RFF file
 gse : output coordinate system
 toto_gse.rff : name of a new RFF file

Example :

RPC_change_coordinate_system GEOS1_POS_19770713.rff gse GEOS1_POS_19770713_gse.rff

8.9 Waves polarisation computation

8.9.1 Transformation in MFAV coordinate system

The calculation of the polarization of the waves requires that we first go to the MFAV (Magnetic Field Aligned) frame [5]. This is done by the above command :

RPC_vectime_to_mfav_GEOULF file_ULF.rff file_ULF_mfav.rff file_MAG.rff

Example :

RPC_vectime_to_mfav_GEOULF GEOS1_ULF_VTL2_19770713.rff GEOS1_ULF_VTL2_19770713_mfav.rff GEOS1_MAG_VTL2_19770713_srv.rff

Warning ! This is only possible if the MAG file is in SRV system. As originally it is in VDH reference, the conversion is done by the command:

RPC_vectime_vdh_to_srv_GEOMAG GEOS1_MAG_VTL2_19770713.rff GEOS1_MAG_VTL2_19770713_srv.rff

8.9.2 Calculation and plot of the polarization parameters

Once the ULF calibrated waveform file is in the MFA frame, we make the spectrogram using the generic command:

RPC_vectime_to_spectro.

Example :

RPC_vectime_to_spectro GEOS1_ULF_VTL2_19770713_mfa.rff GEOS1_ULF_SPL2_19770713_mfa.rff 512 512 t

The visualization of the spectrogram in the reference line of force can already give information, especially if it is represented in circular coordinates « Left, Right, Z » [9] by the command :

RPC_visu_spectro GEOS1_ULF_SPL2_19770713_mfa.rff 0 0 0. 0. 0. LR 0.2 2.

The polarization parameters calculated by method [6] are obtained by the command :

RPC_spectro_to_polar VTL2_mfa.rff copolar.resu

With :

VTL2_mfa.rff : name of an input RFF VTL2 file in MFA

copolar.resu : file name for polar results

Finally, the polarization parameters are displayed using the command:

RPC_visu_polar copolar.resu datiso1 datiso2 threshold f1 f2

With :

datiso1 datiso2 : iso date/time first and end of desired time section (0 0 for all)

threshold: for visualization ex: -6.1

f1 f2 : frequency bounds for visualizations

The "threshold" value should be chosen with care, because it allows the polarization of the phenomenon to be isolated from that of the surrounding noise. Various tests may be necessary.

Example :

RPC_visu_polar copolar.resu 0 0 -6.1 0. 2.

8.10 Data position visualization

8.10.1 2D-Visualization

Viewing a VecTime file such as GEOS1_POS_19770713.rff is done by the generic command :

RPC_visu_vectime GEOS1_POS_19770713.rff 0 0

These files must of course, in addition to the date, contain the requested time slot, otherwise the plot will be incomplete.

We will see in the next chapter that the use of RPC scripts will simplify this procedure.

8.10.2 3D-visualization

On the same principle as the 2D view, we can view in 3D a VecTime rff file by the generic procedure:

```
RPC_visu_vectime_3D GEOS1_POS_19770713.rff 0 0
```

8.10.3 3D-visualization in GSE

You can change the coordinate system of any VecTime file via Rocotlib [4] with the command:

```
RPC_change_coordinate_system toto.rff gse toto_gse.rff
```

So just do:

```
RPC_change_coordinate_system GEOS1_POS_19770713.rff gse GEOS1_POS_19770713_gse.rff
```

then:

```
RPC_visu_vectime_3D GEOS1_POS_19770713_gse.rff 0 0
```

8.10.4 Transform positions in another system

To pass a position file, initially in GEO, in another marker, for example GSE, we will use the generic procedure previously seen, and we will launch for example the command :

```
RPC_change_coordinate_system CLU2_AUX_VTL2_POS_20010923_000000_20010923_235959.rff \
gse \
CLU2_AUX_VTL2_POS_20010923_000000_20010923_235959_GSE.rff
```

8.11 Compute orbital parameters

If we have a position file over at least a little more than a half-revolution, we can calculate all the parameters of the orbit by:

```
RPC_compute_sat_orbit_param pos_file.rff
```

The results can be used by the following command, which calculates a trajectory according to Kepler's laws as a function of the orbit parameters.

8.12 Compute an elliptic trajectory

From the previous parameters, we can calculate the trajectory over 2 revolutions by:

```
RPC_compute_sat_trajectory Apo, Per, PerTime, a1,a2,a3, b1,b2,b3,dt, rff_out,scale
```

With :

Apo : Apogee in km, real*8
 Per : Perigee in km, real*8
 PerTime : Perigee Time, ISO format
 a1,a2,a3 : direction of semi-major axis, in GEI system
 : (from ellipse center to perigee, unit vector)
 b1,b2,b3 : direction of semi-minor axis, in GEI system
 : (a,b vector in the sens of the motion, unit vector)
 dt : time resolution for output positions
 rff_out : output rff_file
 scale : factor to apply to x,y,z

example:"

```

RPC_compute_sat_trajectory 124252.735203443d0 26321.1522350593d0 \
                             2001-02-24T15:17:23.000Z \
                             -0.9493840 0.3011743 0.0892410 \
                             0.0820478 -0.0364760 0.9959607 5.0 \
                             trajectory.rff \
                             1.
  
```

The resulting trajectory file can be viewed by:

```
RPC_visu_vectime_3D trajectory.rff
```

Chapter 9

USING GEOS SCRIPTS

Scripts are bash files that string together multiple RPC commands, to avoid the tedious task of typing them all one after the other. The RPC package ships with a number of scripts that perform the most common processing. The user can use this as inspiration to write their own treatments.

9.1 Draw a waveform

9.1.1 `do_VT_plot_GEOULF.sh`

Examples :

```
do_VT_plot_GEOULF.sh 1 19770713 121000 123000
do_VT_plot_GEOULF.sh 2 19780810 060000 190000
```

The first one give the plot shown on fig 9.1.

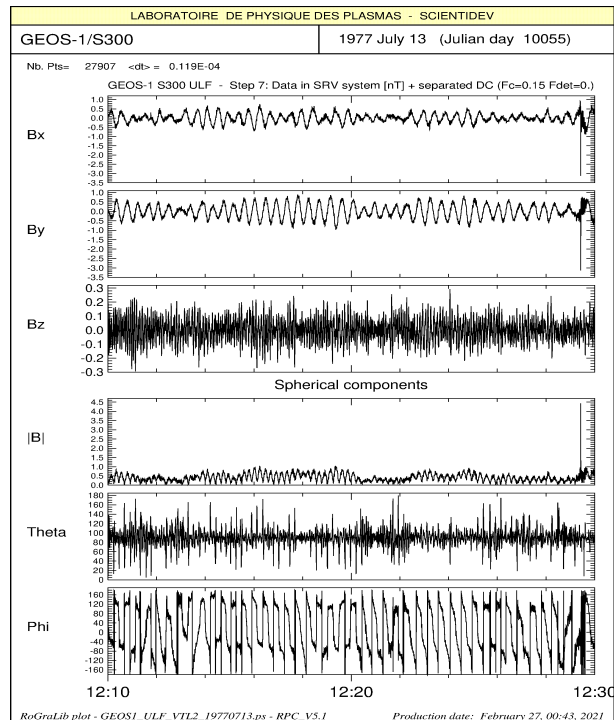


FIG. 9.1: Plotting waveform with `do_VT_plot_GEOULF.sh` script.

9.2 Make a spectrogramme

9.2.1 do_SP_plot_GEOULF.sh

Example :

```
do_SP_plot_GEOULF.sh 1 19770713 110000 150000 512 0. 2.0 t XY 1. -7. 1. y
```

```
do_SP_plot_GEOULF.sh 2 19780810 180000 210000 512 0. 11.0 t LR 1. -7. -0.5 y
```

arguments :date, H_debut, H_fin, sat, BitRate, repere, Nbp_FFT, F1, F2, Apod, XY ou LR, F1_p_int, seuil_min, seuil_max

The script :

```
do_SP_plot_GEOULF.sh 1 19770713 120000 150000 512 0. 2.0 t XY 0.5 -7. 1.
```

is equivalent to sequencing the following commands by hand :

```
RPC_get_data_VTL_GEOULF 1 1977 07 13
```

```
RPC_vectime_to_spectro GEOS1_ULF_VTL2_19770713.rff GEOS1_ULF_SPL2_19770713.rff 512 512 t
```

```
datiso1='RPC_date_time_to_datiso 19770713 120000
```

```
datiso2='RPC_date_time_to_datiso 19770713 150000'
```

```
RPC_visu_spectro GEOS1_ULF_SPL2_19770713.rff $datiso1 $datiso2 0. 2. -7. 1. XY 0.5 2.
```

```
RPC_visu_ave_spectrum GEOS1_ULF_SPL2_19770713.rff $datiso1 $datiso2 0. 2. -7. 1. XY y y
```

The first example give the following plot of fig. 9.2. Note the last argument set to "y" which allow to plot the Fci gyrofrequency on the spectrograms, as well as the position of S/C in thge bottom of the figure.

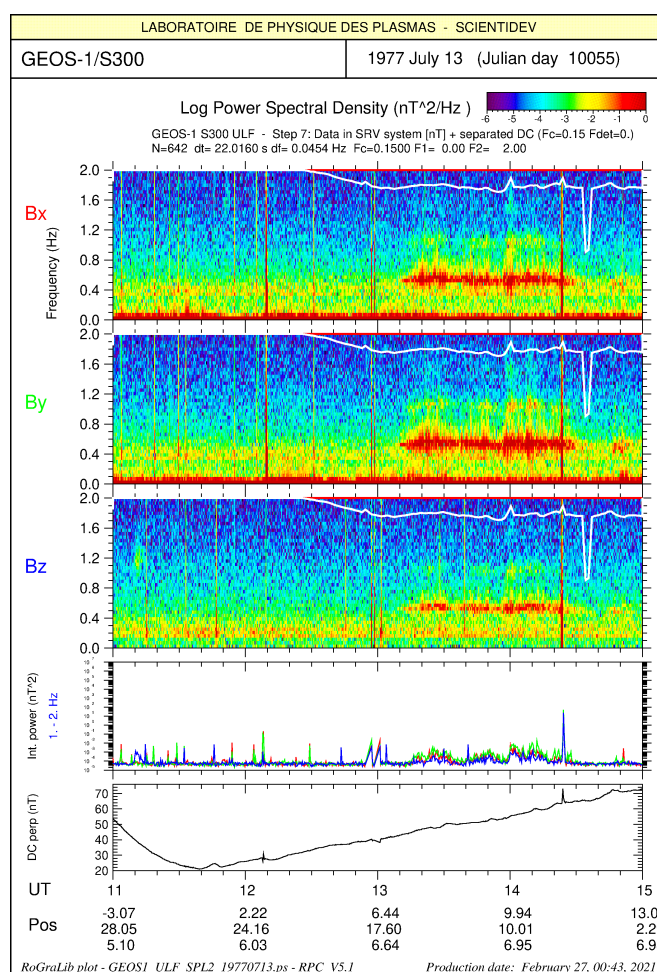


FIG. 9.2: ULF spectrogram plot done with with do_SP_plot_GEOULF.sh script.

9.2.2 do_SP_plot_GEOMAG.sh

Example :

```
do_SP_plot_GEOMAG.sh 2 19780810 000000 235959 64 0. 0.02 t XY 0.002 -1. 3.
```

arguments :date, H_debut, H_fin, sat, mode, repere, Nbp_FFT, F1, F2, Apod, XY ou LR, F1_p_int, seuil_min, seuil_max

This script give the plot of fig. 9.3

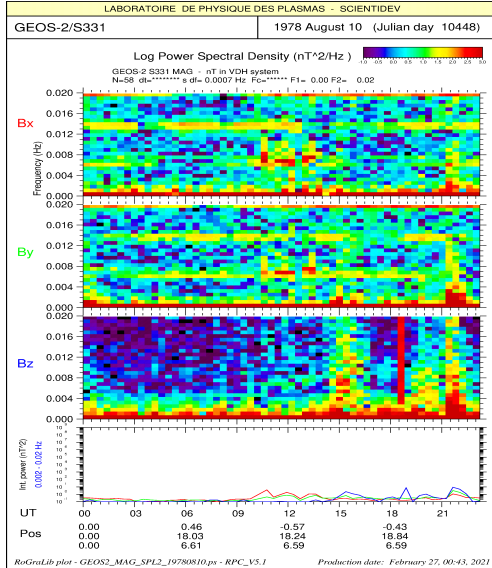


FIG. 9.3: MAG spectrogram plot done with script `do_SP_plot_GEOMAG.sh`.

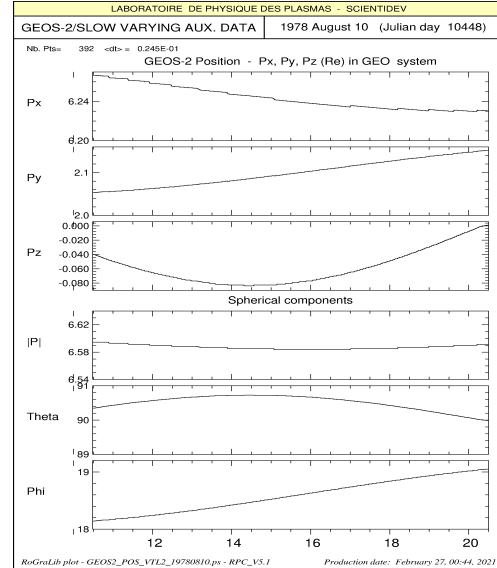


FIG. 9.4: 2D plot done with script `do_2D_plot_GEOPOS.sh`.

9.3 Plotting satellite trajectory

9.3.1 do_2D_plot_GEOPOS.sh

Example :

```
do_VT_plot_2D_GEOPOS.sh 1 19770713 000000 235000
```

is equivalent to :

```
RPC_get_data_GEOPOS 1 1977 07 13
datiso1='RPC_date_time_to_datiso 19770713 000000'
datiso2='RPC_date_time_to_datiso 19770713 235000'
RPC_visu_vectime GEOS2_POS_VTL2_19770713.rff $datiso1 $datiso2
```

result is shown fig. 9.4

9.3.2 do_3D_plot_GEOPOS.sh

Example :

```
do_3D_plot_GEOPOS.sh 1 19770713 000000 235000
```

is equivalent to :

```
RPC_get_data_GEOPOS 1 1977 07 13
datiso1='RPC_date_time_to_datiso 19770713 000000'
datiso2='RPC_date_time_to_datiso 19770713 235000'
RPC_visu_vectime_3D GEOS1_POS_VTL2_19770713.rff $ $datiso1 $datiso2
RPC_change_coordinate_system GEOS1_POS_19770713.rff gse GEOS1_POS_19770713_gse.rff
RPC_visu_vectime_3D GEOS1_POS_19770713_gse.rff 0 0
```

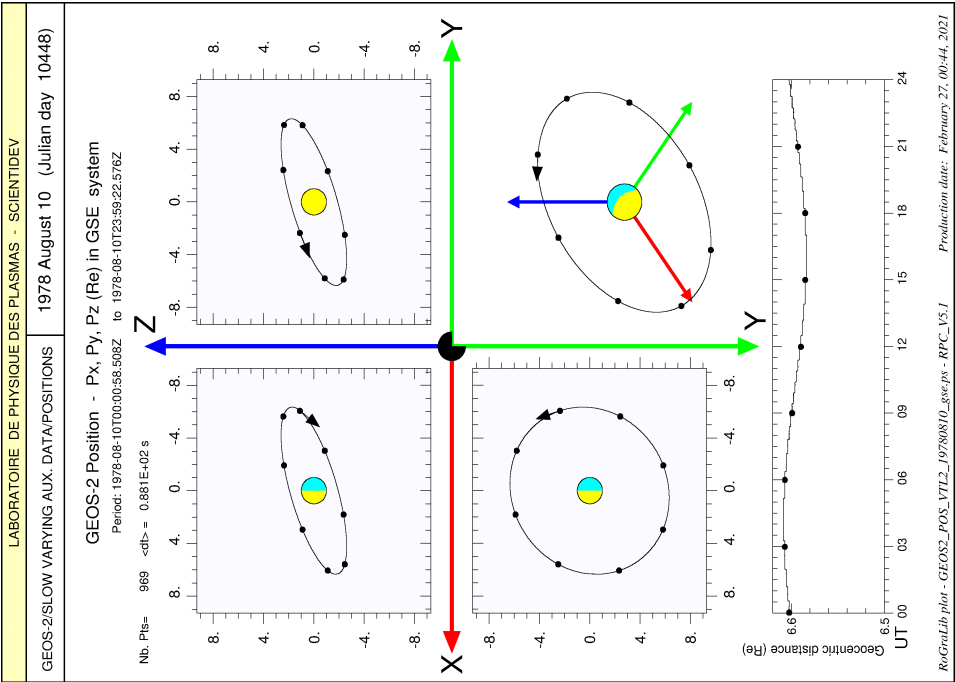


Fig. 9.6: 3-D plot in GSE system done with script `do_3D_plot_GEOPOS.sh`

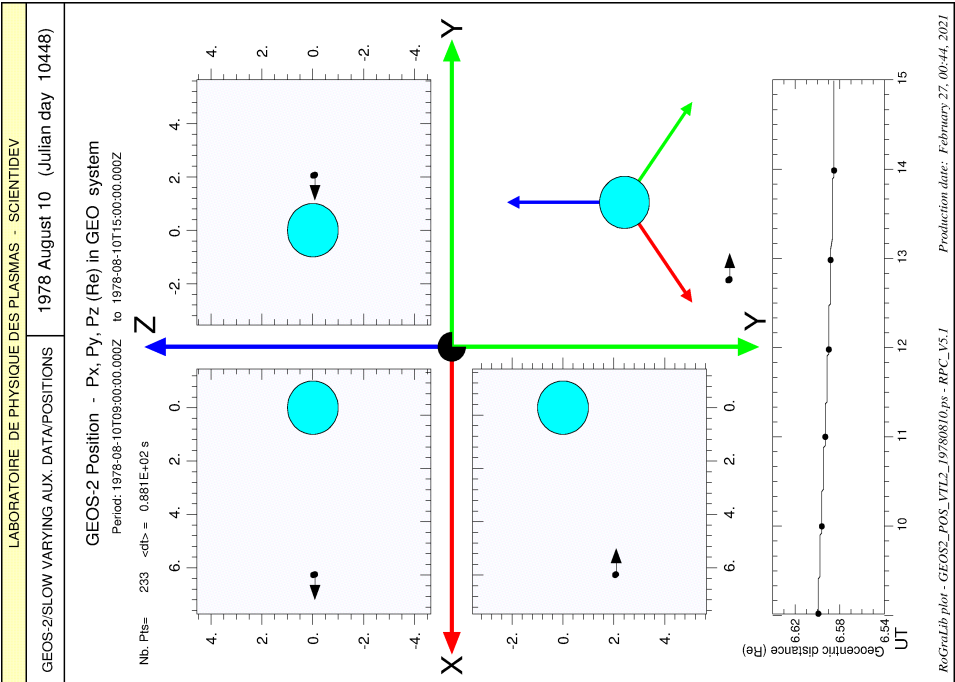


Fig. 9.5: 3-D plot in GEO systemdone with script `do_3D_plot_GEOPOS.sh`. Note that in GEO system, GEOS-2 is syncrone with the Earth, so do not move.

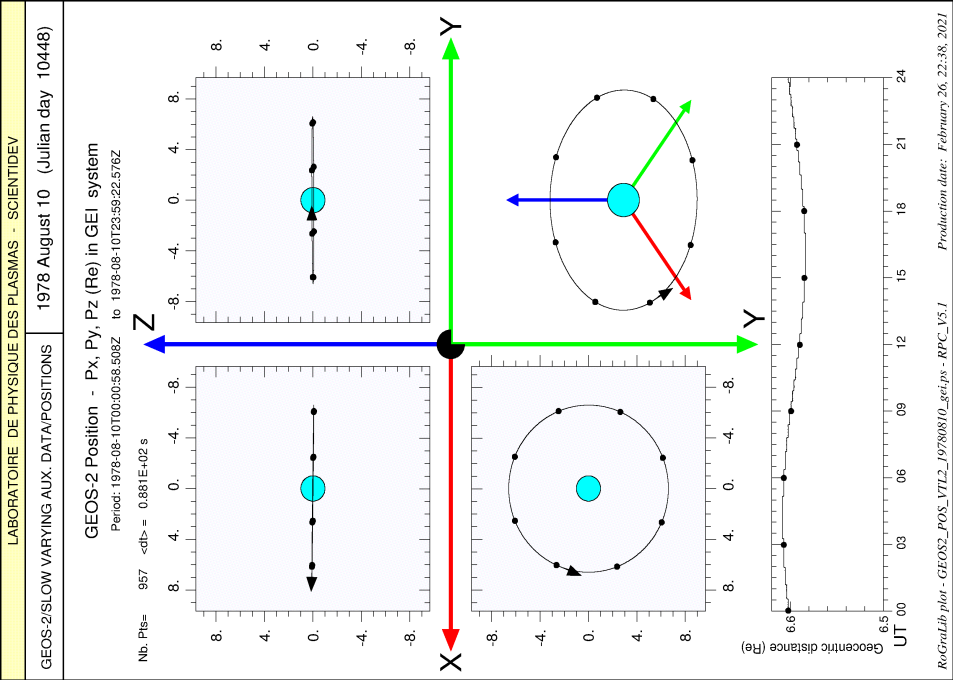


Fig. 9.8: 3-D plot of GEOS-2 in GEI system

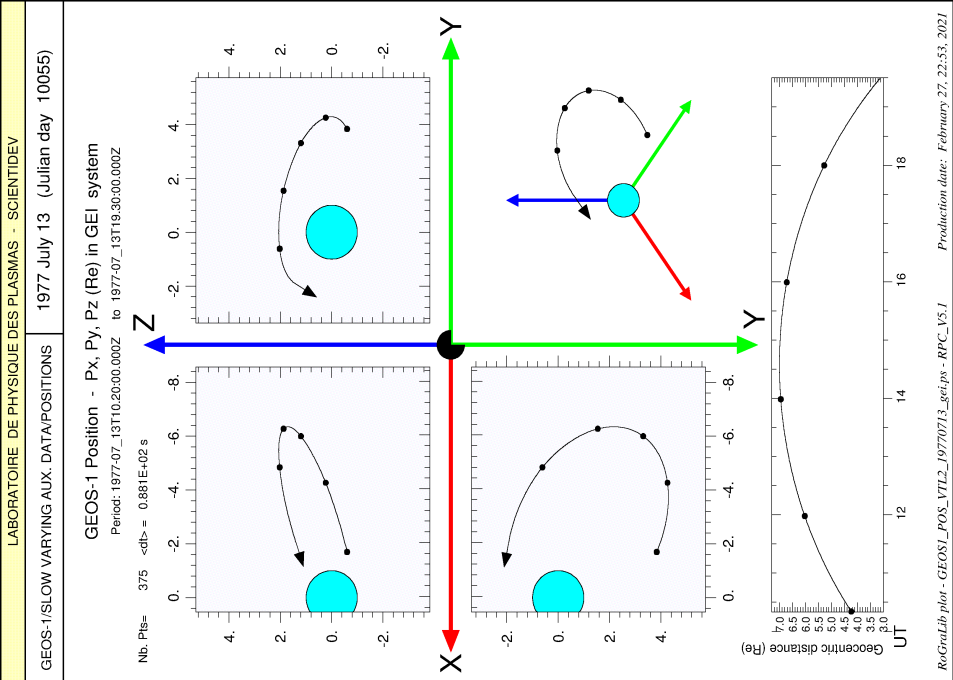


Fig. 9.7: 3-D plot of GEOS-1 in GEI system

9.4 Calculate and visualize waves polarization

9.4.1 do_Polar_plot_GEOULF.sh

Example :

```
do_Polar_plot_GEOULF.sh 19770713 1 120000 150000 512 0. 2.0 tXY 1. -2. 1.
do_Polar_plot_GEOULF.sh 19780810 2 180000 210000 512 0. 11.0 tXY 1. -4. 1.
```

The first command is equivalent to :

```
RPC_get_data_VTL2_GEOULF 1 1977 07 13
RPC_get_data_GEOMAG 1 1977 07 13
RPC_vectime_vdh_to_srv_GEOMAG GEOS1_MAG_VTL2_19770713.rff GEOS1_MAG_VTL2_19770713_srv.rff
datiso1='RPC_date_time_to_datiso 19770713 120000
datiso2='RPC_date_time_to_datiso 19770713 150000

RPC_reduce_time_vectime GEOS1_ULF_VTL2_19770713.rff GEOS1_ULF_VTL2_19770713_red.rff datiso1 datiso2
RPC_vectime_to_mfav_GEOULF GEOS1_ULF_VTL2_19770713.rff GEOS1_MAG_VTL2_19770713_srv.rff GEOS1_ULF_VTL2_19770713_mfav.rff
RPC_vectime_to_spectro GEOS1_ULF_VTL2_19770713_mfav.rff GEOS1_ULF_SPL2_19770713_mfav.rff 512 512 t
RPC_visu_spectro GEOS1_ULF_SPL2_19770713_mfav.rff $datiso1 $datiso2 0. 2. -7. 1. LR 0.5 2.
RPC_spectro_to_polar GEOS1_ULF_SPL2_19770713_mfav.rff copolar.resu
RPC_visu_polar copolar.resu datiso1 datiso2 -3.1 0.5 10. -6. 0.
```

In this example we see here the usefulness of scripts: we execute with a single command line a tedious process to write.

Fig. 9.9 and 9.10 show interest of MFA (Magnetic field aligned) system to a first view of wave polarisation, in particular when we replace the XY components by the circular Left and Right component.

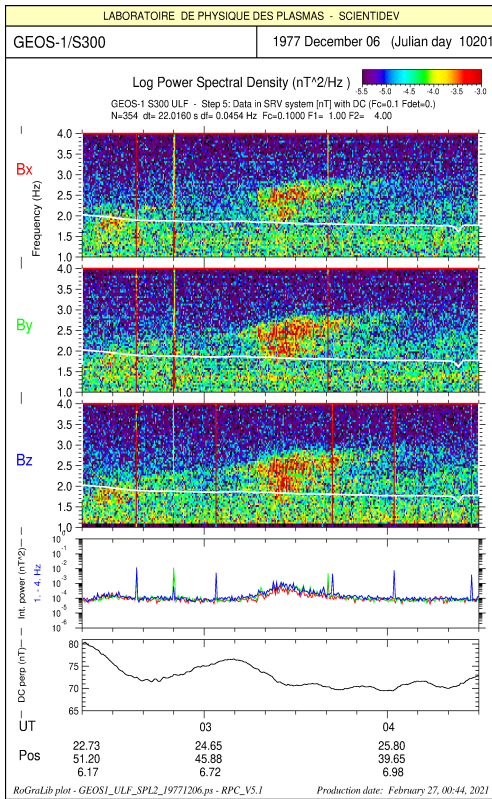


FIG. 9.9: Spectrogram in fixed system, where XYZ are not related to the DC magnetic field.

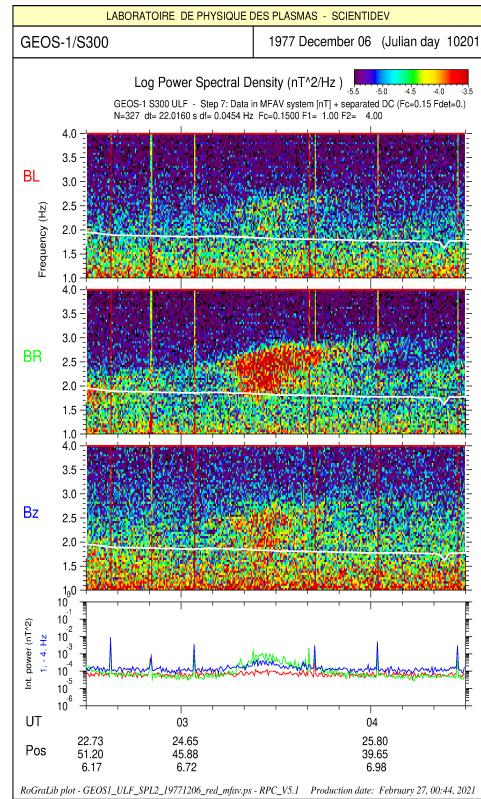


FIG. 9.10: Spectrogram in MFA system, where Z is aligned with the DC magnetic field

9.4.2 Interpretation of polarization parameters

The polarization parameters are shown on Fig. 9.11. Interpretation is following:

- eccentricity : green, so ~ 0.5 , \Rightarrow circular polarization.
- theta K : dark blue, so $\sim 30^\circ$, \Rightarrow *right circular polarization with K parallel to Bo*.
- theta Major axis : green, so $\sim 90^\circ$, \Rightarrow perpendicular to Bo, consistent with the direction of K.

Therefore the circular right polarization was already seen on previous fig. 9.10.

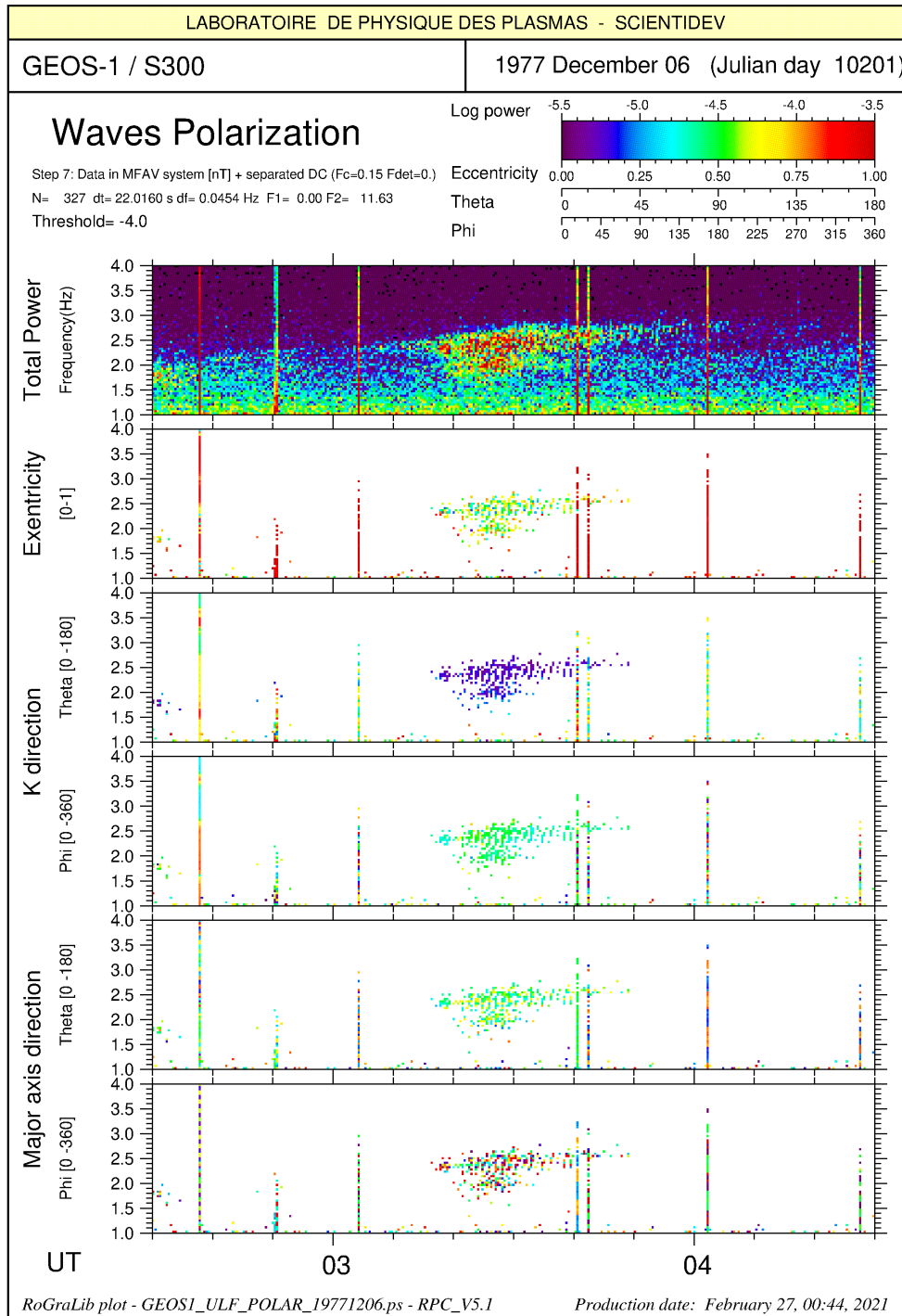


FIG. 9.11: Polarisation parameters in MFA system

9.4.3 Two other examples of wave polarization

9.4.3.1 Low frequency linear polarization

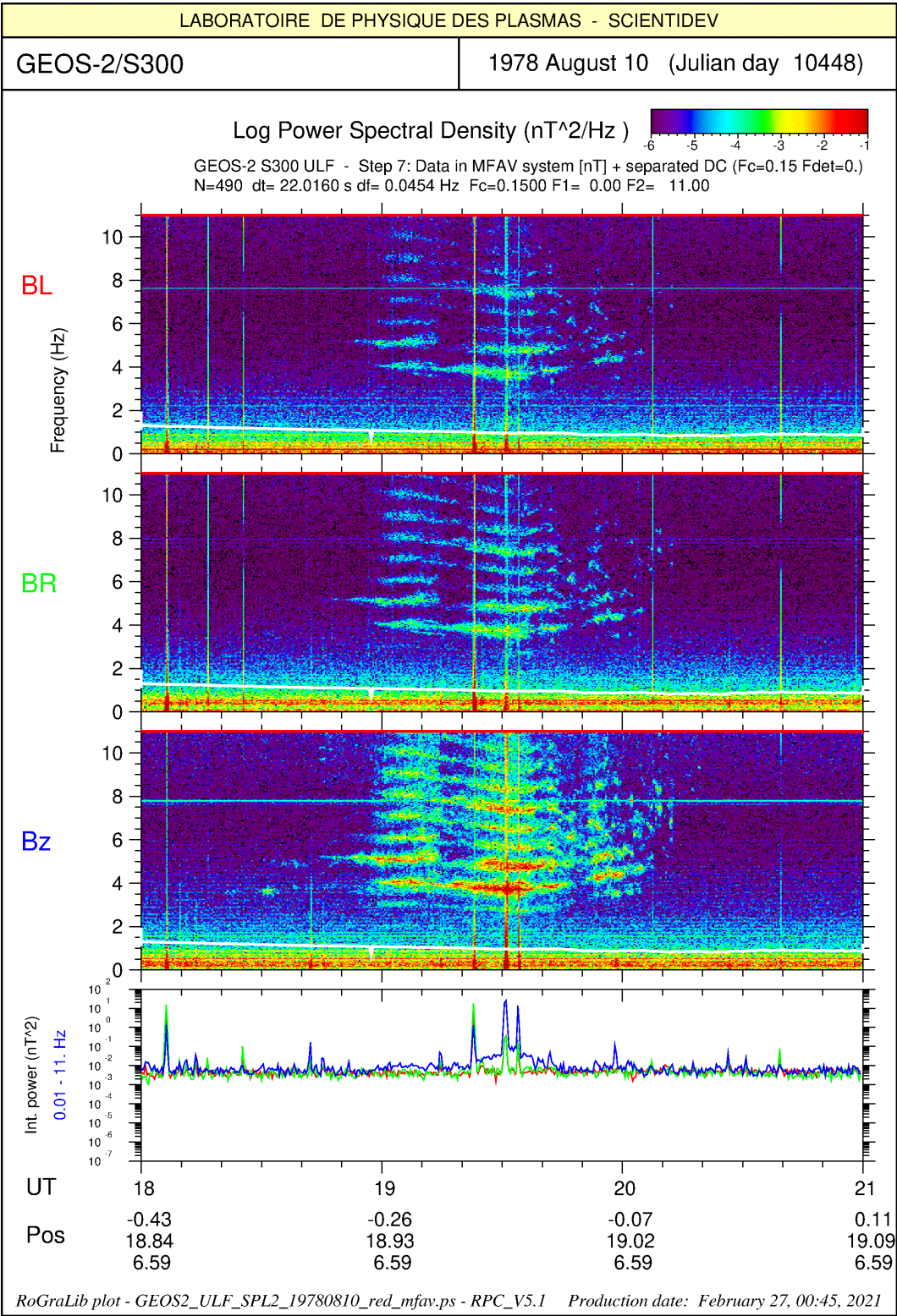


FIG. 9.12: Example of low frequency linear polarisation aligned with the B DC magnetic field

The polarization parameters are shown on Fig. 9.13. Interpretation is following:

eccentricity:	red	so ~ 1	\Rightarrow linear polarization (degenerated ellipse).
theta K:	green	so $\sim 90^\circ$	\Rightarrow linear polarization with $K \perp$ to B_0.
theta major axis:	red or dark purple	so ~ 0 or 180°	\Rightarrow major axis aligned to B_0 , consistent with the direction of K.

Therefore the linear polarization was already seen on previous fig. 9.12.

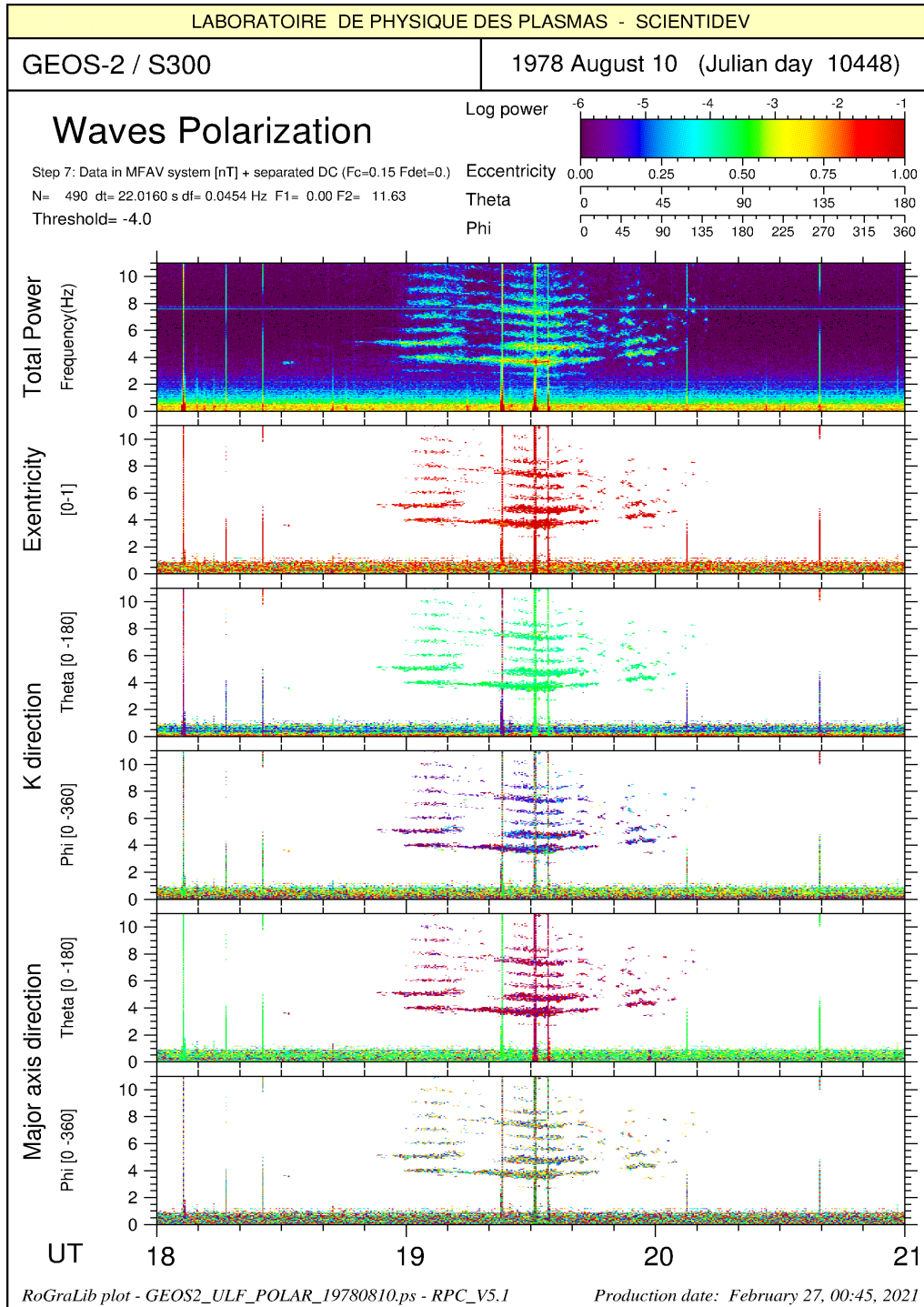


FIG. 9.13: Polarization parameter of linear polarisation aligned with the B DC magnetic field

9.4.3.2 Hight frequency linear polarization

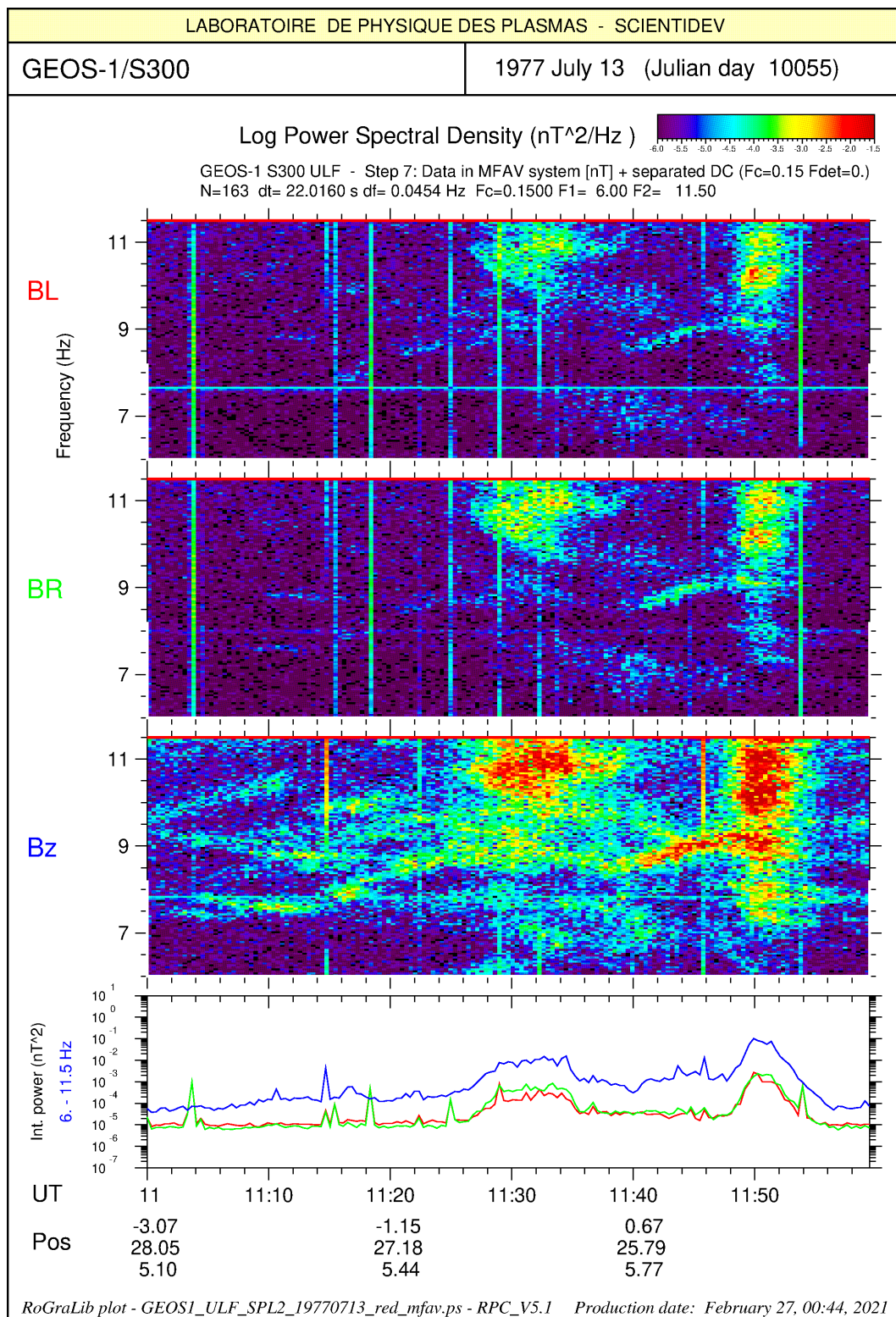


FIG. 9.14: Example of hight frequency linear polarisation aligned with the B DC magnetic field

The polarization parameters are shown on Fig. 9.13. Interpretation is following:

eccentricity:	red	so ~ 1	\Rightarrow linear polarization (degenerated ellipse).
theta K:	green	so $\sim 90^\circ$	\Rightarrow linear polarization with $K \perp$ to B_0.
theta major axis:	red or dark purple	so ~ 0 or 180°	\Rightarrow major axis aligned to B_0 , consistent with the direction of K.

Therefore the linear polarization was already seen on previous fig. 9.14.

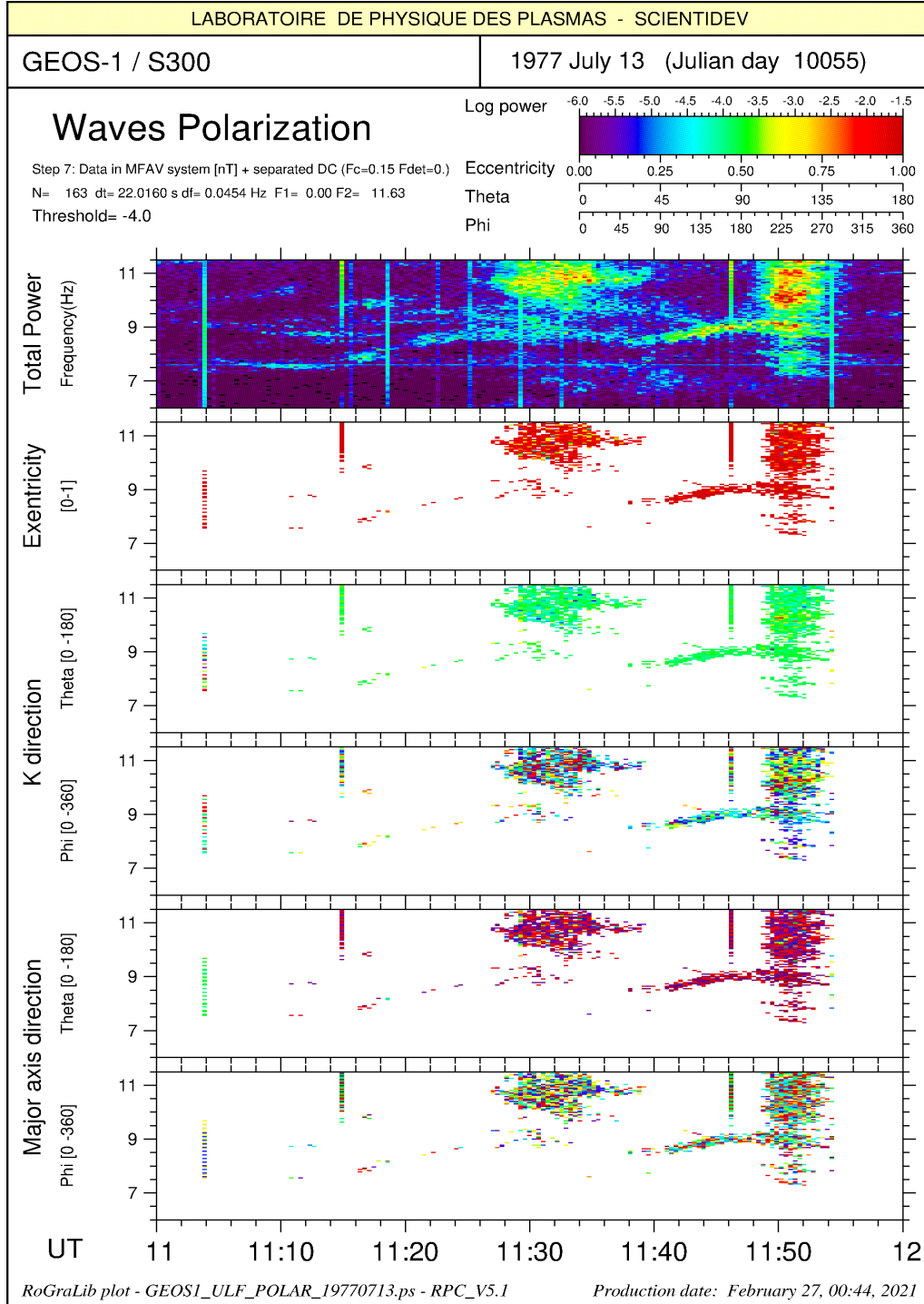


FIG. 9.15: Polarization parameter of linear polarisation aligned with the B DC magnetic field

9.5 ULF and Magnetometer comparison

The rotation of the ULF antennas in the spin plane makes it possible to measure the two components of the continuous field in this plane [10] and therefore to be able to compare them with those of the magnetometer transformed in the same frame. The following script performs this work, and plots these two components on the same graph:

9.5.1 do_compare_ULF_MAG_GEOS.sh

Examples:

```
do_compare_ULF_MAG_GEOS.sh 1 19770713 120000 123000
do_compare_ULF_MAG_GEOS.sh 2 19780810 192800 193000
```

The first command is equivalent to :

```
RPC_get_data_VTL2_GEOULF 1 1977 07 13
RPC_get_data_GEOMAG 1 1977 07 13
RPC_add_DxDy_to_BxBy GEOS1_ULF_VTL2_19770713.rff
RPC_vectime_vdh_to_srv_GEOMAG GEOS1_MAG_VTL2_19770713.rff GEOS1_MAG_VTL2_19770713_srv.rff
datiso1='RPC_date_time_to_datiso 19770713 120000'
datiso2='RPC_date_time_to_datiso 19770713 150000'
RPC_visu_2_vectime GEOS1_ULF_VTL2_19770713.rff GEOS1_MAG_VTL2_19770713_srv.rff $datiso1 $datiso2
```

Fig. 9.16 above show the result. ULF data are in black, while MAG data are in red. The agreement is rather good, and of course ULF fluctuations are visible on the S300 experiment.

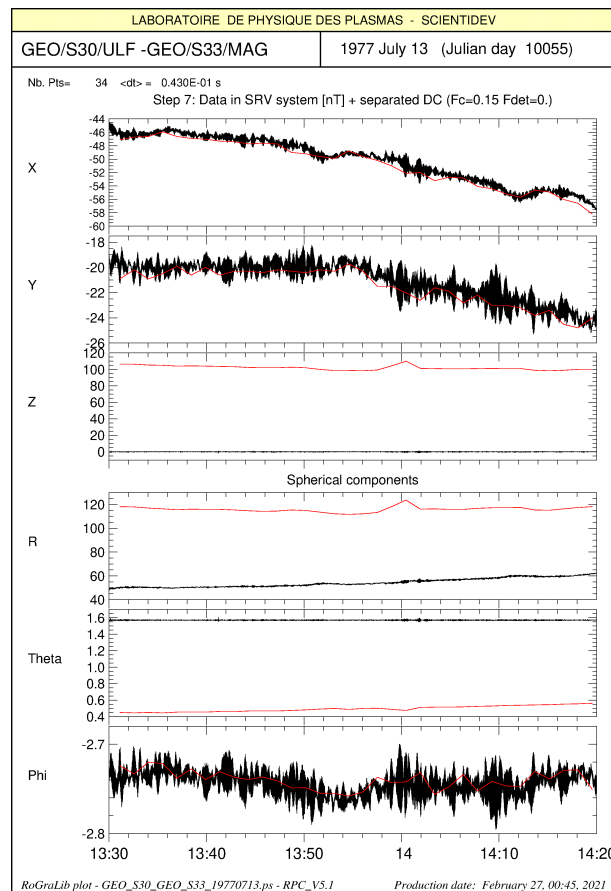


FIG. 9.16: Comparison between two components of the DC field measured by ULF and MAG experiment

Another example of DC field comparison on Fig 9.17

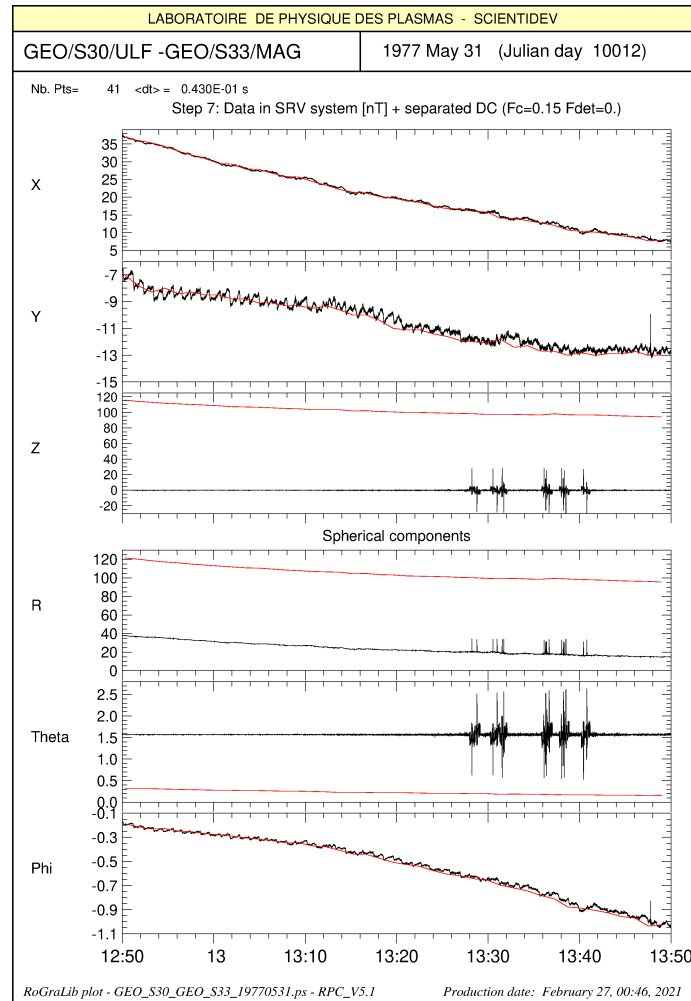


FIG. 9.17: Comparison between two components of the DC field measured by ULF and MAG experiment

9.6 Calculate the orbital parameters of a satellite

9.6.1 do_simulated_orbit_CLUPOS.sh

Example :

```
do_simulated_orbit_CLUPOS.sh 2 20040129 0 0
```

This rather complex script is not detailed here; it can be viewed in the scripts directory. It connects two treatments:

- On the one hand it uses the procedure **RPC_compute_sat_orbit_param** on a GEOS-2 position file in order to calculate the orbital parameters.
- On the other hand it uses these orbital parameters in the procedure **RPC_compute_sat_trajectory** to recalculate the full orbit over a period.

The two figures 9.18 and 9.19 above show the measured GEOS-2 trajectory in the GEI system and the re-computed trajectory in the same system derived from the orbital parameters provided by the procedure **RPC_compute_sat_orbit_param**. Results are perfectly similar.

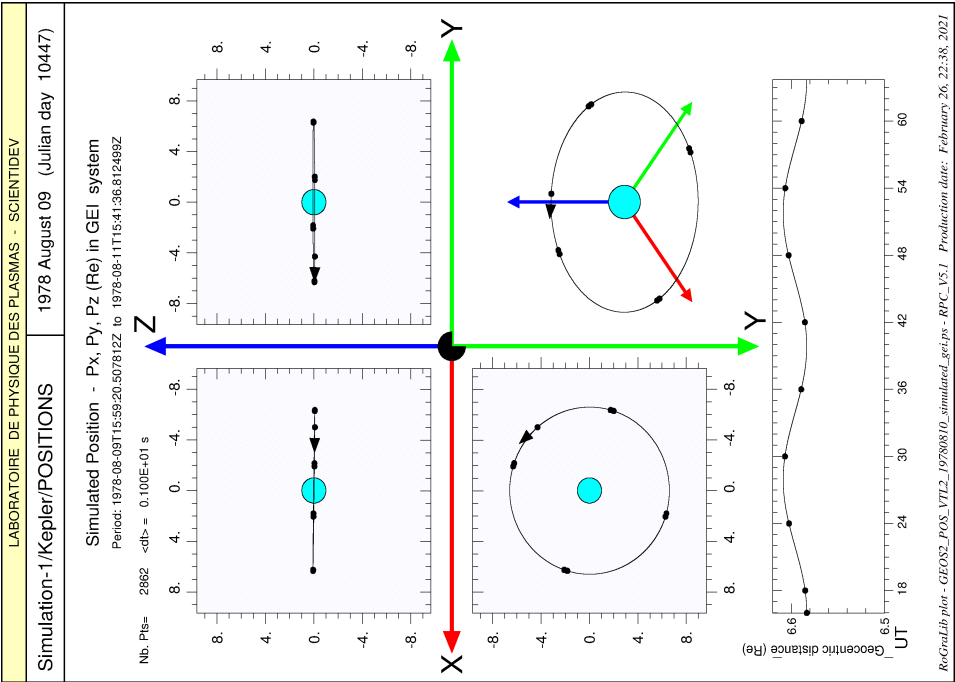


Fig. 9.19: recomputed positions of GEOS-1 from estimated orbital parameters

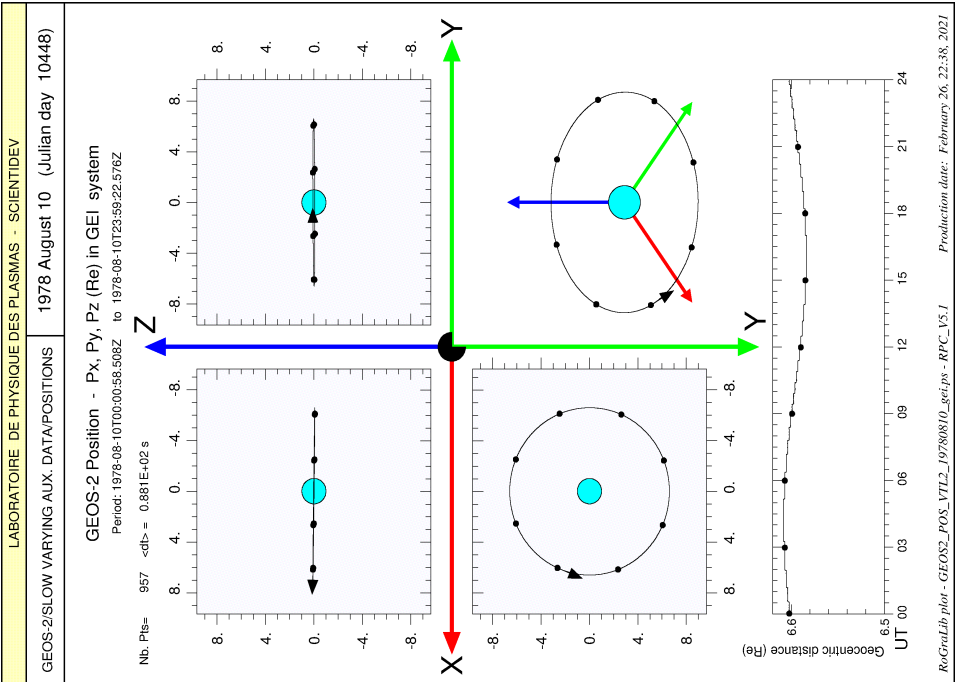


Fig. 9.18: Measured positions of GEOS-1 in GEI system

Chapter 10

TESTS OF GEOS SCRIPTS

The tests of the RPC commands are made from the scripts, which, as we have seen, successively launch several RPC commands for a particular treatment. For each script it was made a test, which are summarized here. Most of the graphical results have been used to illustrate this part.

10.1 test_do_VT_plot_GEOS.sh

This test runs the following scripts: **do_VT_plot_GEOULF.sh** 1 19770713 121000 123000

```
do_VT_plot_GEOULF.sh 2 19780810 192800 193000
do_VT_plot_GEOMAG.sh 1 19770713 000000 233000
do_VT_plot_GEOMAG.sh 2 19780810 060000 190000
do_GEOPOS_VT_plot_2D.sh 2 19780810 102800 203000
```

10.2 test_do_SP_plot_GEOS.sh

This test runs the following scripts:

```
do_SP_plot_GEOULF.sh 1 19770713 120000 150000 512 0. 2.0 t XY 1. -7. 1.
do_SP_plot_GEOULF.sh 2 19780810 180000 210000 512 0. 11.0 t LR 1. -7. -0.5
do_SP_plot_GEOMAG.sh 2 19780810 000000 235959 64 0. 0.02 t XY 0.002 -1. 3.
```

10.3 test_do_POS_3D_plot_GEOS.sh

This test runs the following scripts:

```
do_3D_plot_GEOPOS.sh 2 19780810 0 0
do_3D_plot_GEOPOS.sh 2 19780810 090000 150000
```

10.4 test_do_Polar_plot_GEOS.sh

This test runs the following scripts:

```
do_Polar_plot_GEOULF.sh 1 19770713 120000 150000 512 0. 2.0 t XY 1. -2. 1.
do_Polar_plot_GEOULF.sh 2 19780810 180000 210000 512 0. 11.0 t XY 1. -4. 1.
```

10.5 test_do_ULF_MAG_plot_GEOS.sh

This test runs the following scripts:

```
do_GEOS_ULF_MAG_plot.sh 1 19770713 133000 142000
do_GEOS_ULF_MAG_plot.sh 2 19780810 193000 203000
do_GEOS_ULF_MAG_plot.sh 2 19770531 130000 150000
do_GEOS_ULF_MAG_plot.sh 2 19780102 020000 030000
```

10.6 test_do_join_vectime.sh

This test is used to verify the RPC procedure It performs the following operations :RPC_join_vectime. This allows two vectime files to be merged, thus making it possible for example to view a spectrogram over two days, and in particular around midnight, while the data files are only given over a full day.

It performs the following operations :

```
RPC_get_data_GEOULF  VTL2 2 1980 06 06
if test $? != 0 ; then echo $alarm ; exit 1 ; fi

RPC_get_data_GEOULF  VTL2 2 1980 06 07
if test $? != 0 ; then echo $alarm ; exit 2 ; fi

RPC_join_vectime  GEOS2_ULF_VTL2_19800606.rff GEOS2_ULF_VTL2_19800607.rff GEOS2_ULF_VTL2_198006_06_07.rff
if test $? != 0 ; then echo $alarm ; exit 3 ; fi

RPC_reduce_time_vectime  GEOS2_ULF_VTL2_198006_06_07.rff
                        GEOS2_ULF_VTL2_198006_06_07_red.rff
                        1980-06-06T22:00:00.000Z  1980-06-07T02:00:00.000Z
if test $? != 0 ; then echo $alarm ; exit 4 ; fi

RPC_vectime_to_spectro  GEOS2_ULF_VTL2_198006_06_07_red.rff GEOS2_ULF_SPL2_198006_06_07_red.rff 512 512 t
if test $? != 0 ; then echo $alarm ; exit 5 ; fi

RPC_visu_spectro  GEOS2_ULF_SPL2_198006_06_07_red.rff 0 0 0 0 0 XY 0.5 0.
if test $? != 0 ; then echo $alarm ; exit 6 ; fi
```

We then obtain a spectrogram between June 6 at 10 p.m. and June 7 at 2 a.m., as can be seen in the examples in the following chapter.

10.7 Test of GEOS scripts

This script launch a fuul and long series of tests. First, you have to position yourself in the directory test/test_GEOS. Then, to erase the old tests, we can run the command:

```
clean_all_tests.sh
```

Only the following launch shells will remain:

```
test_do_2D_plot_GEOPOS.sh
test_do_3D_plot_GEOPOS.sh
test_do_compare_ULF_MAG_GEOS.sh
test_do_join_vectime.sh
test_do_Polar_plot_GEOULF.sh
test_do_SP_plot_from_VTL1_GEOS.sh
test_do_SP_plot_GEOS.sh
test_do_VT_plot_GEOS.sh
```

as well as the shell which launches all the tests:

```
run_test_demo.sh
```

10.8 run_test_demo.sh

This shell launch a series of scripts, to check its validity. Time execution is shortness that the previous one, and can be used as a demo of what the scripts can do. List of the scripts launched by this "super-script" is given above.

```

line='-----'

echo ' ====='
echo ' Tests of GEOS script do_VT_plot_GEOS.sh '
echo ' ====='
echo
echo ' Compute and plot ULF or MG waveforms and 2D positions '
echo
echo $line ; echo 'do_VT_plot_GEOULF.sh 1 19770713 121000 123000'
echo $line ; do_VT_plot_GEOULF.sh 1 19770713 121000 123000 > /dev/null
echo $line ; do_VT_plot_GEOULF.sh 2 19780810 192800 193000 > /dev/null
echo $line ; echo 'do_VT_plot_GEOMAG.sh 1 19770713 000000 233000'
echo $line ; do_VT_plot_GEOMAG.sh 1 19770713 000000 233000 > /dev/null
echo $line ; echo 'do_VT_plot_GEOMAG.sh 2 19780810 060000 190000'
echo $line ; do_VT_plot_GEOMAG.sh 2 19780810 060000 190000 > /dev/null
echo $line ; echo 'do_2D_plot_GEOPOS.sh 2 19780810 102800 203000'
echo $line ; do_2D_plot_GEOPOS.sh 2 19780810 102800 203000 > /dev/null

echo
echo ' ====='
echo ' Tests of GEOS script do_SP_plot_GEOS.sh '
echo ' ====='
echo
echo ' Compute and plot ULF or MAG spectrograms '
echo
echo $line ; echo 'do_SP_plot_GEOULF.sh 1 19770713 110000 150000 512 0. 2.0 t XY 1. -6. 0.'
echo $line ; do_SP_plot_GEOULF.sh 1 19770713 110000 150000 512 0. 2.0 t XY 1. -6. 0. > /dev/null
echo $line ; do_SP_plot_GEOULF.sh 2 19780810 180000 210000 512 0. 11.0 t LR 1. -6. -1. > /dev/null
echo $line ; echo 'do_SP_plot_GEOMAG.sh 2 19780810 000000 235959 64 0. 0.02 t XY 0.002 -1. 3.'
echo $line ; do_SP_plot_GEOMAG.sh 2 19780810 000000 235959 64 0. 0.02 t XY 0.002 -1. 3. > /dev/null
echo
echo ' ====='
echo ' Tests of GEOS script do_SP_plot_from_VTL1_GEOULF.sh '
echo ' ====='
echo
echo ' Compute and plot ULF or MAG spectrograms '
echo
echo $line ; echo 'do_SP_plot_from_VTL1_GEOULF.sh 1 19771206 022000 043000 512 1. 4.0 t XY 1. -5.5 -3.'
echo $line ; do_SP_plot_from_VTL1_GEOULF.sh 1 19771206 022000 043000 512 1. 4.0 t XY 1. -5.5 -3. > /dev/null
echo
echo ' ====='
echo ' Tests of GEOS script do_2D_plot_GEOPOS.sh '
echo ' ====='
echo
echo $line ; echo 'do_2D_plot_GEOPOS.sh 2 19780810 102800 203000'
echo $line ; do_2D_plot_GEOPOS.sh 2 19780810 102800 203000 > /dev/null
echo
echo ' ====='
echo ' example of using script do_3D_plot_GEOPOS.sh '
echo ' ====='
echo
echo $line ; echo 'do_3D_plot_GEOPOS.sh 1 19780102 000000 080000 '

```

```

echo $line ;      do_3D_plot_GEOPOS.sh 1 19780102 000000 080000 > /dev/null
echo $line ; echo 'do_3D_plot_GEOPOS.sh 2 19780810 0 0 ' ;
echo $line ;      do_3D_plot_GEOPOS.sh 2 19780810 0 0 > /dev/null
echo $line ; echo 'do_3D_plot_GEOPOS.sh 2 19780810 090000 150000 '
echo $line ;      do_3D_plot_GEOPOS.sh 2 19780810 090000 150000 > /dev/null
echo
echo ' ====='
echo ' Tests of GEOS script do_Polar_plot_GEOULF.sh '
echo ' ====='
echo
echo ' Compute and plot Wave polarisation of GEOS/ULF'
echo
echo '          satnum  yymdd  T1  T2  N_FFT  f1  f2  apod  flpow  Pmin  Pmax  seuil '
echo $line ;      do_Polar_plot_GEOULF.sh 1 19770713 110000 150000 512 .0 2. t 1. -5. 0 -3. > /dev/null
echo $line ; echo 'do_Polar_plot_GEOULF.sh 1 19770713 110000 120000 512 6.0 11.5 t 1. -6. -1.5 -4.'
echo $line ;      do_Polar_plot_GEOULF.sh 1 19770713 110000 120000 512 6.0 11.5 t 1. -6. -1.5 -4. > /dev/null
echo $line ; echo 'do_Polar_plot_GEOULF.sh 1 19771206 023000 043000 512 1.0 4.0 t 1. -5.5 -3.5 -4.'
echo $line ;      do_Polar_plot_GEOULF.sh 1 19771206 023000 043000 512 1.0 4.0 t 1. -5.5 -3.5 -4. > /dev/null
echo $line ; echo 'do_Polar_plot_GEOULF.sh 2 19780810 180000 210000 512 0. 11.0 t 1. -6. -1. -4.'
echo $line ;      do_Polar_plot_GEOULF.sh 2 19780810 180000 210000 512 0. 11.0 t 1. -6. -1. -4. > /dev/null
echo
echo ' ====='
echo ' Tests of GEOS script do_compare_ULF_MAG.sh '
echo ' ====='
echo
echo ' compare GEOS ULF +Despin data with MAG'
echo
echo $line ; echo 'do_compare_ULF_MAG_GEOS.sh 1 19770713 133000 142000'
echo $line ;      do_compare_ULF_MAG_GEOS.sh 1 19770713 133000 142000 > /dev/null
echo $line ;      do_compare_ULF_MAG_GEOS.sh 2 19780810 193000 203000 > /dev/null
echo $line ; echo 'do_compare_ULF_MAG_GEOS.sh 1 19770531 125000 135000'
echo $line ;      do_compare_ULF_MAG_GEOS.sh 1 19770531 125000 135000 > /dev/null
echo $line ;      do_compare_ULF_MAG_GEOS.sh 1 19780102 010000 024600 > /dev/null

```

Execution can be launch by **run_test_demo_with_log.sh** wich allow to keep a copy of the script in a file. The user can look at the screen to check that all are running correctly, or look at the log file **run_test_demo.log** which is given above.

```

=====
Tests of GEOS script do_VT_plot_GEOS.sh
=====

Compute and plot ULF or MG waveforms and 2D positions
-----
do_VT_plot_GEOULF.sh 1 19770713 121000 123000
-----
RPC_get_data_GEOULF          : NORMAL TERMINATION - time exe= 2 s.
STOP visu_vectime.exe        : NORMAL TERMINATION
RPC_visu_vectime              : NORMAL TERMINATION - time exe= 11 s.
-----
do_VT_plot_GEOMAG.sh 1 19770713 000000 233000
-----
RPC_get_data_GEOMAG          : NORMAL TERMINATION - time exe= 0 s.
STOP visu_vectime.exe        : NORMAL TERMINATION
RPC_visu_vectime              : NORMAL TERMINATION - time exe= 0 s.
-----
do_VT_plot_GEOMAG.sh 2 19780810 060000 190000
-----
RPC_get_data_GEOMAG          : NORMAL TERMINATION - time exe= 0 s.
STOP visu_vectime.exe        : NORMAL TERMINATION
RPC_visu_vectime              : NORMAL TERMINATION - time exe= 0 s.
-----

```



```

do_2D_plot_GEOPOS.sh 2 19780810 102800 203000
-----
RPC_get_data_GEOPOS                : NORMAL TERMINATION - time exe= 0 s.
STOP visu_vectime.exe              : NORMAL TERMINATION
RPC_visu_vectime                   : NORMAL TERMINATION - time exe= 0 s.

=====
Tests of GEOS script do_SP_plot_GEOS.sh
=====

Compute and plot ULF or MAG spectrograms

-----
do_SP_plot_GEOULF.sh 1 19770713 110000 150000 512 0. 2.0 t XY 1. -6. 0.
-----
RPC_get_data_GEOULF                : NORMAL TERMINATION - time exe= 1 s.
STOP vectime_to_spectro.exe        : NORMAL TERMINATION
RPC_vectime_to_spectro             : NORMAL TERMINATION - time exe= 12 s.
STOP visu_spectro.exe              : NORMAL TERMINATION
RPC_visu_spectro                   : NORMAL TERMINATION - time exe= 2 s.
STOP visu_ave_spectrum.exe         : NORMAL TERMINATION
RPC_visu_ave_spectrum              : NORMAL TERMINATION - time exe= 1 s.

-----
do_SP_plot_GEOMAG.sh 2 19780810 000000 235959 64 0. 0.02 t XY 0.002 -1. 3.
-----
RPC_get_data_GEOMAG                : NORMAL TERMINATION - time exe= 0 s.
STOP vectime_to_spectro.exe        : NORMAL TERMINATION
RPC_vectime_to_spectro             : NORMAL TERMINATION - time exe= 0 s.
STOP visu_spectro.exe              : NORMAL TERMINATION
RPC_visu_spectro                   : NORMAL TERMINATION - time exe= 0 s.
STOP visu_ave_spectrum.exe         : NORMAL TERMINATION
RPC_visu_ave_spectrum              : NORMAL TERMINATION - time exe= 0 s.

=====
Tests of GEOS script do_SP_plot_from_VTL1_GEOULF.sh
=====

-----
do_SP_plot_from_VTL1_GEOULF.sh 1 19771206 022000 043000 512 1. 4.0 t XY 1. -5.5 -3.
-----
RPC_get_data_GEOULF                : NORMAL TERMINATION - time exe= 0 s.
STOP vectime_L1_to_spectro_L2_GEOULF : NORMAL TERMINATION
RPC_vectime_L1_to_spectro_L2_GEOULF : NORMAL TERMINATION - time exe= 14 s.
STOP visu_spectro.exe              : NORMAL TERMINATION
RPC_visu_spectro                   : NORMAL TERMINATION - time exe= 1 s.
STOP visu_ave_spectrum.exe         : NORMAL TERMINATION
RPC_visu_ave_spectrum              : NORMAL TERMINATION - time exe= 1 s.

=====
Tests of GEOS script do_2D_plot_GEOPOS.sh
=====

-----
do_2D_plot_GEOPOS.sh 2 19780810 102800 203000
-----
RPC_get_data_GEOPOS                : NORMAL TERMINATION - time exe= 0 s.
STOP visu_vectime.exe              : NORMAL TERMINATION
RPC_visu_vectime                   : NORMAL TERMINATION - time exe= 0 s.

=====
example of using script do_3D_plot_GEOPOS.sh
=====

-----
do_3D_plot_GEOPOS.sh 1 19780102 000000 080000
-----
RPC_get_data_GEOPOS                : NORMAL TERMINATION - time exe= 0 s.

```

```

STOP visu_vectime_3D.exe           : NORMAL TERMINATION
RPC_visu_vectime_3D               : NORMAL TERMINATION - time exe= 0 s.
STOP change_coordinate_system.exe : NORMAL TERMINATION
RPC_change_coordinate_system       : NORMAL TERMINATION - time exe= 0 s.
STOP visu_vectime_3D.exe           : NORMAL TERMINATION
RPC_visu_vectime_3D               : NORMAL TERMINATION - time exe= 0 s.

-----
do_3D_plot_GEOPOS.sh 2 19780810 0 0
-----
RPC_get_data_GEOPOS               : NORMAL TERMINATION - time exe= 0 s.
STOP visu_vectime_3D.exe           : NORMAL TERMINATION
RPC_visu_vectime_3D               : NORMAL TERMINATION - time exe= 0 s.
STOP change_coordinate_system.exe : NORMAL TERMINATION
RPC_change_coordinate_system       : NORMAL TERMINATION - time exe= 1 s.
STOP visu_vectime_3D.exe           : NORMAL TERMINATION
RPC_visu_vectime_3D               : NORMAL TERMINATION - time exe= 0 s.

-----
do_3D_plot_GEOPOS.sh 2 19780810 090000 150000
-----
RPC_get_data_GEOPOS               : NORMAL TERMINATION - time exe= 0 s.
STOP visu_vectime_3D.exe           : NORMAL TERMINATION
RPC_visu_vectime_3D               : NORMAL TERMINATION - time exe= 0 s.
STOP change_coordinate_system.exe : NORMAL TERMINATION
RPC_change_coordinate_system       : NORMAL TERMINATION - time exe= 0 s.
STOP visu_vectime_3D.exe           : NORMAL TERMINATION
RPC_visu_vectime_3D               : NORMAL TERMINATION - time exe= 0 s.

=====
Tests of GEOS script do_Polar_plot_GEOULF.sh
=====

Compute and plot Wave polarisation of GEOS/ULF

          satnum   yymmdd   T1    T2    N_FFT   f1   f2   apod f1p Pmi Pma seuil
-----
do_Polar_plot_GEOULF.sh 1 19770713 110000 120000 512 6.0 11.5 t 1. -6. -1.5 -4.
-----
RPC_get_data_GEOULF               : NORMAL TERMINATION - time exe= 0 s.
RPC_get_data_GEOMAG               : NORMAL TERMINATION - time exe= 0 s.
STOP reduce_time_vectime.exe       : NORMAL TERMINATION
RPC_reduce_time_vectime           : NORMAL TERMINATION - time exe= 1 s.
STOP vectime_vdh_to_srv_GEOMAG.exe : NORMAL TERMINATION
RPC_vectime_vdh_to_srv_GEOMAG     : NORMAL TERMINATION - time exe= 0 s.
STOP vectime_to_mfav_GEOULF.exe    : NORMAL TERMINATION
RPC_vectime_to_mfav_GEOULF        : NORMAL TERMINATION - time exe= 4 s.
STOP vectime_to_spectro.exe        : NORMAL TERMINATION
RPC_vectime_to_spectro            : NORMAL TERMINATION - time exe= 1 s.
STOP visu_spectro.exe              : NORMAL TERMINATION
RPC_visu_spectro                  : NORMAL TERMINATION - time exe= 0 s.
STOP spectro_to_polar.exe          : NORMAL TERMINATION
RPC_spectro_to_polar              : NORMAL TERMINATION - time exe= 0 s.
STOP visu_polar.exe                : NORMAL TERMINATION
RPC_visu_polar                    : NORMAL TERMINATION - time exe= 1 s.

-----
do_Polar_plot_GEOULF.sh 1 19771206 023000 043000 512 1.0 4.0 t 1. -5.5 -3.5 -4.
-----
RPC_get_data_GEOULF               : NORMAL TERMINATION - time exe= 1 s.
RPC_get_data_GEOMAG               : NORMAL TERMINATION - time exe= 0 s.
STOP reduce_time_vectime.exe       : NORMAL TERMINATION
RPC_reduce_time_vectime           : NORMAL TERMINATION - time exe= 1 s.
STOP vectime_vdh_to_srv_GEOMAG.exe : NORMAL TERMINATION
RPC_vectime_vdh_to_srv_GEOMAG     : NORMAL TERMINATION - time exe= 0 s.
STOP vectime_to_mfav_GEOULF.exe    : NORMAL TERMINATION
RPC_vectime_to_mfav_GEOULF        : NORMAL TERMINATION - time exe= 7 s.
STOP vectime_to_spectro.exe        : NORMAL TERMINATION
RPC_vectime_to_spectro            : NORMAL TERMINATION - time exe= 2 s.

```

```

STOP visu_spectro.exe           : NORMAL TERMINATION
  RPC_visu_spectro              : NORMAL TERMINATION - time exe= 0 s.
STOP spectro_to_polar.exe       : NORMAL TERMINATION
  RPC_spectro_to_polar          : NORMAL TERMINATION - time exe= 2 s.
STOP visu_polar.exe             : NORMAL TERMINATION
  RPC_visu_polar                : NORMAL TERMINATION - time exe= 0 s.
-----
do_Polar_plot_GEOULF.sh 2 19780810 180000 210000 512 0. 11.0 t 1. -6. -1. -4.
-----
  RPC_get_data_GEOULF           : NORMAL TERMINATION - time exe= 3 s.
  RPC_get_data_GEOMAG           : NORMAL TERMINATION - time exe= 0 s.
STOP reduce_time_vectime.exe    : NORMAL TERMINATION
  RPC_reduce_time_vectime       : NORMAL TERMINATION - time exe= 6 s.
STOP vectime_vdh_to_srv_GEOMAG.exe : NORMAL TERMINATION
  RPC_vectime_vdh_to_srv_GEOMAG : NORMAL TERMINATION - time exe= 0 s.
STOP vectime_to_mfav_GEOULF.exe : NORMAL TERMINATION
  RPC_vectime_to_mfav_GEOULF    : NORMAL TERMINATION - time exe= 10 s.
STOP vectime_to_spectro.exe     : NORMAL TERMINATION
  RPC_vectime_to_spectro        : NORMAL TERMINATION - time exe= 4 s.
STOP visu_spectro.exe           : NORMAL TERMINATION
  RPC_visu_spectro              : NORMAL TERMINATION - time exe= 1 s.
STOP spectro_to_polar.exe       : NORMAL TERMINATION
  RPC_spectro_to_polar          : NORMAL TERMINATION - time exe= 2 s.
STOP visu_polar.exe             : NORMAL TERMINATION
  RPC_visu_polar                : NORMAL TERMINATION - time exe= 0 s.

=====
Tests of GEOS script do_compare_ULF_MAG.sh
=====

compare GEOS ULF +Despin data with MAG

-----
do_compare_ULF_MAG_GEOS.sh 1 19770713 133000 142000
-----
  RPC_get_data_GEOULF           : NORMAL TERMINATION - time exe= 1 s.
  RPC_get_data_GEOMAG           : NORMAL TERMINATION - time exe= 0 s.
STOP vectime_vdh_to_srv_GEOMAG.exe : NORMAL TERMINATION
  RPC_vectime_vdh_to_srv_GEOMAG : NORMAL TERMINATION - time exe= 0 s.
STOP add_DxDy_to_Bx_By.exe      : NORMAL TERMINATION
  RPC_add_DxDy_to_BxBy          : NORMAL TERMINATION - time exe= 21 s.
STOP visu_2_vectime.exe         : NORMAL TERMINATION
  RPC_visu_2_vectime            : NORMAL TERMINATION - time exe= 14 s.
-----
do_compare_ULF_MAG_GEOS.sh 1 19770531 125000 135000
-----
  RPC_get_data_GEOULF           : NORMAL TERMINATION - time exe= 1 s.
  RPC_get_data_GEOMAG           : NORMAL TERMINATION - time exe= 0 s.
STOP vectime_vdh_to_srv_GEOMAG.exe : NORMAL TERMINATION
  RPC_vectime_vdh_to_srv_GEOMAG : NORMAL TERMINATION - time exe= 0 s.
STOP add_DxDy_to_Bx_By.exe      : NORMAL TERMINATION
  RPC_add_DxDy_to_BxBy          : NORMAL TERMINATION - time exe= 17 s.
STOP visu_2_vectime.exe         : NORMAL TERMINATION
  RPC_visu_2_vectime            : NORMAL TERMINATION - time exe= 13 s.

Starting time : 2021-02-27 01:43:24
Ending time   : 2021-02-27 01:46:28
Duration      : 184 sec. (3.06 mn.)

```


Part IV

APPLICATION TO GEOS GROUND DATA

Chapter 11

ACCESS TO GEOS GROUND DATA

11.1 Introduction

During the GEOS-1 and GEOS-2 missions, a mobile station was set up on the ground, equipped with alternative magnetometers of the same type as those of the satellites, and positioned at the conjugate points. As for GEOS, they are waveforms in the range 0.1-11.5 Hz. The data of 4 measurement campaigns were found, at the positions and at the following dates:

HUSAFELL (64.40 N, 20.80 W), Jul 4, 1977-Sep 22, 1977.

KITDALEN (69.21 N, 20.18 E), Jan 29, 1980-Mar 21, 1980 & Jan 31, 1981-Mar 28, 1981.

SKIBOTN (69.35 N, 20.36 E), Jan 15, 1982-Mar 07, 1982.

11.2 Downloads RFF data files

GEOS Ground data files are available from the CDPP site: <https://cdpp-archive.cnes.fr/>
Rubrique: European GEOS mission - ground data

The downloaded files are of the following type:

GEOGRD/ULF VTL1 : HUS_VTL1_19770713.rff

GEOGRD/ULF VTL2 : HUS_VTL2_19770713.rff

HUS for HUSAFELL, KIT for KITDALEN and SKI for SKIBOTN.

11.3 Setting up a local database

It is recommended to make a database of the files in RFF format used by the RPC commands. The RPC package comes with a minimal database, just containing samples for testing. These directories can be anywhere on your machine, even on a different disk or on the network as long as its path is correctly indicated in the RPC `RPC_config.bash`.

You just have to copy them into the report, and then the sub-directories will be automatically created when filling with the command **`RPC_put_rff_database`**.

This database is organized as follows :

```

|_ULF
|_VTL1
|_1977
|_1977_07
|_1977_08
|_1977_09
|_1980
|_1980_01
|_1980_02
|_1980_03
|_1981
|_1981_01
|_1981_02
|_1981_03
|_1982
|_1982_01
|_1982_02
|_1982_03
|_VTL2
|_1977
|_1977_07
|_1977_08
|_1977_09
|_1980
|_1980_01
|_1980_02
|_1980_03
|_1981
|_1981_01
|_1981_02
|_1981_03
|_1982
|_1982_01
|_1982_02
|_1982_03

```

11.4 Interest of a local database

RPC processing commands request one or more RFF files as arguments. These files can be in the current directory or elsewhere, if the path is correctly entered there is no problem. But if you are working on many files, it is more pleasant to store them directly in the local database. First, they can be easily found with the commands described in the next chapter, and then this database can be in another directory, a disk« data » for example.

11.5 Commands relating to the database

After downloading an rff file, you can *move the RFF data files in the local data base* by :

```
RPC_put_rff_database name.rff
```

If one wants *copy the RFF files in the local data base*, and not move it, one use the command:

```
RPC_copy_rff_database name.rff
```

Conversely, we can *copy the file in the working directory* by the commands:

```
RPC_get_data_GEOGRD Level year month day
Level : VTL1 or VTL2
```

If we want to know what is in the RFF database, *the list of all RFF files* by experience is obtained by the commands :

```
RPC_list_rff_database GEOGRD ULF
```


Chapter 12

WORKING ON GEOS GROUND RFF FILES

12.1 Main types of RFF files

We will mainly handle files of type VecTime [3] which are vectors indexed by time, and files of type Spectrogram [3] which are series of dated spectra.

To go fast, you can go directly to chapter« USING SCRIPTS »

12.1.1 Example of VecTime file names :

HUS_VTL2_19770713.rff
KIT_VTL2_19770713.rff
SKI_VTL2_19770713.rff

12.1.2 Example of Spectrograms file names :

The nomenclature is like :

HUS_SPL2_19770713.rff

Note that the viewing commands will create files like :

HUS_SPL2_19770713.ps

and for the average spectra :

HUS_SPL2_19770713_ave_spe.ps
HUS_SPL2_19770713_ave_spe.pdf
HUS_SPL2_19770713_ave_spe.png

12.1.3 Checking the validity of an RFF file

The validity of an RFF file can be checked by the command:

RPC_check_rff toto.rff

In the event of non-compliance, it will be reported, and the reasons will be displayed in the check_rff.out file. Note that the validity of all the rff files found in a toto directory can be checked with the command

RPC_check_rff_dir toto_dir

12.2 Special commands for GEOGRD

RPC_menu_GEOGRD

Displays on screen the general menu of commands, as well as special commands to GEOS.

RPC_vectime_calibration_GEOGRD

Calibrate a VTL1 file and produce VTL2

Use:

RPC_vectime_calibration_GEOGRD VTL1.rff VTL2.rff Fdet Fc F1 F2 N-Kern N-shift Apod

With :

VTL1.rff : name of an input vectime RFF file
 VTL2.rff : name of an output vectime RFF file
 Fdet : detrend frequency (0. for classis despin)
 Fc : Frequency cut-off for calibration
 Fmin : frequency min for filtering (Min=0 => Fc)
 Fmax : frequency max for filtering (Max=0 => Nyquist)
 N-Kern : Kernel size
 N-shift : for sliding window
 Apod : trapezium (t) or Gaussian (g)

RPC_vectime_L1_to_spectro_L2_GEOGRD

Produces a calibrated spectrogram from an uncalibrated VTL1 file

Use:

RPC_vectime_L1_to_spectro_L2_GEOGRD VTL1.rff SPL2.rff Fdet Fc F1 F2 N_Kern N_shift Apod

With :

VTL1.rff : name of an input vectime RFF file
 SPL2.rff : name of an output spectrogram RFF file
 Fdet : detrend frequency (0. for classis despin)
 Fc : Frequency cut-off for calibration
 Fmin : frequency min for filtering (Min=0 = Fc)
 Fmax : frequency max for filtering (Max=0 = Nyquist)
 N_Kern : Kernel size
 N_shift : for sliding window
 Apod : trapezium (t) or nothing(n)

Example :

RPC_vectime_L1_to_spectro_L2_GEOGRD HUS_VTL1_19770713.rff HUS_SPL2_19770713.rff 0. 0.1 0.5 0. 512 512 t

RPC_vectime_to_mfav_GEOGRD

Change coordinate of GEOGRD/ULF from NWV to MFAV

Usage :

RPC_vectime_to_mfav_GEOGRD VTL2_ulf_nwv VTL2_ulf_mfav

With :

VTL2_ulf_srv : name of a GEOGRD RFF file
 VTL2_ulf_mfav : name the GEOGRD RFF file in mfav

12.3 Make an ULF spectrogram

12.3.1 From a VTL1 file

Launch the command (detailed in the previous section):

RPC_vectime_L1_to_spectro_L2_GEOGRD VTL1.rff SPL2.rff Fdet Fc F1 F2 N_Kern N_shift Apod

Example :

RPC_vectime_L1_to_spectro_L2_GEOGRD HUS_VTL1_19770713.rff HUS_SPL2_19770713.rff 0. 0.1 0.5 0. 1024 2 g

When the SPL2.rff file is created, it can be viewed with the command:

RPC_visu_spectro SP.rff datiso1 datiso2 f1 f2 pmin pmax XY fi1 fi2

With :

SP.rff : name of a spectro RFF file
 datiso1 datiso2 : iso date/time first and end
 f1 f2 : frequency bounds to plot
 pmin pmax : min max power for dynamic colors
 XY or LR : for XYZ or Left Right Z
 fi1, fi2 : frequency bounds for integrated power

12.3.2 From a VTL2 file

This is done with the generic procedure (valid whatever the VecTime file):

RPC_vectime_to_spectro VT.rff SP.rff N_Kern N_shift Apod

With :

VT.rff : name of an input vectime RFF file
 SP.rff : name of an output spectrogram RFF file
 N_Kern : Kernel size
 N_shift : for sliding window
 Apod : trapezium (t), Gaussian (g) or nothing (n)

Example :

RPC_vectime_to_spectro HUS_VTL2_19770713.rff HUS_SPL2_19770713.rff 512 512 t

The visualization is done with the same command below :

RPC_visu_spectro SP.rff datiso1 datiso2 f1 f2 pmin pmax XY fi1 fi2

12.3.3 Spectrogram visualization

Just run the following generic command:

RPC_visu_spectro SP.rff datiso1 datiso2 f1 f2 pmin pmax XY fi1 fi2

With :

SP.rff : name of a spectro RFF file
 datiso1 datiso2 : iso date/time first and end
 f1 f2 : frequency bounds to plot
 pmin pmax : min max power for dynamic colors
 XY or LR : for XYZ or Left Right Z
 fi1, fi2 : frequency bounds for integrated power

Example :

RPC_visu_spectro toto_SP.rff 1977-07-13T09:00:00.000Z 1977-07-13T11:00:00.000Z 0. 5. 0. 0. XY 0.2 10.

or again:

RPC_visu_spectro toto_SP.rff 0 0 0. 200. -9.6 -3. LR 0. 200.

12.4 Visualize an average spectrum

The file SPL2.rff being created, one can visualize the average spectrum of all the spectra of the SPL2 by the command :

RPC_visu_ave_spectrum SP.rff datiso1 datiso2 f1 f2 pmin pmax

With :

SP.rff : name of a spectro RFF file
 datiso1 datiso2 : iso date/time first and end
 f1 f2 : frequency bounds to plot
 pmin pmax : min max power for dynamic colors (log values)
 XY or LR : for XYZ or Left Right Z
 info_orb,info_FGM : y/n y/n

Example :

RPC_visu_ave_spectrum toto_SP.rff 1977-07-13T09:00:00.000Z 1977-07-13T11:00:00.000Z 0. 5. 0. 0. XY y n

RPC_visu_ave_spectrum toto_SP.rff 0 0 0. 0. 0. 0. XY y n

12.5 Draw a waveform

Just run the generic procedure :

RPC_visu_vectime name.rff

Example :

RPC_visu_vectime HUS_VTL2_19770713.rff \$T1 \$T2

Avec \$T1 et \$T2 variables containing the ISO start and end date of the period to be displayed.

Example: T1=2002-09-23T10:30:00.000Z T2= 2002-09-23T10:33:00.000Z

for [T1 T2] = [0 0] the whole file will be displayed.

12.6 Extract part of a VecTime file

Note that we can also extract the interesting portion of the file before viewing or other. This is done with the command:

RPC_reduce_time_vectime toto_VT.rff toto_VT.rff datiso1 datiso2

With :

toto_VT.rff : name of an input vectime RFF file
 toto_VT.rff : name of an output vectime RFF file
 datiso1 datiso2 : date in ISO format

Example :

RPC_reduce_time_vectime HUS_VTL2_19770713.rff HUS_VTL2_19770713_red.rff 1977-07-13T09:12:00Z 1977-07-13T09:30:00Z

We can then do :

RPC_visu_vectime HUS_VTL2_19770713_red.rff 0 0

The start and end time set to 0 0 means that the entire file will be viewed.

12.7 Waves polarisation computation

12.7.1 Transformation in MFAV coordinate system

The calculation of the polarization of the waves requires that we first go to the MFAV (Magnetic Field Aligned) frame [5]. This is done by the above command :

```
RPC_vectime_to_mfav_GEOGRD file_ULF.rff file_ULF_mfav.rff file_MAG.rff
```

Example :

```
RPC_vectime_to_mfav_GEOGRD HUS_VTL2_19770713.rff HUS_VTL2_19770713_mfav.rff
```

12.7.2 Calculation and plot of the polarization parameters

Once the ULF calibrated waveform file is in the MFA frame, we make the spectrogram using the generic command:

```
RPC_vectime_to_spectro.
```

Example :

```
RPC_vectime_to_spectro HUS_VTL2_19770713_mfa.rff HUS_SPL2_19770713_mfa.rff 512 512 t
```

The visualization of the spectrogram in the reference line of force can already give information, especially if it is represented in circular coordinates « Left, Right, Z » [9] by the command :

```
RPC_visu_spectro HUS_VTL2_19770713_mfa.rff 0 0 0. 0. 0. LR 0.2 2.
```

The polarization parameters calculated by method [6] are obtained by the command :

```
RPC_spectro_to_polar VTL2_mfa.rff copolar.resu
```

With :

```
VTL2_mfa.rff : name of an input RFF VTL2 file in MFA
copolar.resu : file name for polar results
```

Finally, the polarization parameters are displayed using the command:

```
RPC_visu_polar copolar.resu datiso1 datiso2 threshold f1 f2
```

With :

```
copolar.resu : name of file created by RPC_spectro_to_polar
datiso1 datiso2 : iso date/time first and end
threshold for visualization ex : -3.1
f1r f2r : frequency bounds for visualization
p1 p2 : for spectrogram power limits
```

the "threshold" value should be chosen with care, because it allows the polarization of the phenomenon to be isolated from that of the surrounding noise. Various tests may be necessary.

Example :

```
RPC_visu_polar copolar.resu datiso1 datiso2 -3.1 0.5 10. -6. 0.
```


Chapter 13

USING GEOGRD SCRIPTS

Scripts are bash files that string together multiple RPC commands, to avoid the tedious task of typing them all one after the other. The RPC package ships with a number of scripts that perform the most common processing. The user can use this as inspiration to write their own treatments.

13.1 Draw a waveform

13.1.1 do_VT_plot_GEOGRD.sh

Examples :

```
do_VT_plot_GEOGRD.sh 19770706 130200 131000 L1
```

```
do_VT_plot_GEOGRD.sh 19770706 130200 131000 L2
```

These two commands gives the plots shown on fig 13.1.

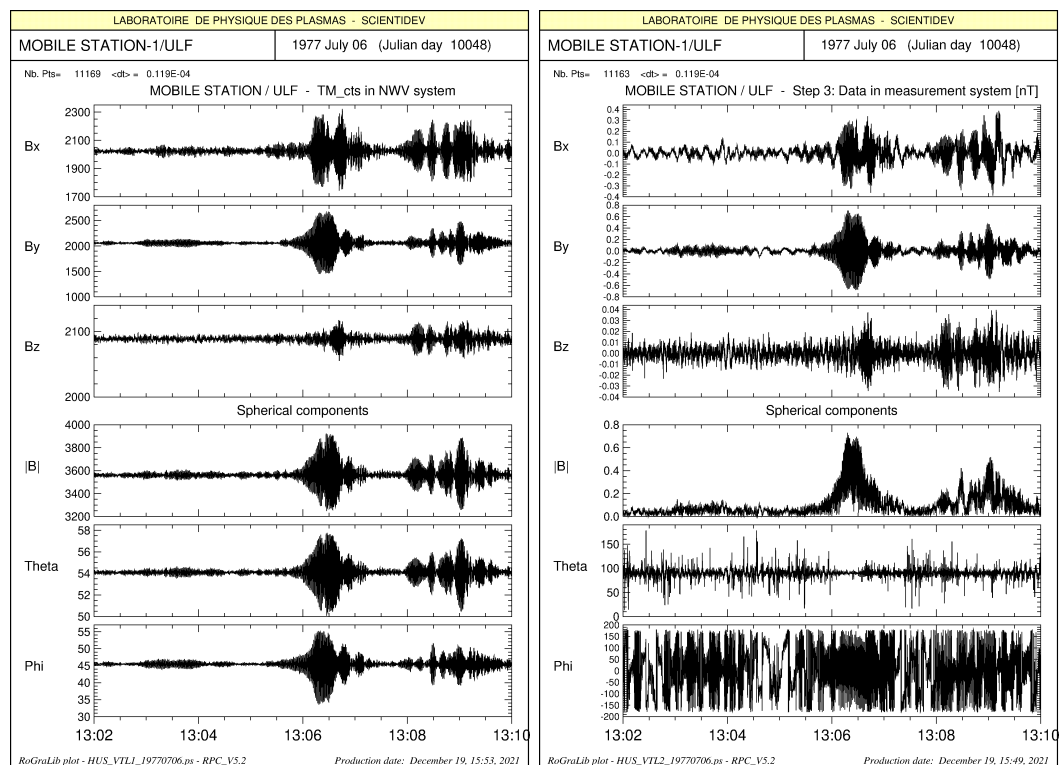


FIG. 13.1: Plotting waveform with `do_VT_plot_GEOGRD.sh` script.

13.2 Make a spectrogramme

13.2.1 do_SP_plot_from_VTL1_GEOGRD.sh

Example :

```
do_SP_plot_from_VTL1_GEOGRD.sh 19770706 120000 150000 512 0. 1.5 t XY 1. -7.2 -2.2
```

arguments :date, H_debut, H_fin, Nbp_FFT, F1, F2, Apod, XY ou LR, F1_p_int, seuil_min, seuil_max

The script :

```
do_SP_plot_from_VTL1_GEOGRD.sh 19770706 120000 150000 512 0. 1.5 t XY 1. -7.2 -2.2
```

is equivalent to sequencing the following commands by hand :

```
RPC_get_data_GEOGRD VTL1 1977 07 06
RPC_vectime_to_spectro HUS_VTL1_19770706.rff HUS_SPL2_19770706.rff 512 512 t
datiso1='RPC_date_time_to_datiso 19770706 120000
datiso2='RPC_date_time_to_datiso 19770706 150000'
RPC_visu_spectro 19770706 $datiso1 $datiso2 0. 1.5 -7.2 -2.2 XY 0. 1.5
RPC_visu_ave_spectrum 19770706 $datiso1 $datiso2 0. 1.5 -7.2 -2.2 XY n n
```

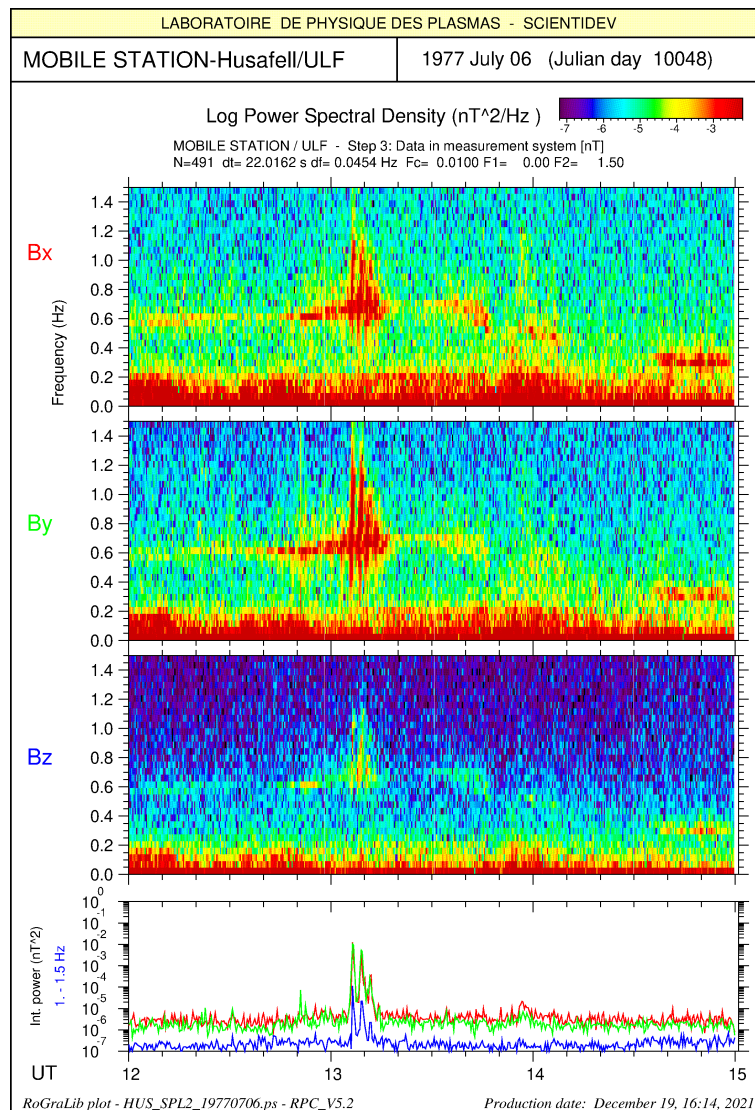


FIG. 13.2: ULF spectrogram plot done with with `do_SP_plot_from_VTL1_GEOGRD.sh` script.

13.2.2 do_SP_plot_from_VTL2_GEOGRD.sh

Example :

do_SP_plot_from_VTL2_GEOGRD.sh 19770706 120000 150000 512 0. 1.5 t XY 1. -7.2 -2.2

arguments :date, H_debut, H_fin, Nbp_FFT, F1, F2, Apod, XY ou LR, F1_p_int, seuil_min, seuil_max

The script :

do_SP_plot_from_VTL2_GEOGRD.sh 19770706 120000 150000 512 0. 1.5 t XY 1. -7.2 -2.2

is equivalent to sequencing the following commands by hand :

RPC_get_data_GEOGRD VTL2 1977 07 06

RPC_vectime_to_spectro HUS_VTL2_19770706.rff HUS_SPL2_19770706.rff 512 512 t

datiso1='RPC_date_time_to_datiso 19770706 120000

datiso2='RPC_date_time_to_datiso 19770706 150000'

RPC_visu_spectro 19770706 \$datiso1 \$datiso2 0. 1.5 -7.2 -2.2 XY 0. 1.5

RPC_visu_ave_spectrum 19770706 \$datiso1 \$datiso2 0. 1.5 -7.2 -2.2 XY n n

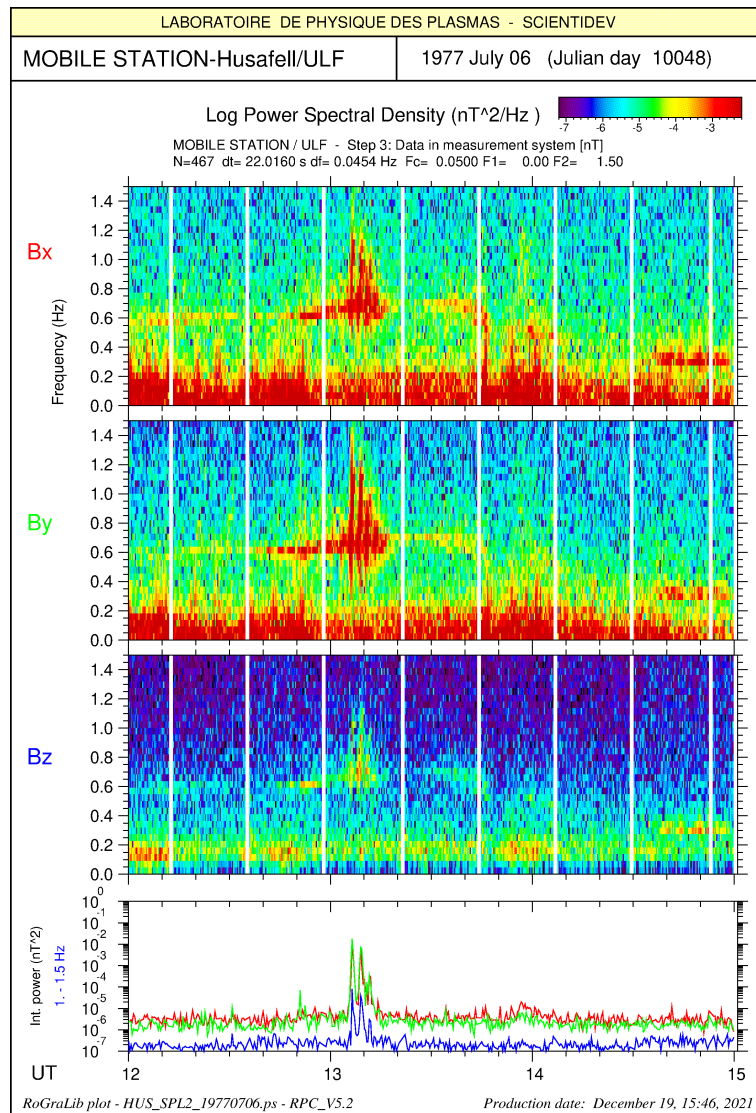


FIG. 13.3: ULF spectrogram plot done with with `do_SP_plot_from_VTL2_GEOGRD.sh` script.

Note that the wave calibration enlarge the data gaps.

13.3 Calculate and visualize waves polarization

13.3.1 do_Polar_plot_GEOGRD.sh

Example :

```
do_Polar_plot_GEOGRD.sh 19770713 1 120000 150000 512 0. 2.0 t XY 1. -2. 1.
do_Polar_plot_GEOGRD.sh 19780810 2 180000 210000 512 0. 11.0 t XY 1. -4. 1.
```

The first command is equivalent to :

```
RPC_get_data_VTL2_GEOGRD 1 1977 07 13
RPC_get_data_GEOMAG 1 1977 07 13
RPC_vectime_vdh_to_srv_GEOMAG GEOS1_MAG_VTL2_19770713.rff GEOS1_MAG_VTL2_19770713_srv.rff
datiso1='RPC_date_time_to_datiso 19770713 120000
datiso2='RPC_date_time_to_datiso 19770713 150000

RPC_reduce_time_vectime GEOS1_ULF_VTL2_19770713.rff GEOS1_ULF_VTL2_19770713_red.rff datiso1 datiso2
RPC_vectime_to_mfav_GEOGRD GEOS1_ULF_VTL2_19770713.rff GEOS1_MAG_VTL2_19770713_srv.rff GEOS1_ULF_VTL2_19770713_mfav.rff
RPC_vectime_to_spectro GEOS1_ULF_VTL2_19770713_mfav.rff GEOS1_ULF_SPL2_19770713_mfav.rff 512 512 t
RPC_visu_spectro GEOS1_ULF_SPL2_19770713_mfav.rff $datiso1 $datiso2 0. 2. -7. 1. LR 0.5 2.
RPC_spectro_to_polar GEOS1_ULF_SPL2_19770713_mfav.rff copolar.resu
RPC_visu_polar copolar.resu 0 0 -3. 0. 2.
```

In this example we see here the usefulness of scripts: we execute with a single command line a tedious process to write.

Fig. 13.4 and 13.5 show two examples of spectrograms in MFAV system, interesting to a first view of wave polarisation, in particular when we replace the XY components by the circular Left and Right component. Note that the B_{DC} being close to the local vertical, there is not so much difference between NVW system and MFAV one.

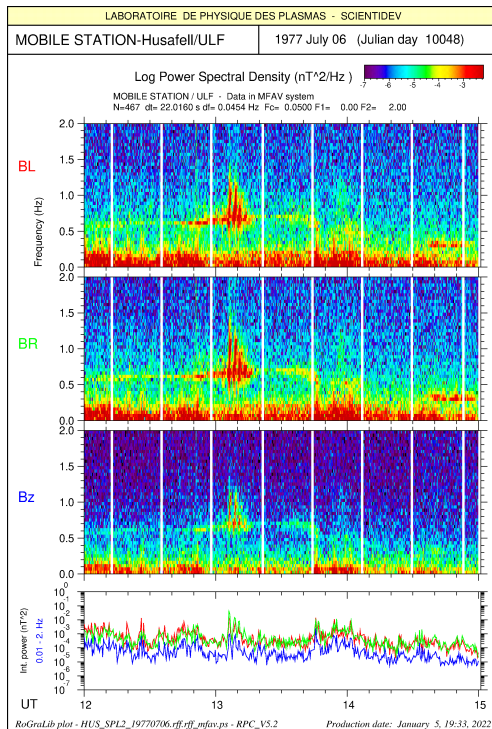


FIG. 13.4: Spectrogram in MFA system, for 0-2 Hz frequency range

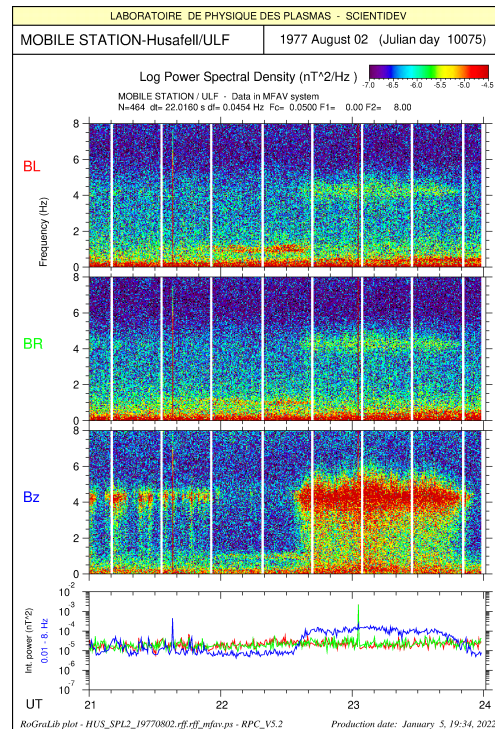


FIG. 13.5: Spectrogram in MFA system, for 0-8 Hz frequency range

13.3.2 Interpretation of polarization parameters

The polarization parameters are shown on Fig. 13.6. Interpretation is following:

- eccentricity : red, so $\sim 1.$, \Rightarrow linear or very elliptical polarization.
- theta K : red or dark blue, so ~ 0 or 180° , \Rightarrow **linear polarization with K parallel to Bo.**
- theta Major axis : green, so $\sim 90^\circ$, \Rightarrow perpendicular to Bo, consistent with the direction of K.

Therefore the linear polarization was already seen on previous fig. 13.4.

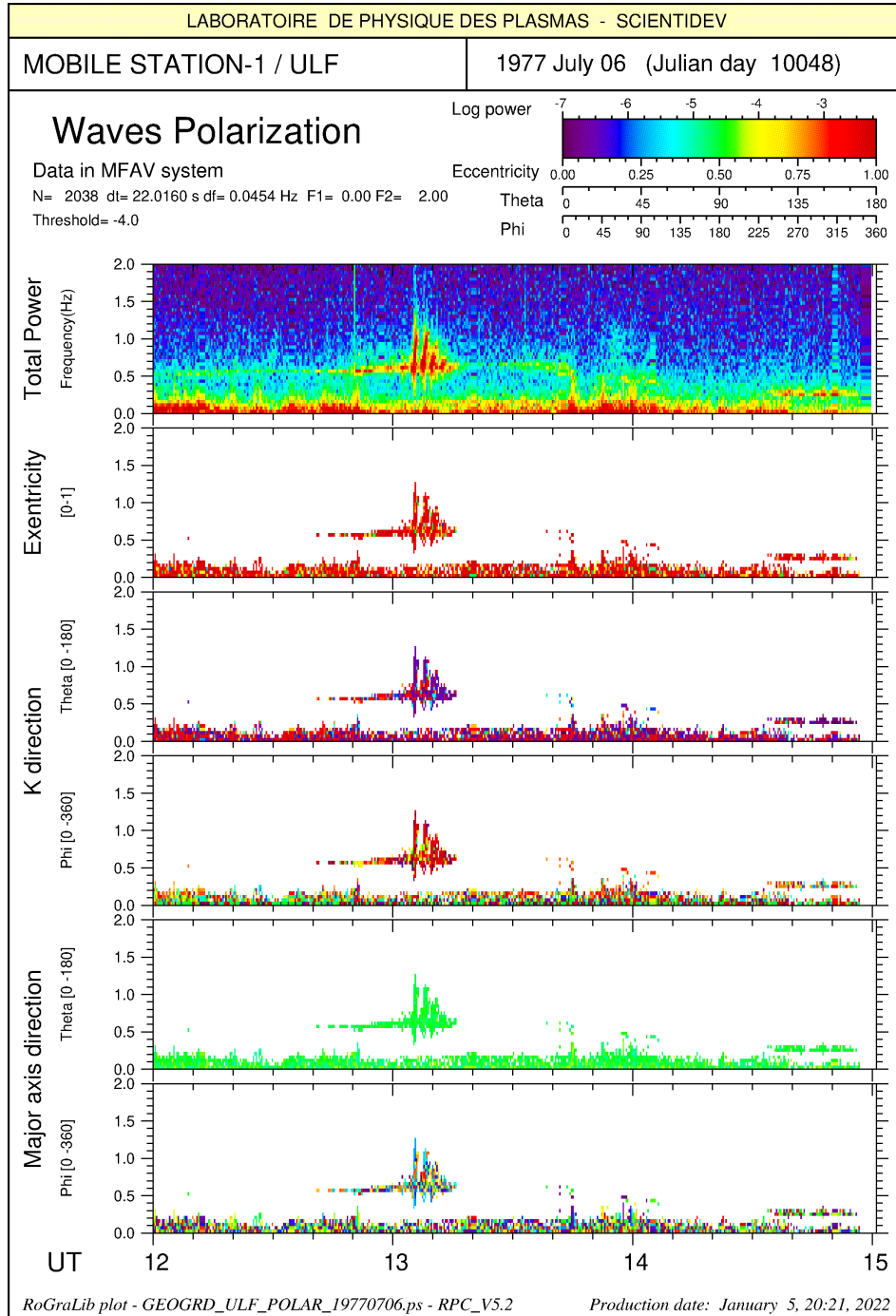


FIG. 13.6: Polarisation parameters in MFA system

13.3.3 Other examples of wave polarization

The polarization parameters are shown on Fig. 13.7. Interpretation is following:

eccentricity:	red	so ~ 1	\Rightarrow linear polarization (degenerated ellipse).
theta K:	green	so $\sim 90^\circ$	\Rightarrow linear polarization with $K \perp$ to B_0.
theta major axis:	red or dark purple	so ~ 0 or 180°	\Rightarrow major axis aligned to B_0 , consistent with the direction of K.

Therefore the linear polarization was already seen on previous fig. 13.4.

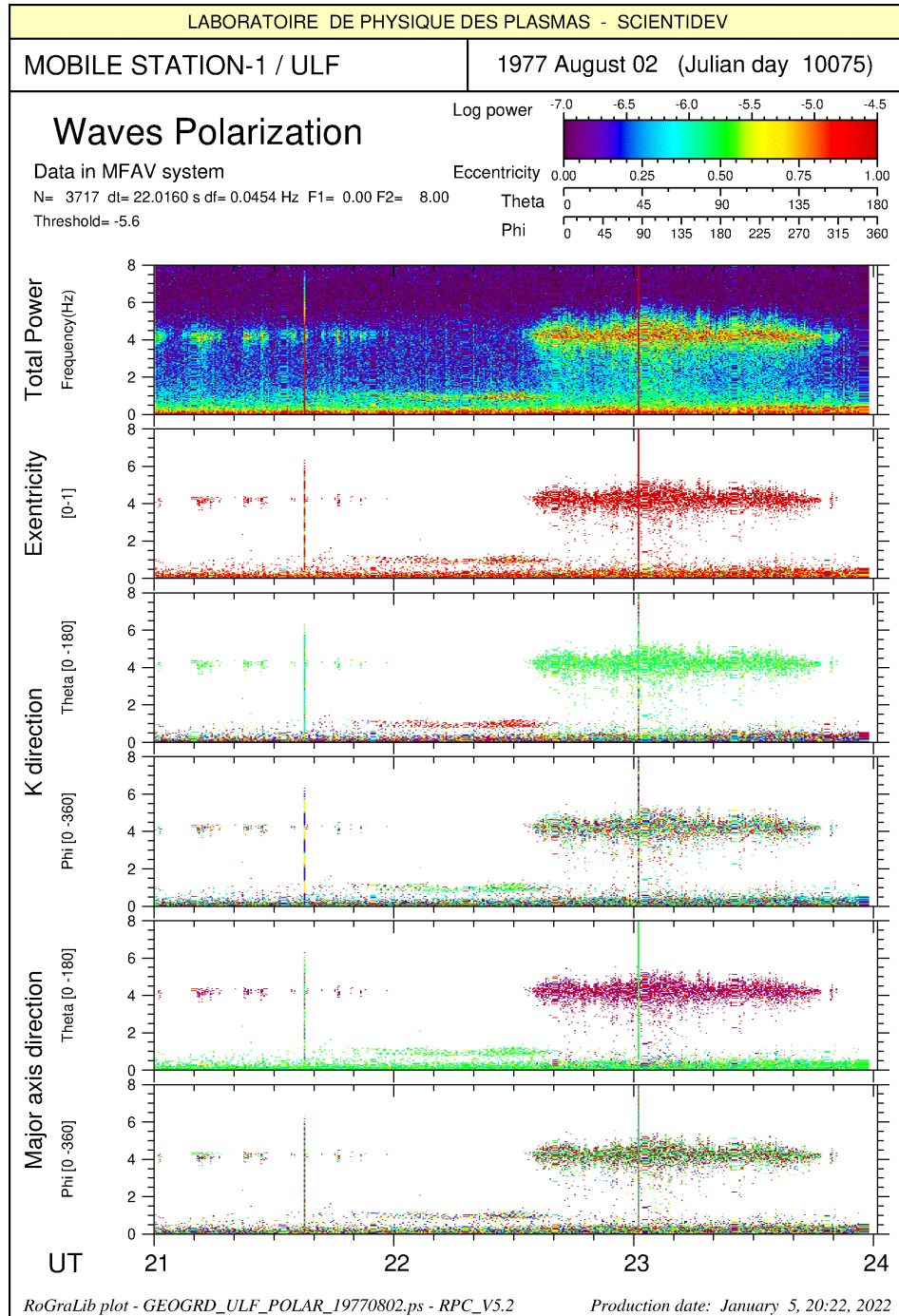


FIG. 13.7: Polarization parameter of linear polarisation aligned with the B DC magnetic field

Chapter 14

TESTS OF GEOGRD SCRIPTS

The tests of the RPC commands are made from the scripts, which, as we have seen, successively launch several RPC commands for a particular treatment. For each script it was made a test, which are summarized here. Most of the graphical results have been used to illustrate this part.

14.1 test_do_VT_plot_GEOGRD.sh

This test runs the following scripts:

```
do_VT_plot_GEOGRD.sh 19770706 130200 131000 L1
do_VT_plot_GEOGRD.sh 19770706 130200 131000 L2
```

14.2 test_do_SP_plot_from_VTL1_GEOGRD.sh

This test runs the following scripts:

```
do_SP_plot_from_VTL1_GEOGRD.sh 19770706 120000 150000 512 0. 1.5 t XY 1. -7.2 -2.2
do_SP_plot_from_VTL1_GEOGRD.sh 19770802 210000 240000 512 0. 8.0 t LR 1. -7.2 -4.0
```

14.3 test_do_SP_plot_from_VTL2_GEOGRD.sh

This test runs the following scripts:

```
do_SP_plot_from_VTL2_GEOGRD.sh 19770706 120000 150000 512 0. 1.5 t XY 1. -7.2 -2.2
do_SP_plot_from_VTL2_GEOGRD.sh 19770802 210000 240000 512 0. 8.0 t LR 1. -7.2 -4.0
```

14.4 test_do_Polar_plot_GEOGRD.sh

This test runs the following scripts:

```
do_Polar_plot_GEOGRD.sh 19770706 120000 150000 512 0. 2.0 t 1. -7. -2.2 -4.
do_Polar_plot_GEOGRD.sh 19770802 210000 240000 512 0. 8.0 t 1. -7. -4.5 -5.6
```

14.5 Test of all GEOGRD scripts

This script launch a full and long series of tests. First, you have to position yourself in the directory test/test_GEOS. Then, to erase the old tests, we can run the command:

```
clean_all_tests.sh
```


Only the following launch shells will remain:

```
clean_all_tests.sh
test_do_VT_plot_GEOGRD.sh
test_do_SP_plot_from_VTL1_GEOGRD.sh
test_do_SP_plot_from_VTL2_GEOGRD.sh
test_do_Polar_plot_GEOGRD.sh
```

as well as the shell which launches all the tests:

```
run_test_demo.sh
```

14.6 run_test_demo.sh

This shell launch a series of scripts, to check its validity. Time execution is shortness that the previous one, and can be used as a demo of what the scripts can do. List of the scripts launched by this "super-script" is given above.

```
line='-----'
echo ' ====='
echo ' Tests of GEOGRD script "do_VT_plot_GEOGRD.sh '
echo ' ====='
echo
echo ' Compute and plot Ground ULF SPL2 spectrogram from VTL1'
echo
echo $line ; echo 'do_VT_plot_GEOGRD.sh 19770706 130200 131000 L1 '
echo $line ; do_VT_plot_GEOGRD.sh 19770706 130200 131000 L1 > /dev/null
echo $line ; echo 'do_VT_plot_GEOGRD.sh 19770706 130200 131000 L2 '
echo $line ; do_VT_plot_GEOGRD.sh 19770706 130200 131000 L2 > /dev/null
echo
echo ' ====='
echo ' Tests of GEOGRD script "do_SP_plot_from_VTL1_GEOGRD.sh '
echo ' ====='
echo
echo ' Compute and plot Ground ULF SPL2 spectrogram from VTL1'
echo
echo $line ; echo 'do_SP_plot_from_VTL1_GEOGRD.sh 19770706 120000 150000 512 0. 1.5 t XY 1. -7.2 -2.2 '
echo $line ; do_SP_plot_from_VTL1_GEOGRD.sh 19770706 120000 150000 512 0. 1.5 t XY 1. -7.2 -2.2 > /dev/null
echo $line ; echo 'do_SP_plot_from_VTL1_GEOGRD.sh 19770802 210000 240000 512 0. 8.0 t LR 1. -7.2 -4.0 '
echo $line ; do_SP_plot_from_VTL1_GEOGRD.sh 19770802 210000 240000 512 0. 8.0 t LR 1. -7.2 -4.0 > /dev/null
echo
echo ' ====='
echo ' Tests of GEOGRD script "do_SP_plot_from_VTL2_GEOGRD.sh '
echo ' ====='
echo
echo ' Compute and plot Ground ULF SPL2 spectrogram from VTL2'
echo
echo $line ; echo 'do_SP_plot_from_VTL2_GEOGRD.sh 19770706 120000 150000 512 0. 1.5 t XY 1. -7.2 -2.2 '
echo $line ; do_SP_plot_from_VTL2_GEOGRD.sh 19770706 120000 150000 512 0. 1.5 t XY 1. -7.2 -2.2 > /dev/null
echo $line ; echo 'do_SP_plot_from_VTL2_GEOGRD.sh 19770802 210000 240000 512 0. 8.0 t LR 1. -7.2 -4.0 '
echo $line ; do_SP_plot_from_VTL2_GEOGRD.sh 19770802 210000 240000 512 0. 8.0 t LR 1. -7.2 -4.0 > /dev/null
echo
echo ' ====='
echo ' Tests of GEOGRD script "do_Polar_plot_GEOGRD.sh '
echo ' ====='
echo
echo ' Compute and plot Ground ULF SPL2 spectrogram from VTL2'
echo
```

```

echo $line ; echo 'do_Polar_plot_GEOGRD.sh 19770706 120000 150000 512 0. 2.0 t 1. -7. -2.2 -4. '
echo $line ; do_Polar_plot_GEOGRD.sh 19770706 120000 150000 512 0. 2.0 t 1. -7. -2.2 -4. > /dev/null
echo $line ; echo 'do_Polar_plot_GEOGRD.sh 19770802 210000 240000 512 0. 8.0 t 1. -7. -4.5 -5.6'
echo $line ; do_Polar_plot_GEOGRD.sh 19770802 210000 240000 512 0. 8.0 t 1. -7. -4.5 -5.6 > /dev/null
echo

```

Execution can be launch by **run_test_demo_with_log.sh** wich allow to keep a copy of the script result in a file. The user can look at the screen to check that all are running correctly, or look at the log file **run_test_demo.log** which is given above.

```

=====
Tests of GEOGRD script "do_VT_plot_GEOGRD.sh
=====

Compute and plot Ground ULF SPL2 spectrogram from VTL1

-----
do_VT_plot_GEOGRD.sh 19770706 130200 131000 L1
-----

RPC_get_data_GEOGRD : NORMAL TERMINATION - time exe= 0 s.
STOP visu_vectime.exe : NORMAL TERMINATION
RPC_visu_vectime : NORMAL TERMINATION - time exe= 10 s.
-----
do_VT_plot_GEOGRD.sh 19770706 130200 131000 L2
-----

RPC_get_data_GEOGRD : NORMAL TERMINATION - time exe= 0 s.
STOP visu_vectime.exe : NORMAL TERMINATION
RPC_visu_vectime : NORMAL TERMINATION - time exe= 12 s.
-----

=====
Tests of GEOGRD script "do_SP_plot_from_VTL1_GEOGRD.sh
=====

Compute and plot Ground ULF SPL2 spectrogram from VTL1

-----
do_SP_plot_from_VTL1_GEOGRD.sh 19770706 120000 150000 512 0. 1.5 t XY 1. -7.2 -2.2
-----

RPC_get_data_GEOGRD : NORMAL TERMINATION - time exe= 0 s.
STOP vectime_L1_to_spectro_L2_GEOGRD : NORMAL TERMINATION
RPC_vectime_L1_to_spectro_L2_GEOGRD : NORMAL TERMINATION - time exe= 12 s.
STOP visu_spectro.exe : NORMAL TERMINATION
RPC_visu_spectro : NORMAL TERMINATION - time exe= 2 s.
STOP visu_ave_spectrum.exe : NORMAL TERMINATION
RPC_visu_ave_spectrum : NORMAL TERMINATION - time exe= 2 s.
-----
do_SP_plot_from_VTL1_GEOGRD.sh 19770802 210000 240000 512 0. 8.0 t LR 1. -7.2 -4.0
-----

RPC_get_data_GEOGRD : NORMAL TERMINATION - time exe= 0 s.
STOP vectime_L1_to_spectro_L2_GEOGRD : NORMAL TERMINATION
RPC_vectime_L1_to_spectro_L2_GEOGRD : NORMAL TERMINATION - time exe= 24 s.
STOP visu_spectro.exe : NORMAL TERMINATION
RPC_visu_spectro : NORMAL TERMINATION - time exe= 4 s.
STOP visu_ave_spectrum.exe : NORMAL TERMINATION
RPC_visu_ave_spectrum : NORMAL TERMINATION - time exe= 3 s.
-----

=====
Tests of GEOGRD script "do_SP_plot_from_VTL2_GEOGRD.sh
=====

Compute and plot Ground ULF SPL2 spectrogram from VTL2

-----
do_SP_plot_from_VTL2_GEOGRD.sh 19770706 120000 150000 512 0. 1.5 t XY 1. -7.2 -2.2
-----

```

```

RPC_get_data_GEOGRD      : NORMAL TERMINATION - time exe= 1 s.
STOP vectime_to_spectro.exe : NORMAL TERMINATION
RPC_vectime_to_spectro    : NORMAL TERMINATION - time exe= 14 s.
STOP visu_spectro.exe     : NORMAL TERMINATION
RPC_visu_spectro          : NORMAL TERMINATION - time exe= 3 s.
STOP visu_ave_spectrum.exe : NORMAL TERMINATION
RPC_visu_ave_spectrum     : NORMAL TERMINATION - time exe= 2 s.
-----
do_SP_plot_from_VTL2_GEOGRD.sh 19770802 210000 240000 512 0. 8.0 t LR 1. -7.2 -4.0
-----
RPC_get_data_GEOGRD      : NORMAL TERMINATION - time exe= 0 s.
STOP vectime_to_spectro.exe : NORMAL TERMINATION
RPC_vectime_to_spectro    : NORMAL TERMINATION - time exe= 25 s.
STOP visu_spectro.exe     : NORMAL TERMINATION
RPC_visu_spectro          : NORMAL TERMINATION - time exe= 4 s.
STOP visu_ave_spectrum.exe : NORMAL TERMINATION
RPC_visu_ave_spectrum     : NORMAL TERMINATION - time exe= 3 s.

=====
Tests of GEOGRD script "do_Polar_plot_GEOGRD.sh
=====

Compute and plot Ground ULF SPL2 spectrogram from VTL2

-----
do_Polar_plot_GEOGRD.sh 19770706 120000 150000 512 0. 2.0 t 1. -7. -2.2 -4.
-----
RPC_get_data_GEOGRD      : NORMAL TERMINATION - time exe= 1 s.
STOP vectime_to_mfav_GEOGRD.exe : NORMAL TERMINATION
RPC_vectime_to_mfav_GEOGRD : NORMAL TERMINATION - time exe= 14 s.
STOP vectime_to_spectro.exe : NORMAL TERMINATION
RPC_vectime_to_spectro    : NORMAL TERMINATION - time exe= 14 s.
STOP visu_spectro.exe     : NORMAL TERMINATION
RPC_visu_spectro          : NORMAL TERMINATION - time exe= 2 s.
STOP visu_ave_spectrum.exe : NORMAL TERMINATION
RPC_visu_ave_spectrum     : NORMAL TERMINATION - time exe= 2 s.
STOP spectro_to_polar.exe  : NORMAL TERMINATION
RPC_spectro_to_polar      : NORMAL TERMINATION - time exe= 15 s.
STOP visu_polar.exe       : NORMAL TERMINATION
RPC_visu_polar            : NORMAL TERMINATION - time exe= 3 s.
-----
do_Polar_plot_GEOGRD.sh 19770802 210000 240000 512 0. 8.0 t 1. -7. -4.5 -5.6
-----
RPC_get_data_GEOGRD      : NORMAL TERMINATION - time exe= 1 s.
STOP vectime_to_mfav_GEOGRD.exe : NORMAL TERMINATION
RPC_vectime_to_mfav_GEOGRD : NORMAL TERMINATION - time exe= 27 s.
STOP vectime_to_spectro.exe : NORMAL TERMINATION
RPC_vectime_to_spectro    : NORMAL TERMINATION - time exe= 25 s.
STOP visu_spectro.exe     : NORMAL TERMINATION
RPC_visu_spectro          : NORMAL TERMINATION - time exe= 4 s.
STOP visu_ave_spectrum.exe : NORMAL TERMINATION
RPC_visu_ave_spectrum     : NORMAL TERMINATION - time exe= 4 s.
STOP spectro_to_polar.exe  : NORMAL TERMINATION
RPC_spectro_to_polar      : NORMAL TERMINATION - time exe= 26 s.
STOP visu_polar.exe       : NORMAL TERMINATION
RPC_visu_polar            : NORMAL TERMINATION - time exe= 5 s.

Starting time : 2022-01-06 02:01:20
Ending time   : 2022-01-06 02:05:56
Duration      : 276 sec. (4.60 mn.)

```


Part V

APPLICATION TO CLUSTER DATA

Chapter 15

ACCESS TO CLUSTER DATA

15.1 Download CEF files

The files in CEF format [2] are downloaded from the CSA with commands like :

<code>RPC_download_data_oneday_t1t2_CLUSTA_dwf</code>	→ a part of one day
<code>RPC_download_data_oneday_t1t2_CLUSTA_cwf</code>	→ a part of one day
<code>RPC_download_data_CLUFGM</code>	→ many days in a single file
<code>RPC_download_data_oneday_t1t2_CLUFGM</code>	→ a part of one day
<code>RPC_download_data_oneday_CLUFGM</code>	→ a full day
<code>RPC_download_data_onemonth_CLUFGM</code>	→ one file per day ¹
<code>RPC_download_data_oneyear_CLUFGM</code>	→ one file per day ¹
<code>RPC_download_data_CLUPOS</code>	→ many days in a single file
<code>RPC_download_data_oneday_t1t2_CLUPOS</code>	→ a part of one day
<code>RPC_download_data_oneday_CLUPOS</code>	→ a full day
<code>RPC_download_data_onemonth_CLUPOS</code>	→ one file per day ¹
<code>RPC_download_data_oneyear_CLUPOS</code>	→ one file per day ¹

15.1.1 Download command examples

The download is done directly from the CSA by the following commands. We give here an example for each dataset.

```
RPC_download_data_oneday_t1t2_CLUSTA_dwf 3 2001 01 10 NBR 00:00:00 02:00:00
RPC_download_data_oneday_t1t2_CLUSTA_cwf 2 2004 01 30 HBR 16:47:00 16:48:00
RPC_download_data_oneday_t1t2_CWF_CLUSTA 3 2001 01 10 ISR2 00:00:00 02:00:00
RPC_download_data_oneday_t1t2_CWF_CLUSTA 3 2001 01 10 GSE 00:00:00 02:00:00

RPC_download_data_oneday_t1t2_CLUFGM      3 2001 01 10 5VPS 00:00:00 02:00:00
RPC_download_data_oneday_t1t2_CLUFGM      1 2001 01 26 FULL 10:30:00 12:00:00

RPC_download_data_oneday_t1t2_CLUPOS      3 2001 01 10 00:00:00 23:59:59
RPC_download_data_oneday_t1t2_CLUPOS      1 2001 01 26 10:30:00 12:00:00
```

¹also convert CEF to RFF and put it into the local database

15.1.2 Example of downloaded files

CLUSTER/STAFF DWF :

C2_CP_STA_DWF_HBR_20040130_164700_20040130_164800.cef
C2_CP_STA_DWF_NBR_20010923_090000_20010923_110000.cef

CLUSTER/STAFF CWF :

C2_CP_STA_CWF_GSE_20010923_090000_20010923_110000.cef
C2_CP_STA_CWF_ISR2_20010923_090000_20010923_110000.cef

CLUSTER/FGM :

C1_CP_FGM_FULL_20010126_103000_20010126_120000.cef
C2_CP_FGM_5VPS_20010923_090000_20010923_110000.cef

CLUSTER/POS in GSE :

C1_CP_AUX_POSGSE_1M_20010126_103000_20010126_120000.cef

15.2 CEF to RFF conversion

RPC commands only work on files in RFF format [3]. After downloading a CEF file, the first thing to do is convert it to rff. This conversion is done by following commands :

RPC_cef_to_rff_CLUSTA_dwf name.cef
RPC_cef_to_rff_CLUSTA_cwf name.cef
RPC_cef_to_rff_CLUFGM name.cef
RPC_cef_to_rff_CLUPOS name.cef

15.2.1 Examples of conversion commands

The examples below are used to convert the CEF files for each of the experiments to RFF format:

RPC_cef_to_rff_CLUSTA_dwf C3_CP_STA_DWF_NBR_20010110_000000_20010110_020000.cef
RPC_cef_to_rff_CLUSTA_cwf C3_CP_STA_CWF_ISR2_20010110_000000_20010110_020000.cef
RPC_cef_to_rff_CLUSTA_cwf C3_CP_STA_CWF_GSE_20010110_000000_20010110_020000.cef

RPC_cef_to_rff_CLUFGM C3_CP_FGM_5VPS_20010110_000000_20010110_020000.cef
RPC_cef_to_rff_CLUFGM C1_CP_FGM_FULL_20010126_103000_20010126_120000.cef

RPC_cef_to_rff_CLUPOS C3_CP_AUX_POSGSE_1M_20010110_000000_20010110_235959.cef

15.2.2 Example of RFF files obtained

CLUSTER/STAFF VTL1 et VTL2

CLU3_STASC_VTL1_NBR_20010110_000000_20010110_020000.rff
CLU2_STASC_VTL1_HBR_20040130_164700_20040130_164800.rff
CLU3_STASC_VTL2_NBR_ISR2_20010110_000000_20010110_020000.rff
CLU3_STASC_VTL2_NBR_GSE_20010110_000000_20010110_020000.rff

CLUSTER/FGM 5VPS et FULL

CLU3_FGM_VTL2_5VPS_20010110_000000_20010110_020000.rff
CLU1_FGM_VTL2_FULL_20010126_103000_20010126_120000.rff

CLUSTER/POS in GSE

LU3_AUX_VTL2_POS_20010110_000000_20010110_235959.rff

15.3 Make a local database

It is recommended to make a base of the RFF files used by RPC commands. We can also, if we want to keep the files CEF from download commands, make a CEF base for the CEF files.

The RPC package comes with two minimum databases, containing just examples for testing. These directories can be anywhere on your machine, even on a different disk or on the network as long as its path is correctly indicated in the **RPC_config.bash**. You just have to copy them by the following commands, and then the sub-directories will be automatically created.

RPC_put_cef_database file.cef and **RPC_put_rff_database** file.rff.

15.3.1 The CEF database

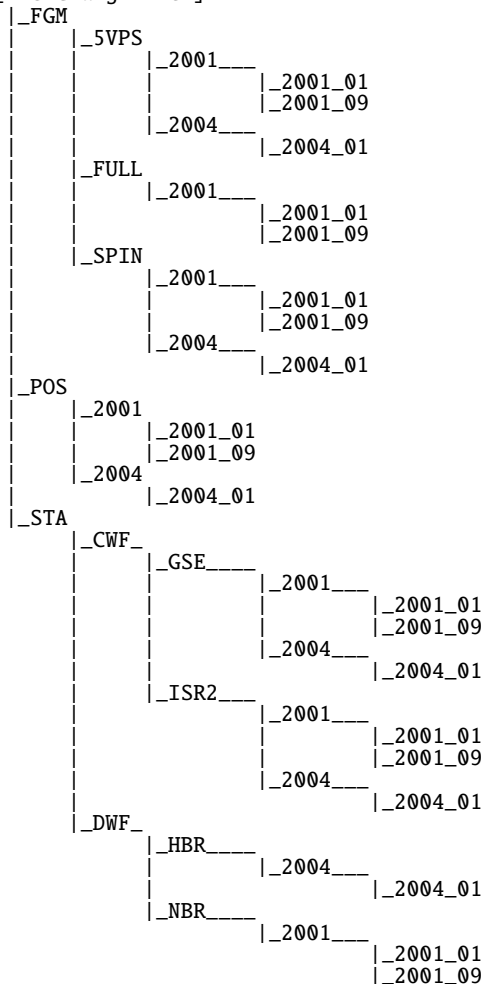
CEF files obtained by downloading can be put directly in the local base by the above commands. However, as soon as CEF files have been converted to RFF, it is no longer useful to keep it.

This base is therefore not mandatory.

The CEF database is organized as follows:

CEF_database_CLUSTER

[xleftmargin=2cm]



CWF.cef may contain mixed NBR and / or HBR.
FGM and POS are both in GSE.

15.3.2 The RFF database

This is the base that is used by the RPC commands. It is filled to from the CEF to RFF conversion commands using the commands that we just saw.

The RFF database is organized as follows:

RFF_database_CLUSTER

```
[xleftmargin=2cm]
|_FGM
|   |_5VPS
|       |_2001____|_2001_01
|               |_2001_09
|       |_2004____|_2004_01
|   |_FULL
|       |_2001____|_2001_01
|               |_2001_09
|   |_SPIN
|       |_2001____|_2001_01
|               |_2001_09
|       |_2004____|_2004_01
|_POS
|   |_2001
|       |_2001_01
|       |_2001_09
|   |_2004
|       |_2004_01
|_STA
|   |_VTL1
|       |_HBR____|_2004____|_2004_01
|       |_NBR____|_2001____|_2001_01
|               |_2001_09
|   |_VTL2
|       |_HBR____|_GSE____|_2004____|_2004_01
|               |_ISR2____|_2004____|_2004_01
|       |_NBR____|_GSE____|_2001____|_2001_01
|               |_2001_09
|               |_ISR2____|_2001____|_2001_01
|               |_2001_09
|_VIT
|   |_2004
|       |_2004_01
```

The command **RPC_cef_to_rff_CWF_CLUSTER** separates the two modes which can be contained in the single CEF file and produce two HBR file and NBR ; If one of them does not contain data, it is destroyed at the production.

15.3.3 Interest of a local database

RPC processing commands request one or more RFF files as arguments. These files can be in the current directory or elsewhere, if the path is correctly entered there is no problem. But if you are working on many files, it is more pleasant to store them directly in the local database. First, they can be easily found with the commands described in the next chapter, and then this database can be in another directory, a disk « data » for example.

The local database is also used for access to two simultaneous experiments, for example when we want to trace the gyrofrequency of protons (from FGM data) or the position of satellites (from POS) on a spectrogram of STA data for example .

15.4 Database commands

After downloading an rff file, you can *move it into the local RFF database* by the command :

```
RPC_put_rff_database name.rff
```

The same command is valid for VecTime from STA, FGM and POS.

If one wants *copy the RFF file into the database*, and not move it, we will use the command:

```
RPC_copy_rff_database name.rff
```

As before, the same command is valid for VecTime from STA, FGM and POS.

Conversely, we can *copy the desired file into the current directory* by the commands :

```
RPC_get_data_CLUSTA_VTL1 SatNum year month day BitRate
RPC_get_data_CLUSTA_VTL2 SatNum year month day BitRate frame
                        BitRate: NBR ou HBR ;   frame: GSE ou ISR2

RPC_get_data_CLUFGM  SatNum year month day
RPC_get_data_CLUFGM_4sat      year month day
RPC_get_data_CLUPOS   SatNum year month day
RPC_get_data_CLUPOS_4sat      year month day

RPC_get_data_CLUVIT   SatNum year month day
RPC_get_data_CLUVIT_4sat      year month day

RPC_get_data_CLUGEOM      year month day
```

If we want to know what is in the RFF database, *the list of all RFF files* by experience is obtained by the commands :

```
RPC_list_rff_database CLUSTER STA
RPC_list_rff_database CLUSTER FGM
RPC_list_rff_database GEOS   ULF
RPC_list_rff_database GEOS   MAG
```

ou encore

```
RPC_list_rff_database CLUSTER ALL
```

To compress all the CEF files in the database:

```
RPC_zip_cef_database_CLUSTER
```

If you want to delete all the CEF files from the database:

```
RPC_purge_cef_database_CLUSTER
```


Chapter 16

WORKING ON CLUSTER RFF FILES

16.1 Main types of CLUSTER RFF files

We will mainly handle files of type VecTime [3] which are vectors indexed by time, and files of type Spectrogram [3] which are series of dated spectra.

To go fast, you can go directly to chapter " USING SCRIPTS "

16.1.1 Example of VecTime file names

```
CLU2_STASC_VTL1_HBR_20040130_164700_20040130_164800.rff
CLU2_STASC_VTL2_HBR_GSE_20040130_164700_20040130_164800.rff
CLU2_STASC_VTL2_HBR_ISR2_20040130_164700_20040130_164800.rff

CLU2_FGM_VTL2_SPIN_20010923_090000_20010923_110000.rff
CLU2_FGM_VTL2_5VPS_20010923_090000_20010923_110000.rff
CLU1_FGM_VTL2_FULL_20010126_103000_20010126_120000.rff

CLU1_AUX_VTL2_POS_20010126_103000_20010126_120000.rff
CLU1_AUX_VTL2_VIT_20010126_103000_20010126_120000.rff
```

VTL1 (VecTime Level 1) are STAFF-SC telemetry files.

VTL2 (VecTime Level 2) are calibrated STAFF-SC waveforms.

FGM data and position data are calibrated and in GSE system.

16.1.2 Example of Spectrograms file names

The nomenclature is like :

```
CLU2_STASC_SPL2_NBR_GSE_20010923_090000_20010923_110000.rff
```

Note that the viewing commands will create files like :

```
CLU2_STASC_SPL2_NBR_GSE_20010923_090000_20010923_110000.ps
CLU2_STASC_SPL2_NBR_GSE_20010923_090000_20010923_110000.pdf
CLU2_STASC_SPL2_NBR_GSE_20010923_090000_20010923_110000.png
```

And for average spectra :

```
CLU2_STASC_SPL2_NBR_GSE_20010923_090000_20010923_110000_ave_spe.ps
CLU2_STASC_SPL2_NBR_GSE_20010923_090000_20010923_110000_ave_spe.pdf
CLU2_STASC_SPL2_NBR_GSE_20010923_090000_20010923_110000_ave_spe.pdf
```

16.1.3 Checking the validity of an RFF file

The validity of an RFF file can be checked by the `RPC_check_rff` command:

RPC_check_rff nom.rff

In the event of non-compliance, it will be reported, and the reasons will be displayed in the `check_rff.out` file. Note that the validity of all the `rff` files found in a `toto` directory can be checked with the command **RPC_check_rff_dir toto_dir**

16.2 Special commands for CLUSTER

16.2.1 Special commands for all CLUSTER experiments

RPC_menu_CLUSTER

Displays on screen the general menu of commands, as well as special commands to CLUSTER

RPC_give_spin_dir_CLUSTER

give spin direction in GEI system, use STAFF VTL1 or VTL2 data files.

Usage :

RPC_give_spin_dir_CLUSTER SatNum year month day

RPC_visu_spectro_4Bz

visualization of 4 spectrogrammes Bz of 4 S/C on a single plot

Usage :

RPC_visu_spectro_4Bz SP1.rff SP2.rff SP3.rff SP4.rff datiso_deb datiso_fin f1 f2 pmin pmax fi1 fi2

datiso_deb and datiso_fin could be set to 0 0 to take all the files

RPC_visu_vectime_4sat

visualization of 4 VT files for 4 S/C on a single plot

Usage :

RPC_visu_vectime_4sat POS1.rff POS2.rff POS3.rff POS4.rff datiso_deb datiso_fin

datiso_deb and datiso_fin could be set to 0 0 to take all the files

RPC_visu_vectime_3D_4sat

3D visualization of 4 VT files for 4 S/C on a single plot

Usage :

RPC_visu_vectime_3D_4sat POS1.rff POS2.rff POS3.rff POS4.rff datiso_deb datiso_fin

datiso_deb and datiso_fin could be set to 0 0 to take all the files

With :

SP1-4.rff	: names of 4 spectro RFF files
datiso1 datiso2	: iso date/time first and end
f1 f2	: frequency bounds to plot
pmin pmax	: min max power for dynamic colors
fi1, fi2	: frequency bounds for integrated power

RPC_list_cef_database_CLUSTER

list of all cef files in cef data base

Usage :

RPC_list_cef_database_CLUSTER experi

With :

experi : STA, FGM or POS

RPC_purge_cef_database_CLUSTER

purge of all cef files in cef data base

Usage :

RPC_purge_cef_database_CLUSTER experi

With :

experi : STA, FGM or POS

RPC_put_cef_database_CLUSTER

put file in the right place in data base

Usage :

RPC_put_cef_database_CLUSTER toto.cef

With :

toto.cef : cef file (STA, FGM or POS); may be a toto.cef.gz file.

RPC_zip_cef_database_CLUSTER

zip all cef files in cef data base

Usage :

RPC_zip_cef_database_CLUSTER

Requires no argument.

16.2.2 Special commands for CLUSTER/STAFF**RPC_download_data_oneday_t1t2_DWF_CLUSTER**

download STA/DWF data from CAA a day between t1-t2

Usage :

RPC_download_data_oneday_t1t2_DWF_CLUSTER SatNum year month day Mode t1 t2

Ex :

RPC_download_data_oneday_t1t2_DWF_CLUSTER 3 2001 09 23 NBR 09:00:00 11:00:00

RPC_download_data_oneday_t1t2_CWF_CLUSTER

download STA/CWF data from CAA a day between t1-t2

Usage :

RPC_download_data_oneday_t1t2_CWF_CLUSTER SatNum year month day Mode t1 t2

Ex :

RPC_download_data_oneday_t1t2_CWF_CLUSTER 3 2001 09 23 NBR 09:00:00 11:00:00

RPC_cef_to_rff_CLUSTER_dwf

convert a STAFF DWF.cef file into a VTL1.rff

Usage :

RPC_cef_to_rff_CLUSTER_dwf toto.cef

Ex :

RPC_cef_to_rff_CLUSTER_dwf C2_CP_STA_DWF_NBR__20010923_090000_20010923_110000.cef

Will give :

CLU2_STASC_VTL1_NBR_20010923_090000_20010923_110000.rff

RPC_cef_to_rff_CLUSTER_cwf

convert a STAFF CWF.cef file into a VTL1.rff

Usage :

RPC_cef_to_rff_CLUSTER_cwf toto.cef

Ex :

RPC_cef_to_rff_CLUSTER_cwf C2_CP_STA_CWF_ISR2__20010923_090000_20010923_110000.cef

Will give :

CLU2_STASC_VTL2_NBR_ISR2_20010923_090000_20010923_110000.rff

RPC_get_data_CLUSTER_VTL1

get CLUSTER/STA VTL1 from data base

Usage :

RPC_get_data_CLUSTER_VTL1 SatNum year month day BitRate

Ex :

RPC_get_data_CLUSTER_VTL1 3 2001 09 23 NBR

RPC_get_data_CLUSTER_VTL2

get CLUSTER/STA VTL2 from data base

Usage :

RPC_get_data_CLUSTER_VTL2 SatNum year month day BitRate frame

Ex :

RPC_get_data_CLUSTER_VTL2 3 2001 09 23 NBR ISR2

RPC_vectime_calibration_CLUSTER

Calibrate a VTL1 file and produce VTL2

Usage:

RPC_vectime_calibration_CLUSTER VTL1.rff VTL2.rff Fdet Fc F1 F2 Step N-Kern N-shift Apod

With :

VTL1.rff : name of an input vectime RFF file
 VTL2.rff : name of an output vectime RFF file
 Fdet : detrend frequency (0. for classis despin)
 Fc : Frequency cut-off for calibration
 Fmin : frequency min for filtering (Min=0 => Fc)
 Fmax : frequency max for filtering (Max=0 => Nyquist)
 Step : Processing steps asked (1-8)
 N-Kern : Kernel size
 N-shift : for sliding window
 Apod : trapezium (t) or Gaussian (g)

Comments on processing steps:

- 1 : Volts, spinning system, with DC field
- 2 : Volts, spinning system, without DC field
- 3 : nTesla, spinning system, without DC field
- 4 : nTesla, fixed SR2 system, without DC field
- 5 : nTesla, fixed SR2 system, with DC field
- 6 : nTesla, fixed SR2 system, only DC-field
- 7 : nTesla, fix. ISR2 system + Dx,Dy in sperate fields
- 8 : nTesla, fixed GSE system, without DC field

RPC_vectime_L1_to_spectro_L2_CLUSTER

Produces a calibrated spectrogram from an uncalibrated VTL1 file

Usage:

RPC_vectime_L1_to_spectro_L2_CLUSTER VTL1.rff SPL2.rff Fdet Fc F1 F2 Step N_Kern N_shift Apod

With :

VTL1.rff : name of an input vectime RFF file
 SPL2.rff : name of an output spectrogram RFF file
 Fdet : detrend frequency (0. for classis despin)
 Fc : Frequency cut-off for calibration
 Fmin : frequency min for filtering (Min=0 = Fc)
 Fmax : frequency max for filtering (Max=0 = Nyquist)
 Step : Processing steps asked (1-8)
 N_Kern : Kernel size
 N_shift : for sliding window
 Apod : trapezium (t) or nothing(n)

Example :

RPC_vectime_L1_to_spectro_L2_CLUSTER CLUS1_ULF_VTL1_19770713.rff GEOS1_ULF_SPL2_19770713.rff 0. 0.1 0.5 0. 4 1024 2 g

RPC_vectime_to_mfa_CLUSTERA

Change coordinate of CLUSTER/STA from GSE to MFA

Usage :

RPC_vectime_to_mfa_CLUSTERA VTL2_STA_gse VTL2_FGM_gse VTL2_STA_mfa

With :

VTL2_ulf_srv : name of a STA RFF file
 VTL2_mag_srv : name of a FGM RFF file
 VTL2_ulf_mfav : name the STA RFF file in mfa

RPC_add_DxDy_to_BxBy

Add Dx and Dy to Bx and By on a 5 dimension VT file (Bx, By, Bz, Dx, Dy)

Usage : RPC_add_DxDy_to_BxBy file1 file2

with:

file1 : input VTL2 file with 5 comp., Bx,By,Bz,Dx,Dy
 file2 : output VTL2 file with 3 comp., Bx+Dx, By+Dy,Bz

Example : RPC_add_DxDy_to_BxBy CLU2_STASC_VTL2_NBR_ISR2_20010923.rff CLU2_STASC_VTL2_NBR_ISR2_20010923_DxDy.rff

16.2.3 Special commands for CLUSTER/FGM**RPC_download_data_oneday_t1t2_CLUFGM**

get FGM data from CAA a day between t1-t2

Usage :

RPC_download_data_oneday_t1t2_CLUFGM SatNum year month day Mode t1 t2

Ex :

RPC_download_data_oneday_t1t2_CLUFGM 3 2001 09 23 5VPS 09:00:00 11:00:00

RPC_download_data_oneday_CLUFGM

get entire day of FGM data from CAA

Usage :

RPC_download_data_oneday_CLUFGM SatNum year month day Mode

Ex :

RPC_download_data_oneday_CLUFGM 1 2001 09 23 SPIN

RPC_download_data_onemonth_CLUFGM

download CLUFGM.rff files 1 month 4 sat from CAA

Usage :

RPC_download_data_onemonth_CLUFGM year month Mode

Ex :

RPC_download_data_onemonth_CLUFGM 2009 12 SPIN

RPC_download_data_oneyear_CLUFGM

download entire year of FGM data from CAA

Usage :

RPC_download_data_oneyear_CLUFGM year Mode

Ex :

RPC_download_data_oneyear_CLUFGM 2009 SPIN

RPC_download_data_CLUFGM

download FGM data from CAA for several days

Usage :

RPC_download_data_CLUFGM SatNum date_iso1 date_iso2 mode

Ex :

RPC_download_data_CLUFGM 3 2001-09-01T09:00:00.OOOZ 2001-09-01T09:00:00.OOOZ SPIN

RPC_cef_to_rff_CLUFGM

convert a fgm.cef file into a fgm_VT.rff file

Usage :

RPC_cef_to_rff_CLUFGM toto.cef

RPC_get_data_CLUFGM

get CLUSTER/FGM RFF file from data base

Usage :

RPC_get_data_CLUFGM SatNum year month day mode

Ex :

RPC_get_data_CLUFGM 3 2001 09 23 5VPS

RPC_get_data_CLUFGM_4sat

get 4 CLUSTER/FGM RFF files from data base

Usage :

RPC_get_data_CLUFGM_4sat year month day mode

Ex :

RPC_get_data_CLUFGM_4sat 2001 09 23 5VPS

RPC_get_indexed_data_CLUFGM

get CLUSTER/FGM indexed data (RFF without meta data)

Usage :

RPC_get_indexed_data_CLUFGM SatNum year month day mode

Ex :

RPC_get_indexed_data_CLUFGM 3 2001 09 23 SPIN

RPC_vectime_gse_to_isr2_CLUFGM

Change coordinate of CLUS/FGM from GSE to ISR2

Usage :

RPC_vectime_gse_to_isr2_CLUFGM VTL2_FGM_gse VTL2_FGM_isr2

With :

VTL2_FGM_gse : input file of a FGM RFF file in GSE

VTL2_FGM_isr2 : output file of a FGM RFF file in ISR2

RPC_alitime_4_vectime

Time alignment of 4 rff vectime files

Usage :

RPC_alitime_4_vectime VT1.rff VT2.rff VT3.rff VT4.rff

RPC_compute_curl_div_4sat

Compute curl(B) and div(B) from MAG and POS data (see details of used method in [14].)

Usage :

RPC_compute_curl_div_CLUFGM MAG1 MAG2 MAG3 MAG4 POS1 POS2 POS3 POS4 (RFF files)

Create file compute_curl_div_4sat.resu

RPC_visu_curl_div_4sat

Visualization of compute_curl_div_4sat.resu

Usage :

RPC_visu_curl_div_4sat iso_date1 iso_date2

Ex :

RPC_visu_curl_div_4sat 2001-09-01T09:00:00.OOOZ 2001-09-01T09:30:00.OOOZ

16.2.4 Special commands for CLUSTER/POS

RPC_download_data_oneday_t1t2_CLUPOS

get Pos. and Vel. from CAA a day between t1-t2

Usage :

RPC_download_data_oneday_t1t2_CLUPOS SatNum year month day t1 t2

Ex :

RPC_download_data_oneday_t1t2_CLUPOS 3 2001 09 23 09:00:00 11:00:00

RPC_download_data_oneday_CLUPOS

get entire day of POS data from CAA

Usage :

RPC_download_data_oneday_CLUPOS SatNum year month day

Ex :

RPC_download_data_oneday_CLUPOS 1 2001 09 23

RPC_download_data_onemonth_CLUPOS

download CLUPOS.rff files 1 month 4 sat from CAA

Usage :

RPC_download_data_onemonth_CLUPOS year month

Ex :

RPC_download_data_onemonth_CLUPOS 2009 12

RPC_download_data_oneyear_CLUPOS

download entire year of POS data from CAA

Usage :

RPC_download_data_oneyear_CLUPOS year

Ex :

RPC_download_data_oneyear_CLUPOS 2009

RPC_download_data_CLUPOS

download POS data from CAA for several days

Usage :

RPC_download_data_CLUPOS SatNum date_iso1 date_iso2 mode

Ex :

RPC_download_data_CLUPOS 3 2001-09-01T09:00:00.OOOZ 2001-09-01T09:00:00.OOOZ SPIN

RPC_cef_to_rff_CLUPOS

convert a POS.cef file into a POS_VT.rff file

Usage :

RPC_cef_to_rff_CLUPOS toto.cef

RPC_get_data_CLUPOS

get CLUSTER/POS RFF file from data base

Usage :

RPC_get_data_CLUPOS SatNum year month day

Ex :

RPC_get_data_CLUPOS 3 2001 09 23

RPC_get_data_CLUPOS_4sat

get 4 CLUSTER/POS RFF files from data base

Usage :

RPC_get_data_CLUPOS_4sat year month day

Ex :

RPC_get_data_CLUPOS_4sat 2001 09 23

RPC_get_data_CLUVIT

get CLUSTER/VIT RFF file from data base

Usage :

RPC_get_data_CLUVIT SatNum year month day

Ex :

RPC_get_data_CLUVIT 3 2001 09 23

RPC_get_data_CLUVIT_4sat

get 4 CLUSTER/VIT RFF files from data base

Usage :

RPC_get_data_CLUVIT_4sat year month day

Ex :

RPC_get_data_CLUVIT_4sat 2001 09 23

RPC_get_data_CLUGEOM

compute all geometric parameter of the CLUSTER tetrahedron. Create file **get_data_CLUGEOM.resu**

Usage :

RPC_get_data_CLUGEOM 2001 09 23

RPC_get_indexed_data_CLUPOS

get CLUSTER/POS indexed data (RFF without meta data)

Usage :

RPC_get_indexed_data_CLUPOS SatNum year month day

Ex :

RPC_get_indexed_data_CLUPOS 3 2001 09 23

RFF POS files, such as CLU1_AUX_VTL2_POS_20010126_103000_20010126_120000.rff can be used to do 3D visualization, by using:

RPC_visu_vectime_3D CLU1_AUX_VTL2_POS_20010126_103000_20010126_120000.rff datiso_deb datiso_fin
or

RPC_visu_vectime_3D_4sat POS1.rff POS2.rff POS3.rff POS4.rff datiso_deb datiso_fin
datiso_deb and datiso_fin could be set to 0 0 to take all the files

RPC_visu_CLUGEOM

visualization of a CLUGEOM cluster position files (file **get_data_CLUGEOM.resu**)

Usage :

RPC_visu_CLUGEOM

Create file CLUGEOM.ps (and PNG & PDF according choice done in RPC_config)

16.3 Make a STAFF spectrogram

16.3.1 From a VTL1 file

Launch the command (detailed in the previous chapter):

RPC_vectime_L1_to_spectro_L2_CLUSTA VTL1.rff SPL2.rff Fdet Fc F1 F2 Step N_Kern N_shift Apod

Example :

RPC_vectime_L1_to_spectro_L2_CLUSTA CLU2_STASC_VTL1_NBR_20010923_090000_20010923_110000.rff \
CLU2_STASC_SPL1_NBR_20010923_090000_20010923_110000\
0. 0.1 0.5 0. 4 1024 2 g

16.3.2 From a VTL2 file

This is done with the generic procedure (valid whatever the VecTime file):

RPC_vectime_to_spectro VT.rff SP.rff N_Kern N_shift Apod

With :

VT.rff : name of an input vectime RFF file
 SP.rff : name of an output spectrogram RFF file
 N_Kern : Kernel size
 N_shift : for sliding window
 Apod : trapezium (t), Gaussian (g) or nothing (n)

Example :

```
RPC_vectime_to_spectro CLU2_STASC_VTL2_NBR_GSE_20010923_090000_20010923_110000.rff \
                      CLU2_STASC_SPL2_NBR_GSE_20010923_090000_20010923_110000.rff \
                      512 512 t
```

16.3.3 Spectrogram visualization

Just run the following generic command:

RPC_visu_spectro SP.rff datiso1 datiso2 f1 f2 pmin pmax XY fi1 fi2 y/n

With :

SP.rff : name of a spectro RFF file
 datiso1 datiso2 : iso date/time first and end
 f1 f2 : frequency bounds to plot
 pmin pmax : min max power for dynamic colors
 XY or LR : for XYZ or Left Right Z
 fi1, fi2 : frequency bounds for integrated power
 y/n : y or no for adding on the plot gyrofrequency and spacecraft positions.

Example :

```
RPC_visu_spectro toto_SP.rff 1977-07-13T09:00:00.000Z 1977-07-13T11:00:00.000Z 0. 5. 0. 0. XY 0.2 10. y
```

or again:

```
RPC_visu_spectro toto_SP.rff 0 0 0. 200. -9.6 -3. LR 0. 200. y
```

This command will add on the plot of the gyrofrequencies and the satellite positions, *provided that the MAG and position files are present in the database*

The frequencies are plotted according to the formulas:

$$F_{ci} = |B_o| * 1.52E-2 \text{ (Bo en nT)} \quad FLH = \sqrt{1836.} * 6.2832 * F_{ci}$$

16.4 Make a FGM spectrogram

Since the data is already of the calibrated type, it suffices to launch the generic procedure :

RPC_vectime_to_spectro VT.rff SP.rff N_Kern N_shift Apod

Example :

```
RPC_vectime_to_spectro CLU2_FGM_VTL2_FULL_20010923_090000_20010923_110000.rff \
                      CLU2_FGM_SPL2_FULL_20010923_090000_20010923_110000.rff \
                      512 512 t
```

We then visualize this spectrogram by the command :

RPC_visu_spectro SP.rff datiso1 datiso2 f1 f2 pmin pmax XY fi1 fi2 y

Example :

```
RPC_visu_spectro CLU2_FGM_SPL2_FULL_20010923_090000_20010923_110000.rff \
                 2001-09-23T09:00:00.000Z 2001-09-23T11:00:00.000Z \
                 0. 2.1 0. 0. XY 0.2 2.
```

or, for the full duration of the file :

```
RPC_visu_spectro CLU2_FGM_SPL2_FULL_20010923_090000_20010923_110000.rff 0 0 0. 0. 0. XY 0.2 2.
```

16.5 Visualize an average spectrum

The file SPL2.rff being created, one can visualize the average spectrum of all the spectra of the SPL2 by the command :

```
RPC_visu_ave_spectrum SP.rff datiso1 datiso2 f1 f2 pmin pmax XY
```

With :

```
SP.rff      : name of a spectro RFF file
datiso1 datiso2 : iso date/time first and end
f1 f2      : frequency bounds to plot
pmin pmax  : min max power for dynamic colors (log values)
XY or LR   : for XYZ or Left Right Z
```

Example :

```
RPC_visu_ave_spectrum toto_SP.rff 1977-07-13T09:00:00.000Z 1977-07-13T11:00:00.000Z 0. 5. 0. 0. XY y n
```

```
RPC_visu_ave_spectrum toto_SP.rff 0 0 0. 0. 0. 0. XY y n
```

16.6 Draw a waveform

Just run the generic procedure :

```
RPC_visu_vectime name.rff
```

Example :

```
RPC_visu_vectime CLU2_FGM_VTL2_5VPS_20010923_090000_20010923_110000 $T1 $T2
```

With \$T1 et \$T2 des variables contenant la date ISO de début et de fin de période à visualiser.

Example: T1=2002-09-23T10:30:00.000Z T2= 2002-09-23T10:33:00.000Z
for [T1 T2] = [0 0] the whole file will be displayed.

16.7 Extract part of a VecTime file

Note that we can also extract the interesting portion of the file before viewing or other. This is done with the command:

```
RPC_reduce_time_vectime toto_VT.rff toto_VT.rff datiso1 datiso2
```

With :

```
toto_VT.rff : name of an input vectime RFF file
toto_VT.rff : name of an output vectime RFF file
datiso1 datiso2 : date in ISO format
```

Example :

```
RPC_reduce_time_vectime CLU2_STASC_VTL2_NBR_ISR2_20010923_090000_20010923_110000.rff \
CLU2_STASC_VTL2_NBR_ISR2_20010923_091000_20010923_093000.rff \
2001-09-23T09:12:00Z 2001-09-23T09:30:00Z
```

We can then do :

```
RPC_visu_vectime CLU2_STASC_VTL2_NBR_ISR2_20010923_091000_20010923_093000.rff 0 0
```

The start and end time set to 0 0 means that the entire file will be viewed.

16.8 Change coordinate system

You can change the coordinate system of any VecTime file via Rocotlib [4] with the command:

RPC_change_coordinate_system toto.rff gsm toto_gsm.rff

With :

toto.rff : name of a RFF file
gse : output coordinate system
toto_gse.rff : name of a new RFF file

Example :

```
RPC_change_coordinate_system CLU2_AUX_VTL2_POS_20010923_000000_20010923_235959.rff \
                             gsm \
                             CLU2_AUX_VTL2_POS_20010923_000000_20010923_235959_gsm.rff
```

16.9 Waves polarisation computation

16.9.1 Transformation in MFA coordinate system

The calculation of the polarization of the waves requires that we first go to the MFA (Magnetic Field Aligned) frame [5]. This is done by the above command :

RPC_vectime_to_mfa_CLUSTA file_STA.rff file_STA_mfa.rff file_FGM.rff

Example :

```
RPC_vectime_to_mfa_CLUSTA CLU2_STASC_VTL2_NBR_GSE_20010923_090000_20010923_110000.rff \
                          CLU2_STASC_VTL2_NBR_GSE_20010923_090000_20010923_110000_mfa.rff \
                          CLU2_FGM_VTL2_SPIN_20010923_090000_20010923_110000.rff
```

Warning ! This is only possible if the CLUSTA file is in GSE system, as CLUFGM file.

16.9.2 Calculation and plot of the polarization parameters

Once the ULF calibrated waveform file is in the MFA frame, we make the spectrogram using the generic command:

RPC_vectime_to_spectro.

Example :

```
RPC_vectime_to_spectro CLU2_FGM_VTL2_FULL_20010923_090000_20010923_110000_mfa.rff \
                      CLU2_FGM_SPL2_FULL_20010923_090000_20010923_110000_mfa.rff \
                      512 512 t
```

The visualization of the spectrogram in the reference line of force can already give information, especially if it is represented in circular coordinates « Left, Right, Z » [9] by the command :

RPC_visu_spectro CLU2_FGM_SPL2_FULL_20010923_090000_20010923_110000_mfa.rff 0 0 0. 0. 0. LR 0.2 2.

The polarization parameters calculated by method [6] are obtained by the command :

RPC_spectro_to_polar VTL2_mfa.rff polar.resu

With :

VTL2_mfa.rff : name of an input RFF VTL2 file in MFA
polar.resu : file name for polar results

16.9.3 Plot of polarisation parameters

Finally, the polarization parameters are displayed using the `command`:

RPC_visu_polar copolar.resu datiso1 datiso2 threshold f1 f2

With :

datiso1 datiso2 : iso date/time first and end of desired time section (0 0 for all)

threshold: for visualization ex: -6.1

f1 f2 : frequency bounds for visualizations

The "threshold" value should be chosen with care, because it allows the polarization of the phenomenon to be isolated from that of the surrounding noise. Various tests may be necessary.

Example :

RPC_visu_polar copolar.resu 0 0 -6.1 0. 2.

16.10 Curl(B) and Div(B) computation

*Note that all this work can be done by the script "**do_compute_curl_div_CLUFGM.sh**" detailed in section 17.9.2.*

16.10.1 Time alignment

Before Curl and Div computation it is necessary to have the 4 components of the magnetic field at the same time, as well as the 4 positions of the 4 Spacecrafts.

This is done by the command:

RPC_alitime_4_vectime file1 file2 file3 file4

file1 file2 file3 file4 are FGM VTL2 RFF files.

16.10.2 Computation

Computation of curl(B) and div(B) is done by the command::

RPC_compute_curl_div_4sat SC1_MAG.rff SC2_MAG.rff SC3_MAG.rff SC4_MAG.rff
SC1_POS.rff SC2_POS.rff SC3_POS.rff SC4_POS.rff

This program creates the file **compute_curl_div_4sat.resu** which contains many useful quantities (see section 17.9.2)

16.10.3 Visualization

The command **RPC_visu_polar** reads the file **compute_curl_div_4sat.resu** and creates two plots:

- One plot visualizing POS and MAG data, in GSE or GSE according to the chosen option above,
- A second plot, visualizing current density \vec{J} ; (\vec{J}, \vec{B}) angle, ratio $\text{div}(\mathbf{B})/\text{curl}(\mathbf{B})$, Elongation, planarity, inter distances.

Examples of plots are given in section 17.9.2.

16.11 Data position visualization

16.11.1 2D-Visualization

Viewing a VecTime file such as `CLU2_AUX_VTL2_POS_20010923_000000_20010923_235959.rff` is done by the generic command :

```
RPC_visu_vectime_4sat CLU2_AUX_VTL2_POS_20010923_000000_20010923_235959.rff 0 0
```

And to view the 4 satellites, we will use the command:

```
RPC_visu_vectime_4sat CLU1_AUX_VTL2_POS_20010923_000000_20010923_235959.rff \
CLU2_AUX_VTL2_POS_20010923_000000_20010923_235959.rff \
CLU3_AUX_VTL2_POS_20010923_000000_20010923_235959.rff \
CLU4_AUX_VTL2_POS_20010923_000000_20010923_235959.rff 0 0
```

We can of course do the same with speed files like this : `CLU1_AUX_VTL2_VIT_20010923_000000_20010923_235959.rff`

These files must of course, in addition to the date, contain the requested time slot, otherwise the plot will be incomplete.

We will see in the next chapter that the use of RPC scripts will simplify this procedure.

16.11.2 3D-visualization

On the same principle as the 2D view, we can view in 3D a VecTime rff file by the generic procedure:

```
RPC_visu_vectime_3D CLU2_AUX_VTL2_POS_20010923_000000_20010923_235959.rff
```

To have the 4 satellites in the same figure, we will use the command :

```
RPC_visu_vectime_3D_4sat CLU1_AUX_VTL2_POS_20010923_000000_20010923_235959.rff \
CLU2_AUX_VTL2_POS_20010923_000000_20010923_235959.rff \
CLU3_AUX_VTL2_POS_20010923_000000_20010923_235959.rff \
CLU4_AUX_VTL2_POS_20010923_000000_20010923_235959.rff 0 0
```

16.11.3 Transform positions in another system

To pass a position file, initially in GSE, in another marker, for example GSM, we will use the generic procedure previously seen , and we will launch for example the command :

```
RPC_change_coordinate_system CLU2_AUX_VTL2_POS_20010923_000000_20010923_235959.rff \
gsm \
CLU2_AUX_VTL2_POS_20010923_000000_20010923_235959_GSM.rff
```

16.12 Compute orbital parameters

If we have a position file over at least a little more than a half-revolution, we can calculate all the parameters of the orbit by:

```
RPC_compute_sat_orbit_param pos_file.rff
```

The results can be used by the following command, which calculates a trajectory according to Kepler's laws as a function of the orbit parameters.

16.13 Compute an elliptic trajectory

From the previous parameters, we can calculate the trajectory over 2 revolutions by:

RPC_compute_sat_trajectory Apo, Per, PerTime, a1,a2,a3, b1,b2,b3,dt, rff_out,scale

With :

Apo : Apogee in km, real*8
 Per : Parigee in km, real*8
 PerTime : Perigee Time, ISO format
 a1,a2,a3 : direction of semi-major axis, in GEI system
 : (from ellipse center to perigee, unit vector)
 b1,b2,b3 : direction of semi-minor axis, in GEI system
 : (a,b vector in the sens of the motion, unit vector)
 dt : time resolution for output positions
 rff_out : output rff_file
 scale : factor to apply to x,y,z

example:

```
RPC_compute_sat_trajectory 124252.735203443d0 26321.1522350593d0 \
2001-02-24T15:17:23.000Z \
-0.9493840 0.3011743 0.0892410 \
0.0820478 -0.0364760 0.9959607 5.0 \
trajectory.rff \
1.
```

The resulting trajectory file can be viewed by:

RPC_visu_vectime_3D trajectory.rff

Chapter 17

USING CLUSTER SCRIPTS

Scripts are bash files that string together multiple RPC commands, to avoid the tedious task of typing them all one after the other. The RPC package ships with a number of scripts that perform the most common processing. The user can use this as inspiration to write their own treatments. Each script has its test program, explaining its use. Most of plot of this chapter has been made with scripts.

The scripts generally do the data acquisition from the database, then perform operation such spectrogram computation, and so on, then visualize the results. To plot a waveform, the script is very short. To plot a spectrogram, it is more long. To plot a polarisation plot, the script is rather long, and particularly usefull.

17.1 Draw a waveform

17.1.1 do_VT_plot_CLUSTA.sh

Example :

```
do_VT_plot_CLUSTA.sh 2 20010923 092000 093000 NBR ISR2
```

This script is equivalent to sequencing the following commands by hand :

```
RPC_get_data_CLUSTA_VTL2 2 2001 09 23 NBR ISR2  
RPC_visu_vectime CLU2_STASC_VTL2_NBR_ISR2_20010923_090000_20010923_110000 \  
2001-09-23T09:20:00.000Z 2001-09-23T09:30:00.000Z
```

Another examples :

```
do_VT_plot_CLUSTA.sh 2 20010923 092000 093000 NBR GSE  
do_VT_plot_CLUSTA.sh 2 20040130 164710 164730 HBR GSE
```

Figure 17.1 hereafter show an example.

17.1.2 do_VT_plot_CLUFGM.sh

Examples :

```
do_VT_plot_CLUFGM.sh 2 20010923 092000 093000 FULL  
do_VT_plot_CLUFGM.sh 2 20010923 092000 093000 5VPS  
do_VT_plot_CLUFGM.sh 2 20040130 164700 164800 5VPS
```

Figure 17.2 hereafter show an example.

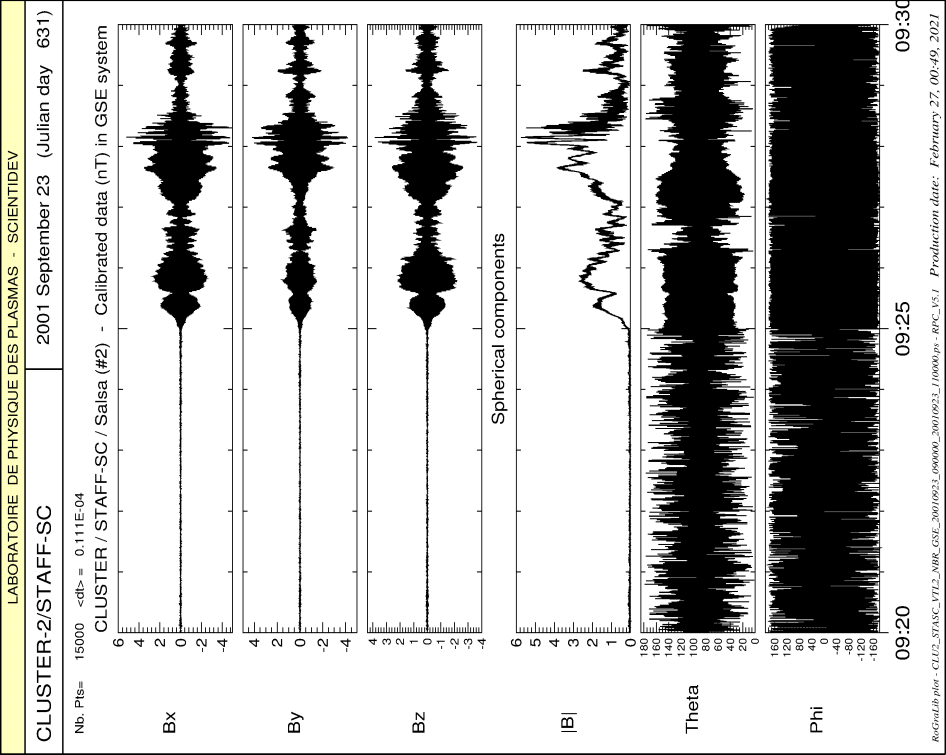


Fig. 17.1: Waveform plot of STAFF data (Normal Bit Rate) with do_VT_plot_CLUSTA.sh script

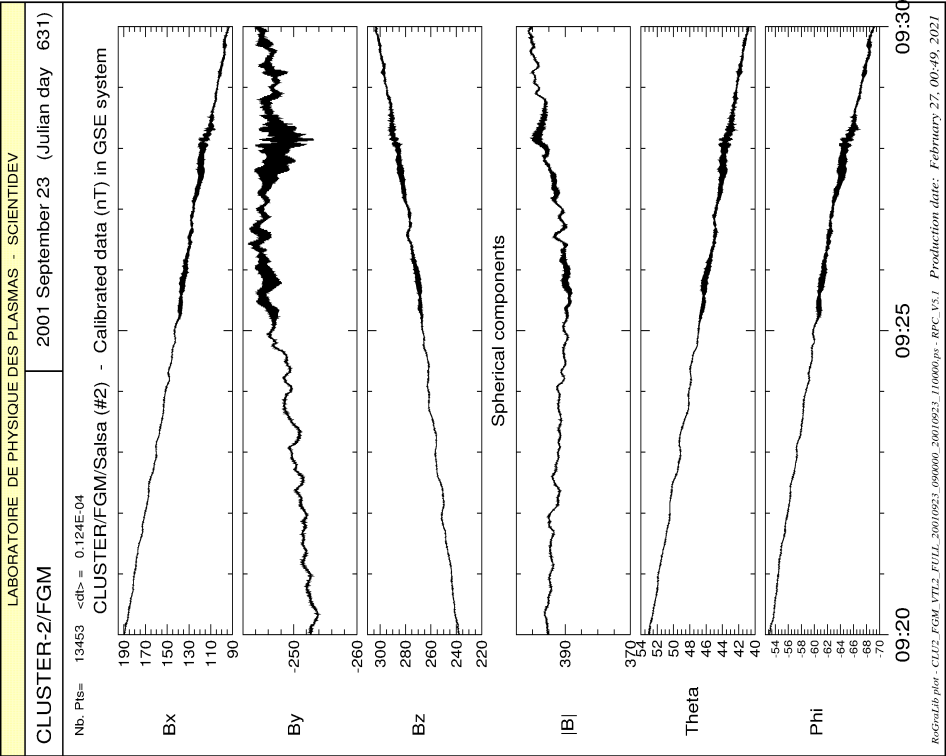


Fig. 17.2: Waveform plot of FGM (full resolution) data with do_VT_plot_CLUFGM.sh script

17.2 Make a spectrogramme

17.2.1 do_SP_plot_CLUSTA.sh

Examples :

```
do_SP_plot_CLUSTA.sh 2 20010923 090000 103000 NBR ISR2 512 0. 8.0 t XY 1. -6. 1.
do_SP_plot_CLUSTA.sh 2 20010923 090000 103000 NBR GSE 512 0. 2.5 t XY 1. -6. 1.
do_SP_plot_CLUSTA.sh 2 20040130 164700 164800 HBR GSE 128 0. 225. t XY 1. -10.-3.
```

arguments :

date, H_debut, H_fin, sat, BitRate, repere, Nbp_FFT, F1, F2, Apod, XY ou LR, F1_p_int, seuil_min, seuil_max

The script :

```
do_SP_plot_CLUSTA.sh 2 20010923 090000 103000 NBR ISR2 512 0.8 0 t XY 1.-6. 1.
```

is equivalent to sequencing the following commands by hand :

```
RPC_get_data_CLUSTA_VTL2 2 2001 09 23 NBR ISR2
RPC_vectime_to_spectro CLU2 STASC_VTL2_NBR_ISR2_20010923_090000_20010923_110000 \
CLU2 STASC_SPL2_NBR_ISR2_20010923_090000_20010923_110000 \
Nbp_FFT Nbp_shift apod
datiso1='RPC_date_time_to_datiso 20010923 090000'
datiso2='RPC_date_time_to_datiso 20010923 110000'
RPC_visu_spectro CLU2 STASC_SPL2_NBR_ISR2_20010923_090000_20010923_110000 \
$datiso1 $datiso2 0. 8. 0. 0. XY 1. -6. 1. y
RPC_visu_ave_spectrum CLU2 STASC_SPL2_NBR_ISR2_20010923_090000_20010923_110000 \
$datiso1 $datiso2 0. 8. -6. 1. XY yy
```

Therefore one execute with a single command line a process tedious to write.

Figure 17.3 hereafter show an example of spectrogram plot and average spectrum plot. Note that we have superimposed the plot of the Fci gyrofrequency on the spectrogram, as well as the position of the Spacecrafts, thank to the last parameter set to 'y' at the command `RPC_visu_spectro` .

17.2.2 do_SP_plot_CLUFGM.sh

Example :

```
do_SP_plot_CLUFGM.sh 2 20010923 090000 103000 FULL 512 0. 8.0 n XY 0.02 -7. 0.
do_SP_plot_CLUFGM.sh 2 20010923 090000 103000 5VPS 256 0. 2.5 n XY 0.02 -5. 2.
```

arguments :

date, H_debut, H_fin, sat, mode, repere, Nbp_FFT, F1, F2, Apod, XY ou LR, F1_p_int, seuil_min, seuil_max

This script is built as the previous `do_SP_plot_CLUSTA.sh`. Figure 17.4 hereafter show an example of spectrogram plot and average spectrum plot, with the same option "y" for Fci and spacecrafts positions plots.

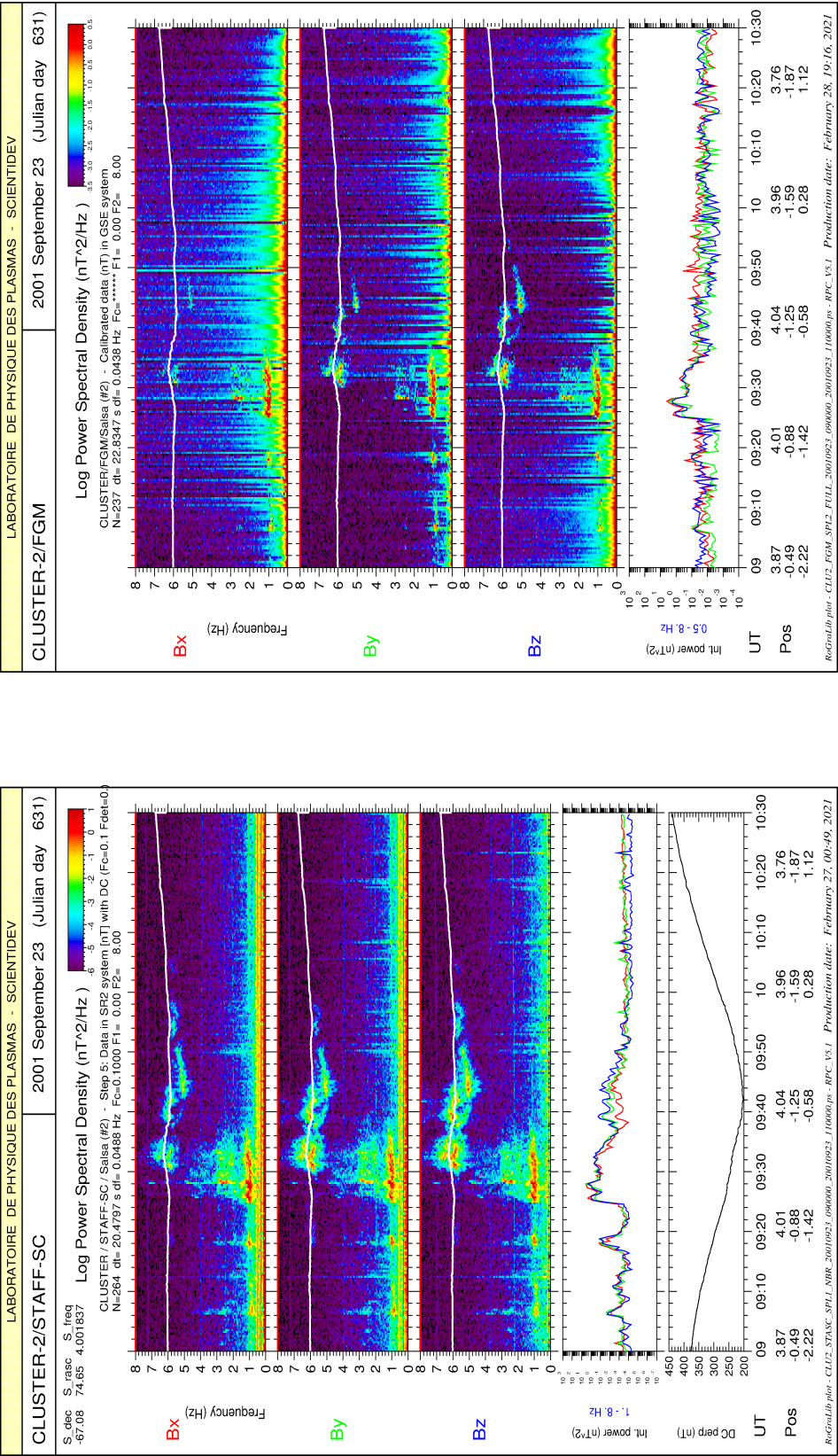


Fig. 17.3: Spectrogram plot of STAFF data (Normal Bit Rate) with
do_SP_plot_CLUSTA.sh script

Fig. 17.4: Spectrogram plot of FGM (full resolution) data with
do_SP_plot_CLUFGM.sh script

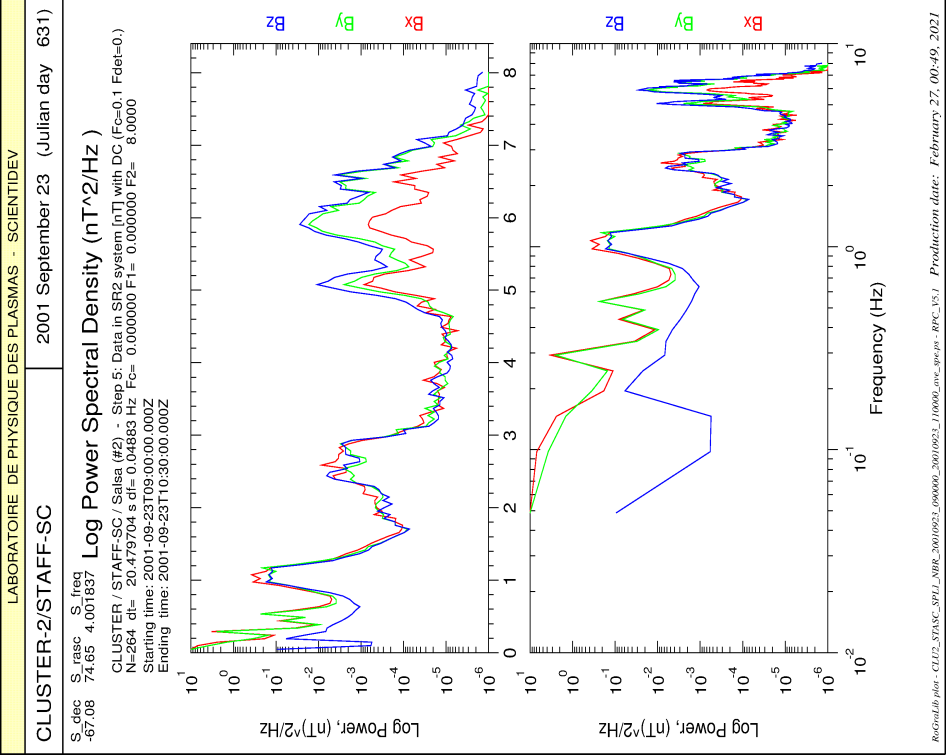


Fig. 17.5: Average plot of STAFF data (Normal Bit Rate) with `do_SP_plot_CLUSTER.sh` script

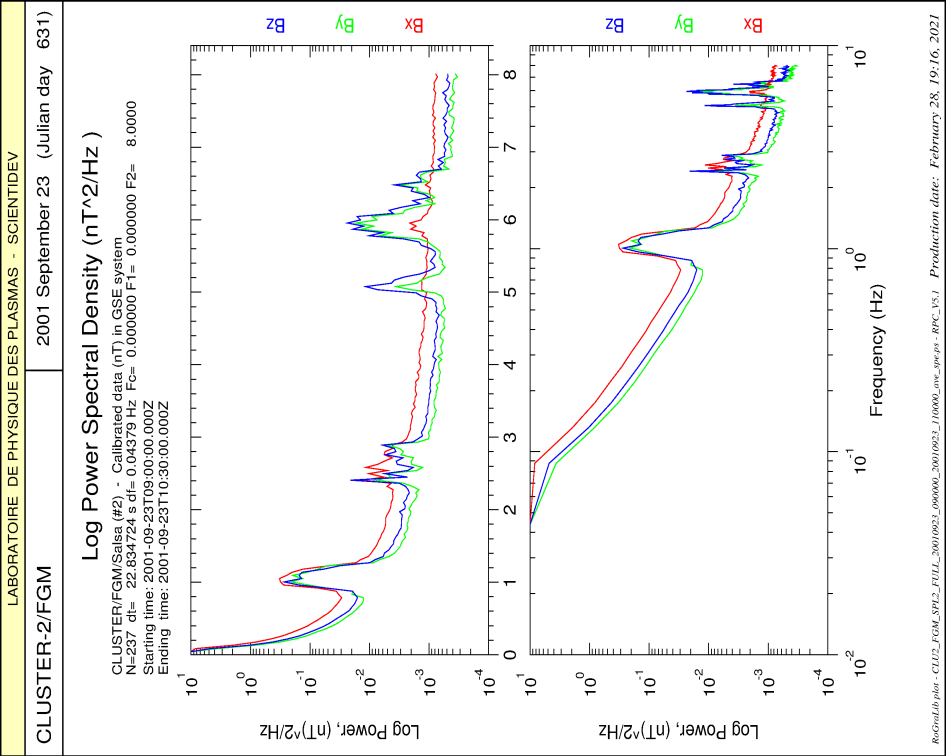


Fig. 17.6: Average plot of FGM (full resolution) data with `do_SP_plot_CLUSTER.sh` script

17.3 Spectrogram of 4 S/C for Bz component

It can be useful to compare spectrograms of STAFF or FGM for the 4 spacecrafts. The following scripts allow this comparison, by displaying the 4 Bz components of STAFF-SC or FGM.

17.3.1 do_SP_plot_CLUSTA_4Bz

Example:

```
do_SP_plot_CLUSTA_4Bz.sh 20010923 090000 103000 NBR GSE 512 0. 8.0 t -6. 1. 0.5 8.
```

This script is too long to be written here use the following commands, applied to the 4 S/C:

```
RPC_get_data_CLUSTA_VTL2
RPC_vectime_to_spectro
RPC_visu_spectro_4Bz
```

17.3.2 Result

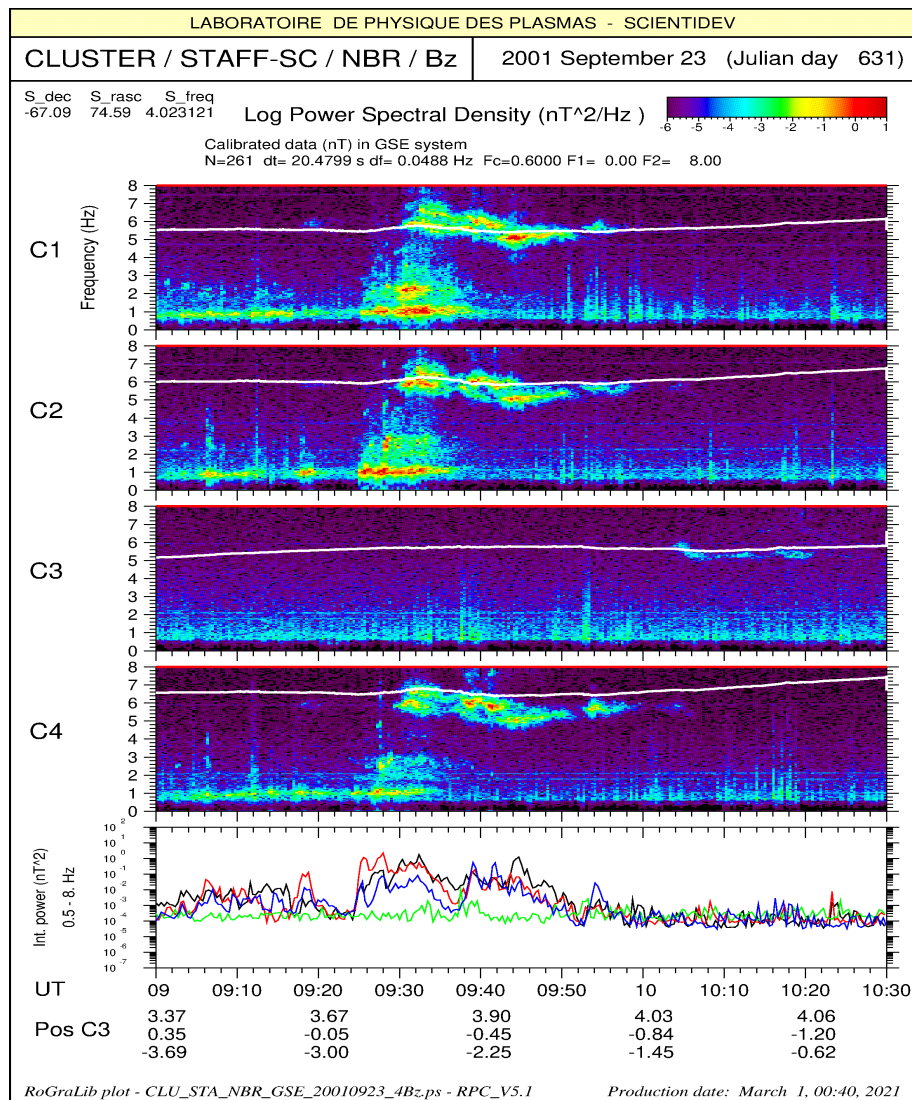


FIG. 17.7: 4Bz STAFF spectrogram plot provided by do_SP_plot_4Bz.sh script

17.3.3 do_SP_plot_CLUFGM_4Bz

Example:

```
do_SP_plot_CLUFGM_4Bz.sh 20010923 090000 103000 FULL 512 0. 8.0 t -3.5 0.5 0.02 8.0
do_SP_plot_CLUFGM_4Bz.sh 20010923 090000 103000 5VPS 256 0. 2.5 t -3.5 0.5 0.02 2.5
```

This script is too long to be written here use to the following commands, applied to the 4 S/C:

```
RPC_get_data_CLUFGM_VTL2
RPC_vectime_to_spectro
RPC_visu_spectro_4Bz
```

17.3.4 Result

Figure 17.8 and 17.9 show two example of FM spectrogrammes, for FULL mode and 5VPS mode.

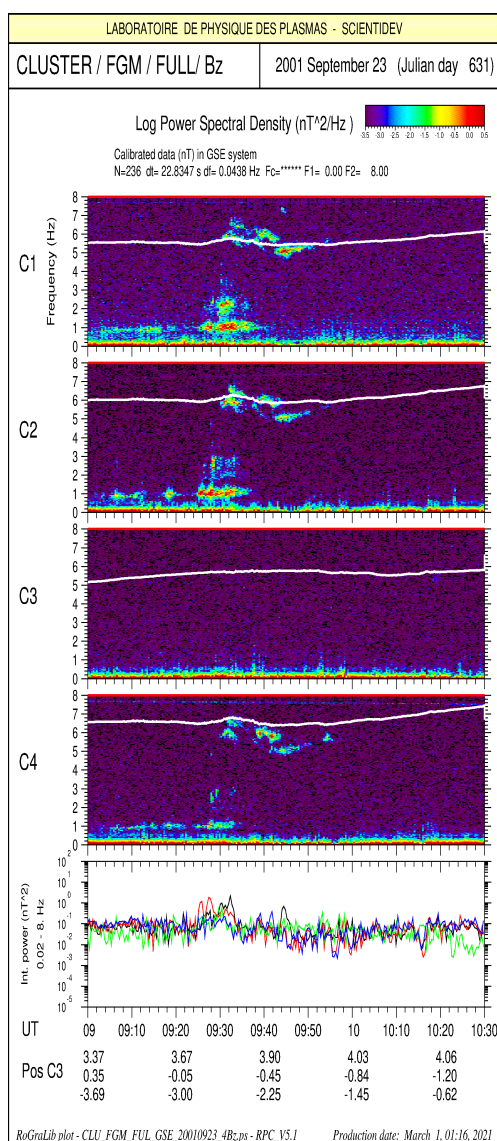


FIG. 17.8: 4Bz FGM (FULL) spectrogram plot provided by do_SP_plot_4Bz.sh script

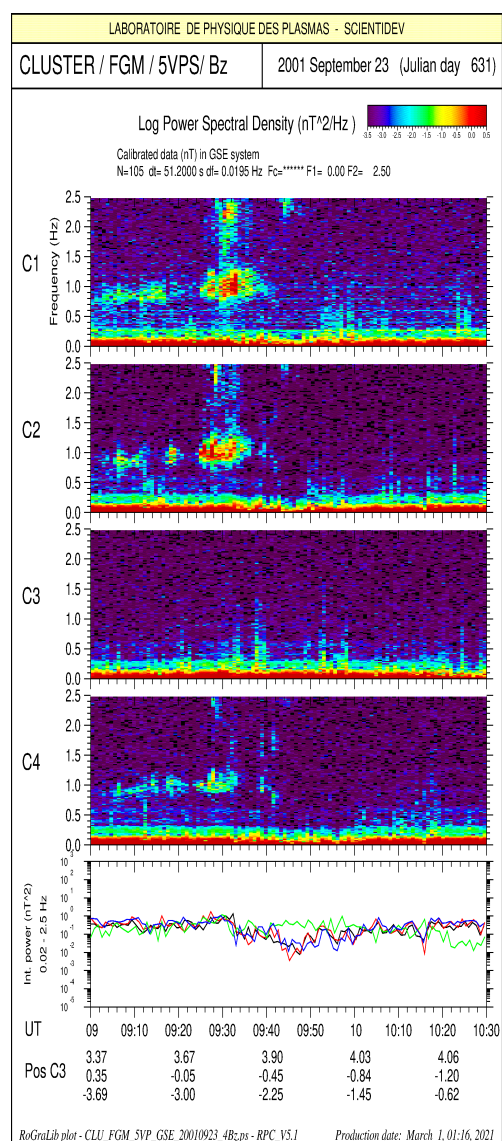


FIG. 17.9: 4Bz FGM (5VPS) spectrogram plot provided by do_SP_plot_4Bz.sh script

17.4 Plot the trajectory of the 4 satellites

17.4.1 do_2D_plot_CLUPOS.sh

Example :

```
do_2D_plot_CLUPOS.sh 20010923 090000 100000
```

is equivalent to :

```
RPC_get_data_CLUPOS_4sat 2001 09 23
datiso1='RPC_date_time_to_datiso 20010923 090000'
datiso2='RPC_date_time_to_datiso 20010923 100000 '
RPC_visu_vectime_4sat   CLU1_AUX_VTL2_POS_20010923_000000_20010923_235959.rff \
                           CLU2_AUX_VTL2_POS_20010923_000000_20010923_235959.rff \
                           CLU3_AUX_VTL2_POS_20010923_000000_20010923_235959.rff \
                           CLU4_AUX_VTL2_POS_20010923_000000_20010923_235959.rff \
                           $datiso1 $datiso2
```

Figures 17.12 and 17.13 hereafter show example of 2-D plots of the 4 CLUSTER S/C for two different scales.

17.4.2 do_3D_plot_CLUPOS.sh

Example :

```
do_3D_plot_CLUPOS.sh 20010923 090000 100000
```

is equivalent to :

```
RPC_get_data_CLUPOS_4sat 2001 09 23
datiso1='RPC_date_time_to_datiso 20010923 090000'
datiso2='RPC_date_time_to_datiso 20010923 100000 '
RPC_visu_vectime_3D_4sat CLU1_AUX_VTL2_POS_20010923_000000_20010923_235959.rff \
                           CLU2_AUX_VTL2_POS_20010923_000000_20010923_235959.rff \
                           CLU3_AUX_VTL2_POS_20010923_000000_20010923_235959.rff \
                           CLU4_AUX_VTL2_POS_20010923_000000_20010923_235959.rff \
                           $datiso1 $datiso2
```

Figures 17.12 and 17.13 hereafter show example of 3-D plots of the 4 CLUSTER S/C for two different scales.

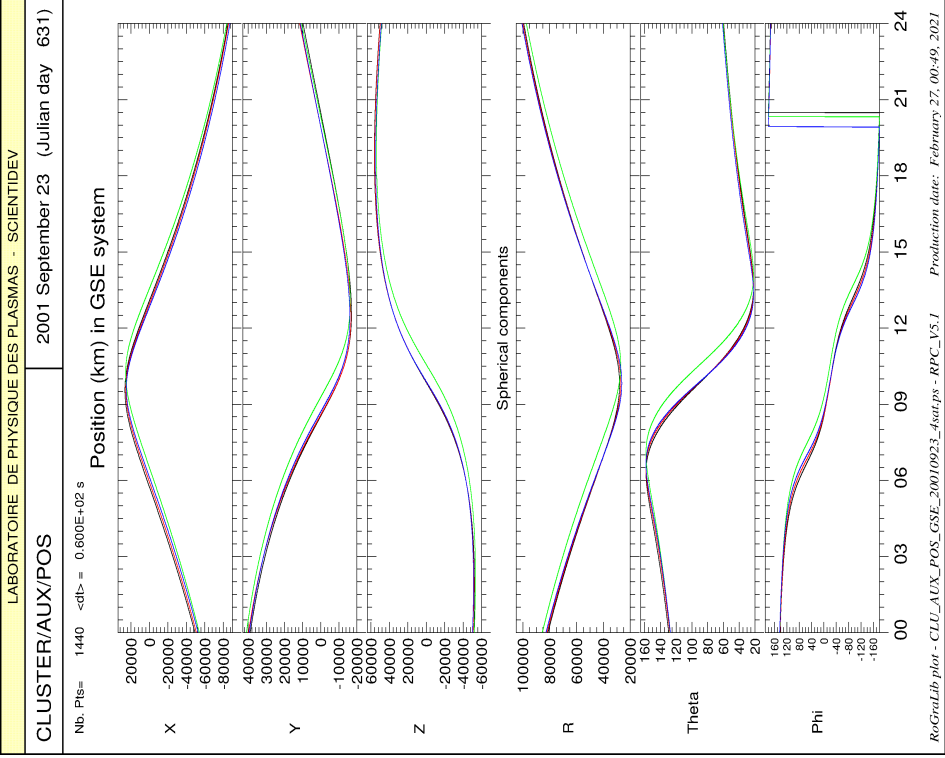


Fig. 17.10: 2-D plot of CLUSTER positions (4 S/C, 24 hours) with `do_2D_plot_CLUSTER.sh` script

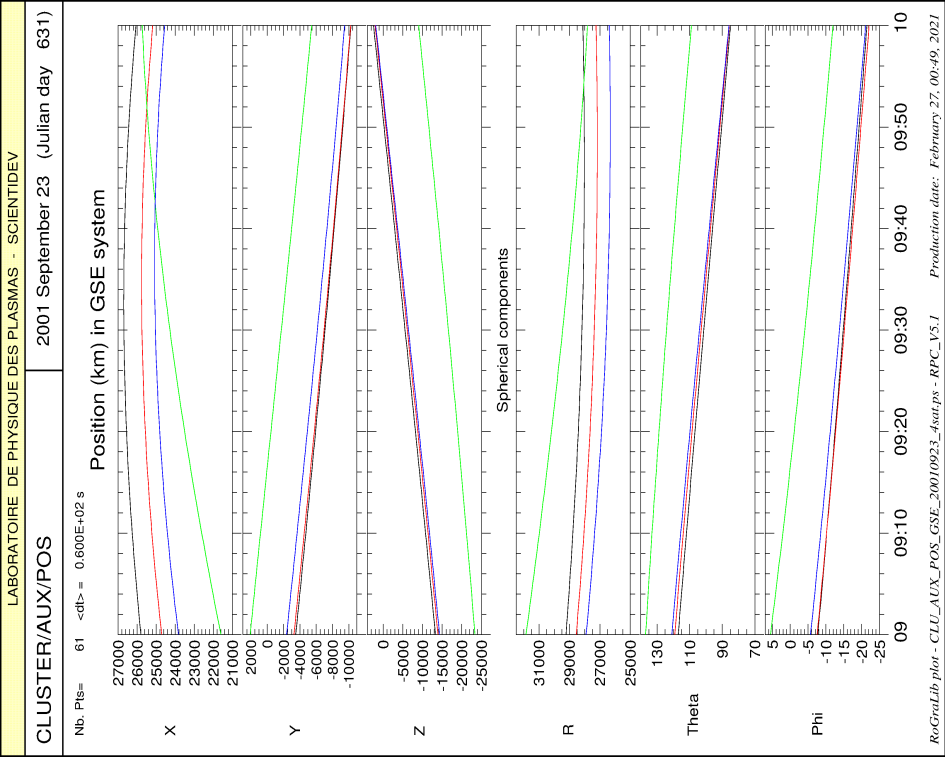


Fig. 17.11: 2-D plot of CLUSTER positions (4 S/C, 1 hour) with `do_2D_plot_CLUSTER.sh` script

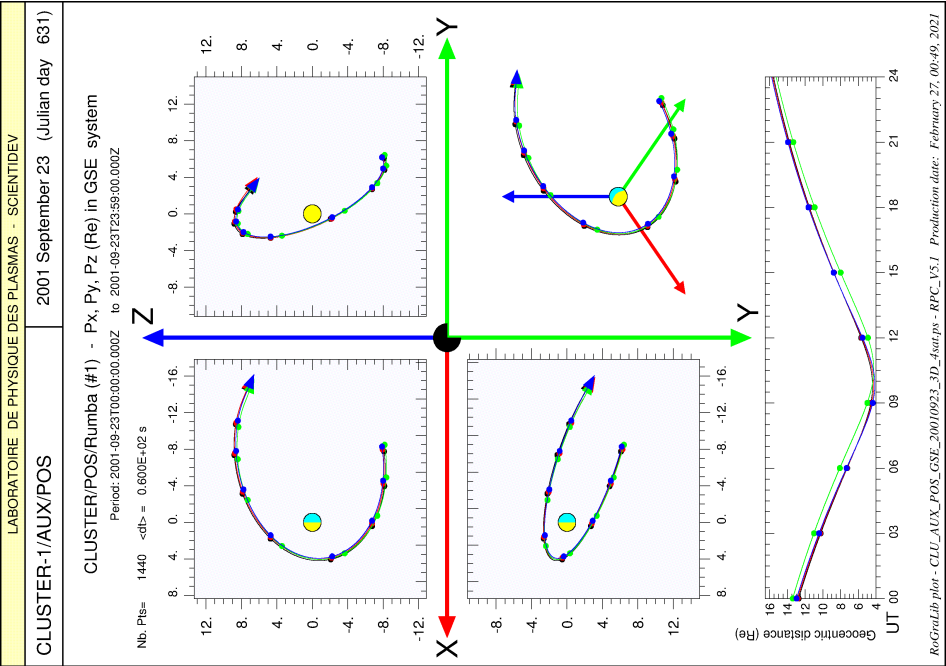


Fig. 17.12: 3-D plot of CLUSTER positions (4 S/C, 24 hours) with do_3D_plot_CLUSTER.sh script

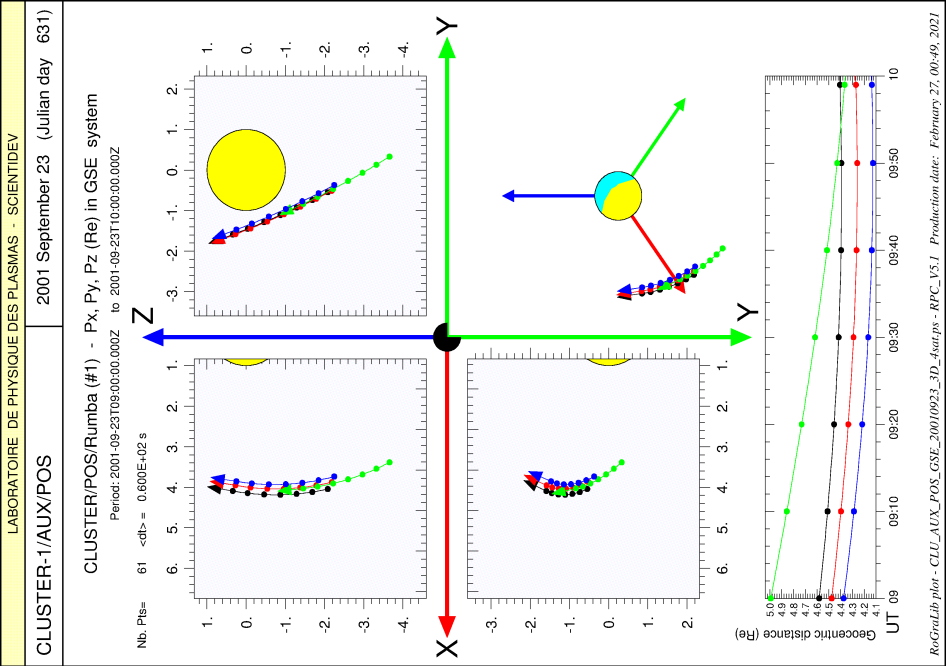


Fig. 17.13: 3-D plot of CLUSTER positions (4 S/C, 1 hours) with do_3D_plot_CLUSTER.sh script

17.5 Calculate and visualize waves polarization

17.5.1 do_Polar_plot_CLUSTA.sh

Examples :

```
do_Polar_plot_CLUSTA.sh 2 20010923 090000 103000 NBR GSE 512 0. 8.0 t XY 1. -5.
do_Polar_plot_CLUSTA.sh 2 20040130 164700 164750 HBR GSE 128 50. 225. t XY 50. -5.5
```

The second command is equivalent to :

```
RPC_get_data_CLUSTA_VTL2 2 2004 01 30 HBR GSE
RPC_get_data_CLUFGM 2 2004 01 30 SPIN
RPC_vectime_to_mfa_CLUSTA CLU2_STASC_VTL2_HBR_GSE_20040130_164700_20040130_164800.rff \
                           CLU2_STASC_VTL2_HBR_GSE_20040130_164700_20040130_164800_mfa.rff \
                           CLU2_FGM_VTL2_SPIN_20040130_163000_20040130_170000
RPC_vectime_to_spectro CLU2_STASC_VTL2_HBR_GSE_20040130_164700_20040130_164800_mfa.rff \
                       CLU2_STASC_SPL2_HBR_GSE_20040130_164700_20040130_164800_mfa.rff \
                       128 128 t
RPC_get_data_CLUPOS_4sat 2001 09 23
datiso1='RPC_date_time_to_datiso 20040130 164700'
datiso2='RPC_date_time_to_datiso 20040130 164800 '
RPC_visu_spectro CLU2_STASC_SPL2_HBR_GSE_20040130_164700_20040130_164800_mfa.rff \
                 $datiso1 $datiso2 50. 225. -9. -3. LR 50. 225. y
RPC_spectro_to_polar CLU2_STASC_SPL2_HBR_GSE_20040130_164700_20040130_164800_mfa.rff copolar.resu
RPC_visu_polar copolar.resu 0 0 -5.5 50. 225.
```

In this example we see the usefulness of scripts: we execute with a single command line a tedious process to write.

17.5.2 Example of result in HBR

Following figures show result for the event of 2004, january 30. On figure 17.14 and 17.15 we can see spectrograms in GSE and MFA system. Note that one have the FLH Low Hybrid Frequency plotted in red superimposed to the spectrogram. Accordind frequency range, Fci or FLH are plotted with the formulas:

$$Fci = |Bo| * 1.52E-2 \text{ (Bo en nT)} \quad FLH = \sqrt{1836.} * 6.2832 * Fci$$

Polarisation parameters are given in FIG. 17.16. Interpretation is following:

eccentricity :	green,	so ~ 0.5,	⇒ circular polarization.
theta K :	dark purple,	so ~ 30 °,	⇒ right circular polarization with K parallel to Bo.
theta Major axis :	green,	so ~ 90 °,	⇒ perpendicular to Bo, consistent with the direction of K.

17.5.3 Example of result in NBR

Following figures show result for the event of 2001, september 23. On figure 17.17 we can see spectrogram in MFA system. Note that one have the Fci Frequency plotted in white superimposed to the spectrogram.

Polarisation parameters are given in FIG. 17.18. Interpretation is following:

eccentricity :	red,	so ~ 1,	⇒ linear polarization.
theta K :	green,	so ~ 90 °,	⇒ linear polarization with K ⊥ to Bo.
theta Major axis :	red or dark blue,	so ~ 0 or 180 °,	⇒ parallel to Bo, consistent with the direction of K.

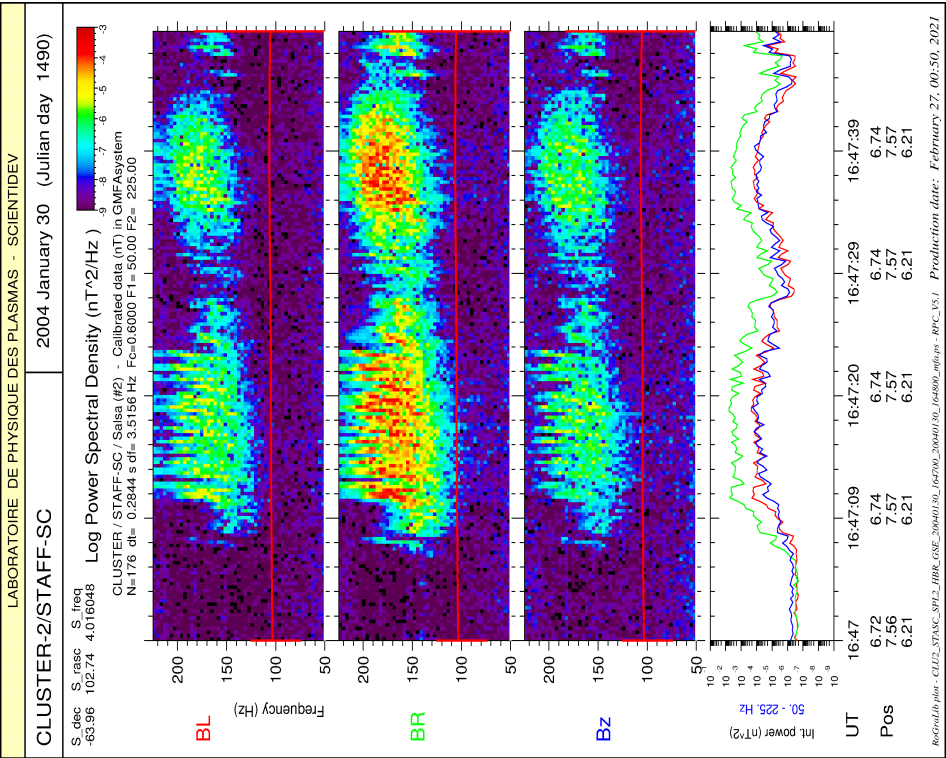


Fig. 17.15: Spectrogram of STAFF-SC data in MFA provided by do_Polar_plot_CLUSTERA.sh script

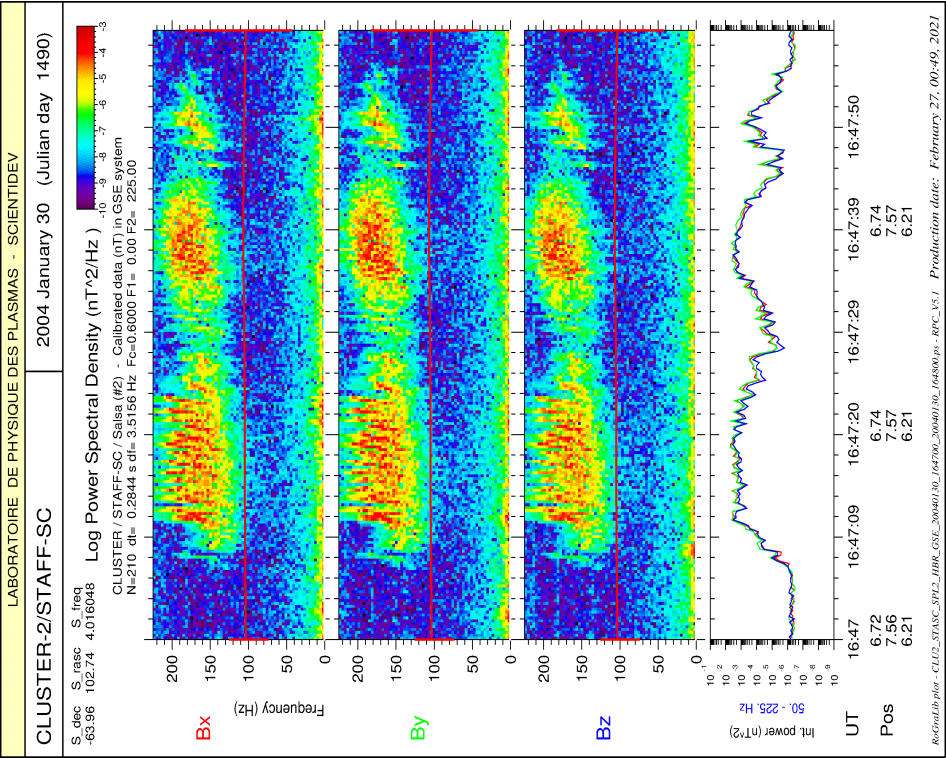


Fig. 17.14: Spectrogram of STAFF-SC data in GSE provided by do_Polar_plot_CLUSTERA.sh script

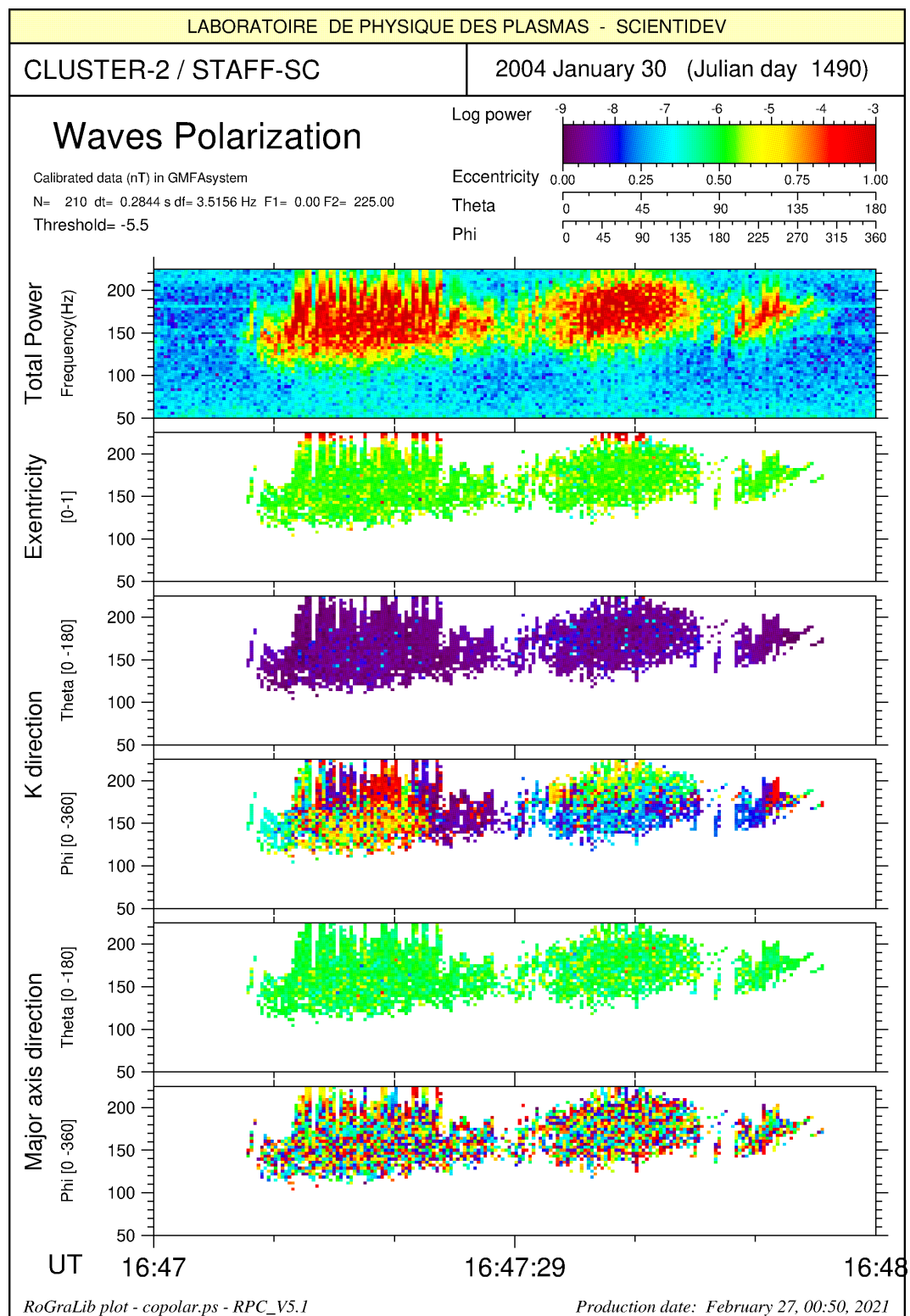


FIG. 17.16: Polarization parameters in MFA provided by `do_Polar_plot_CLUSTA.sh` script; see interpretation in section 17.5.2

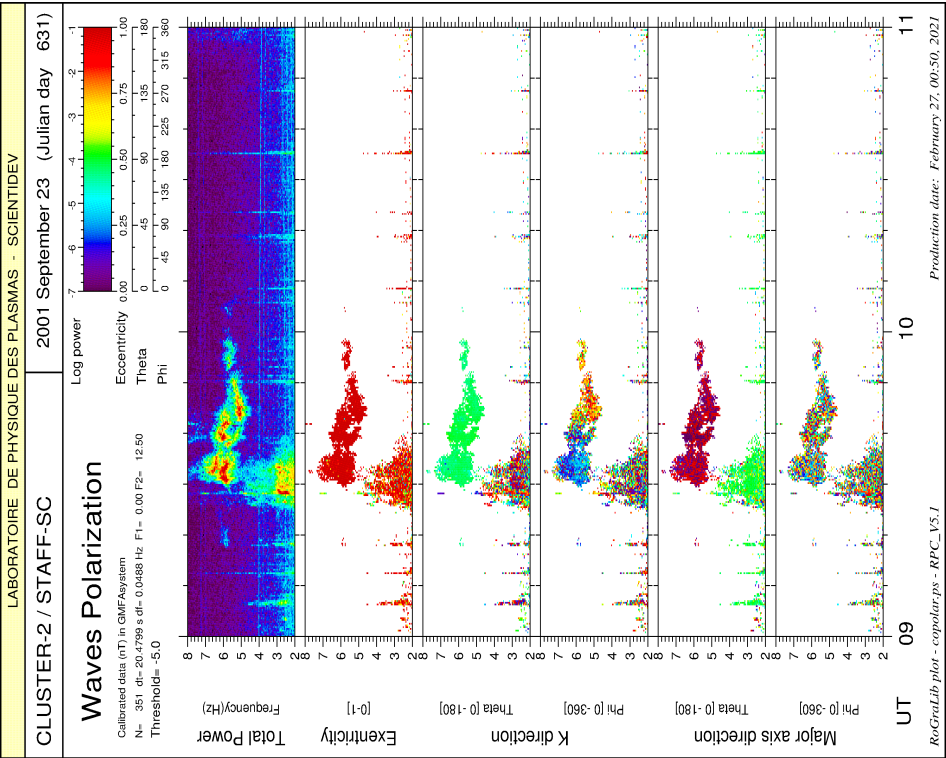


Fig. 17.17: Spectrogram of STAFF-SC data in GSE provided by `do_Polar_plot_CLUSTA.sh` script

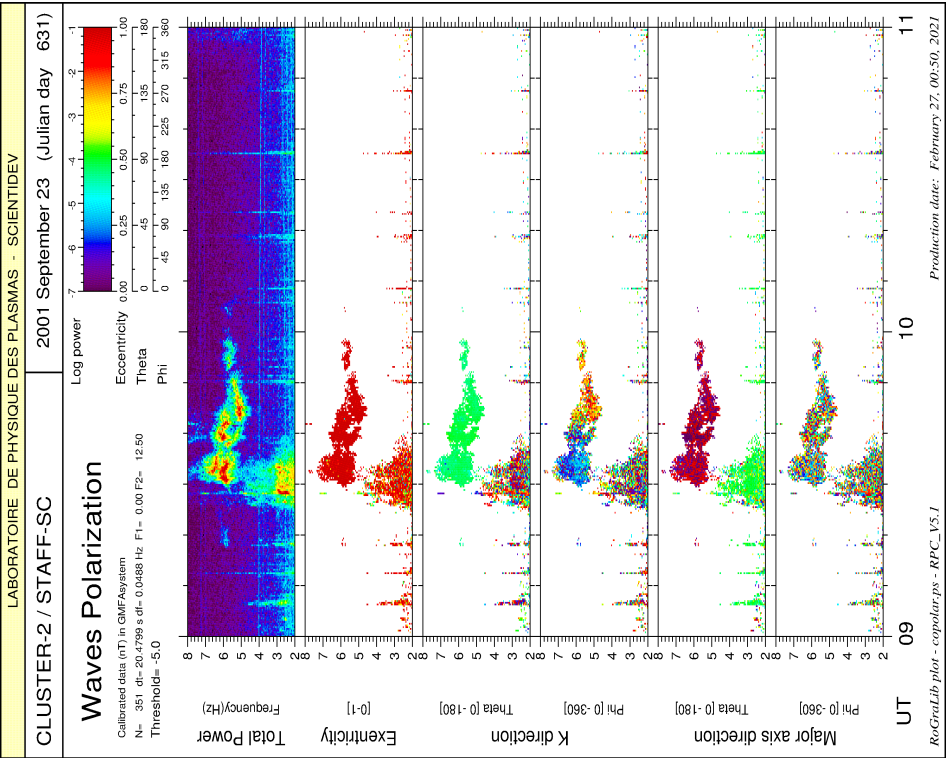


Fig. 17.18: Polarization parameters in MFA provided by `do_Polar_plot_CLUSTA.sh` script; see interpretation in section 17.5.3

17.6 STAFF and FGM comparison

The rotation of the STAFF ULF antennas in the spin plane makes it possible to measure the two components of the continuous field in this plane [10] and therefore to be able to compare them with those of the magnetometer transformed in the same frame. The following script performs this work, and plots these two components on the same graph.

17.6.1 do_compare_STA_FGM_CLUSTER.sh

Examples:

```
do_compare_STA_FGM_CLUSTER.sh 2 20010923 092000 093000 NBR ISR2
do_compare_STA_FGM_CLUSTER.sh 3 20010110 000000 020000 NBR
```

This script is quite detailed, and is not explained here. It can be viewed in the test directory. It uses the following commands:

```
RPC_get_data_CLUSTA_VTL2
RPC_get_data_CLUFGM
RPC_add_DxDy_to_BxBy
RPC_vectime_gse_to_isr2
RPC_visu_2_vectime
```

17.6.2 Examples of STAFF/FGM DC field comparison

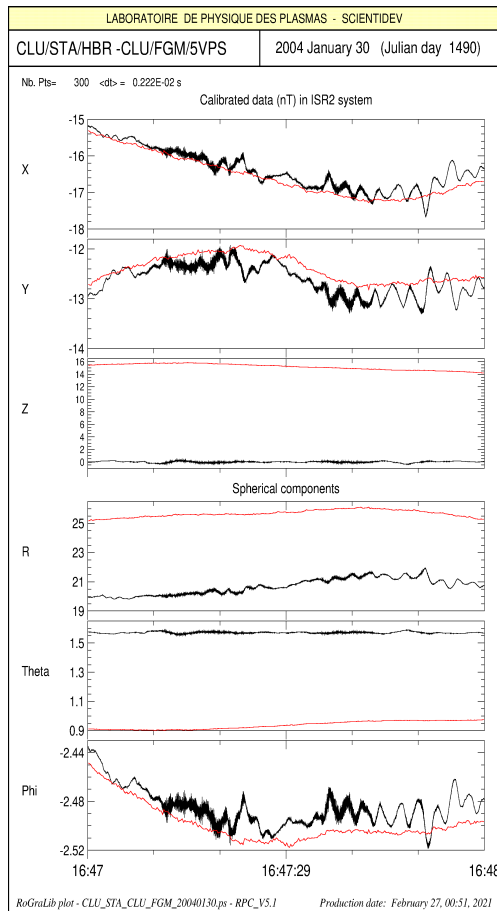


FIG. 17.19: STAFF/FGM comparison of two components of DC magnetic field

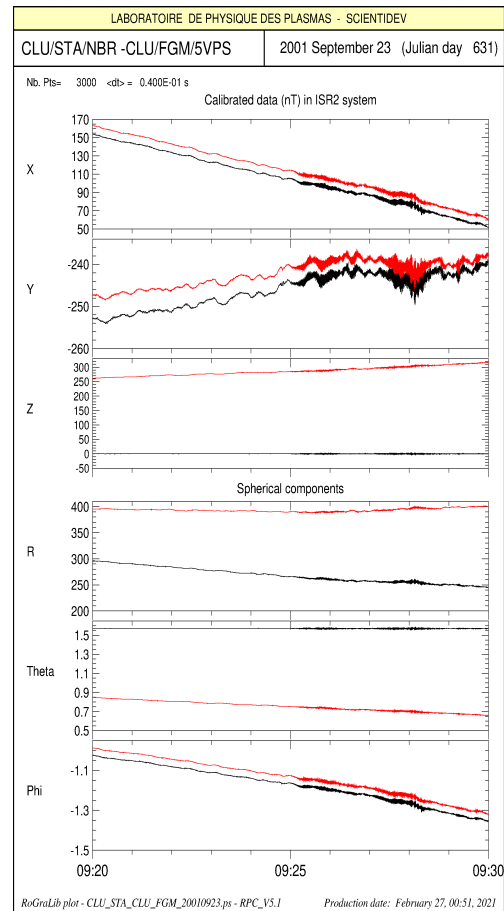


FIG. 17.20: Another example

17.7 Calculate the orbital parameters of a satellite

17.7.1 do_simulated_orbit_CLUPOS.sh

Example :

```
do_simulated_orbit_CLUPOS.sh 2 20040129 0 0
```

This rather complex script is not detailed here; it can be viewed in the scripts directory. It connects two treatments:

- On the one hand it uses the procedure **RPC_compute_sat_orbit_param** on a GEOS-2 position file in order to calculate the orbital parameters.
- On the other hand it uses these orbital parameters in the procedure **RPC_compute_sat_trajectory** to recalculate the full orbit over a period.

17.7.2 Results

Figure 17.21 show the CLUSTER trajectory of S/C 2 and the re-computed trajectory from the orbital parameters. CLUSTER data are in blue, while theoretical values are in red. Curves are perfectly superimposed.

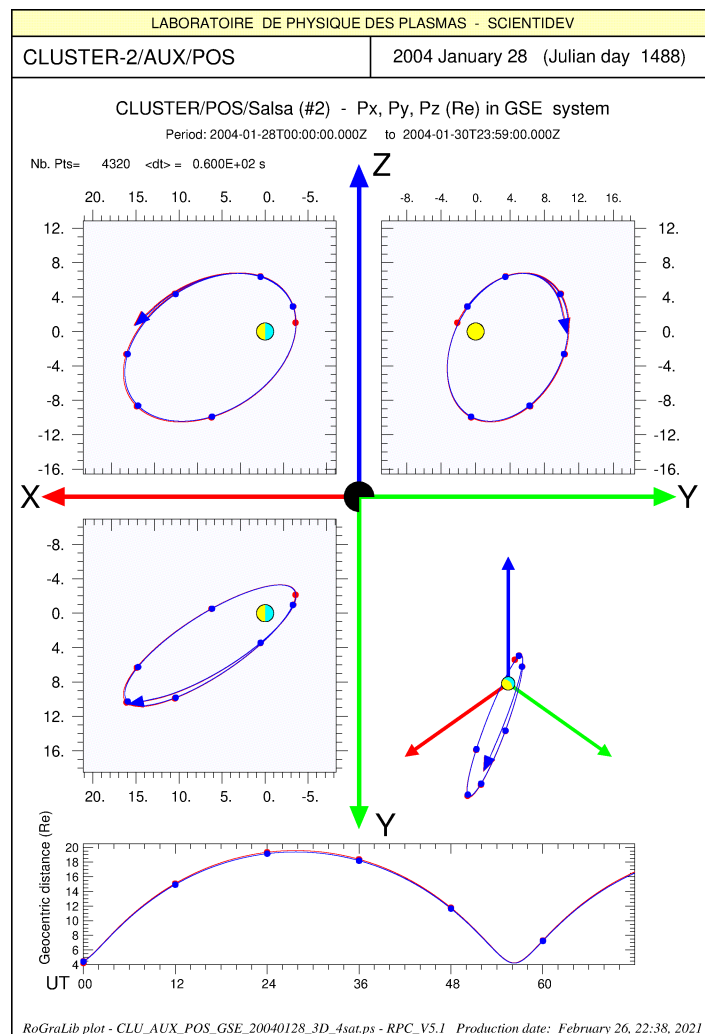


FIG. 17.21: Comparison of CLUSTER orbit data and re-computed simulated orbit

17.8 CLUSTER tetrahedron geometry

17.8.1 Geometry parameters computation

This is done by the command **RPC_get_data_CLUGEOM** which create the file **get_data_CLUGEOM.resu**.

Example:

```
RPC_get_data_CLUGEOM 2001 09 23
```

and the **get_data_CLUGEOM.resu** file looks like:

```
# START_HEADER
File type : get_data_CLUGEOM.resu
Creator   : get_data_CLUGEOM.exe
Version   : April 2001
Author    : P. Robert, CNRS/CETP
Coordinates: GSE
Nbloc     : 144
Content   : see following line
JD        YY MM DD hh mm ss s/c X (km) Y Z
          s/c X (km) Y Z in centre of mass
          Dij, 6 values
          Si , 4 values
          a,b,c
          Elong, tetE, phiE
          Plana, tetP, phiP
-----
# END_HEADER
631 2001 09 23 00 00 00 1 -49317.19 38843.54 -51677.20
                               2 -50778.10 39846.40 -50619.64
                               3 -54016.06 40982.33 -52098.46
                               4 -52751.73 39373.12 -50169.30
                               2063.59 5179.89 3788.18 3736.52 2078.94 2812.42
                               0.520295E+07 0.158762E+07 0.319829E+07 0.289127E+07
                               5426.26 2331.55 967.59
                               0.57032 86.61 -20.58
                               0.58500 67.13 70.85
631 2001 09 23 00 10 00 1 -48092.09 38495.34 -51838.40
                               2 -49570.36 39489.37 -50795.39
...
631 2001 09 23 23 50 00 1 -84357.52 9309.78 50558.55
                               2 -85046.93 11145.10 50304.59
                               3 -83024.45 10101.44 48716.89
                               4 -86557.44 10770.22 49081.88
                               1976.91 2407.39 3025.40 2774.96 1979.19 3614.20
                               0.361100E+07 0.192798E+07 0.232630E+07 0.272223E+07
                               3781.07 2303.50 1471.68
                               0.39078 93.12 -19.18
                               0.36111 68.02 69.56
```

17.8.2 Geometry parameters visualization

This is done by the command **RPC_visu_CLUGEOM** which read the file **get_data_CLUGEOM.resu**.

Results are shown on figure [17.22](#)

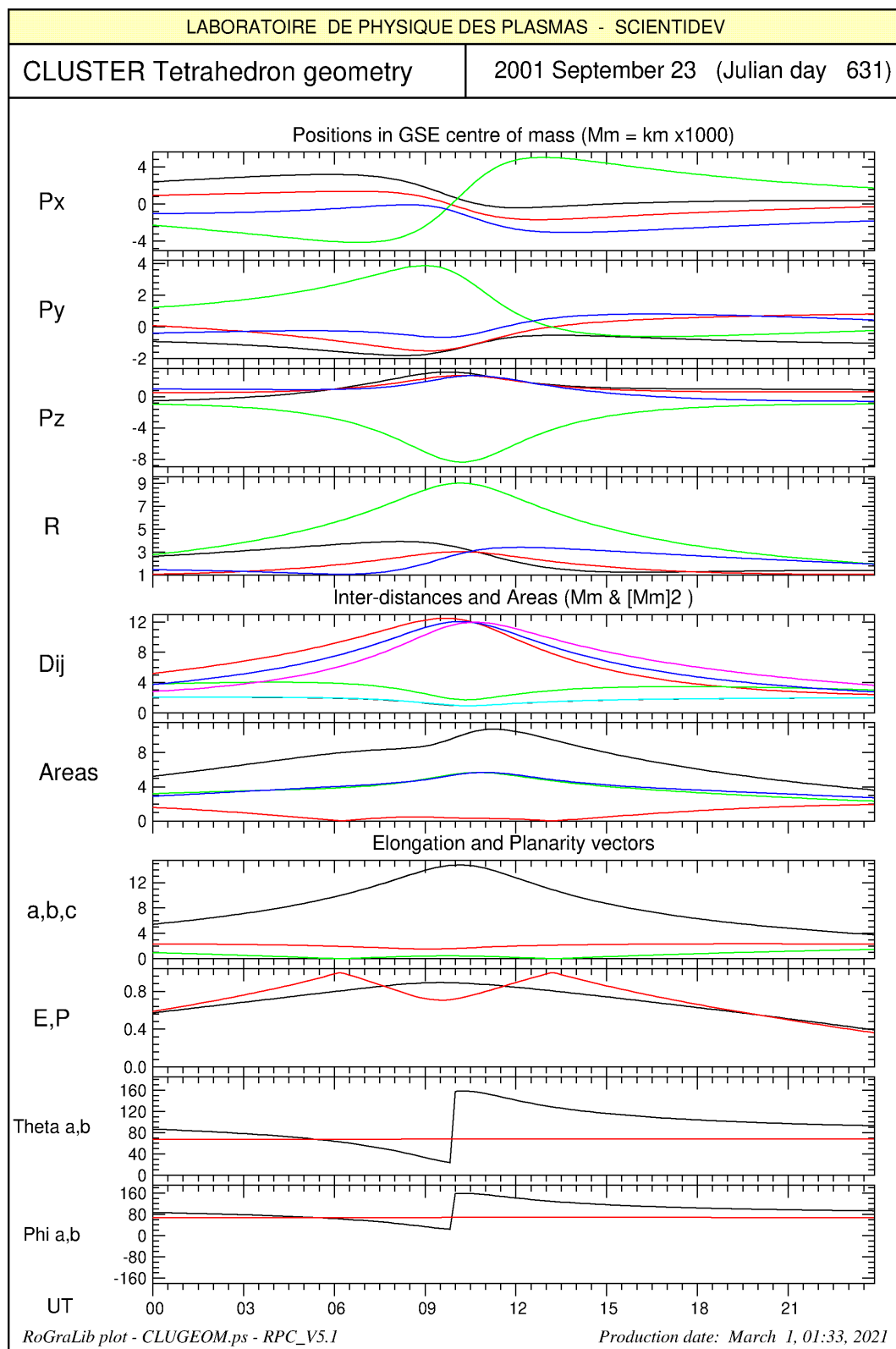


FIG. 17.22: Cluster geometry parameters

17.9 Curl(B) and Div(B) computation

17.9.1 Introduction to the method

Computation of curl(B) and div(B) is done by the command::

```
RPC_compute_curl_div_4sat SC1_MAG.rff SC2_MAG.rff SC3_MAG.rff SC4_MAG.rff
                          SC1_POS.rff SC2_POS.rff SC3_POS.rff SC4_POS.rff
```

This command use the Ampere's law to compute Curl(B) as:

$$\oint \vec{B}(M) \cdot d\vec{l} = \mu_0 I$$

And to compute div(B) we use the divergence law, or Green-Ostrogradski law, as:

$$\iiint_V \vec{\nabla} \cdot \vec{B} dV = \oint_{\partial V} \vec{B} \cdot d\vec{S}$$

with $\mu_0 = 4\pi \times 10^{-7} T \cdot m/A$

This command create a file **compute_curl_div_4sat.resu** which contains all data about the fields, curl, div etc.

The validity of the computation has been checked on simulated data in section 4.3, chapter 6.

Nevertheless, to apply this method (or any other one) to experimental data, it is necessary to have the 4 components of the magnetic field at the same time, as well as the 4 positions of the 4 Spacecrafts.

So, the script **do_compute_curl_and_div_CLUFGM.sh** which performs this computation use a time alignment command:

```
RPC_alitime_4_vectime file1 file2 file3 file4
```

This command, applied to 4 FGM files, interpolate each point to ensure that all point have now the same time reference within the 4 files. So, we obtain a set of FGM data with a same time stamp which are read by the computation program which provide estimate of Curl(B) and div(B). For a given FGM time, the program search the corresponding position in the POS file, by interpolate the POS data.

17.9.2 do_compute_curl_div_CLUFGM.sh script

This command use the 4 time aligned FGM data, and the 4 time aligned POS data.

One run it by:

```
do_compute_curl_div_CLUFGM.sh 2001 09 23 SPIN GSM
```

As said above, this script use the following commands:

```
RPC_get_data_CLUFGM_4sat
RPC_alitime_4_vectime
```

and run the **compute_curl_div_4sat.exe** program.

This program create the file **compute_curl_div_4sat.resu** which contains many quantities:

- the 4 \vec{B}_{ij} and the 4 P_{ij} positions in GSE system.
- the estimated $\vec{\nabla} \times \vec{B}$ and $\vec{\nabla} \cdot \vec{B}$,
- the radius of curvature of the mean $\langle \vec{B} \rangle$,
- the Elongation and Planarity parameters,
- the parallel and perpendicular component of the current density \vec{J} by respect to $\langle \vec{B} \rangle$,
- the (\vec{J}, \vec{B}) angle,
- the vector $\vec{N} = \vec{J} \times \vec{B}$,
- the normal to the osculator plane,

17.9.3 RPC_visu_curl_div_4sat command

This command read the file **compute_curl_div_4sat.resu** and create two plots:

- One plot visualizing POS and MAG data, in GSE or GSE according choosed option above,
- A second plot, visualising current density \vec{J} ; (\vec{J}, \vec{B}) angle, ration $\text{div}(\vec{B})/\text{curl}(\vec{B})$, Elongation, planarity, interdistances.

Example are given in the following figures.

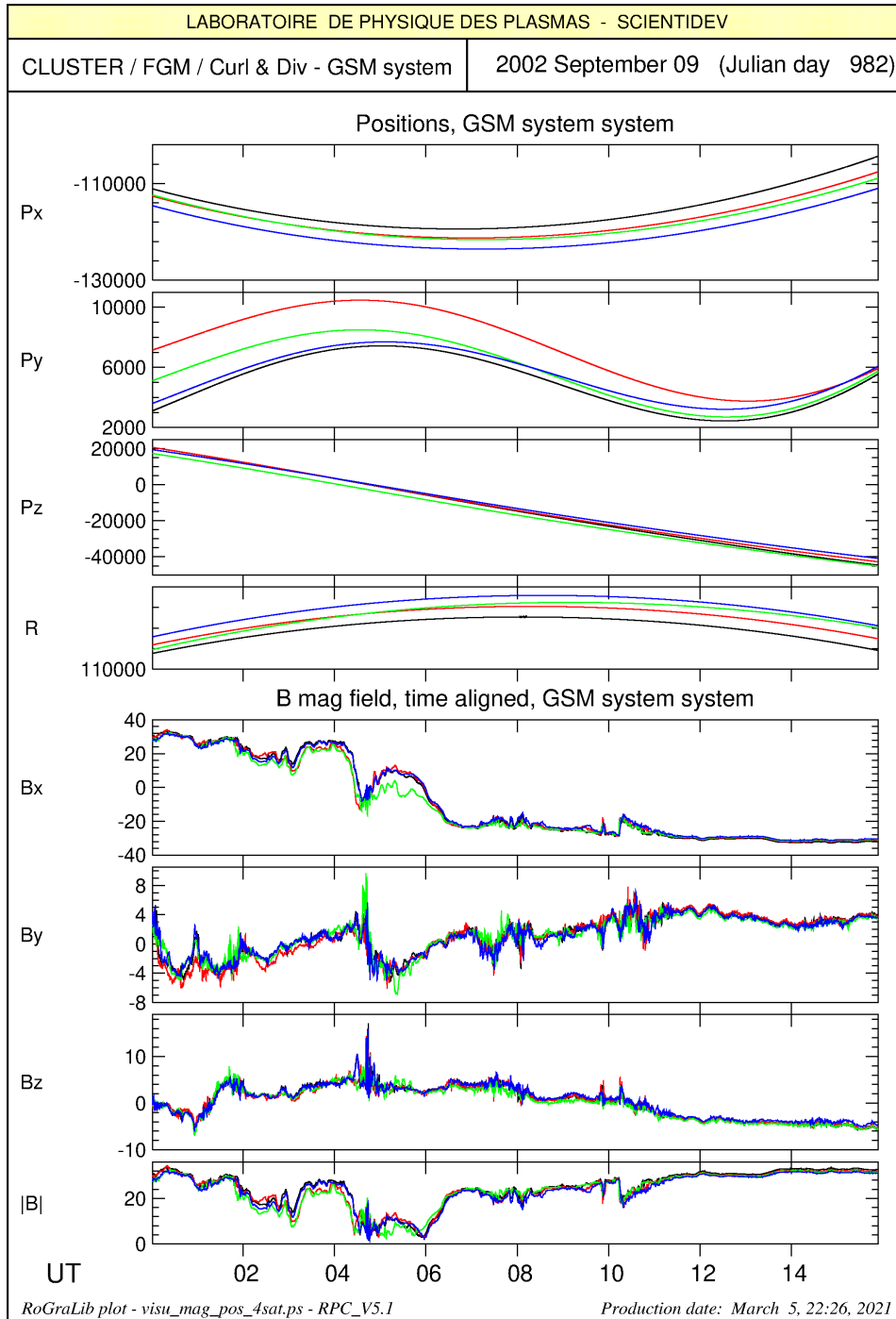


FIG. 17.23: Time aligned magnetic fields and positions (ex. 1)

Note that on figure 17.23 the CLUSTER S/C was in the magnetotail, in the dark side, and we found on fig. 17.24 a current density toward the Y of GSM axis, and \sim perpendicular to the magnetic field. When the current is high, the ration $\text{div}(\mathbf{B})/\text{curl}(\mathbf{B})$ is low, which is rather reassuring.

On the contrary, on figure 17.26 we found a current density along X axis, so parallel to the magnetic field.

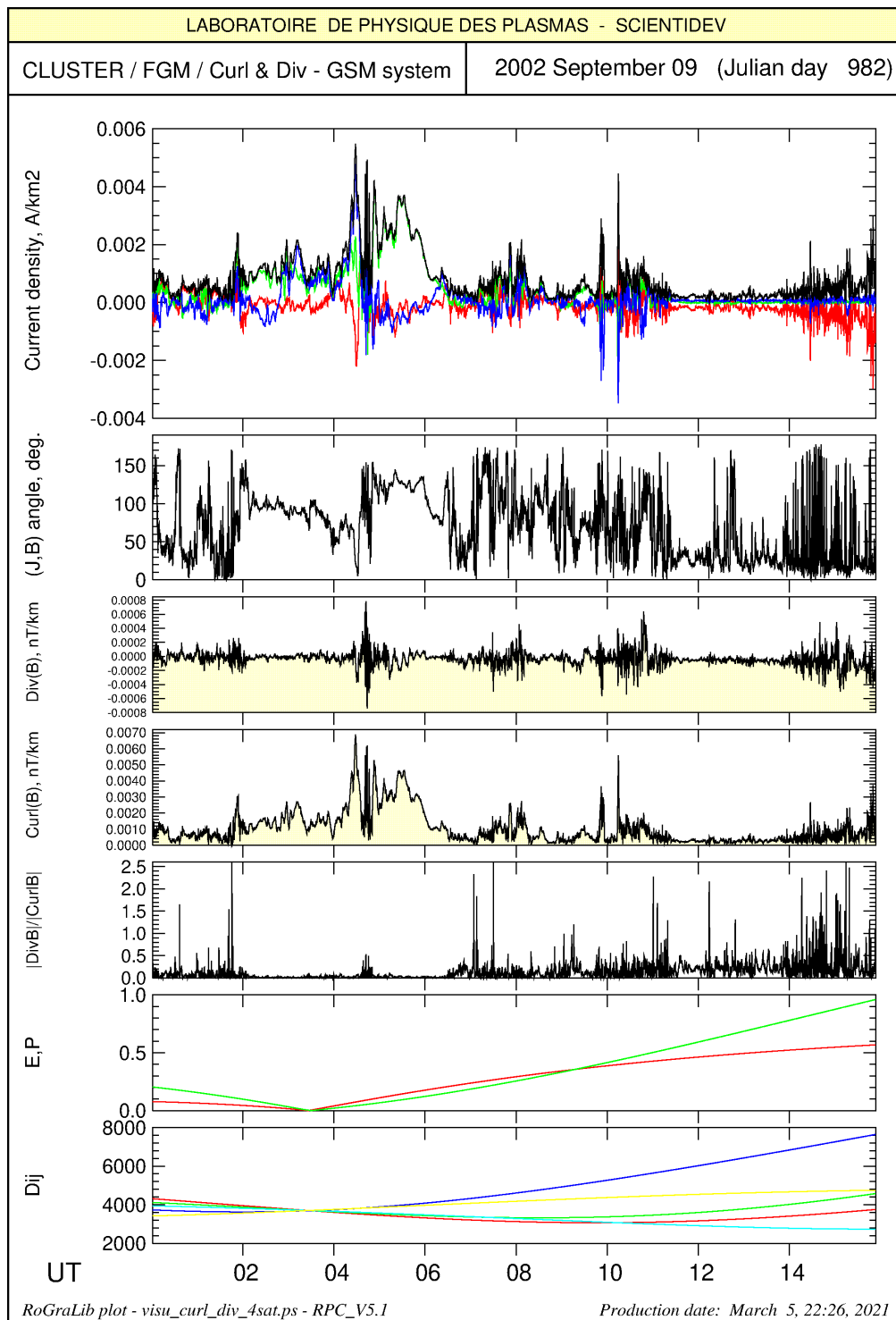


FIG. 17.24: Estimated current density and other parameters (ex. 1)

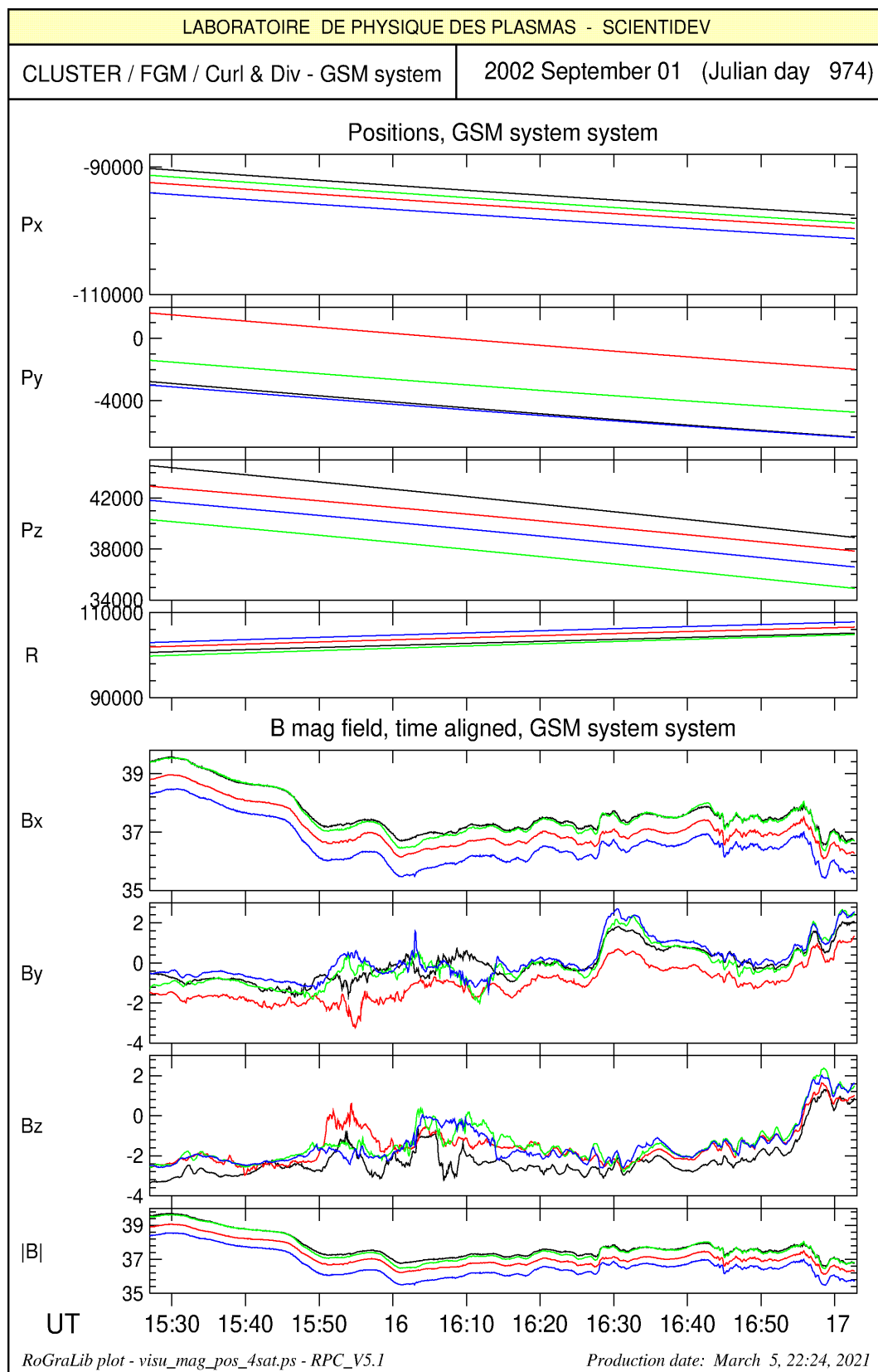


FIG. 17.25: Time aligned magnetic fields and positions (ex. 2)

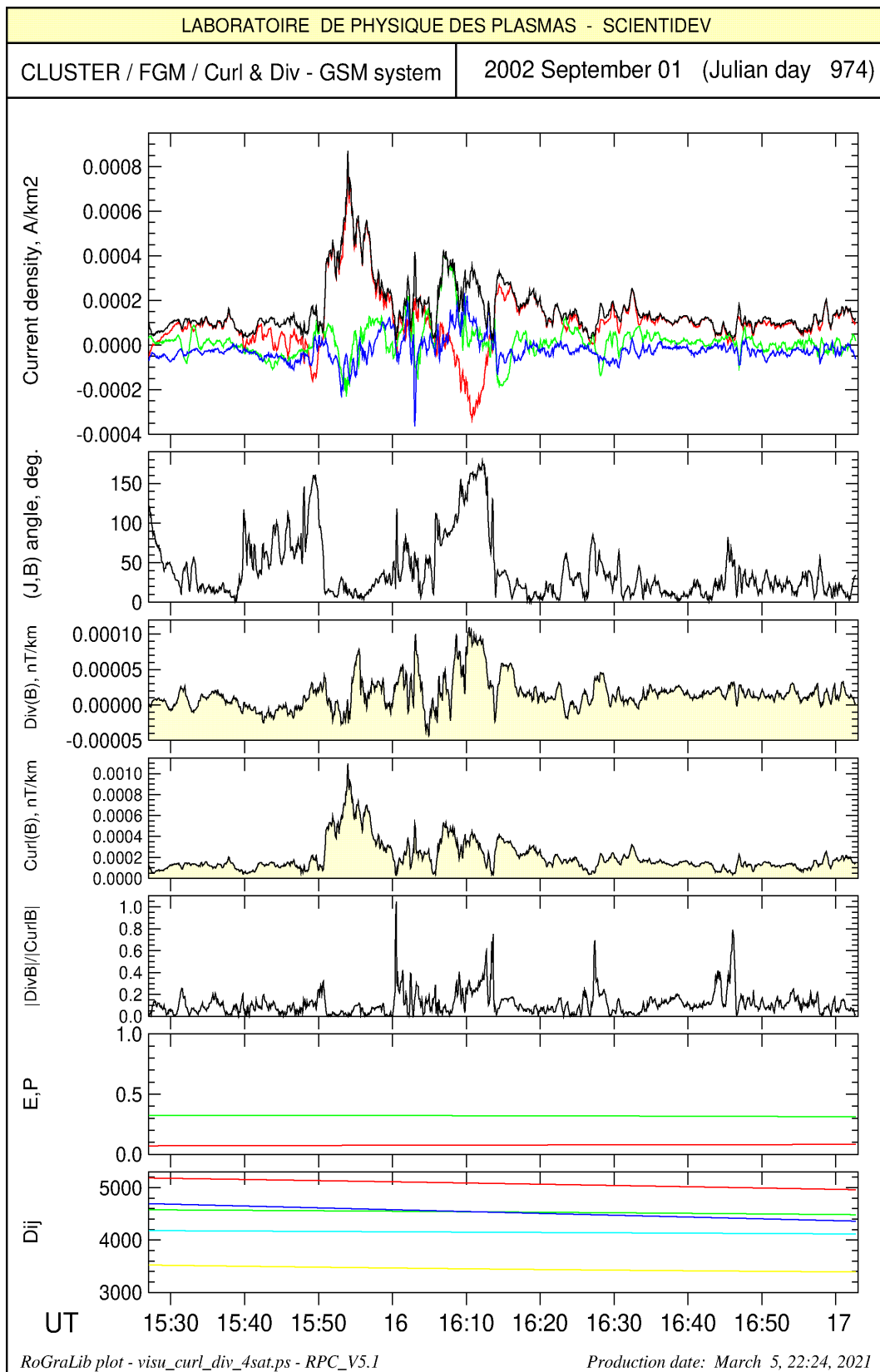


FIG. 17.26: Estimated current density and other parameters (ex. 2)

Chapter 18

TESTS OF CLUSTER SCRIPTS

The tests of the RPC commands are made from the scripts, which, as we have seen, successively launch several RPC commands for a particular treatment. For each script it was made a test, which are summarized here.

18.1 test_do_VT_plot_CLUXXX.sh

This test runs the following scripts, on STAFF and FGM data.

```
do_VT_plot_CLUSTA.sh 20010923 092000 093000 2 NBR ISR2
do_VT_plot_CLUSTA.sh 20010923 092000 093000 2 NBR GSE
do_VT_plot_CLUSTA.sh 20040130 164710 164730 2 HBR GSE

do_VT_plot_CLUFGM.sh 20010923 092000 093000 2 FULL
do_VT_plot_CLUFGM.sh 20010923 092000 093000 2 5VPS
do_VT_plot_CLUFGM.sh 20040130 164700 164800 2 5VPS
```

18.2 test_do_SP_plot_CLUXXX.sh

This test runs the following scripts:

```
do_SP_plot_CLUSTA.sh 20010923 090000 103000 2 NBR ISR2 512 0. 8.0 t XY 1. -6. 1.
do_SP_plot_CLUSTA.sh 20010923 090000 103000 2 NBR GSE 512 0. 2.5 t XY 1. -6. 1.
do_SP_plot_CLUSTA.sh 20040130 164700 164800 2 HBR GSE 128 0. 225. t XY 1. -10. -3.

do_SP_plot_CLUFGM.sh 20010923 090000 103000 2 FULL 512 0. 8.0 n XY 0.02 -7. 0.
do_SP_plot_CLUFGM.sh 20010923 090000 103000 2 5VPS 256 0. 2.5 n XY 0.02 -5. 2.
```

18.3 test_do_SP_plot_CLUXXX_4Bz.sh

This test runs the following scripts:

```
do_SP_plot_CLUSTA_4Bz.sh 20010923 090000 103000 NBR GSE 512 0. 8.0 t -6. 1. 0.5 8.
do_SP_plot_CLUFGM_4Bz.sh 20010923 090000 103000 FULL 512 0. 8.0 t -5. 1. 0.02 8.0
do_SP_plot_CLUFGM_4Bz.sh 20010923 090000 103000 5VPS 256 0. 2.5 t -5. 1. 0.02 2.5
```

18.4 test_do_SP_plot_CLUSTA_from_VTL1.sh

This test runs the following scripts:

```
do_SP_plot_CLUSTA_from_VTL1.sh 20010923 090000 103000 2 NBR 512 0. 8.0 †XY 1. -6. 1.
do_SP_plot_CLUSTA_from_VTL1.sh 20040130 164700 164800 2 HBR 128 0. 225. †XY 1. -10. -3.
```

18.5 test_do_Polar_plot_CLUSTA.sh

This test runs the following scripts:

```
do_Polar_plot_CLUSTA.sh 20010923 090000 103000 2 NBR GSE 512 2. 8.0 †XY 1. -5. -6. 0.
do_Polar_plot_CLUSTA.sh 20040130 164700 164750 2 HBR GSE 128 50. 225. †XY 50. -5.5 -6. 0.
```

18.6 test_do_2D_plot_CLUPOS.sh

This test runs the following scripts:

```
do_2D_plot_CLUPOS.sh 20010923 0 0
do_2D_plot_CLUPOS.sh 20010923 090000 100000
```

18.7 test_do_3D_plot_CLUPOS.sh

This test runs the following scripts:

```
do_3D_plot_CLUPOS.sh 20010923 0 0
do_3D_plot_CLUPOS.sh 20010923 090000 100000
```

18.8 test_do_compare_STA_FGM_CLUSTER.sh

This test runs the following scripts:

```
do_compare_STA_FGM_CLUSTER.sh 2 20010923 092000 093000 NBR
do_compare_STA_FGM_CLUSTER.sh 3 20010110 000000 020000 NBR
do_compare_STA_FGM_CLUSTER.sh 2 20040130 164700 164800 HBR
```

18.9 test_CLUGEOM.sh

This test runs the following commands:

```
RPC_get_data_CLUGEOM 2001 09 23
RPC_visu_CLUGEOM
```

18.10 test_curl_div_CLUFGM.sh

This test runs the following commands:

```
do_compute_curl_div_CLUFGM.sh 2001 09 23 2001-09-23T09:10:00.000Z 2001-09-23T10:50:00.000Z SPIN GSM
RPC_visu_curl_div_4sat
```


18.11 Tests of all CLUSTER scripts

First, you have to position yourself in the directory `test/test_CLUSTER`. Then, to erase the old tests, we can run the command:

```
clean_all_tests.sh
```

Only the following launch shells will remain:

```
test_do_VT_plot_CLUXXX.sh
test_do_SP_plot_CLUXXX.sh
test_do_SP_plot_CLUXXX_4Bz.sh
test_do_SP_plot_CLUSTA_from_VTL1.sh
test_do_2D_plot_CLUPOS.sh
test_do_3D_plot_CLUPOS.sh
test_do_Polar_plot_CLUSTA.sh
test_do_compare_STA_FGM_CLUSTER.sh
test_CLUGEOM.sh
test_curl_div_CLUFGM.sh
```

as well as the shells which launches all the tests:

```
run_test_demo.sh
run_test_demo_with_log.sh
```

We recommend to use the last one, which create a log file. You can read this log after the end of all test, rather than watch the screen during execution. This command perform all the tests, and therefore take a little time. Each test will create its own directory of results. The results will not be displayed on the screen, they will be redirected to a file in each of the test directories. Only the *stderr* will be displayed on the screen, with a copy in the log file, in the form given below, which allows you to verify that all the tests have been executed correctly.

This shell launch a series of scripts, to check its validity. Time execution is shortness that the previous one, and can be used as a demo of what the scripts can do. List of the scripts launched by this "super-script" is given above.

```
=====
Tests of CLUSTER script do_VT_plot_CLUXXX.sh
=====
Compute and plot STA or FGM waveforms
-----
do_VT_plot_CLUSTA.sh 20010923 092000 093000 2 NBR ISR2
-----
RPC_get_data_CLUSTA_VTL2          : NORMAL TERMINATION - time exe= 0 s.
STOP visu_vectime.exe             : NORMAL TERMINATION
RPC_visu_vectime                   : NORMAL TERMINATION - time exe= 3 s.
-----
do_VT_plot_CLUSTA.sh 20010923 092000 093000 2 NBR GSE
-----
RPC_get_data_CLUSTA_VTL2          : NORMAL TERMINATION - time exe= 0 s.
STOP visu_vectime.exe             : NORMAL TERMINATION
RPC_visu_vectime                   : NORMAL TERMINATION - time exe= 3 s.
-----
do_VT_plot_CLUSTA.sh 20040130 164710 164730 2 HBR GSE
-----
RPC_get_data_CLUSTA_VTL2          : NORMAL TERMINATION - time exe= 0 s.
STOP visu_vectime.exe             : NORMAL TERMINATION
RPC_visu_vectime                   : NORMAL TERMINATION - time exe= 1 s.
-----
do_VT_plot_CLUFGM.sh 20010923 092000 093000 2 FULL
-----
```

```

RPC_get_data_CLUFGM          : NORMAL TERMINATION - time exe= 0 s.
STOP visu_vectime.exe        : NORMAL TERMINATION
RPC_visu_vectime             : NORMAL TERMINATION - time exe= 2 s.
-----
do_VT_plot_CLUFGM.sh 20010923 092000 093000 2 5VPS
-----
RPC_get_data_CLUFGM          : NORMAL TERMINATION - time exe= 0 s.
STOP visu_vectime.exe        : NORMAL TERMINATION
RPC_visu_vectime             : NORMAL TERMINATION - time exe= 0 s.
-----
do_VT_plot_CLUFGM.sh 20040130 164700 164800 2 5VPS
-----
RPC_get_data_CLUFGM          : NORMAL TERMINATION - time exe= 0 s.
STOP visu_vectime.exe        : NORMAL TERMINATION
RPC_visu_vectime             : NORMAL TERMINATION - time exe= 0 s.
=====
Tests of CLUSTER script do_2D_plot_CLUPOS.sh
=====
-----
do_2D_plot_CLUPOS.sh 20010923 0 0
-----
RPC_get_data_CLUPOS_4sat     : NORMAL TERMINATION - time exe= 0 s.
STOP visu_vectime_4sat.exe   : NORMAL TERMINATION
RPC_visu_vectime_4sat        : NORMAL TERMINATION - time exe= 0 s.
-----
do_2D_plot_CLUPOS.sh 20010923 090000 100000
-----
RPC_get_data_CLUPOS_4sat     : NORMAL TERMINATION - time exe= 0 s.
STOP visu_vectime_4sat.exe   : NORMAL TERMINATION
RPC_visu_vectime_4sat        : NORMAL TERMINATION - time exe= 0 s.
=====
Tests of CLUSTER script do_3D_plot_CLUPOS.sh
=====
-----
do_3D_plot_CLUPOS.sh 20010923 0 0
-----
RPC_get_data_CLUPOS_4sat     : NORMAL TERMINATION - time exe= 0 s.
STOP visu_vectime_3D_4sat.exe : NORMAL TERMINATION
RPC_visu_vectime_3D_4sat     : NORMAL TERMINATION - time exe= 0 s.
-----
do_3D_plot_CLUPOS.sh 20010923 090000 100000
-----
RPC_get_data_CLUPOS_4sat     : NORMAL TERMINATION - time exe= 0 s.
STOP visu_vectime_3D_4sat.exe : NORMAL TERMINATION
RPC_visu_vectime_3D_4sat     : NORMAL TERMINATION - time exe= 0 s.
=====
Tests of CLUSTER script do_SP_plot_CLUXXX.sh
=====
Compute and plot STA or FGM spectrograms
-----
do_SP_plot_CLUSTA.sh 20010923 090000 103000 2 NBR ISR2 512 0. 8.0 t XY 1. -6. 0.
-----
RPC_get_data_CLUSTA_VTL2     : NORMAL TERMINATION - time exe= 0 s.
STOP vectime_to_spectro.exe   : NORMAL TERMINATION
RPC_vectime_to_spectro        : NORMAL TERMINATION - time exe= 3 s.
STOP visu_spectro.exe        : NORMAL TERMINATION
RPC_visu_spectro              : NORMAL TERMINATION - time exe= 0 s.
STOP visu_ave_spectrum.exe    : NORMAL TERMINATION
RPC_visu_ave_spectrum         : NORMAL TERMINATION - time exe= 0 s.
-----
do_SP_plot_CLUSTA.sh 20010923 090000 103000 2 NBR GSE 512 0. 2.5 t XY 1. -6. 0.
-----
RPC_get_data_CLUSTA_VTL2     : NORMAL TERMINATION - time exe= 0 s.
STOP vectime_to_spectro.exe   : NORMAL TERMINATION
RPC_vectime_to_spectro        : NORMAL TERMINATION - time exe= 2 s.

```

```

STOP visu_spectro.exe           : NORMAL TERMINATION
RPC_visu_spectro                : NORMAL TERMINATION - time exe= 0 s.
STOP visu_ave_spectrum.exe      : NORMAL TERMINATION
RPC_visu_ave_spectrum           : NORMAL TERMINATION - time exe= 0 s.
=====
do_SP_plot_CLUSTA.sh 20040130 164700 164800 2 HBR GSE 128 0. 225. t XY 50. -10. -3.
=====
RPC_get_data_CLUSTA_VTL2       : NORMAL TERMINATION - time exe= 0 s.
STOP vectime_to_spectro.exe    : NORMAL TERMINATION
RPC_vectime_to_spectro         : NORMAL TERMINATION - time exe= 0 s.
STOP visu_spectro.exe          : NORMAL TERMINATION
RPC_visu_spectro               : NORMAL TERMINATION - time exe= 1 s.
STOP visu_ave_spectrum.exe     : NORMAL TERMINATION
RPC_visu_ave_spectrum          : NORMAL TERMINATION - time exe= 0 s.
=====
do_SP_plot_CLUFGM.sh 20010923 090000 103000 2 FULL 512 0. 8.0 n XY 0.5 -4.0 1.
=====
RPC_get_data_CLUFGM           : NORMAL TERMINATION - time exe= 0 s.
STOP vectime_to_spectro.exe    : NORMAL TERMINATION
RPC_vectime_to_spectro         : NORMAL TERMINATION - time exe= 2 s.
STOP visu_spectro.exe          : NORMAL TERMINATION
RPC_visu_spectro               : NORMAL TERMINATION - time exe= 1 s.
STOP visu_ave_spectrum.exe     : NORMAL TERMINATION
RPC_visu_ave_spectrum          : NORMAL TERMINATION - time exe= 0 s.
=====
do_SP_plot_CLUFGM.sh 20010923 090000 103000 2 5VPS 256 0. 2.5 n XY 0.5 -4.0 1.
=====
RPC_get_data_CLUFGM           : NORMAL TERMINATION - time exe= 0 s.
STOP vectime_to_spectro.exe    : NORMAL TERMINATION
RPC_vectime_to_spectro         : NORMAL TERMINATION - time exe= 1 s.
STOP visu_spectro.exe          : NORMAL TERMINATION
RPC_visu_spectro               : NORMAL TERMINATION - time exe= 0 s.
STOP visu_ave_spectrum.exe     : NORMAL TERMINATION
RPC_visu_ave_spectrum          : NORMAL TERMINATION - time exe= 0 s.
=====
Tests of CLUSTER script do_SP_plot_CLUSTA_from_VTL1.sh
=====
Compute and plot STA spectrograms from VTL1 data
=====
do_SP_plot_CLUSTA_from_VTL1.sh 20010923 090000 103000 2 NBR 512 0. 8.0 t XY 1. -6. 1.
=====
RPC_get_data_CLUSTA_VTL1      : NORMAL TERMINATION - time exe= 0 s.
STOP vectime_L1_to_spectro_L2_CLUSTA : NORMAL TERMINATION
RPC_vectime_L1_to_spectro_L2_CLUSTA : NORMAL TERMINATION - time exe= 2 s.
STOP visu_spectro.exe          : NORMAL TERMINATION
RPC_visu_spectro               : NORMAL TERMINATION - time exe= 0 s.
STOP visu_ave_spectrum.exe     : NORMAL TERMINATION
RPC_visu_ave_spectrum          : NORMAL TERMINATION - time exe= 0 s.
=====
do_SP_plot_CLUSTA_from_VTL1.sh 20040130 164700 164800 2 HBR 128 0. 225. t XY 1. -10. -3.
=====
RPC_get_data_CLUSTA_VTL1      : NORMAL TERMINATION - time exe= 0 s.
STOP vectime_L1_to_spectro_L2_CLUSTA : NORMAL TERMINATION
RPC_vectime_L1_to_spectro_L2_CLUSTA : NORMAL TERMINATION - time exe= 0 s.
STOP visu_spectro.exe          : NORMAL TERMINATION
RPC_visu_spectro               : NORMAL TERMINATION - time exe= 0 s.
STOP visu_ave_spectrum.exe     : NORMAL TERMINATION
RPC_visu_ave_spectrum          : NORMAL TERMINATION - time exe= 1 s.
=====
Tests of CLUSTER script do_SP_plot_CLUXXX.sh
=====
Compute and plot STA or FGM spectrograms
=====
do_SP_plot_CLUSTA_4Bz.sh 20010923 090000 103000 NBR GSE 512 0. 8.0 t -6. 1. 0.5 8.
=====

```

```

RPC_get_data_CLUSTA_VTL2      : NORMAL TERMINATION - time exe= 0 s.
RPC_get_data_CLUSTA_VTL2      : NORMAL TERMINATION - time exe= 0 s.
RPC_get_data_CLUSTA_VTL2      : NORMAL TERMINATION - time exe= 0 s.
RPC_get_data_CLUSTA_VTL2      : NORMAL TERMINATION - time exe= 0 s.
STOP vectime_to_spectro.exe    : NORMAL TERMINATION
RPC_vectime_to_spectro         : NORMAL TERMINATION - time exe= 3 s.
STOP vectime_to_spectro.exe    : NORMAL TERMINATION
RPC_vectime_to_spectro         : NORMAL TERMINATION - time exe= 2 s.
STOP vectime_to_spectro.exe    : NORMAL TERMINATION
RPC_vectime_to_spectro         : NORMAL TERMINATION - time exe= 3 s.
STOP vectime_to_spectro.exe    : NORMAL TERMINATION
RPC_vectime_to_spectro         : NORMAL TERMINATION - time exe= 2 s.
STOP visu_spectro_4Bz.exe      : NORMAL TERMINATION
RPC_visu_spectro_4Bz          : NORMAL TERMINATION - time exe= 1 s.
=====
do_SP_plot_CLUFGM_4Bz.sh 20010923 090000 103000 FULL 512 0. 8.0 t -4. 1. 0.02 8.0
=====
RPC_get_data_CLUFGM           : NORMAL TERMINATION - time exe= 0 s.
RPC_get_data_CLUFGM           : NORMAL TERMINATION - time exe= 0 s.
RPC_get_data_CLUFGM           : NORMAL TERMINATION - time exe= 0 s.
RPC_get_data_CLUFGM           : NORMAL TERMINATION - time exe= 0 s.
STOP vectime_to_spectro.exe    : NORMAL TERMINATION
RPC_vectime_to_spectro         : NORMAL TERMINATION - time exe= 3 s.
STOP vectime_to_spectro.exe    : NORMAL TERMINATION
RPC_vectime_to_spectro         : NORMAL TERMINATION - time exe= 2 s.
STOP vectime_to_spectro.exe    : NORMAL TERMINATION
RPC_vectime_to_spectro         : NORMAL TERMINATION - time exe= 2 s.
STOP vectime_to_spectro.exe    : NORMAL TERMINATION
RPC_vectime_to_spectro         : NORMAL TERMINATION - time exe= 2 s.
STOP visu_spectro_4Bz.exe      : NORMAL TERMINATION
RPC_visu_spectro_4Bz          : NORMAL TERMINATION - time exe= 1 s.
=====
do_SP_plot_CLUFGM_4Bz.sh 20010923 090000 103000 5VPS 256 0. 2.5 t -4. 1. 0.02 2.5
=====
RPC_get_data_CLUFGM           : NORMAL TERMINATION - time exe= 0 s.
RPC_get_data_CLUFGM           : NORMAL TERMINATION - time exe= 0 s.
RPC_get_data_CLUFGM           : NORMAL TERMINATION - time exe= 0 s.
RPC_get_data_CLUFGM           : NORMAL TERMINATION - time exe= 0 s.
STOP vectime_to_spectro.exe    : NORMAL TERMINATION
RPC_vectime_to_spectro         : NORMAL TERMINATION - time exe= 1 s.
STOP vectime_to_spectro.exe    : NORMAL TERMINATION
RPC_vectime_to_spectro         : NORMAL TERMINATION - time exe= 0 s.
STOP vectime_to_spectro.exe    : NORMAL TERMINATION
RPC_vectime_to_spectro         : NORMAL TERMINATION - time exe= 1 s.
STOP vectime_to_spectro.exe    : NORMAL TERMINATION
RPC_vectime_to_spectro         : NORMAL TERMINATION - time exe= 0 s.
STOP visu_spectro_4Bz.exe      : NORMAL TERMINATION
RPC_visu_spectro_4Bz          : NORMAL TERMINATION - time exe= 1 s.
=====
Tests of CLUSTER script do_Polar_plot_CLUSTA.sh
=====
Compute and plot CLUSTA polarization
=====
do_Polar_plot_CLUSTA.sh 20010923 090000 103000 2 NBR GSE 512 2. 8.0 t XY 1. -5. -6. 0.
=====
RPC_get_data_CLUSTA_VTL2      : NORMAL TERMINATION - time exe= 0 s.
RPC_get_data_CLUFGM           : NORMAL TERMINATION - time exe= 0 s.
STOP vectime_to_mfa_CLUSTA.exe : NORMAL TERMINATION
RPC_vectime_to_mfa_CLUSTA      : NORMAL TERMINATION - time exe= 6 s.
STOP vectime_to_spectro.exe    : NORMAL TERMINATION
RPC_vectime_to_spectro         : NORMAL TERMINATION - time exe= 3 s.
STOP visu_spectro.exe          : NORMAL TERMINATION
RPC_visu_spectro               : NORMAL TERMINATION - time exe= 0 s.
STOP spectro_to_polar.exe       : NORMAL TERMINATION
RPC_spectro_to_polar           : NORMAL TERMINATION - time exe= 2 s.

```

→ *Toc*

```

RPC_change_coordinate_system      : NORMAL TERMINATION - time exe= 0 s.
STOP_change_coordinate_system.exe : NORMAL TERMINATION
RPC_change_coordinate_system      : NORMAL TERMINATION - time exe= 0 s.
STOP_change_coordinate_system.exe : NORMAL TERMINATION
RPC_change_coordinate_system      : NORMAL TERMINATION - time exe= 0 s.
STOP_compute_curl_div_4sat.exe    : NORMAL TERMINATION
do_compute_curl_div_CLUFGM.sh     : NORMAL TERMINATION - time exe= 2 s.

```

```

-----
RPC_visu_curl_div_4sat 0 0
STOP_visu_curl_div_4sat.exe       : NORMAL TERMINATION
RPC_visu_curl_div_4sat           : NORMAL TERMINATION - time exe= 1 s.

```

```

=====
Tests of CLUSTER script do_compare_STA_FGM_CLUSTER.sh
=====

```

```

compare CLUSTER/STAFF +Despin data with MAG

```

```

-----
do_compare_STA_FGM_CLUSTER.sh 2 20010923 092000 093000 NBR

```

```

-----
RPC_get_data_CLUSTA_VTL2          : NORMAL TERMINATION - time exe= 0 s.
RPC_get_data_CLUFGM              : NORMAL TERMINATION - time exe= 0 s.
STOP_vectime_gse_to_isr2.exe      : NORMAL TERMINATION
RPC_vectime_gse_to_isr2           : NORMAL TERMINATION - time exe= 1 s.
STOP_add_DxDy_to_Bx_By.exe        : NORMAL TERMINATION
RPC_add_DxDy_to_BxBy             : NORMAL TERMINATION - time exe= 4 s.
STOP_visu_2_vectime.exe           : NORMAL TERMINATION
RPC_visu_2_vectime                : NORMAL TERMINATION - time exe= 3 s.

```

```

-----
do_compare_STA_FGM_CLUSTER.sh 3 20010110 000000 020000 NBR

```

```

-----
RPC_get_data_CLUSTA_VTL2          : NORMAL TERMINATION - time exe= 0 s.
RPC_get_data_CLUFGM              : NORMAL TERMINATION - time exe= 0 s.
STOP_vectime_gse_to_isr2.exe      : NORMAL TERMINATION
RPC_vectime_gse_to_isr2           : NORMAL TERMINATION - time exe= 1 s.
STOP_add_DxDy_to_Bx_By.exe        : NORMAL TERMINATION
RPC_add_DxDy_to_BxBy             : NORMAL TERMINATION - time exe= 3 s.
STOP_visu_2_vectime.exe           : NORMAL TERMINATION
RPC_visu_2_vectime                : NORMAL TERMINATION - time exe= 9 s.

```

```

-----
do_compare_STA_FGM_CLUSTER.sh 2 20040130 164700 164800 HBR

```

```

-----
RPC_get_data_CLUSTA_VTL2          : NORMAL TERMINATION - time exe= 0 s.
RPC_get_data_CLUFGM              : NORMAL TERMINATION - time exe= 0 s.
STOP_vectime_gse_to_isr2.exe      : NORMAL TERMINATION
RPC_vectime_gse_to_isr2           : NORMAL TERMINATION - time exe= 0 s.
STOP_add_DxDy_to_Bx_By.exe        : NORMAL TERMINATION
RPC_add_DxDy_to_BxBy             : NORMAL TERMINATION - time exe= 1 s.
STOP_visu_2_vectime.exe           : NORMAL TERMINATION
RPC_visu_2_vectime                : NORMAL TERMINATION - time exe= 1 s.

```

```

Starting time : 2021-03-06 01:58:57
Ending time   : 2021-03-06 02:01:01
Duration      : 124 sec. (2.06 mn.)

```

Part VI

ANNEXES

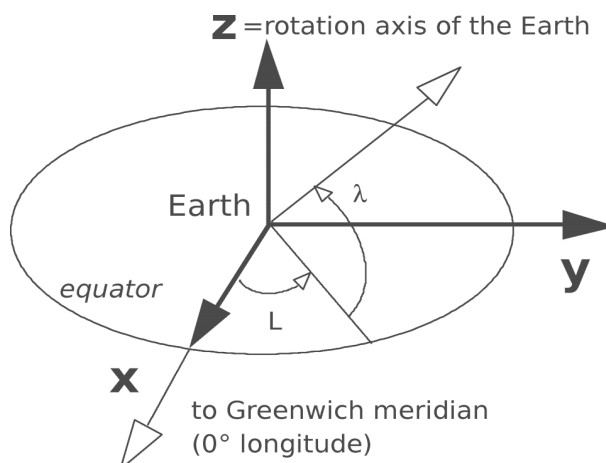
Chapter 19

DEFINITION OF USED COORDINATE SYSTEMS

Definition of all coordinate systems can be found in [4], as well as the matrix transformation. One recall here the definition of the system used by the RPC command.

19.1 The Geographic System (GEO)

The Geographic System (GEO) is used for GEOS POS data. The diagram below recalls its definition.



The Z-axis is parallel to the rotation axis of the Earth.

The X and Y axes are included in the equator plane.

The X axis is pointing from the centre of the Earth to the Greenwich meridian (0° longitude).

The GEO system is fixed with the rotating Earth.

One can define the *Longitude* L and the *Latitude* λ as:

$$\text{Longitude } L = \tan^{-1}(V_Y/V_X)$$

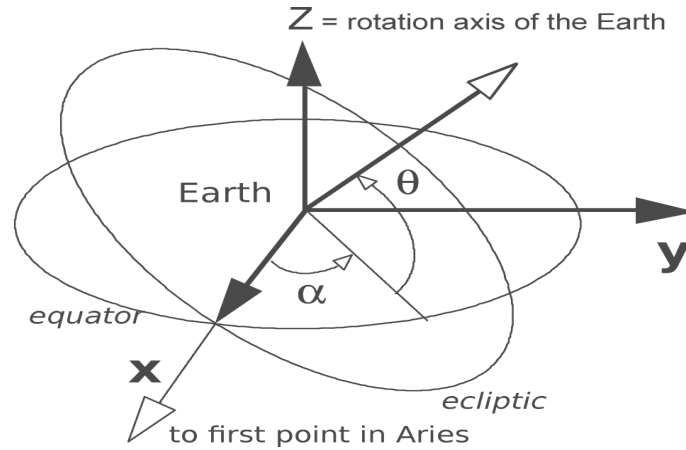
$$\text{with } \begin{array}{ll} L \text{ in } [0^\circ, 180^\circ] & \text{for } V_Y > 0 \\ L \text{ in } [180^\circ, 360^\circ] & \text{for } V_Y < 0 \end{array}$$

$$\text{Latitude } \lambda = \sin^{-1}(V_Z/V)$$

$$\text{with } \lambda \text{ in } [-90^\circ, +90^\circ]$$

19.2 The Geocentric Equatorial Inertial system (GEI)

The Geocentric Equatorial Inertial system (GEI) is a fixed system frequently used to plot the spacecraft trajectory. The diagram below recall its definition.



The Z-axis is parallel to the rotation axis of the Earth.

The X-axis is defined by the intersection of the equator plane and the ecliptic plane, and is pointing towards the first point of Aries (Sun position at the vernal equinox).

One can define the *right ascension* α and the *declination* θ as:

$$\text{right ascension } \alpha = \tan^{-1}(V_Y/V_X)$$

$$\text{with } \alpha \text{ in } [0^\circ, 180^\circ] \text{ for } V_Y > 0$$

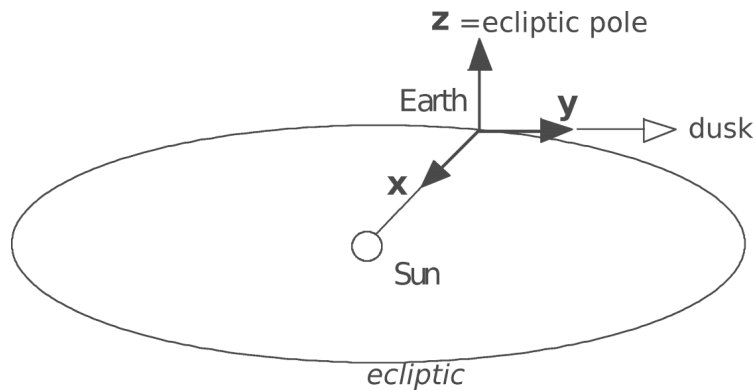
$$\alpha \text{ in } [180^\circ, 360^\circ] \text{ for } V_Y < 0$$

$$\text{declination } \theta = \sin^{-1}(V_Z/V)$$

$$\text{with } \theta \text{ in } [-90^\circ, +90^\circ]$$

19.3 The Geocentric Solar Ecliptic system (GSE)

The Geocentric Solar Ecliptic system (GSE) is used to plot data, very usefull thank to the x axis toward the Sun. CLUSTER position and CLUSTER/FGM data are provided in this system. The diagram below recall its definition.



The X-axis is pointing from the Earth towards the Sun.

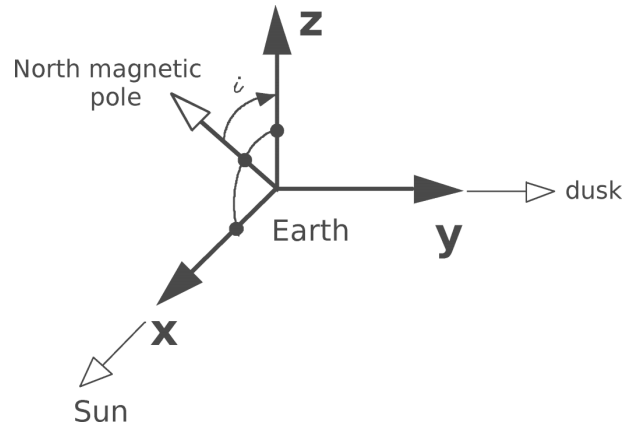
The X-axis and the Y-axis are include in the ecliptic plane.

The Y-axis is pointing toward the dusk, opposing to the planetary motion.

The Z-axis is parallel to the ecliptic pole. The GSE system has a yearly rotation with respect to the inertial system.

19.4 The Geocentric Solar Magnetospheric system (GSM)

The Geocentric Solar Magnetospheric system (GSM) is used to plot data related to the direction of the Sun and direction of the magnetic field. Useful to plot the estimated current density. The diagram below recall its definition.



The X-axis is pointing from the Earth towards the Sun.

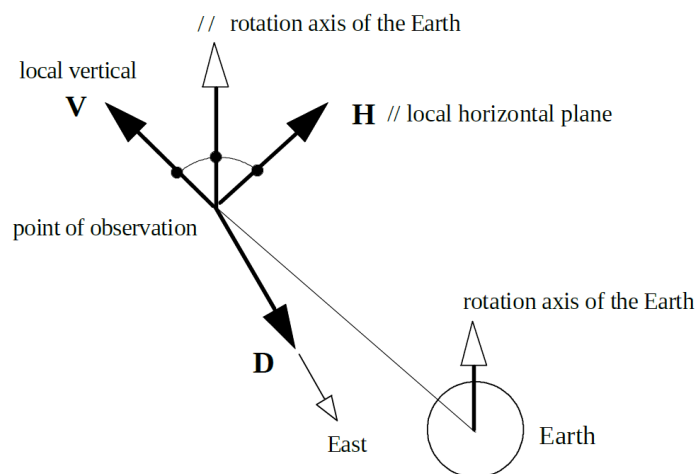
The X-Z plane contains the dipole axis.

The Y-axis is perpendicular to the Earth's magnetic dipole, towards the dusk and include in the magnetic equator plane.

The positive Z-axis is chosen to be in the same sense as the northern magnetic pole; the dipole tilt angle i is positive when the north magnetic pole is tilted towards the Sun. In addition to a yearly period due to the motion of the Earth about the Sun, the GSM system rocks about the Solar direction with a 24 h period.

19.5 The Vertical Dusk Horizontal system (VDH)

The Vertical-Dusk-Horizontal system is used by GEOS data. GEOS magnetometer data are in the VDH system. The diagram below recall its definition.



19.6 The Sensor Coordinate System (SCS)

This is the system where the original signal is measured (see Figure 19.1 below). This system could be a non perfect orthogonal system.

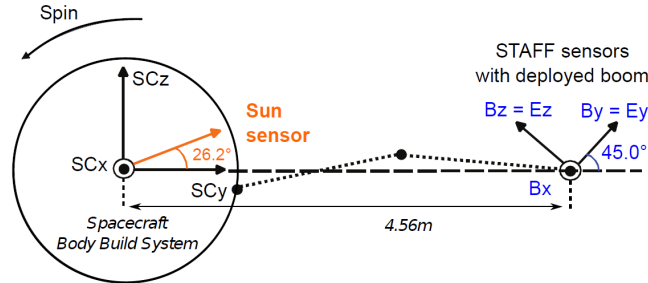


FIG. 19.1: Correspondance between Body Build System and STAFF Sensors System.

19.7 The Orthogonal Sensor System (OSS)

This is a Cartesian orthogonal coordinate system. The original sensor system can be a non orthogonal system, the first step is to transform the data vector in an orthogonal coordinate system: Z axis being the reference of the new Orthogonal Sensor System. The corresponding matrix, called “SCS_to_OSS”, close to a unit matrix, is required and must be applied: values are supposed to be constant in time. Nevertheless, in a first time, taking into account the low deviation of the sensor to an orthogonal system for CLUSTER/STAFF (0.2°), this correction is not applied and the matrix is set to unity matrix.

$$SCS \sim \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} OSS$$

If the user wants to do this correction, he can use the formulas given in section 21.3 which allows the transformation from a non orthogonal system to an orthogonal one.

19.8 The Data Sensor System (DSS)

The Body Build System (BBS, see next section) is a system fixed to the geometry of the spacecraft, and is used as the spacecraft system reference for all the experiments. Generally, for most of spacecraft missions, the Z axis is close to the maximum principal inertia axis also called the spin axis (for spin stabilized spacecraft). Nevertheless, for CLUSTER, this axis has been defined as the X axis (see Figure 19.1).

In all our data, the convention taken is Z=spin axis. It means that we have an intermediate coordinate system, called Data Sensor System (DSS) which corresponds to the previous OSS, but where the axes are permuted, to make Z close to the spin axis. By respect to the Figure 19.1, becomes Y, Z, X in DSS. This permutation is obtained by the following matrix:

$$OSS = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} DSS$$

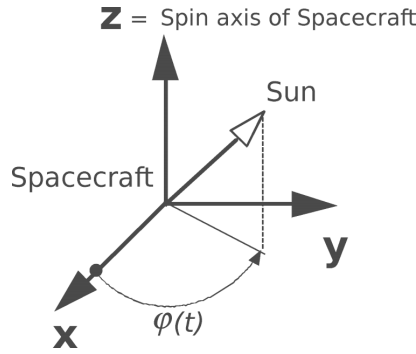
19.9 The Body Build System (BBS)

In the case of CLUSTER, the Z axis of the Data Sensor System is close to the X axis of the BBS system, but the misalignment angle is not easy to determine. It is also true for the small angle between this X_{BBS} and the true spin axis (precession and nutation motions). Nevertheless, an estimate of the cumulative angle is done in next subsection. Here, we neglect this small misalignment and assume $Z_{DDS} = X_{BBS}$. In all cases, 2 other axis may be rotated by an important angle (see Figure 19.1). The corresponding matrix is required, called "DSS_to_BBS": values are supposed to be constant. Practically, for the STAFF search coils of CLUSTER, this matrix is a rotation matrix of $\alpha = 45^\circ$.

$$DSS = \begin{pmatrix} 0 & 0 & 1 \\ \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \end{pmatrix} BBS$$

19.10 The Spin Reference System (SRS or SR)

The Spin Reference system is a spinning local system close to the measurement antenna of a spacecraft. It is used to calibrate the search-coils data before transform it in the fixed SR2 system.



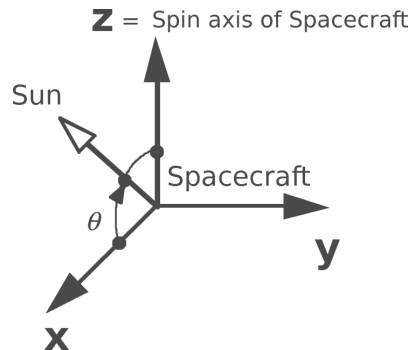
The Z-axis is the spin axis of the spacecraft.

The X-axis and Y-axis are perpendicular to the spin axis, and rotate at the spin frequency of the spacecraft.

The definition of the SR system need the knowledge of the spin axis in a fixed frame of reference as the GEI inertial system, and the value of the spin phase φ at a given time.

19.11 The spin reference2 system (SR2)

The Spin Reference 2 is a fixed system usefull for the spacecraft data processing. It is also called SCS, as "Spacecraft-Sun system", or DS system (Despun Satellite).

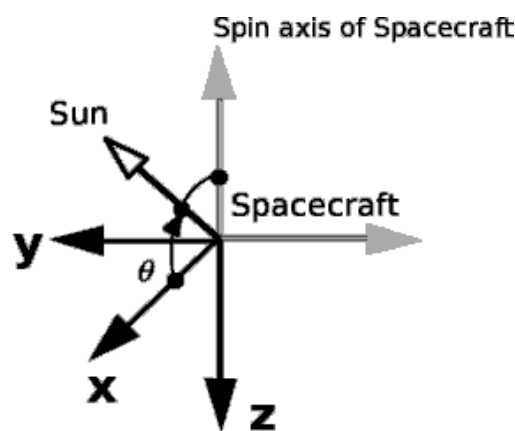


The Z-axis is the spin axis of the spacecraft.
 The X-Z plane contains the direction of the Sun.
 The X-axis is towards the day side.
 The Y-axis is perpendicular to the spacecraft-Sun line.

The SR2 system rotates with the same period than the orbital period of the spacecraft with respect to the inertial system, while the declination θ varies continuously.

19.12 The Inverse SR2 system (ISR2)

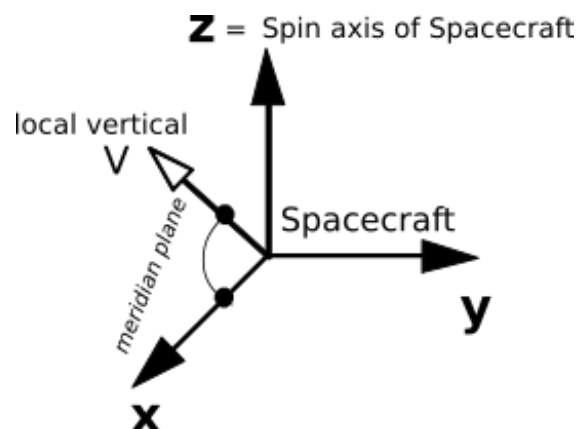
The Inverse-Spin-Reference-2 system is used for The CLUSTER/STAFF-SC calibrated waveform. The diagram below recall its definition.



This system is equivalent to the SR2 system where the Z and Y axis has inverse sign. This system is useful for CLUSTER, where the Z axis of ISR2 system is close to the Z axis of the GSE system, so ISR2 is a rather good approximation of the GSE system, and does not requires knowledge of spin direction in GSE system.

19.13 The Spin Reference Vertical system (SRV)

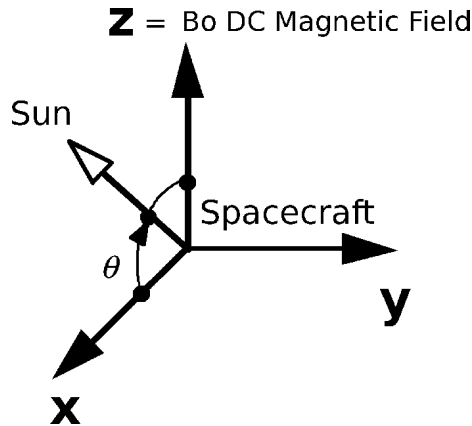
The Spin-Reference-Vertical system is used for GEOS / ULF calibrated data. The diagram below recall its definition.



In the SRV frame the Z axis is aligned with the spin axis, while X is in the plane of the geographic meridian (H, V) of the VDH frame

19.14 The Magnetic Field Aligned system (MFA)

The Magnetic Field Aligned system [5] was introduced for studying the polarization of waves. Indeed, most plane waves are characterized by their direction of rotation around the magnetic field, and by the angle between the normal to the wave plane and the main field. It is used for the analysis of CLUSTER and GEOS data, or any other wave experiment. The diagram below recall its definition.



The Z axis is along the continuous magnetic field.

The X-Z plane contains the direction of the Sun.

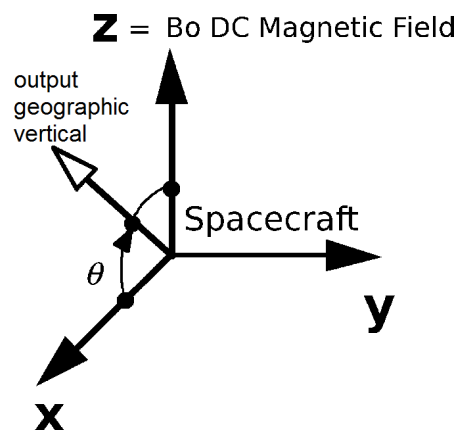
The X axis is towards the day side.

The Y axis is perpendicular to the satellite-Sun line.

It is a useful system for physics, but the B_0 DC magnetic field must be known, as its temporal variation. The MFA system moves continuously with the temporal variation of the continuous magnetic field.

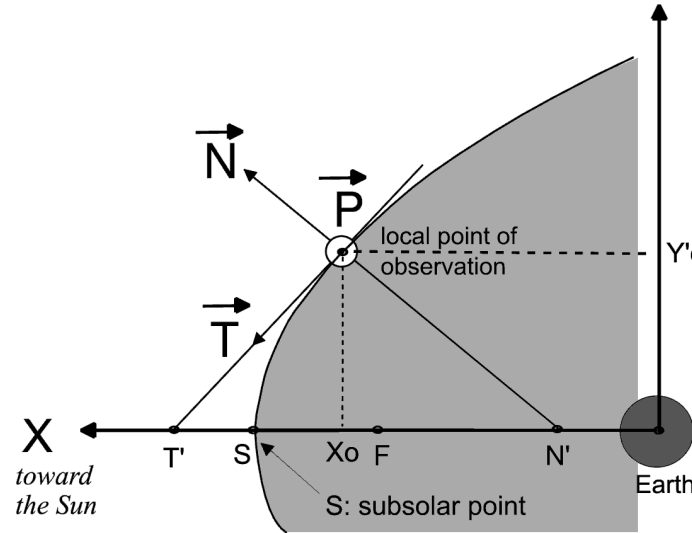
19.15 The Magnetic Field Aligned Vertical system (MFAV)

The Magnetic Field Aligned Vertical system was introduced for the polarization analysis of GEOS ULF waves. It is similar to MFA, except that the XZ plane contains the outgoing geographic vertical instead of the direction of the Sun. The diagram below recall its definition.

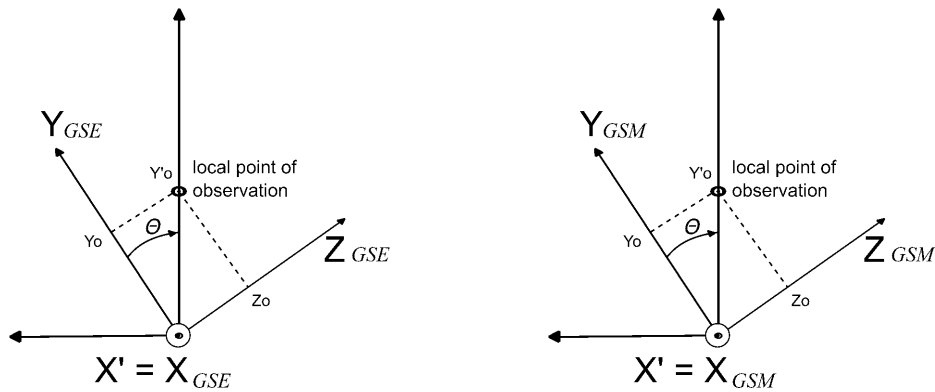


19.16 The Tangente Paraboloid Normal system (TPN)

The Tangente-paraboloid-Normal system has been introduced by the author in 2005, for magnetopause studies. Transformation is available in Rocotlib library [4].



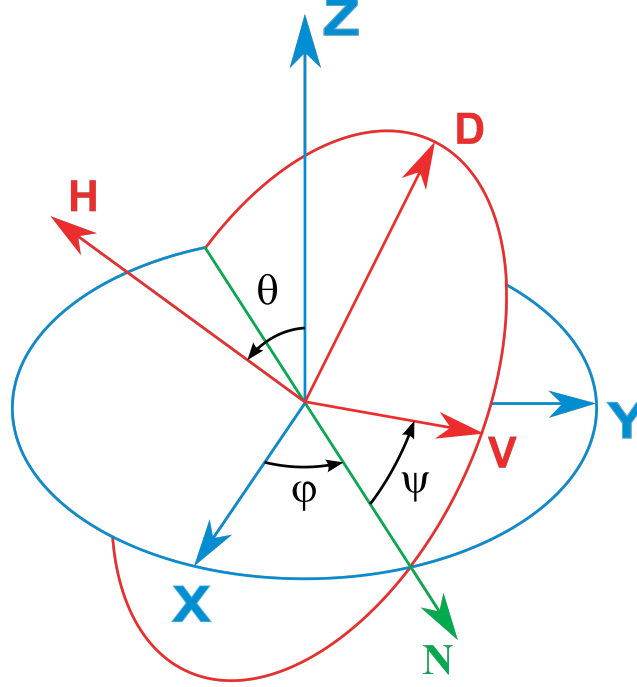
When the spacecraft cross the magnetopause, this one is modeled by a paraboloid around the Sun-Earth direction. The \vec{N} vector is the output normal to the plane tangent to the paraboloid, at the local point of observation. The \vec{T} vector is the tangent in the X-N plane, toward the submit of the paraboloid, while the \vec{P} vector is the tangent perpendicular to the X-N or T-N plane, in the direct sens T, \vec{P}, N .



When magnetopause crossing data are analysed by the Minimum Variance Analysis method, the \vec{N} vector of the TPN system is often close the MVA eigen vector corresponding to the minimum eigenvalue.

19.17 The Euler's angles

Euler angles are used to calculate the passage between the GEOS rotating antenna frame and the reference VDH frame. They are defined according to the diagram below :



The VDH coordinate system is the fixed system, in which we define the system in rotation XYZ by means of the 3 Euler angles : θ is the precession, φ the nutation and ψ the proper rotation, also called gyration.

19.18 Transformation matrix VDH-SRV

This transformation is used to make the comparisons between the perpendicular components of the continuous field measured by the search-coils of GEOS and the measurements of the magnetometer. The passage from the XYZ reference to the VDH reference is done by means of 3 rotations :

- a rotation A around Z (spin axis) and angle φ brings X to the axis of the nodes N, i.e. X'
- a rotation B around N and an angle θ brings Z to H, let Z'
- a rotation C around Z' and angle ψ take X' to V

The transformation matrix from XYZ to VDH is therefore the following product $R = CBA$:

$$R = \begin{pmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} \cos \varphi & -\sin \varphi & 0 \\ \sin \varphi & \cos \varphi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

and in developed form:

$$R = \begin{pmatrix} \cos \psi \cos \varphi - \sin \psi \cos \theta \sin \varphi & -\cos \psi \sin \varphi - \sin \psi \cos \theta \cos \varphi & \sin \psi \sin \theta \\ \sin \psi \cos \varphi + \cos \psi \cos \theta \sin \varphi & -\sin \psi \sin \varphi + \cos \psi \cos \theta \cos \varphi & -\cos \psi \sin \theta \\ \sin \theta \sin \varphi & \sin \theta \cos \varphi & \cos \theta \end{pmatrix}$$

The inverse matrix is $R^{-1} = A^{-1}B^{-1}C^{-1}$ and the inverse of each matrix is calculated simply by changing the sign of the angle. But it is much easier to take the transposed matrix directly.

Chapter 20

CALIBRATION OF SPECTRA AND WAVEFORMS

20.1 Introduction

The calibration of data from a search-coil, whose transfer function is not linear in frequency, is an old problem dating from the first experiments of this type (see [9], Robert, 1971). The calibration of the spectra, if the principle seems simple (correction of the amplitudes of each line of the spectrum by the value of the transfer function at this frequency), the implementation is more delicate because the magnetic sensors are in a reference frame. rotation (spined satellite) whose frequency is included in the frequency band of the instrument. In addition, these sensors rotate in a continuous field 10 to 1000 times greater than the amplitude of the waves to be detected.

The production of calibrated spectra is done batchwise, window after window, as explained below. It leads to the production of daily files of type SPL2.rff.

Calibrating a "continuous" waveform is much more difficult, as explained in chapter 20.4 These calibrations have been validated by fine comparisons with data from the FGM magnetometer, during numerous "cross-calibration meetings" (see [11]), summarized in the "Calibration Report" of the CSA [12], Robert 2012] as well. as in the article [13], Robert et al, 2014]. We recall here in detail the calibration of spectra and waveforms.

20.2 Spectra calibration

20.2.1 Theory

In theory, the calibration of the spectra comes down to a complex FFT of a portion of a waveform (calibration window), which is divided by the complex value of the search-coil transfer function, for each frequency. , and component by component, in the rotating frame of the sensors.

In theory, it's easy. In practice, certain precautions must be taken.

In practice, the original signal $X_{k(Volt)}$ consists of a large sinusoidal signal at the spin frequency f_{spin} 0.25 Hz (or with a Tspin period 4 s), due to the rotation of the sensors in a high DC field (10 to 1000 nT), while the fluctuations which are superimposed on this "carrier" sinusoid are of the order of 10 to 1000 times weaker (a few nT).

Figure 20.1 below illustrates this principle. In addition, a Fourier transform over a period N assumes a periodic signal Or, the truncation of the sinusoid on the edges of the window introduces large discontinuities, which generate high frequency components in the spectrum, which have no physical meanings (see [9], Robert, 1971 and [12], Robert 2012).

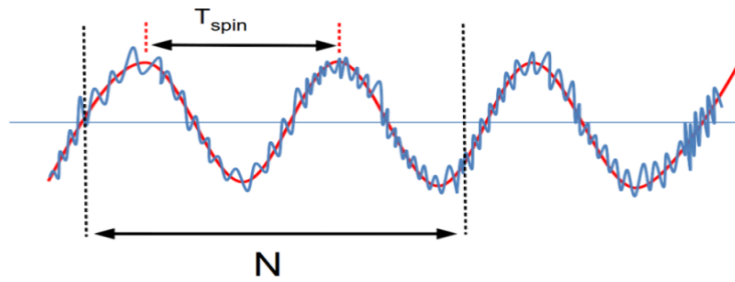


FIG. 20.1: Useful signal superimposed on the spin sinusoid for a rotating search-coil

20.2.2 The successive steps of the calibration

A minimum of four processing steps are needed to obtain a spectrum calibrated in a fixed “de-Dopplerized” frame of reference in the rotation of the satellite.

• Step 1: conversion to Volt

Telemetry data are integer counts, encoded on 16 bits and therefore in the range 0 - 65535. The dynamic being 10 V and the offset 5V (values written in the rff file) the passage of the telemetry shots in Volt is done by the rule of three:

$$X_i(tn)_V = X_i(tn)_c * dyna - offset$$

In the calibration program, this results in the following simple instructions:

```
proc_vec(1,:) = proc_vec(1,:)*dyna -offset
proc_vec(2,:) = proc_vec(2,:)*dyna -offset
proc_vec(3,:) = proc_vec(3,:)*dyna -offset
```

• Step 2 : despin

Thus, the first step consists in removing this large sinusoid with a specialized subroutine which calculates the amplitude and the phase of this sinusoid, whose frequency we must know, but which appears in VTL1. Then we removes it by routine desinus.f90 and desinus_D.f90). This step results in the code by the following instructions (N_Kern is the size of the calibration window):

```
call desinus(procvec(1,:),NKern,samrate,spinrate,modd1,phad1)
call desinus(procvec(2,:),NKern,samrate,spinrate,modd2,phad2)
call desinus(procvec(3,:),NKern,samrate,spinrate,modd3,phad3)
```

In this step, we also possibly apply the “detrend” procedure explained in section 20.3 which is translated into the code by the instructions:

```
do i=1,3
!   trend computation
  RAID_tmp= proc_vec(i,:)
  call lissage(RAID_tmp,N_Kern,nbp_liss,ierr)
  if (ierr /= 0) then
    RAID_tmp= 0.
  endif
!   subtract trend to the original signal
  proc_vec(i,:)= proc_vec(i,:) -RAID_tmp
end do
```

• **Complement to step 2: calculation of the DC field in the spin plane**

When the despin procedure is applied by the desinus routine, the amplitude in Volt and the phase of the spin sinusoid are recovered at the output for each component, that is to say \tilde{B}_x , $\tilde{B}\varphi_x$ and \tilde{B}_y , $\tilde{B}\varphi_y$ and that of the component \tilde{B}_z , $\tilde{B}\varphi_z$, which, although weak, will allow the calculation of the tilt angle (see next section). The values \tilde{B}_x , and \tilde{B}_y are two different estimates of the perpendicular field B_\perp , in uncalibrated value (in V), as well as $\tilde{B}\varphi_x$ and $\tilde{B}\varphi_y$ are two estimates of the phase φ in the rotating frame SR.

The calculation of the DC field in the spin plane, with a rotating frame SR2 as with a fixed frame SR2 leads to results recalled below.

Amplitude and phase of B_\perp in SR rotating frame:

From the two estimates \tilde{B}_x and \tilde{B}_y the amplitude (in V) we can calculate two estimates $B_{\perp x}$ and $B_{\perp y}$ of the amplitude in nT by the formulas:

$$B_{\perp x} = \tilde{B}_x \frac{1}{|\alpha_x(f_s)|} \quad B_{\perp y} = \tilde{B}_y \frac{1}{|\alpha_y(f_s)|}$$

Where $|\alpha_x(f_s)|$ and $|\alpha_y(f_s)|$ are the moduli of the transfer function at the spin frequency for each of the two components X and Y. It is natural to take as the value of B_\perp the mean value:

$$B_\perp = \frac{B_{\perp x} + B_{\perp y}}{2}$$

Similarly, two different estimates of the angle φ are obtained by correcting the phases measured by the transfer function:

$$\varphi_x = \tilde{\varphi}_x - \text{phase}(\alpha_x(f_s))$$

$$\varphi_y = \tilde{\varphi}_y - \text{phase}(\alpha_y(f_s))$$

For the phase we decide to take φ_x as a reference phase (this choice is arbitrary, we could have taken φ_y). The corresponding part of the code for estimating B_\perp the SR system is as follows:

```
!      Spin amplitude in nT, after transfert function correction

Bperp_1= mod_d1/mod_sr1
Bperp_2= mod_d2/mod_sr2
Bperp_3= mod_d3/mod_sr3

!      Bperp_1 and Bperp_2 should be equal, assuming DC field constant
!      over window duration. Phase difference should be 90. degrees.
!      To avoid modulation at twice spin frequency on DC field estimate,
!      one force theses assumptions.

Bperp_A= ( Bperp_1 +Bperp_2 )/2.

phacor_1= pha_d1 -pha_sr1
phacor_2= pha_d2 -pha_sr2
phacor_3= pha_d3 -pha_sr3
```

Tests:

In principle Bx and By should be equal, although the transfer functions are slightly different. Likewise the difference between φ_x and φ_y should be 90 °. We can therefore perform a test to see if these

assumptions are true or not. The measurement of the differences will make it possible to estimate the "misalignment angle" (see section 21.2).

The tests are just done by printing the test values:

```
!   X is taken as phase reference: arbitrary choice, one could prefer Y
diff= phacor_2 - phacor_1   ! must be > 0 and close to 90.
if (pr_out) then
  write(*,*) '   Calibration of spin signal ... '
  write(*,*) '   Spin amplitude and phase in nT and degrees : '
  write(*,*) '   B perp . measured from Bx (nT): ', Bperp_1, phacor_1
  write(*,*) '   B perp . measured from By (nT): ', Bperp_2, phacor_2
  write(*,*) '   B perp . measured from Bz (nT): ', Bperp_3, phacor_3
  write(*,*) '   '
  write(*,*) '   B perp . average (nT): ', Bperp_A
  write(*,*) '   Bxy phase difference (nT): ', diff
  write(*,*) '   Phase reference (nT): ', phacor_1
endif
```

Amplitude and phase of $B_{\perp x}$ in rotating frame SR2:

To obtain the Bx and By values of the DC field in the SR2 frame, it suffices to make an ψ angle rotation and we have:

$$\psi = \omega_s t + \varphi_s + \alpha$$

For more details, see also step 4 below. The values retained for the continuous field in the spin plane are then:

$$B_x^{DC} = B_{\perp} \sin(\varphi_x + \psi)$$

$$B_y^{DC} = B_{\perp} \sin(\varphi_y + \psi)$$

By this method, we have preserved the fact that the amplitude is the same on the two components, and that they are out of phase by 90°. And a little further the calculation of the components BDCx and BDCy in SR2:

```
!   Computation of BX,By DC in the spin plane of SR2
!   (used both by L5 and L6)

if (spin_phase >= -360.) then
  phicr= spin_phase -depif*float(ii-1)/pisd +45.
  Bdc_x= Bperp_A*sin((phacor_1 -phicr)*pisd)
  Bdc_y= Bperp_A*cos((phacor_1 -phicr)*pisd)
  else
  Bdc_x= Bperp_A
  Bdc_y= 0.
endif
```

• Another complement to step 2: calculation of the misalignment angle

The "misalignment angle" is the (small) angle between the true axis of rotation of the satellite and the Z antenna of the search-coils. The complete calculation is given in chapter 9.4. Its value is deduced from the amplitudes and phase of the spin sinusoid calculated on the 3 components.

We have:

$$\theta_z = \sin^{-1} \left(\frac{a_z \sqrt{2}}{a_x^2 + a_y^2 + a_z^2} \right)$$

This calculation is always done in step 2, using the following lines of codes:

```

!      Computation of misalignment angle (done at this step 2 only)

R1_tmp= sqrt(Bperp_1**2 +Bperp_2**2 +Bperp_3**2)

if (abs(R1_tmp).gt.1.e-10) then
  R2_tmp= Bperp_3*sqrt(2.)/R1_tmp
  if (abs(R2_tmp).le.1.) then
    misal_angle=asin(R2_tmp)*180./acos(-1.)
  else
    misal_angle= 999.99
  endif
else
  misal_angle=0.
endif

```

•Step 3: calibration in spinning system

The second step is to first apply a weighting function to the window, before the FFT, in order to avoid residual discontinuities at the edges. This is done after centering the signal on zero of course.

The weighting function should maintain the general shape of the signal but also make the edges zero. From experience, choosing a very long trapeze works well. The edges of the trapezoid can also be slightly rounded as shown in Figure 13 below. It is possible to apply other weighting windows, as will be seen in section 20.2.3.



FIG. 20.2: *trapezoidal apodization of the signal for the calibration of a spectrum*

Then, we can see that an estimate of the calibrated spectrum (in nT) is obtained by:

$$\tilde{X}_n = \frac{1}{\alpha_n} \frac{1}{N} \sum_{-N/2}^{N/2} X_k W_k e^{-2i\pi n k}$$

Where α_n is the antenna transfer function, set over the entire frequency range. In the programming of this formula, one generally uses a library FFT, or that written in the RPC software. Since the signal is real, its spectrum is symmetrical, that is, the negative frequency part of the spectrum is the conjugate of the positive frequency part. If we do not intend to do an inverse FFT then, we can be satisfied with the positive frequencies to apply the correction of the transfer function and therefore simply take the first $N/2$ points of the complex spectrum. But in this case we must not forget a factor of 2 in the calculation of the power spectral density (PSD), which is expressed as follows:

$$DS P_n(nT^2) = \frac{2(\tilde{X}_n)^2}{\delta f}$$

On the other hand, if we then plan to do an inverse FFT, we must be careful to take all the frequencies when we apply the correction of the transfer function, and therefore to define it on the negative frequencies, from the part of conjugate positive frequencies. For more details, refer to chapter 21.4

In practice, this is always the case, because the spectrum obtained above is in the "SR" frame, therefore in the rotating frame of the antennas, which is of little interest except for testing. However, to make the change of reference to switch to "SR2", also called "Despun", it is necessary to switch to waveform.

In the spectra calibration program, the deconvolution is done as follows:

```
!   Deconvolution :
!   waveform centering(done in step 2), weighting window, FFT,
!   transfer function correction, filtering, inverse FFT
!   The input waveform must be despined before (see step 2)

do i=1,3

!       weighting window
proc_vec(i,:)=proc_vec(i,:)*Weight(:)

!       time to frequency domain
RA1D_tmp=0.
call fftpat_XY(proc_vec(i,:),RA1D_tmp,N_Kern,DSS,DCS,M_Kern,1)

!       transfer function correction

do j=1,N_Kern
    ctra=cmplx(proc_vec(i,j),RA1D_tmp(j))*TFcor(j,i)
    proc_vec(i,j)=real(ctra)
    RA1D_tmp(j)=aimag(ctra)
enddo

!       back to time domain
call fftpat_XY(proc_vec(i,:),RA1D_tmp,N_Kern,DSS,DCS,M_Kern,-1)

!       test on the residu on imaginary part
par=SUM(proc_vec(i,:)**2)
pai=SUM(RA1D_tmp(:)**2)

pispr=pai/par
if(pispr.gt.2.0e-4) write(*,'(a,i8,a,e11.3,2a)') ' -> WARN: win #', &
    i_win, ' Imag / Real = ',pispr, ' ',trim(data_index(1))

!   on charge le bloc calibre mais avec le passage a zero pour le 1er pt
R1_tmp=proc_vec(i,1)
R2_tmp=proc_vec(i,N_Kern)
proc_vec(i,:)= proc_vec(i,:) -(R1_tmp +R2_tmp)/2.

if (pr_out)
    write(*,*) ' Force 1st and last point to be close to zero '
    write(*,*) ' pts (1)= ', R1_tmp
    write(*,*) ' pts (N)= ', R2_tmp
    call casinus( proc_vec(i,:),N_Kern,sam_rate,spin_rate, &
        R1_tmp,R2_tmp,' rest of spin on '//sensor(i))
endif
enddo
```

• Step 4: switch to SR2

This step is done on the waveform calibrated by the SCS spinning system of the antennas (Sensor Coordinate System). These waveforms calibrated by sequence of N points (with apodization on the edges) are obtained from the spectrum calibrated above after an inverse FFT (on all frequencies, negative as well as positive). In section 21.3, we will see that we can neglect the depointing matrix, and confuse the SCS frame with the SRS (Spin Reference System) frame.

In this case the product of the various coordinate change matrices is reduced to a simple rotation in the spin plane of the ψ angle. As previously we have:

$$\psi = \omega_s t + \varphi_s + \alpha$$

In this formula, $\omega_s 2\pi f_s$ where f_s is the spin frequency, t is the time measured from the start of the window, i.e. $t = (n - 1)\delta t$ with $\delta t = 1/f_e$ where f_e is the sampling frequency of the satellite considered (25 rr 450 Hz depending on the NBR or HBR mode). φ_s is the "spin phase angle" read in each block of the VTL1 file. For CLUSTER/STAFF $\alpha = 45$ deg (see section 19.6 and figure 19.1).

The transformation from SR to SR2 in the spin plane is therefore:

$$\begin{aligned} X_{SR2} &= X_{SR} \cos(\psi) - Y_{SR} \sin(\psi) \\ Y_{SR2} &= Y_{SR} \sin(\psi) + X_{SR} \cos(\psi) \end{aligned}$$

In the code, this step corresponds to the following instructions:

```

if (spin_phase >= -360.) then
!   one process only the points that we will keep
!   reminder: depif=mod(pi2*spin_rate/sam_rate,pi2) = 2*PI*Fs*dT

do i= i1,i2
!   spin phase shifted of 45 d. according experiment sensor position / Sun sensor

    phicr= -(spin_phase +45.)*pisd -mod(depif*float(i-ii),pi2)
    sinphi=sin(phicr)
    cosphi=cos(phicr)

    R1_tmp= cosphi*proc_vec(1,i) +sinphi*proc_vec(2,i)
    R2_tmp= -sinphi*proc_vec(1,i) +cosphi*proc_vec(2,i)

    proc_vec(1,i)=R1_tmp
    proc_vec(2,i)=R2_tmp

!   if spin_phase is interpolated into the windows, compute
!   the value and modify status

    if (data_exind(i)(11:11) == 'N') then
        data_exind(i)(11:11)= 'R'
        phicr=sngl(mod(dble(spin_phase)+Ddepif*dble(i-ii)*180.D0/Pi, 360.D0))
        write(data_exind(i),manda_param%index_extension_format) data_exind(i),phicr
    endif
enddo

else
write(*, '(a,i8,a,a)') ' -> WARN: win #',i_win, ' nospin phase ', &
trim(data_index(i1))

proc_vec(1,:)= -0.1000E+31
proc_vec(2,:)= -0.1000E+31
endif

```

• Step 5: adding the DC field on X and Y

We saw in step 2 that we could calculate the two components Bx and By of the DC field in the spin plane in frame SR2. In this step, we therefore add these two values to the waveform values simply by the following code:

```

if (spin_phase >= -360.) then
do i= i1,i2
    proc_vec(1,i)= proc_vec(1,i) +Bdc_x
    proc_vec(2,i)= proc_vec(2,i) +Bdc_y
enddo
endif

```

• Step 6: control mode for the study of DC

For this particular mode, steps 3, 4 and 5 have been skipped. We only keep one point per window, which contains the DC field on x and y, and the angle misalignment on Z. The corresponding code portion is as follows:

```
!    only one point for each windows

    manda_param%block_number= N_win

    proc_vec(1,i1)= Bdc_x
    proc_vec(2,i1)= Bdc_y
    proc_vec(3,i1)= misal_angle
```

• Step 7: Calculation in ISR2

This is the same step as step 4, but the result is in ISR2, and no longer in SR2. The corresponding code portion is as follows:

```
!    SR2 to ISR2

    if (spin_phase >= -360.) then

!        proc_vec(1,:) no chge / S4
        proc_vec(2,:)= -proc_vec(2,:)
        proc_vec(3,:)= -proc_vec(3,:)
    endif
```

• Step 8: Calculation in GSE

This is the same step as step 4, but the waveforms are calculated in GSE. The corresponding code portion is as follows:

```
    if (spin_phase >= -360.) then
!        GSE spin axis is supposed to not change during a spectra (~seconds)
!        to reduce CPU time (one day~ 360/365, so 1 mn ~ 6 E-4 deg.)
!        Rocotlib time dependant matrix computation

        call decode_datiso(data_index(i1),iyear,imon,iday,ih,im,is,ims,imc)
        call ctimpar(iyear,imon,iday,ih,im,is)

!        spin axis in gse

        call tgeigse(sxgei,sygei,szgei,sxgse,sygse,szgse)

        if (pr_out) then
            write(*, '(1x,a,3f10.4)') 'Spin dans le gse, x y z : ',sxgse,sygse,szgse
        endif

        do i= i1,i2

!            transform data in GSE

            call tsr2gse(proc_vec(1,i),proc_vec(2,i),proc_vec(3,i),sxgse,sygse,szgse,&
                R1_tmp,R2_tmp,R3_tmp)

            proc_vec(1,i)=R1_tmp
            proc_vec(2,i)=R2_tmp
            proc_vec(3,i)=R3_tmp

        enddo

        else
            proc_vec(1,:)= -0.1000E+31
            proc_vec(2,:)= -0.1000E+31
            proc_vec(3,:)= -0.1000E+31
        endif
```

20.2.3 Calculation of the complex spectrum

Depending on the chosen calibration level, we now have a calibrated waveform (for steps ≥ 3), in the reference frame we have chosen, and over the period T corresponding to the duration of the calibration window (the N_Kern points). We must now calculate the complex spectrum.

To do this, we start by centering the signal by subtracting the average value (essential to make a correct apodization), then we multiply it by the chosen apodization function. We then fear going into the frequency domain by an improved FFT where the sine tables have been pre-calculated in order to save CPU time. The value of continuous, subtracted before the calculation of the FFT, is then added in the zero frequency. Finally, we normalize the spectrum so that energy is conserved.

This operation is done using the following portion of code:

```
!   compute spectra (amplitude and phase in degrees) for each component
do i=j1,j2

    R1_tmp=sum(proc_vec(i,:))/float(N_Kern)
    proc_vec(i,:)= proc_vec(i,:) -R1_tmp    ! subtract continue
    proc_vec(i,:)= proc_vec(i,)* Weight(:) ! weighting function
    RA1D_tmp=0.

    call fftpat_XY(proc_vec(i,:),RA1D_tmp(:),N_Kern,DSS,DCS,M_Kern, 1)

    proc_vec(i,1)= R1_tmp    ! re_add continue on real part for f=0
    proc_vec(i+ncomp1,:)=RA1D_tmp(:) ! store imaginary part

!   normalisation for energy conservation (we took only 1/2 spectrum)
!   rappel: pui_w is the effective amplitude of the window

    proc_vec(i,2:N_Kern/2)=proc_vec(i,2:N_Kern/2)*1.4142137/pui_W

enddo

if (cal_step == 7) then
!   le continue est mis dans la partie reelle de la frequence zero
    proc_vec(1,1)= Bdc_x
    proc_vec(2,1)= -Bdc_y
    proc_vec(1+ncomp1,1)=0.
    proc_vec(2+ncomp1,1)=0.
endif
endif
```

The spectrum is then written in a temporary file, before transfer to the “INDEXED_DATA” field of the RFF file using the following instructions:

```
!   write spectra as axr,axI,ayR,ayI,azR,azI

do j=1,N_Kern/2
    if(j < N_Kern/2) then
        write(tmp_file_unit,ou_data_format) &
            (proc_vec(i,j),proc_vec(i+ncomp1,j), i=1,ncomp1)
    else
        write(tmp_file_unit,ou_data_format_L) &
            (proc_vec(i,j),proc_vec(i+ncomp1,j), i=1,ncomp1)
    endif
enddo
```

20.3 Vectime_L1_to_spectro_L2 parameters

This procedure therefore takes as input the waveforms of level L1 (not calibrated) and produces a calibrated spectrogram, that is to say a series of spectra calibrated according to the method described above. We saw in chapter 20 that the parameters of this procedure were as follows:

Use:

RCL_vectime_L1_to_spectro_L2 VTL1.rff SPL2.rff Fdet Fc F1 F2 Step N-Kern N_shift Apod

VTL1.rff	: name of an input vectime RFF file
SPL2.rff	: name of an output spectrogram RFF file
Fdet	: detrend frequency (0. for classis despin)
Fc	: Frequency cut-off for calibration
Fmin	: frequency min for filtering (Min=0 => Fc)
Fmax	: frequency max for filtering (Max=0 => Nyquist)
Step	: Processing steps asked (1-8)
	1: Volts, spinning system, with DC field
	2: Volts, spinning system, without DC field
	3: nTesla, spinning system, without DC field
	4: nTesla, fixed SR2 system, without DC field
	5: nTesla, fixed SR2 system, with DC field
	6: nTesla, fixed SR2 system, only DC-field
	7: nTesla, fixed ISR2 system
	8: nTesla, fixed GSE system
N_Kern	: Kernel size
N_shift	: for sliding window
Apod	: trapezium (t) or nothing(n)

Certain precautions should of course be taken with regard to the parameters to be applied, depending on what you are looking for and the type of signal at the input.

• File name

The names of the files VTL1.rff and SPL2.rff are free, as long as they have the suffix rff. For example, we can have a name like the following for VTL1:

CLU3_STASC_VTL1_NBR_20021009_03-12.rff

For consistency, it is fashionable to name SPL2 in a similar way, for example:

CLU3_STASC_SPL2_NBR_20021009_03-12.rff

The VTL1 file must be of the VecTime class and contain the necessary metadata. The SPL2 file will be generated by preserving these metadata, and modifying or adding those that are necessary. Like all RCL procedures, the HISTORY parameter is completed.

• Frequency of "detrend"

"Detrend" is a method of smoothing waveforms, removing low frequencies. The principle is as follows:

We perform a smoothing on Nlis points of a waveform of Nbp points. Each point of the smoothed waveform takes the average value of the Nlis / 2 points on each side. When the average on the window of Nlis point is made, we shift this window by one point and we start again. All of these average values, which we can call a smooth waveform, represent the "trend" of the signal, that is to say its average evolution over the sequence considered. It no longer contains any frequencies greater than $F_{det} = Nlis / Fe$, where Fe is the sampling frequency. Obviously we must have $Nlis < Nbp / 2$.

The smoothed waveform is subtracted from the original signal. The signal obtained is freed from low frequencies, below Fdet, known as the "detrend frequency".

The below illustrates this principle. In black and above, we have the original signal, whose trend we calculated, in red. At the bottom, we get the signal from which we have subtracted its trend.

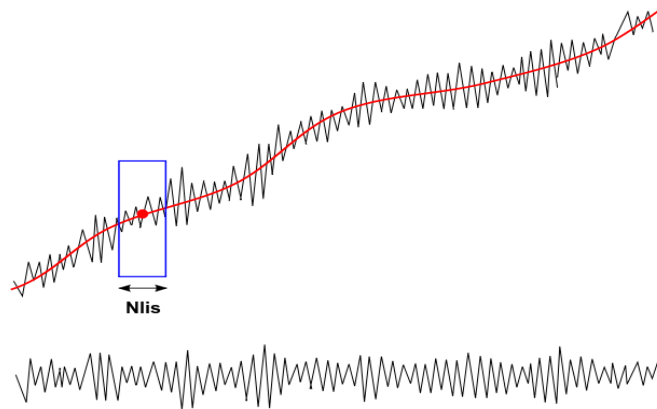


FIG. 20.3: Principle of "detrend"

This method is particularly useful in two cases:

- It allows to calibrate the VTL1 data over short periods, less than that of the spin, for which the classical despin method can no longer be applied. Indeed, if we try to calibrate on only a portion of the spin sinusoid (see Figure 13), we can no longer calculate the spin sinusoid. On the other hand, one can carry out a detrend which by smoothing will approximate the tendency of the signal, that is to say a portion of the sinusoid, and will withdraw it.
- In case of strong DC field in the spin plane, it improves the despin function, and can be used together. In addition, the spin sinusoid can be saturated, and if the saturation is not too great, we can remove the saturated sine by the detrend method, and calibrate the high frequency portions on the unsaturated sides of the sine wave. In HBR mode, this can be useful to recover (short) interesting signal portions, and allows to calibrate the data over durations corresponding to an arch of the spin sinusoid.

Keep in mind that unlike despin, detrend not only removes the signal components at the spin frequency, but all low frequencies. Since the sensitivity of the search-coil is not optimum at low frequencies, this disadvantage can be considered minor. In the mass production chain for the CSA, this option is not used.

• Cutoff frequency

During calibration, we divide the amplitude in volts of a frequency line by the value of the transfer function at that frequency. In the case of a search-coil, the transfer function at zero frequency is zero. We define the cutoff frequency by the value below which the transfer function has no meaning. The calibrated waveforms are therefore filtered below this frequency. It is recalled that this frequency is given in the rotating frame of the antennas. A typical value for STAFF is 0.1 Hz. It is not recommended to go below.

• Min and max frequencies

These are the frequencies between which we calibrate. The calibrated signal is therefore filtered between these terminals, still in the rotating frame. If we mention (0., 0.) the signal will be filtered between F_c and the Nyquist frequency (12.5 or 225 Hz depending on the mode).

• The steps of calibration

They are described in section 20.2.2. Values 1 to 3 are only used for data verification and verification of calibration steps.

- Option 4 is the most common, and is useful when you want to calculate and view spectra or spectrograms.
- Option 5, the DC value is added over the X and Y components of the waveforms. They are therefore no longer centered on zero, and cannot be changed reference. This is the option that is used when making direct comparisons with FGM data, in burst mode (see 13, Robert 2014).
- Option 6 is used to check the stability of the estimated Bperp field during despin.
- Options 7 and 8 are the production options for the “quick look” visualizations in ISR2 and the spectra delivered to the CSA in GSE.

• The size of the calibration kernel

This is an important parameter, which determines both the temporal resolution of the spectra and the quality of the calibration. To choose this value, a compromise has to be made.

- A high value will give a very fine frequency resolution, and therefore a good quality of the calibration since the transfer function varies nonlinearly with the frequency. On the other hand, if one makes a spectrum over a long duration, one supposes during the despin that the perpendicular continuous field does not vary, or little. If it does not, the despin will be bad, and there will be spin residues in the calibrated spectrum.
- A short value is well suited to transient events, varying rapidly over time. But you will have to stay above one or two spin periods to have an effective despin. Below, you will have to play with the detrend. On the other hand, we will then have a bad frequency resolution, which will not be suitable for phenomena of the pseudo-monochromatic wave type. Finally, the transfer function will be applied to wide lines, therefore with poor calibration quality.

In all the cases one has $\Delta t \Delta f = 1$ and it is necessary to play with this compromise. For the production of the spectra of routines, delivered to the CSA, one chose $\Delta t = 10$ seconds, in NBR as in HBR (respectively 256 and 4096 points). This value allows a correct spinning (over two spin periods) and a frequency resolution which makes it possible to correctly follow the variations of the transfer function.

In the case of specific scientific applications, it may be necessary to change these values. For example, for the study of a high frequency whistle in HBR (towards 150 Hz, of width 50 Hz) and relatively transient (10 seconds) we can chose 128 points ($\delta t = 0.28$ sec. and $\delta f = 3.5$ Hz) and used a frequency of detrend of 50 Hz.

For a pseudo-monochromatic wave in NBR, extending over 20 to 30 minutes around 3 Hz in DC field varying little we can chose 512 points (5 spin periods) so $\delta t = 10.48$ sec and $\delta f = 0.049$ Hz which makes it possible to highlight the structures of the wave packets.

• The size of the window movement

In the spectrum calculation program, an "overlapping" parameter is provided, that is to say the possibility of overlapping the spectra, making it possible to artificially increase the time resolution without altering the frequency resolution. This option is managed by the parameter Nshift, making it possible to calculate a spectrum after a displacement of Nshift point, by choosing of course $Nshift \leq Nkern$. However, this option has never been validated and it is not recommended to use it.

• The type of apodization

Several types of waveform apodization are possible before spectrum calculation. In RCL 2.2, the available types are as follows, with their recommendation.

- "No" no apodization: rectangular window, recommended for transient signals whose duration is less than the length of the window or for the separation of two very close frequencies and also with very close amplitudes.
- "Tr" trapezoidal window as in Figure 15: recommended for most spectrograms. This is the one used to produce the SPL2 spectra, then CS delivered to the CSA. Each flank represents $Nf = 1/16$ of the total number of points.

- "Rr" rounded rectangle: improved version of the trapezoid, but which eats a little more signal (see fig. 20.1). The edges are calculated on $Nf = 1/16$ and represent the first and second $1/4$ period of a sine.

- "Rt" rounded trapezoid: derivative version of the trapezoid. The edges are calculated on $Nf = 1/16$ and represent the first and second $1/4$ period of a \sin^2 .

- "Ri" window of Riesz. Its formula is:

$$W_k = 1 - \left(\frac{2(k-1)}{N} - 1 \right)^2$$

- "Ha" Hann (called Hanning) window, generally used if the shape of the signal is not known, and recommended for pseudo-monochromatic waves or sine wave superimpositions, or for narrowband random signals. Its formula is:

$$W_k = 0.5 \left(1 - \cos\left(\frac{2\pi k}{N}\right) \right)$$

- "KB" Kaiser-Bessel window, suitable for the separation of two frequencies that are very close but whose amplitudes are very different. Its formula is complicated, especially since it involves a zero-order Bessel function of the first kind and a parameter α to choose from. This parameter characterizes the compromise between the width of the main lobe and the secondary lobes. A reasonable value is $\alpha = 3$. In this case, a sufficient approximation is given by the following formula ([http://www.diracdelta.co.uk/science/source/k/a/kaiser-bessel window](http://www.diracdelta.co.uk/science/source/k/a/kaiser-bessel%20window)):

$$W_k = 0.40243 - 0.49804 \cos\left(\frac{2\pi k}{N}\right) + 0.09831 \cos\left(\frac{4\pi k}{N}\right) - 0.00122 \cos\left(\frac{6\pi k}{N}\right)$$

- "BH" Blackman-Harris window. It is close to that of Kaiser-Bessel. Its formula is:

$$W_k = 0.35875 - 0.48829 \cos\left(\frac{2\pi k}{N}\right) + 0.14128 \cos\left(\frac{4\pi k}{N}\right) - 0.01168 \cos\left(\frac{6\pi k}{N}\right)$$

- "G" "custom" Gaussian window used for continuous calibration (see next chapter). It is given by

$$W_k = e^{-x^2}$$

with $x = ((k - N/2) - 0.5) / \sigma$ and we take $\sigma = \frac{N/2}{\sqrt{-\ln 10^{-3}}}$

The value -0.5 is put to symmetrize the Gaussian with the vertex between $N/2$ and $N/2 + 1$.

• Shape and normalization of weighting windows

Figure 20.4 below gives the form of some common weighting functions. However, in order to conserve energy, these functions should be normalized before use by a coefficient that normalizes the area to 1, so that there is no loss of energy.

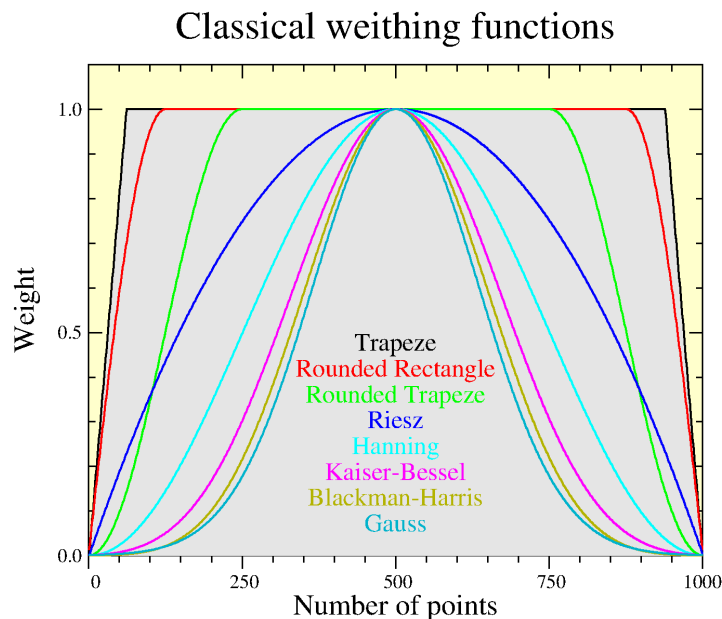


FIG. 20.4: Shape of common weighting functions

The normalization coefficients to be applied are as follows:

	Area	Corr. factor
Trapeze	0.9380000	1.066098
Rounded Rectangle	0.9101529	1.098716
Rounded Trapeze	0.7510000	1.331558
Riesz	0.6666663	1.500001
Hanning	0.4999999	2.000000
Kaiser-Bessel	0.4024300	2.484904
Blackman-Harris	0.3587499	2.787457
Gauss	0.3371234	2.966273

20.4 Continuous waveform calibration

20.4.1 Theory

The method of calibrating continuous waveforms is actually derived from those of calibrating spectra. The detail of this calibration method is described in [13, Robert, 2013] as well as its validation by comparison with FGM data in "Full Resolution".

To obtain the calibrated waveform from the calibrated spectrum, in theory, it would suffice to do an inverse FFT. However, the resulting waveform is marred by a discontinuity at the edges, by resetting the ends. To avoid this, the principle of continuous waveform calibration (as opposed to "discontinuous" when $N_Shift = N_Kern$) is as follows:

- We calibrate a spectrum which starts at time t_1 , over a duration T . The apodization window used is a Gaussian, defined in chapter 20, section 20.3.
- We do the reverse FFT, but we only keep the central points of the calibrated waveform, i.e. N_shift points, which correspond to the top of the Gaussian
- We shift the time t_1 by an interval δ corresponding to N_Shift points, and we again calibrate a window of duration $T = N_Kern/fe$

In practice, the optimal calibration is obtained for $N_Shift = 2$. On a window of N_Kern points (1024 for example) one thus keeps only the 2 points of the top of the Gaussian.

20.4.2 The successive steps of calibration

They are in fact the same as for the calibration of the spectra, with a few details:

- **In step 3**, the calibration is now done by an optimized routine `DECONVO_C3` from the `lib_deconvo` library. The following instructions will be found in the code:

```
!   Deconvolution :
!   waveform centering(done in step 2), weighting window, FFT,
!   transfer function correction, filtering, inverse FFT
!   The input waveform must be despined before (see step 2)

!   weighting window
      CSpec(1,:) = cmplx(proc_vec(1,:) * Weight(:), 0.)
      CSpec(2,:) = cmplx(proc_vec(2,:) * Weight(:), 0.)
      CSpec(3,:) = cmplx(proc_vec(3,:) * Weight(:), 0.)

!   time to frequency domain, transfer function correction,
!   back to time domain
      call DECONVO_C3(CSpec, TFCor, N_Kern, DSS, DCS, M_Kern)

      proc_vec(1,:) = real(CSpec(1,:))
      proc_vec(2,:) = real(CSpec(2,:))
      proc_vec(3,:) = real(CSpec(3,:))
```

- **Still in step 3**, after calibration, the energy due to the Gaussian weighting is normalized:

```
! We do an inverse weighting on the guarded points, so as not to
! assign the module and create harmonics to N_Shift
! that for the Gaussian case, with N_Shift < N_Kern

      if (apod == 'g') then
        do j = i1, i2
          proc_vec(i, j) = proc_vec(i, j) / Weight(j)
        enddo
      endif
```

- **In step 7**, we add to the 3 components calculated in step 4, two components giving the value of the DC on the components X and Y, all in reference `ISR2`:

```
!   SR2 to ISR2

      if (spin_phase >= -360.) then
        do i = i1, i2
          proc_vec(2, i) = -proc_vec(2, i)
          proc_vec(3, i) = -proc_vec(3, i)
          proc_vec(4, i) = Bdc_x
          proc_vec(5, i) = -Bdc_y
        enddo
      endif

      ncomp = 5
```


Chapter 21

USEFUL INFORMATIONS

21.1 Estimation of the DC field in the spin plane

21.1.1 Problem

The rotation of the magnetic sensors in the high direct field generates a strong sinusoidal component, of amplitude B_{\perp} in the spin plane. It is therefore possible, by measuring the amplitude and the phase of this sinusoid, to recalculate the components B_x and B_y of this field, in the rotating frame of reference SR as in the fixed frame of reference SR2.

When the despin procedure is applied by the desinus routine (see code in src/lib_deconvo.f90) the amplitude in Volts and the phase of the spin sinusoid are recovered at the output for each component, ie $\tilde{B}_x, \tilde{\varphi}_x$ and $\tilde{B}_y, \tilde{\varphi}_y$ and that of the component $B_z, \tilde{B}_z, \tilde{\varphi}_z$ which, although small, will allow the calculation of the tilt angle (see chapter 21.2.1).

The values \tilde{B}_x et \tilde{B}_y are two different estimates of the perpendicular field, B_{\perp} , in uncalibrated value (in V), as well as $\tilde{\varphi}_x$ and $\tilde{\varphi}_y$ are two estimates of the phase φ in the rotating frame SR.

The Figure 21.1 illustrates this measurement in the rotating frame of the antennas.

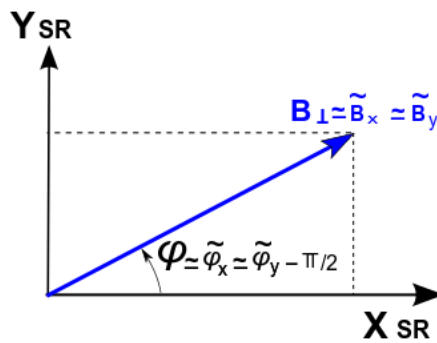


FIG. 21.1: Measurement of the perpendicular field and of the phase in the spin plane.

By associating with these values the complex coefficient of the transfer function at the spin frequency, taking into account on the one hand the position of the antennas with respect to the axes of the " Body-Build System " (see section 19.9) and on the other hand the " spin phase " defined in chapter 21.1, one can reconstitute the components x and y of the field in the fixed reference system SR2.

21.1.2 Amplitude and phase of B_{\perp} in spinning system SR

From the two estimates \tilde{B}_x and \tilde{B}_y of the amplitude (in V) we can calculate two estimates $B_{\perp x}$ and $B_{\perp y}$ of the amplitude in nT by the formulas:

$$B_{\perp x} = \tilde{B}_x \frac{1}{|\alpha_x(f_s)|} B_{\perp y} = \tilde{B}_y \frac{1}{|\alpha_y(f_s)|}$$

Where $|\alpha_x(f_s)|$ and $|\alpha_y(f_s)|$ are the moduli of the transfer function at the spin frequency for each of the two components X and Y.

Likewise, two different estimations of the angle φ are obtained by correcting the phases measured by the transfer function :

$$\varphi_x = \tilde{\varphi}_x - \text{phase}[\alpha_x(f_s)]$$

$$\varphi_y = \tilde{\varphi}_y - \text{phase}[\alpha_y(f_s)]$$

21.1.3 Tests on $B_{\perp x}$ et $B_{\perp y}$

In principle $B_{\perp x}$ and $B_{\perp y}$ should be equal, although the transfer functions are slightly different. Likewise the difference between φ_x and φ_y should be 90° .

We can therefore perform a test to see if these assumptions are true or not. The measurement of the differences will make it possible to estimate the antenna depointing matrix. The complete calculation of the depointing matrix is quite complex but an approach has been done in section 21.3.

On the other hand, insofar as the Z sensor is close to the spin axis, the measurement of the amplitude \tilde{B}_z makes it possible to estimate the “ misalignment angle ”, the calculation of which is given in section 21.2.

21.1.4 Amplitude and phase of B_{\perp} in fixed reference SR2

Insofar as there are two estimations of the field B_{\perp} and of the phase angle φ only one must be proposed. It is natural to take as the value of B_{\perp} the mean value:

$$B_{\perp} = \frac{1}{2} (B_{\perp x} + B_{\perp y})$$

For the phase we decide to take φ_x as reference phase; this choice is arbitrary, we could also take φ_y .

To obtain the values B_x and B_y of the DC field in the SR system it is enough to take an angle rotation ψ . The calculation of this angle is made in the section 20.2.2 and we have :

$$\psi = (\omega_s t + \varphi_s + \alpha).$$

The values retained for the continuous field in the spin plane are then :

$$B_x^{\text{DC}} = B_{\perp} \sin(\varphi_x + \psi)$$

$$B_y^{\text{DC}} = B_{\perp} \cos(\varphi_x + \psi)$$

By this method, we have preserved the fact that the amplitude is the same on the two components, and that they are well out of phase by 90° .

21.2 Estimation of “misalignment angle”

The “misalignment angle” is the small angle between the true rotation axis of the spacecraft and the Z antenna of the experiment (as STAFF).

21.2.1 Definitions

Situation is shown by the Figure 21.2 hereafter : (X,Y,Z) is the SR2 or “Despun” System.

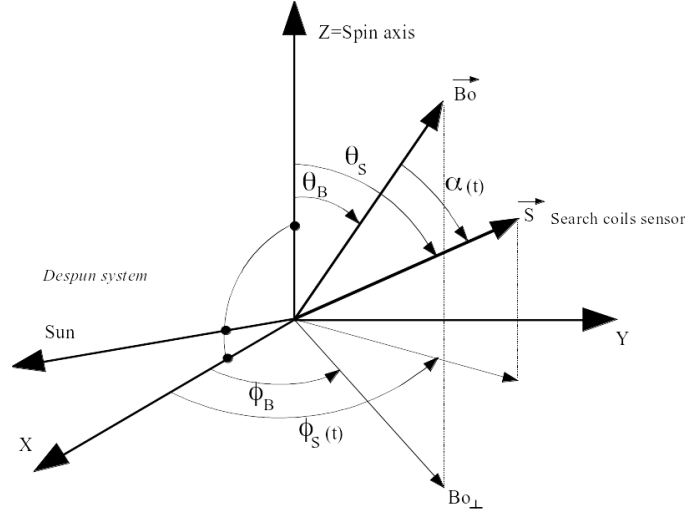


FIG. 21.2: Coordinate system for a spinning spacecraft.

In the SR2, or “Despun” system (DS), the Z axis of the spacecraft is along the spin axis. The sensor \vec{S} (which can be the X, Y, or Z sensors) is rotating around the spin axis with a constant angle θ_S . The DC field \vec{B}_0 is roughly constant on a few period of spin. More often than not, the Z sensor is close to the spin axis ($\theta_z \sim 0^\circ$) and the X and Y sensors are close to the spin plane ($\theta_x \sim \theta_y \sim 90^\circ$).

The θ_z angle is called the misalignment angle. The estimate of this angle is useful to compute the matrix between the sensor system and the spinning system where Z is the spin axis, generally close to the unit matrix.

21.2.2 Signal delivered by a rotating antenna into a DC field

Vectors \vec{S} and \vec{B}_0 have for components:

$$\vec{S} = \begin{pmatrix} \sin \theta_S \cos \phi_S \\ \sin \theta_S \sin \phi_S \\ \cos \theta_S \end{pmatrix} \quad \vec{B}_0 = B_0 \begin{pmatrix} \sin \theta_B \cos \phi_B \\ \sin \theta_B \sin \phi_B \\ \cos \theta_B \end{pmatrix}$$

The signal delivered by the sensor S is $\vec{B}_0 \cos \alpha(t) = \vec{B}_0 \cdot \vec{S}$, so:

$$S(t) = B_0 \sin \theta_B \sin \theta_S \cos(\phi_B - \phi_S(t)) + B_0 \cos \theta_B \cos \theta_S$$

The angles θ_B and θ_S being constant, and the transfer function of a Search-Coil being naught at zero frequency, the signal is reduced at its times variations, and the previous expression becomes:

$$S(t) = B_{o\perp} \sin \theta_S \cos(\phi_B - \phi_S(t)) \text{ with } B_{o\perp} = B_0 \sin \theta_B \text{ (no time-dependent)}$$

So the sensor measure only the component of the DC perpendicular to the spin axis, and this DC field is viewed in the sensor reference frame as a circular wave, left handed polarized, at the spin frequency.

As S sensor is spinning around Z, θ_S is no time-dependant and $\varphi_S(t)$ can be expressed as :

$$\varphi_S(t) = 2\pi F_s t + \varphi_o$$

Therefore we have:

$$S(t) = B_{o\perp} \sin \theta_S \cos (2\pi F_s t + \varphi_o - \varphi_B)$$

So, the $S(t)$ signal is a pure sine wave, with amplitude equal to $B_{o\perp} \sin \theta_S$

21.2.3 Estimate of the misalignment angle θ_z for low value

When we choose the S sensor as the Z sensor, we have on the $z(t)$ component a sine signal with an amplitude $a_z = B_{o\perp} \sin \theta_z$ and a phase of $(\varphi_o - \varphi_B)$. These two quantities can be computed by using the algorithm of the “desinus” software.

For an antenna Z close to the spin axis ($\theta_z \sim 0$, and $\theta_x \sim \theta_y \sim 90^\circ$) and a_x being the amplitude of the corresponding sine on $x(t)$ component, we can estimate θ_z by :

$$\sin \theta_z \simeq a_z / a_x$$

21.2.4 Estimate of the misalignment angle θ_z for high value

So we have (1) :

$$S_x(t) = B_{o\perp} \sin \theta_x \cos (2\pi F_s t + \varphi_{ox} - \varphi_B)$$

$$S_y(t) = B_{o\perp} \sin \theta_y \cos (2\pi F_s t + \varphi_{oy} - \varphi_B)$$

$$S_z(t) = B_{o\perp} \sin \theta_z \cos (2\pi F_s t + \varphi_{oz} - \varphi_B)$$

- If the three-axis of the sensors is an orthogonal system, we have

$$\vec{S}_x \cdot \vec{S}_y = 0$$

$$\vec{S}_y \cdot \vec{S}_z = 0$$

$$\vec{S}_z \cdot \vec{S}_x = 0$$

After computation, we found (2) :

$$\cos (\varphi_x - \varphi_y) = -\cos \theta_x \cos \theta_y / \sin \theta_x \sin \theta_y = -1 / \tan \theta_x \tan \theta_y$$

$$\cos (\varphi_x - \varphi_z) = -\cos \theta_x \cos \theta_z / \sin \theta_x \sin \theta_z = -1 / \tan \theta_x \tan \theta_z$$

$$\cos (\varphi_y - \varphi_z) = -\cos \theta_y \cos \theta_z / \sin \theta_y \sin \theta_z = -1 / \tan \theta_y \tan \theta_z$$

with

$$\varphi_x(t) = 2\pi F_s t + \varphi_{ox} \quad \varphi_y(t) = 2\pi F_s t + \varphi_{oy} \quad \varphi_z(t) = 2\pi F_s t + \varphi_{oz}$$

- Always with the properties of an orthogonal system, after computation we found (3) :

$$\sin (\varphi_z - \varphi_y) = \cos \theta_x / \sin \theta_y \sin \theta_z$$

$$\sin (\varphi_x - \varphi_z) = \cos \theta_y / \sin \theta_z \sin \theta_x$$

$$\sin (\varphi_y - \varphi_x) = \cos \theta_z / \sin \theta_x \sin \theta_y$$

- By using (2) and (3), we can eliminate the φ_i angles, and we obtain a system of 3 equations depending of only θ_i (4) :

$$\begin{aligned} \cos^2 \theta_x &= -\cos(\theta_y - \theta_z)(\theta_y + \theta_z) \quad \sin^2 \theta_x = 1 + \cos(\theta_y - \theta_z)(\theta_y + \theta_z) \\ \cos^2 \theta_y &= -\cos(\theta_z - \theta_x)(\theta_z + \theta_x) \quad \text{so } \sin^2 \theta_y = 1 + \cos(\theta_z - \theta_x)(\theta_z + \theta_x) \\ \cos^2 \theta_z &= -\cos(\theta_x - \theta_y)(\theta_x + \theta_y) \quad \sin^2 \theta_z = 1 + \cos(\theta_x - \theta_y)(\theta_x + \theta_y) \end{aligned}$$

- If now we return to equation (1) we can see that the amplitude of the 3 sine signals are :

$$\begin{aligned} a_x &= B_{o\perp} \sin \theta_x \\ a_y &= B_{o\perp} \sin \theta_y \\ a_z &= B_{o\perp} \sin \theta_z \end{aligned}$$

so, we have (5) :

$$B_{o\perp} = a_x / \sin \theta_x = a_y / \sin \theta_y = a_z / \sin \theta_z$$

- Now, by using (4) and (5), we obtains 3 groups of 3 equations with 3 unknown values (6) :

$$\begin{aligned} \sin^2 \theta_z &= 1 + \cos(\theta_x - \theta_y)(\theta_x + \theta_y) \\ \sin^2 \theta_z &= (a_z/a_x)^2 \sin^2 \theta_x \\ \sin^2 \theta_z &= (a_z/a_y)^2 \sin^2 \theta_y \end{aligned}$$

and we can get easily the two equivalent systems for θ_x and θ_y .

- At last, by solving the 3 previous systems, we can compute the 3 angles θ_x , θ_y , θ_z , we obtains the solutions:

$$\begin{aligned} \sin \theta_x &= a_x \sqrt{2} / (a_x^2 + a_y^2 + a_z^2)^{1/2} \\ \sin \theta_y &= a_y \sqrt{2} / (a_x^2 + a_y^2 + a_z^2)^{1/2} \\ \sin \theta_z &= a_z \sqrt{2} / (a_x^2 + a_y^2 + a_z^2)^{1/2} \end{aligned}$$

We can check that if $\theta_x \simeq \theta_y \simeq 90^\circ$, we found the previous approximation $\sin \theta_z \simeq a_z / a_x$

Likewise, if θ_z is equal to zero, from (5) $\theta_x = \theta_y = 90^\circ$, and $a_x = a_y = B_{o\perp}$

Nevertheless, if θ_z is not equal to zero, and $\theta_x \neq \theta_y$.

Finally, that we call the misalignment angle is θ_z given by:

$$\theta_z = \sin^{-1} \left(\frac{a_z \sqrt{2}}{(a_x^2 + a_y^2 + a_z^2)^{1/2}} \right)$$

21.3 Calculation of the depointing matrix

21.3.1 Passing from non-orthogonal coordinates to an orthogonal system

This chapter is taken from a document by P. Robert dated January 20, 1994. Thank you to G. Chanteur for re-reading, checking and suggesting a clearer presentation than the original one.

21.3.1.1 Position of the problem

The problem is identical to that which arises during the CLUSTER mission to calculate the current measured by the 4 magnetometers arranged in a tetrahedron, using the method based on Ampère's theorem. In this method, one measures the current density J according to 3 non-orthogonal directions which are those of the norms to the 3 chosen faces (among 4 possible combinations) of an unspecified tetrahedron but not degenerated (neither flat, nor linear) by the formula

$$\mu_0 I = \oint \vec{B} \bullet d\vec{l}$$

on each of the chosen faces. Figure 21.4 below summarizes the problem.

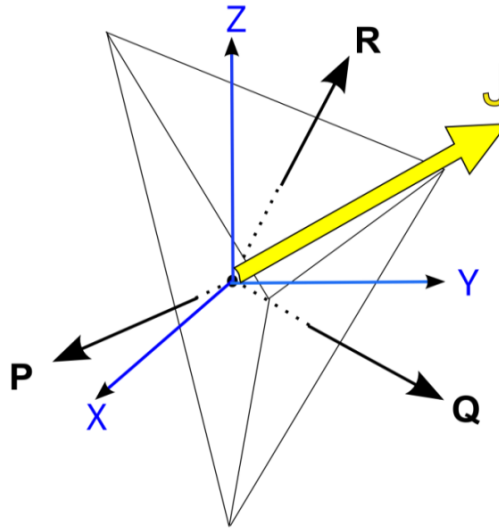


FIG. 21.3: Projection of the vector J onto 3 non-orthogonal vectors (P, Q, R)

he problem here is similar : from 3 measurements B_{pm}, B_{qm}, B_{rm} made by the magnetic sensors on non-orthogonal axes of measurement $\vec{P}, \vec{Q}, \vec{R}$ we want to recalculate the Cartesian components of the orthonormal $\vec{X}, \vec{Y}, \vec{Z}$ i.e. components B_x, B_y, B_z .

In chapter 21.3.1 the non-orthogonal measurement system (P, Q, R) is called « **Sensor Coordinate System** » (SCS), while the reference mark (X, Y, Z) is called « **Orthogonal Sensor System** » (OSS).

21.3.1.2 Switching from a non-orthogonal system to an orthogonal system

One indicates by $\vec{P}, \vec{Q}, \vec{R}$ the normals to the three faces considered, these normals being neither coplanar, nor collinear. Their direction is supposed to be known in the orthonormal coordinate system of reference $\vec{X}, \vec{Y}, \vec{Z}$ as :

$$\vec{P} = \begin{pmatrix} P_x \\ P_y \\ P_z \end{pmatrix} \quad \vec{Q} = \begin{pmatrix} Q_x \\ Q_y \\ Q_z \end{pmatrix} \quad \vec{R} = \begin{pmatrix} R_x \\ R_y \\ R_z \end{pmatrix}$$

The components of \vec{B} in the non-orthogonal coordinate system $\vec{P}, \vec{Q}, \vec{R}$ are :

$$\vec{B} = \begin{pmatrix} B_p \\ B_q \\ B_r \end{pmatrix} = B_p \vec{P} + B_q \vec{Q} + B_r \vec{R}$$

By replacing the vectors in [2] by their expression given in [1] we obtain :

$$\vec{B} = B_p \begin{pmatrix} P_x \\ P_y \\ P_z \end{pmatrix} + B_q \begin{pmatrix} Q_x \\ Q_y \\ Q_z \end{pmatrix} + B_r \begin{pmatrix} R_x \\ R_y \\ R_z \end{pmatrix}$$

Which is expressed in the matrix form by :

$$\vec{B} = \begin{pmatrix} B_x \\ B_y \\ B_z \end{pmatrix} = \begin{pmatrix} P_x & Q_x & R_x \\ P_y & Q_y & R_y \\ P_z & Q_z & R_z \end{pmatrix} \begin{pmatrix} B_p \\ B_q \\ B_r \end{pmatrix}$$

One thus finds the matrix of passage of the non-orthogonal $\vec{P}, \vec{Q}, \vec{R}$ coordinate system to the orthonormal coordinate system $\vec{X}, \vec{Y}, \vec{Z}$

But **BE CAREFUL !** The components B_p, B_q, B_r **ARE NOT** the experimental measurements \vec{B} by the sensors along the axes $\vec{P}, \vec{Q}, \vec{R}$, as can be seen on the figure 21.4.

It will therefore be necessary to proceed otherwise.

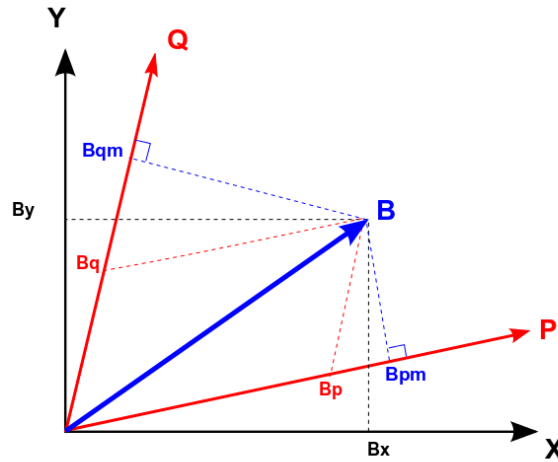


FIG. 21.4: Difference between mathematical components of B and measures

21.3.1.3 Calculation of XYZ components from PQR

If we call B_{pm}, B_{qm}, B_{rm} measured components of \vec{B} in the frame $\vec{P}, \vec{Q}, \vec{R}$ we can write:

$$B_{pm} = \vec{P} \bullet \vec{B} = P_x B_x + P_y B_y + P_z B_z$$

$$B_{qm} = \vec{Q} \bullet \vec{B} = Q_x B_x + Q_y B_y + Q_z B_z$$

$$B_{rm} = \vec{R} \bullet \vec{B} = R_x B_x + R_y B_y + R_z B_z$$

This linear system is written in the matrix form as:

$$\vec{B} = \begin{pmatrix} B_{pm} \\ B_{qm} \\ B_{rm} \end{pmatrix} = \begin{pmatrix} P_x & P_y & P_z \\ Q_x & Q_y & Q_z \\ R_x & R_y & R_z \end{pmatrix} \begin{pmatrix} B_x \\ B_y \\ B_z \end{pmatrix}$$

This corresponds to a matrix of passage $M = (P_i, Q_i, R_i)$ of the values of \vec{B} to the orthonormal system (X, Y, Z) (what we are looking for) towards the values \vec{B} projected on the axes $\vec{P}, \vec{Q}, \vec{R}$ (what we measure). It will therefore be necessary to do the inverse operation.

21.3.2 Definition of the depointing matrix

The matrix $M = (P_i, Q_i, R_i)$ which gives the direction of the sensors in a Cartesian coordinate system is called the “depointing matrix”. In the practical application, it is of course necessary to know this matrix M if we want to be able to make the correction of the misalignment. It can only have been properly measured on the ground before launch.

Note: If the starting measurement system $\vec{P}, \vec{Q}, \vec{R}$ is already orthonormal, and if we have $\vec{P} = \vec{X}$, $\vec{Q} = \vec{Y}$, and $\vec{R} = \vec{Z}$ the misalignment matrix reduced to the unitary matrix and of course :

$$\vec{B} = \begin{pmatrix} B_{pm} \\ B_{qm} \\ B_{rm} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} B_x \\ B_y \\ B_z \end{pmatrix}$$

In the absence of sufficiently precise data on the direction of the antennas compared to the OSS system for the CLUSTER / STAFF sensors, the unit matrix was applied (see [13], Robert, 2013).

21.3.2.1 Misalignment correction matrix

We therefore wish to do the inverse operation, ie to determine \vec{B} in the orthonormal reference XYZ where the vectors $\vec{P}, \vec{Q}, \vec{R}$ are known. It is therefore sufficient to calculate the inverse matrix of M, that is to say :

$$M' = \begin{pmatrix} U_p & U_q & U_r \\ V_p & V_q & V_r \\ W_p & W_q & W_r \end{pmatrix} \text{ inverse of } M = \begin{pmatrix} P_x & P_y & P_z \\ Q_x & Q_y & Q_z \\ R_x & R_y & R_z \end{pmatrix}$$

And we will obtain the desired result :

$$\vec{B} = \begin{pmatrix} B_x \\ B_y \\ B_z \end{pmatrix} = \begin{pmatrix} U_p & U_q & U_r \\ V_p & V_q & V_r \\ W_p & W_q & W_r \end{pmatrix} \begin{pmatrix} B_{pm} \\ B_{qm} \\ B_{rm} \end{pmatrix}$$

Matrix M' is called « misalignment correction matrix ».

21.3.2.2 Calculation of the inverse depointing matrix

We proceed in four steps :

1 - calculation of the transposed matrix

$$M^t = \begin{pmatrix} P_x & Q_x & R_x \\ P_y & Q_y & R_y \\ P_z & Q_z & R_z \end{pmatrix}$$

2 - replacement of each term by its determinant, and change of sign for terms whose sum of indices is odd

$$\begin{aligned} U_p' &= (Q_y R_z - Q_z R_y) \\ U_q' &= -(Q_x R_z - Q_z R_x) \\ U_r' &= (Q_x R_y - Q_y R_x) \\ V_p' &= -(P_y R_z - P_z R_y) \\ V_q' &= (P_x R_z - P_z R_x) \\ V_r' &= -(P_x R_y - P_y R_x) \\ W_p' &= (P_y Q_z - P_z Q_y) \\ W_q' &= -(P_x Q_z - P_z Q_x) \\ W_r' &= (P_x Q_y - P_y Q_x) \end{aligned}$$

3 - calculation of the determinant

$$D = P_x Q_y R_z + P_y Q_z R_x + P_z Q_x R_y - P_z Q_y R_x - P_x Q_z R_y - P_y Q_x R_z$$

4 - normalization of each term by the determinant of the transposed matrix

So either :

$$M' = \begin{pmatrix} U_p & U_q & U_r \\ V_p & V_q & V_r \\ W_p & W_q & W_r \end{pmatrix} = \frac{1}{D} \begin{pmatrix} U_p' & U_q' & U_r' \\ V_p' & V_q' & V_r' \\ W_p' & W_q' & W_r' \end{pmatrix}$$

Note: If the starting measurement system $(\vec{P}, \vec{Q}, \vec{R})$ is already orthonormal, the inverse matrix M' is immediately deduced from the matrix M by :

$$M' = M^t = \begin{pmatrix} P_x & Q_x & R_x \\ P_y & Q_y & R_y \\ P_z & Q_z & R_z \end{pmatrix}$$

21.3.3 Example of application: case of Interball electric antennas

The direction of the measurement axes are deduced from the coordinates of the balls in cm in the Cartesian coordinate system X, Y, Z of the satellite (information S. Perault, 2002), namely:

$$B_{1x} = \begin{pmatrix} 2195 \\ 0 \\ -228 \end{pmatrix} B_{2x} = \begin{pmatrix} 2125 \cdot \cos(15^\circ) \\ -2125 \cdot \sin(15^\circ) \\ 228 \end{pmatrix}$$

We deduce the non-normalized P' axis :

$$P' = \begin{pmatrix} 2195 + 2125 \cdot \cos(15^\circ) \\ 2125 \cdot \sin(15^\circ) \\ -456 \end{pmatrix} = \begin{pmatrix} 4247.6 \\ 549.99 \\ -456 \end{pmatrix}$$

P is deduced from P' after normalization, with $N=4307.26$, ie $P = \begin{pmatrix} 0.9860 \\ 0.1277 \\ 0.1059 \end{pmatrix}$

The **Q** and **R** axis being identical to the Y and Z axes, we have :

$$Q = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} R = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

And the depointed correction matrix is therefore :

$$M' = \begin{pmatrix} U_p & U_q & U_r \\ V_p & V_q & V_r \\ W_p & W_q & W_r \end{pmatrix} = \frac{1}{P_x} \begin{pmatrix} 1 & 0 & 0 \\ -P_y & P_x & 0 \\ -P_z & 0 & P_x \end{pmatrix}$$

ie :

$$M' = \begin{pmatrix} 1 & 0 & 0 \\ -0.1295 & 1 & 0 \\ -0.1074 & 0 & 1 \end{pmatrix}$$

It is thus seen that it appears weak diagonal terms but not zero due to the non-orthogonality of the initial sensors reference system.

21.3.4 Experimental estimation of certain terms

According to the conventions of section 21.3.1, the non-orthogonal measurement system (P, Q, R) is called « **Sensor Coordinate System** » (SCS), while the coordinate system (X,Y,Z) is called « **Orthogonal Sensor System** » (OSS). We saw that for CLUSTER / STAFF, we made the approximation :

$$OSS \rightarrow SCS = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

While in the exact calculation it is necessary to replace the unit matrix by the depointing matrix, namely :

$$OSS \rightarrow SCS = \begin{pmatrix} U_p & U_q & U_r \\ V_p & V_q & V_r \\ W_p & W_q & W_r \end{pmatrix}$$

This gives us, for the transition to DSS (the Data Sensor System):

$$DSS \rightarrow OSS = \begin{pmatrix} U_r & U_p & U_q \\ V_r & V_p & V_q \\ W_r & W_p & W_q \end{pmatrix}$$

The transition from DSS to BBS is more difficult. In section 21.3.1 the small angle between the Z axis of the DSS and the X axis of the BBS was neglected by making the assumption $\vec{Z}_{DSS} \simeq \vec{X}_{BBS}$.

In truth, there is also a small misalignment between the axis of the arm carrying the instrument and the Body Build, and the formula of the section 21.3 should be written with an additional correction matrix as :

$$BBS \rightarrow DSS = \begin{pmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \end{pmatrix}$$

Assuming that the passage from the reference of the arm to the reference of the Body Build is between orthogonal reference systems.

In this case the matrix

$$D = \begin{pmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \end{pmatrix}$$

is a rotation matrix close to unity.

So we have $a_x \simeq b_y \simeq c_z \simeq 1$

while $a_y \simeq a_z \simeq \varepsilon$ $b_x \simeq b_z \simeq \varepsilon$ $c_x \simeq c_y \simeq \varepsilon$

In the same way we have :

$$\text{DSS} \rightarrow \text{OSS} = \begin{pmatrix} U_r & U_p & U_q \\ V_r & V_p & V_q \\ W_r & W_p & W_q \end{pmatrix} \simeq \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

We can say that $W_r \simeq U_p \simeq V_q \simeq 1$

while $U_r \simeq U_q \simeq \varepsilon$ $V_r \simeq V_p \simeq \varepsilon$ $W_p \simeq W_q \simeq \varepsilon$

If the state is worth it, one can do the complete calculation in order 1, neglecting the terms in ε^2 in order to find the total correction matrix. We can then compare it with the " misalignment angle " measured experimentally by the formula in section 21.2

Experimentally, the different values of the amplitude of the two components B_{xs}, B_{ys} of the spin sinusoid measured in the SCS frame, as well as the difference of their phase with the theoretical value of 90° , should be able to allow an experimental estimation of the matrix of total depointing : 3 angles to be determined, from 3 parameters, the problem is therefore a priori soluble.

Nevertheless, for CLUSTER / STAFF, the misalignment angle being of the order of 0.5° , and in any case lower than the degree, this complete calculation was not made.

21.4 Small reminders on the Fourier transform

The literature on the Fourier transform is abundant and diverse. What is given here is a simple basic reminder of what one needs to know for the calculation of spectrum or spectrogram.

21.4.1 The Fourier transform in the mathematical sense

The Fourier transform (TF) makes it possible to go from the “ time ” domain , that is to say of a function dependent on time $x(t)$, which is also called “ waveform ”, to the “ frequency ” domain. ", That is to say a function dependent on the frequency $X(f)$, which is called " spectrum ". For this to

be possible, the function $x(t)$ must be integrable on the real \mathbb{R} .

This operation is done by the formula :

$$X(f) = \int_{-\infty}^{+\infty} x(t) e^{-2i\pi ft} dt$$

From the spectrum $X(f)$ we can return to the waveform by :

$$x(t) = \int_{-\infty}^{+\infty} X(f) e^{+2i\pi ft} df$$

Of course, in practice we never know a signal $x(t)$ over an infinite duration. If we assume the function $x(t)$ to be periodic, and of period T , we can calculate its spectrum by :

$$X(f) = \frac{1}{T} \int_{-\frac{T}{2}}^{+\frac{T}{2}} x(t) e^{-2i\pi ft} dt$$

Which already brings us a little closer to the real case. But with the assumptions of periodicity, beware of side effects ! You should know that an abrupt truncation will cause noise over the entire frequency range which will have no meaning (see Figure 21.7).

In practice, the signals are sampled and the discrete TF is used.

21.4.2 The discrete Fourier transform

In digital signal processing, the function $x(t)$ is sampled at the frequency f_e , called “ sampling frequency”, and the function $x(t)$ becomes a set of values x_n . In this case, if the point $n = 0$ is the origin of the times, the point n is located at the time $t_n = nt$ with $t = \frac{1}{f_e}$

Note that the frequency f_e must be chosen in such a way (Shannon’s theorem) that there are no frequencies greater than $f_e/2$ in the original signal $x(t)$. In practice, when digitizing by CAD (Analog / Digital Converter), an anti-aliasing filter is installed in the hardware to avoid this problem.

The discrete Fourier transform is then expressed by :

$$X_k = \frac{1}{N} \sum_{n=0}^{N-1} x_n e^{-2i\pi \frac{nk}{N}}$$

X_k is then the spectrum of the waveform x_n . As the spectrum is discrete, we also speak of a line spectrum.

The inverse transform is of course :

$$x_n = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} X_k e^{-2i\pi \frac{nk}{N}}$$

21.4.3 The " Fast Fourier Transform "

It is a fast algorithm for computing the discrete Fourier transform. The one used in RCLs is of the " Cooley-Tukey " type named after its inventors. It uses the so-called " butterfly " algorithm and remains an extremely fast FFT. Its only defect is to operate on a number of point which is of the form $N = 2^m$, with m integer.

This algorithm was used to write the deconvolution subroutine used in the continuous calibration program (voir 20.4).

21.4.3.1 Negative and positive frequencies

In FFT calculation programs, we generally introduce an input-output argument **Sn** which is a complex array of dimension N. As input it corresponds to a waveform (which can be complex) \mathbf{x}_n , and as output it corresponds to the complex spectrum \mathbf{X}_k .

The lines of the output spectrum are generally classified according to the figure 21.5 below :

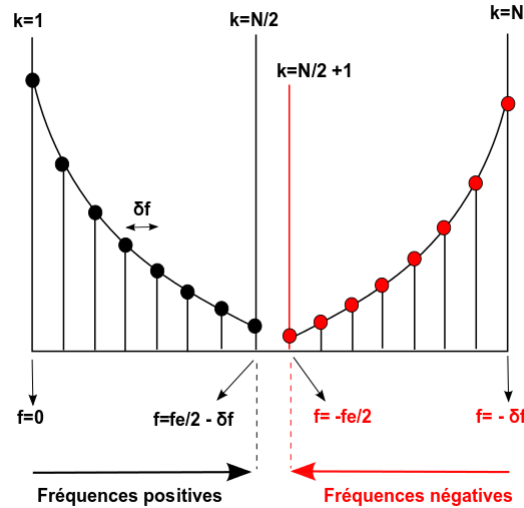


FIG. 21.5: Order of the lines of the spectrum at the output of FFT

21.4.3.2 Some useful relationships

δt : sampling period, δf : spectral resolution, T : Length of the window of N points.

$$t = \frac{1}{f_e} = \frac{T}{N} \quad f = \frac{1}{T} = \frac{f_e}{N} \quad T = Nt = \frac{N}{f_e} f_{\max} = \frac{f_e}{2}$$

f_{\max} , the highest frequency of the spectrum, is called the Nyquist frequency.

21.4.3.3 Normalization test of an FFT

If we take a real function as an input signal, the spectrum must be symmetrical and conjugate, which means that the amplitude on the negative and positive frequencies must be the same, and the phases reversed. To check that an FFT is well normalized, it suffices to take as input signal a pure sine wave, of amplitude $a = 0.5$ as in the example of Figure 21.6 below (the amplitude of the spectrum is given in linear and logarithmic scale).

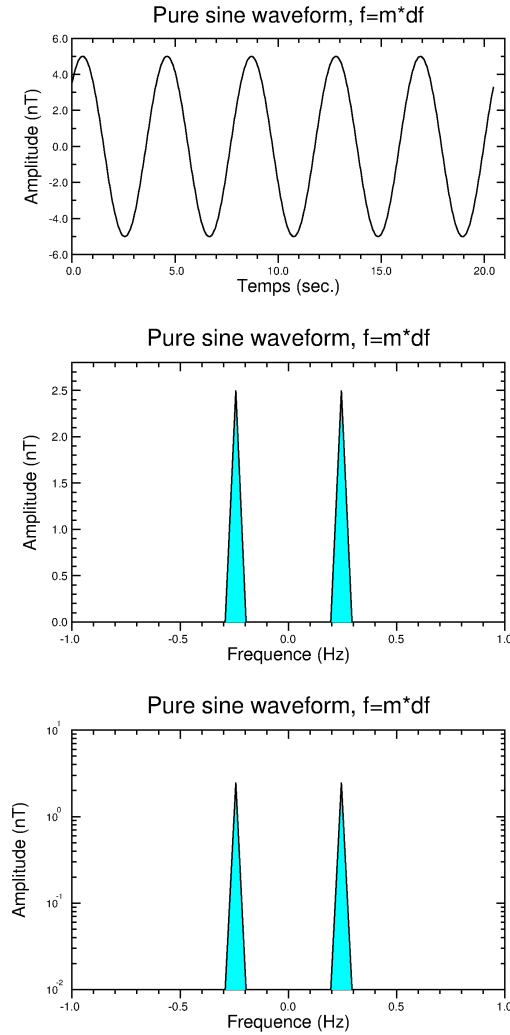


FIG. 21.6: Spectrum of a pure sine wave whose frequency is a multiple of δf

If we take the precaution of taking a frequency f_{test} which is multiple of δf , we respect the hypothesis according to which the signal is periodic (if we repeat it from one window to another, there is continuity).

In this case we obtain two pure lines, at $\pm\delta f$, of the same amplitude equal to $a/2$. We can verify that the energy in time as in frequency is well conserved : in time, the effective amplitude of a sinusoid is either an energy of $\frac{a}{\sqrt{2}}$ soit une énergie de $\frac{a^2}{2}$.

In frequency, we get $2\left(\frac{a}{2}\right)^2$ either good $\frac{a^2}{2}$. This is *Parseval' theorem* :

$$E = \int_{-\infty}^{+\infty} |x(t)|^2 dt = \int_{-\infty}^{+\infty} |X(f)|^2 df.$$

This theorem is sometimes very useful to verify the conservation of energy during complex processing. In this example, we have chosen $f_{\text{test}} = 5 \cdot \delta f = 5 \cdot 0.048828125 = 0.2441406$ Hz and an amplitude of 0.5 nT, and we can check on Figurespp hat we get 2 lines at 0.25 nT.

In this case the phase calculation (which can be verified) also gives the exact original value. This is how the despin software is designed : we use an FFT on a single line (the spin frequency) and

we take as sample length a number of points corresponding exactly to a spin period. We can thus calculate the amplitude and the phase of the spin sinusoid.

21.4.3.4 Edge effects

If we repeat the above test, with a frequency close to but not a multiple of δf , the signal periodicity hypothesis is no longer verified, and there are “breaks” of the signal from one window to another. The main consequence is that, on the one hand, the energy is distributed over all the frequencies, and that on the other hand the spectrum that we obtain is no longer exactly what we expected, as shown in Figure 21.7. However, in all cases the total energy of the signal is always conserved, which means that the amplitude of the lines is reduced.

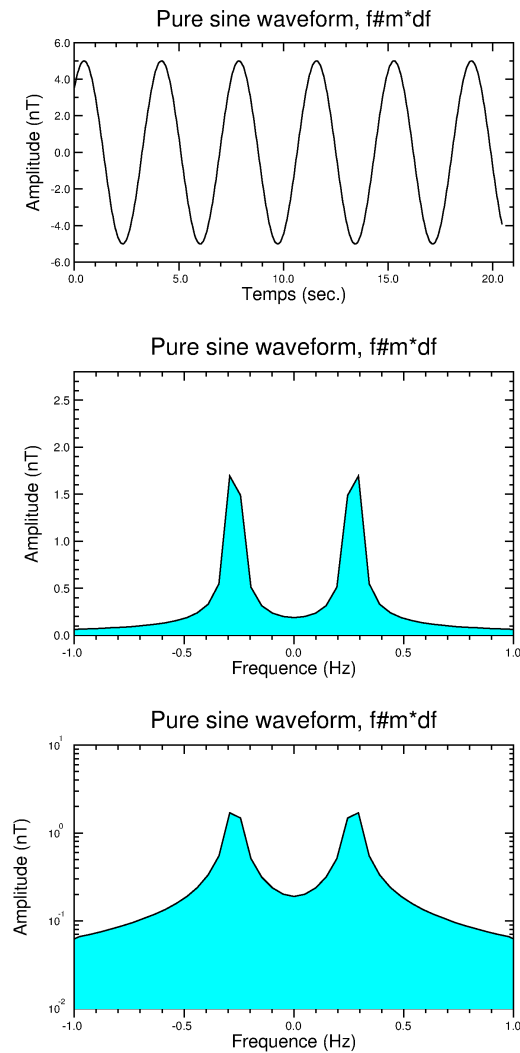


FIG. 21.7: Spectrum of a pure sine wave whose frequency IS NOT a multiple of δf

21.5 Usefulness of circular components

21.5.1 Définition

Recall that an ellipse describing a wave vector rotating to the right by example, can be decomposed into the sum of two circles one of which turns at left, and the other to the right, the latter having a higher radius. Which means that a plane elliptical wave can be decomposed into two circular waves right and left planes. The main interest of this decomposition is to be able to determine the polarization of a plane wave in the plane perpendicular to the magnetic field, in order to easily determine the mode (left or right). The introduction of these circular components was made by K. Kodera in his university thesis on the analysis of non-stationary geophysical signals (Thesis University of Paris 1976).

21.5.2 Passage in left and right circular components

Left and right spectra can be directly obtained by the complex FFT of the signal $S_{xy}(t) = x(t) + iy(t)$. the left mode is given by the negative frequencies, and the right mode by the positive frequencies. For more details, see [20], Robert, 1979. Figure 21.8 below summarizes the interest of this decomposition.

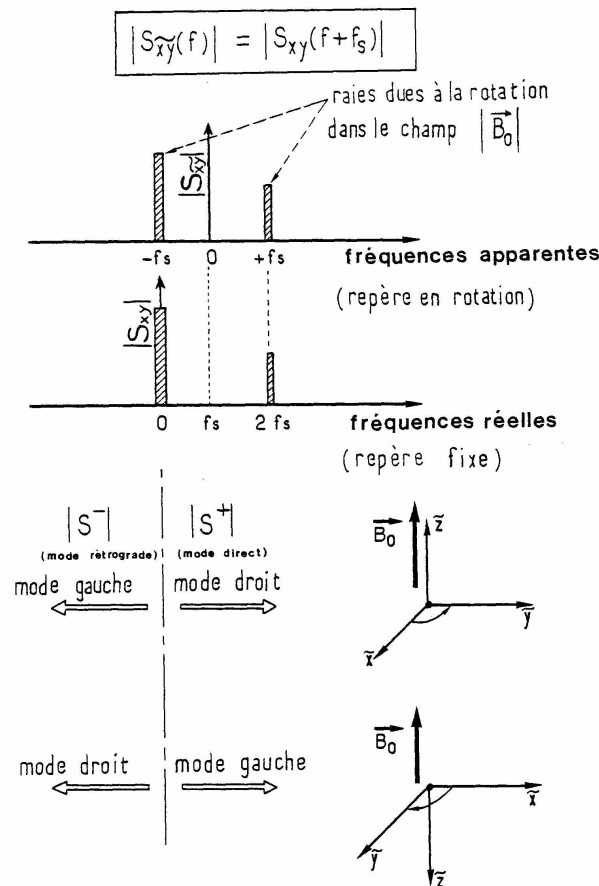


FIG. 21.8: Passage from the rotating reference to a fixed reference by the components left and right circular (Robert, 1971, [20]).

21.5.3 Calculation from the spectra

If we calculate the TF of $S_{xy}(t) = x(t) + iy(t)$ we obtain :

$$\mathbf{F}(S_{xy}(t)) = \mathbf{F}(x(t)) + i\mathbf{F}(y(t)) = X(f) + iY(f)$$

As the TF of a real function is symmetric, we can obtain the spectra left mode and right mode only from positive frequencies of the spectra $X(f)$ and $Y(f)$ by:

$$\begin{aligned} R_+(f) &= (X_+ + iY_+)/2 \\ G_+(f) &= (X_+^* + iY_+^*)/2 \end{aligned}$$

X_+^* et Y_+^* being the conjugates of the part of the positive frequencies of the spectra $X(f)$ and $Y(f)$. These formulas are used in the `RPC_visu_ave_spectrum.pro` procedure with the option "LR".

21.5.4 Consequences of the rotation on the spectrum

In the space of left and right circular coordinates, the Doppler effect of the satellite's rotation results in a simple frequency translation, as shown in Figure 21.9 below :

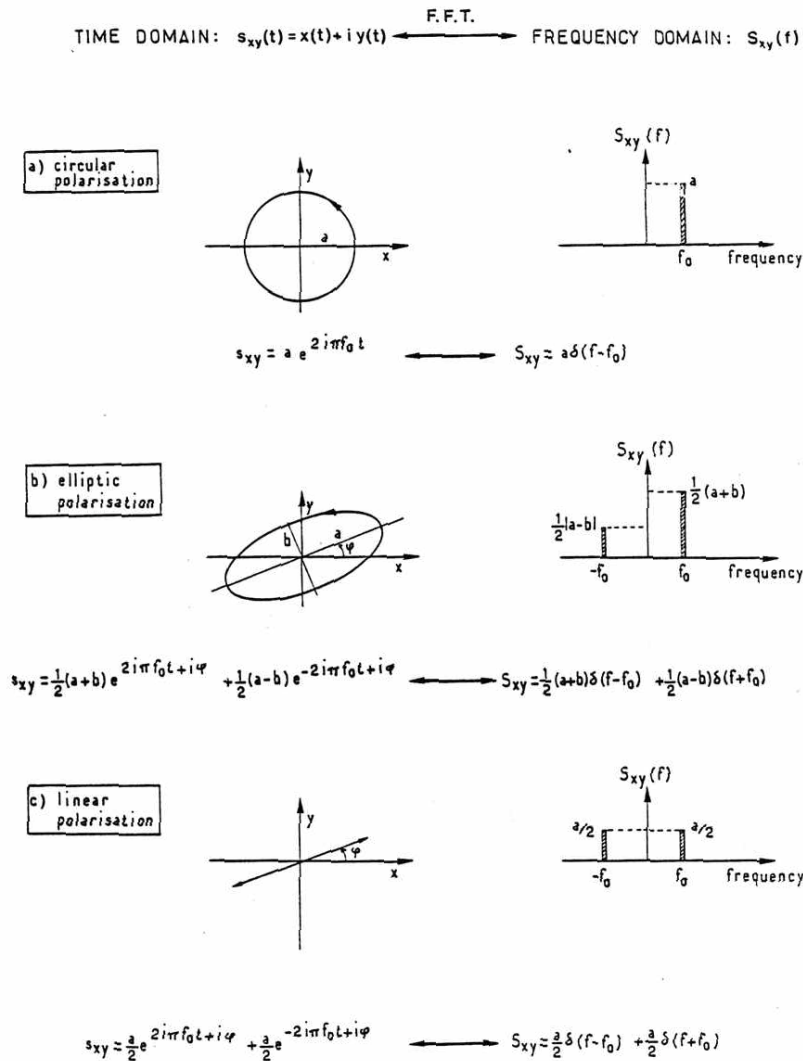


FIG. 21.9: Fourier transform of the complex signal $S_{xy} = x + iy$ associated with a monochromatic plane wave for 3 particular examples (Robert, 1971, [20]).

Chapter 22

CURLOMETER TECHNIQUE USED FOR CLUSTER

22.1 Introduction

To calculate rotational and divergence, it is necessary to have the 4 measurements of \vec{B}_{ij} and the 4 positions P_{ij} measured on the same timeline (i=1,3 j=1,4). It is therefore necessary to interpolate the values of the field, and to bring them back to the same common time. The same operation must be done with the position data, to synchronise them to the same timeline as the magnetic field.

This operation is done by the script **do_compute_curl_div_CLUFGM.sh**.

This script use the following commands:

RPC_get_data_CLUPOS_4sat to get the 4 positions from the CLUSTER data base
RPC_get_data_CLUFGM_4sat to get the 4 values of the DC magnetic field

Then we use the following commands to have the same timeline:

RPC_alitime_4_vectime to have the data time-aligned
RPC_compute_curl_div_4sat to do the computations according the method below.

22.2 Computation Method

The calculation method used for the estimation of curl(B) is that of the classical method of contour integrals on each face of the tetrahedron, by applying Ampere's law on each face:

$$\oint \vec{B}(M) \cdot d\vec{l} = \mu_0 I$$

To apply this formula to a tetrahedron, consider the face formed by vertices (i,j,k). Following the linearity assumption of B, the field between the spacecraft i and j can be expressed by:

$$\vec{B}_{ij} = \vec{B}_i + (\vec{B}_j - \vec{B}_i)l/L_{ij}$$

where L_{ij} is the distance between spacecraft i and j.

So for the line ij we have:

$$\int_0^{L_{ij}} \vec{B}_{ij}(l) \cdot d\vec{l} = \vec{B}_j + (\vec{B}_j - \vec{B}_i) \int_0^{L_{ij}} l d\vec{l} / L_{ij} = (\vec{B}_i + \vec{B}_j) \cdot \vec{L}_{ij} / 2$$

By noting the result S_{ij} , and by doing the same thing for the 3 lines of the (i,j,k) triangle, we obtain:

$$\mu_0 \cdot I_{ijk} = S_{ij} + S_{jk} + S_{ki} \text{ and so we have } J_{ijk} = (S_{ij} + S_{jk} + S_{ki}) / (A_{ijk} \mu_0)$$

where A_{ijk} is the area of the (i,j,k) triangle.

To calculate the density vector \vec{J} , one chooses 3 faces among the 4, and therefore we obtain 3 components of \vec{J} in a non-orthogonal coordinate system. If the tetrahedron is not flat, one carries out the passage in an orthogonal frame by a classical method (see 21.3.1) and one finally obtains the vector \vec{J} in the initial coordinate system.

We thus can obtain 4 possible values for the estimation of the rotational gradient. In practice, when the tetrahedron is not degenerated, these 4 values are extremely close, and we use as final result the average of these 4 estimations.

To compute $\text{div}(\vec{B})$ we use the divergence law, or Green-Ostrogradski law, as:

$$\iiint_V \vec{\nabla} \cdot \vec{B} \, dV = \oint_{\partial V} \vec{B} \cdot d\vec{S}$$

Noting now $\vec{B}_{ijk} = (\vec{B}_i + \vec{B}_j + \vec{B}_k) / 3$

In the same way we have shown that $\int_0^{L_{ij}} \vec{B}_{ij}(l) \cdot d\vec{l} = (\vec{B}_i + \vec{B}_j) \cdot \vec{L}_{ij} / 2$ we can show that on the face of the (i,j,k) triangle we have $\int \vec{B} \cdot d\vec{S} = \vec{B}_{ijk} \cdot \vec{N}_{ijk}$, where \vec{N}_{ijk} is the output normal to the (i,j,k) face.

By dividing by the volume of the tetrahedron, we obtain the contribution of the divergence on each face;

$$D_{ijk} = \vec{B}_{ijk} \cdot \vec{N}_{ijk} / V$$

and finally the total divergence

$$\text{Div}(\vec{B}) = D_{123} + D_{134} + D_{142} + D_{432}$$

This method has been used extensively in all of the many curlometer studies applied to CLUSTER's FGM data. The analysis method to use multipoint magnetometer data appeared a long time before Cluster launch as well as the influence of the shape of the tetrahedron on the accuracy of the measurement of currents, in particular in a recent paper [14].

Another formulation to compute Curl and Div was developed by G. Chanteur based on barycentric coordinates. This elegant method estimate the matrix of gradients, the diagonal terms giving the divergence, while the anti-diagonal terms are used to calculate the rotational gradients. Linear assumption of the magnetic field is done for the two methods.

Chapter 23

USE OF RPC ON OTHER MISSIONS

23.1 ICI-3 and CUSP Rocket

Search-coils have already been onboard Rocket, with, or without success. The short time duration of the flight leads to be lucky or not to see an interesting event. But the purpose here is not science, but showing that the RPC software can be applied to anything of time series data.

So we show here in fig. 23.1 and 23.2 an example of spectrogram made by RPC on the SCM data of two different rockets.

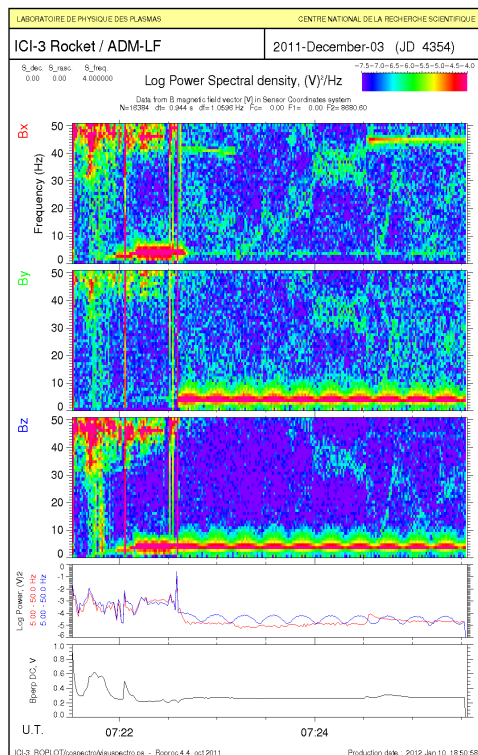


FIG. 23.1: *RPC commands used on ICI-3 Rocket data*

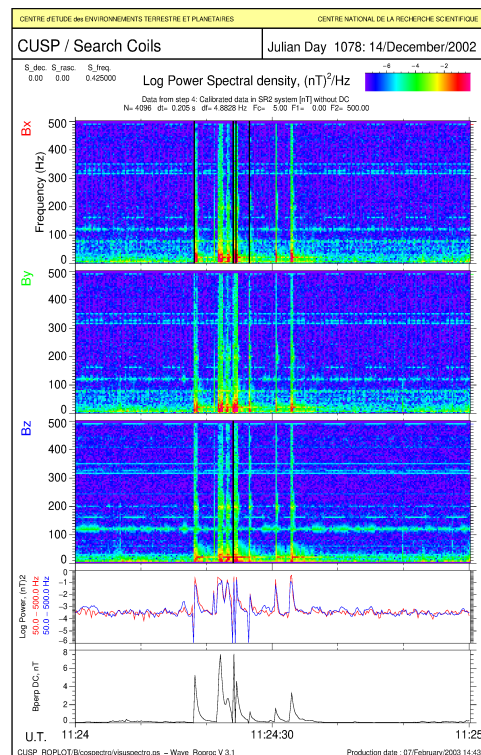


FIG. 23.2: *RPC commands used on CUSP Rocket data*

23.2 CASSINI magnetometer

One show here an example of RPC use on magnetometer data of CASSINI probe, around Jupiter. Thanks to Lina Hadid for providing this data. The RFF vectime file has been created from a flat file by the procedure `RPC_flat_to_rff`.

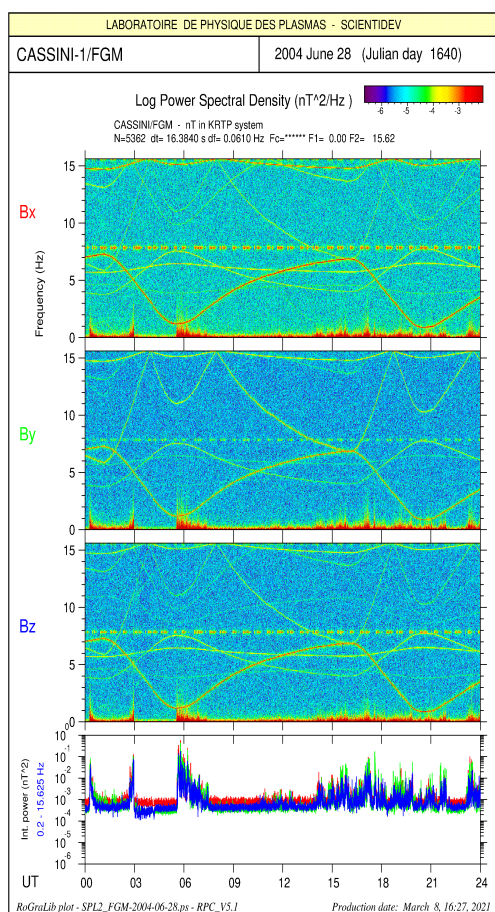


FIG. 23.3: *RPC commands used on CASSINI magnetometer*

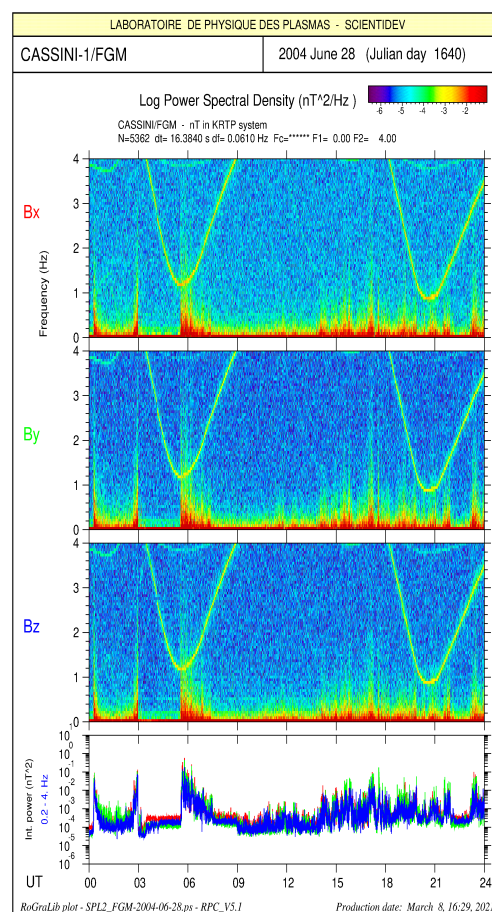


FIG. 23.4: *CASSINI magnetometer (zoom)*

23.3 THEMIS / SCM

23.3.1 Spectrogram at large scale

RPC commands are used to process the burst mode (128 Hz) of the SCM data of Themis #d.

Figure 23.5 show a spectrogram in GSE system of these data, during an interesting time period where we can observe a whistler around 12:30. Since it is an interesting event, a polarization study has been made, and results displayed on next sections.

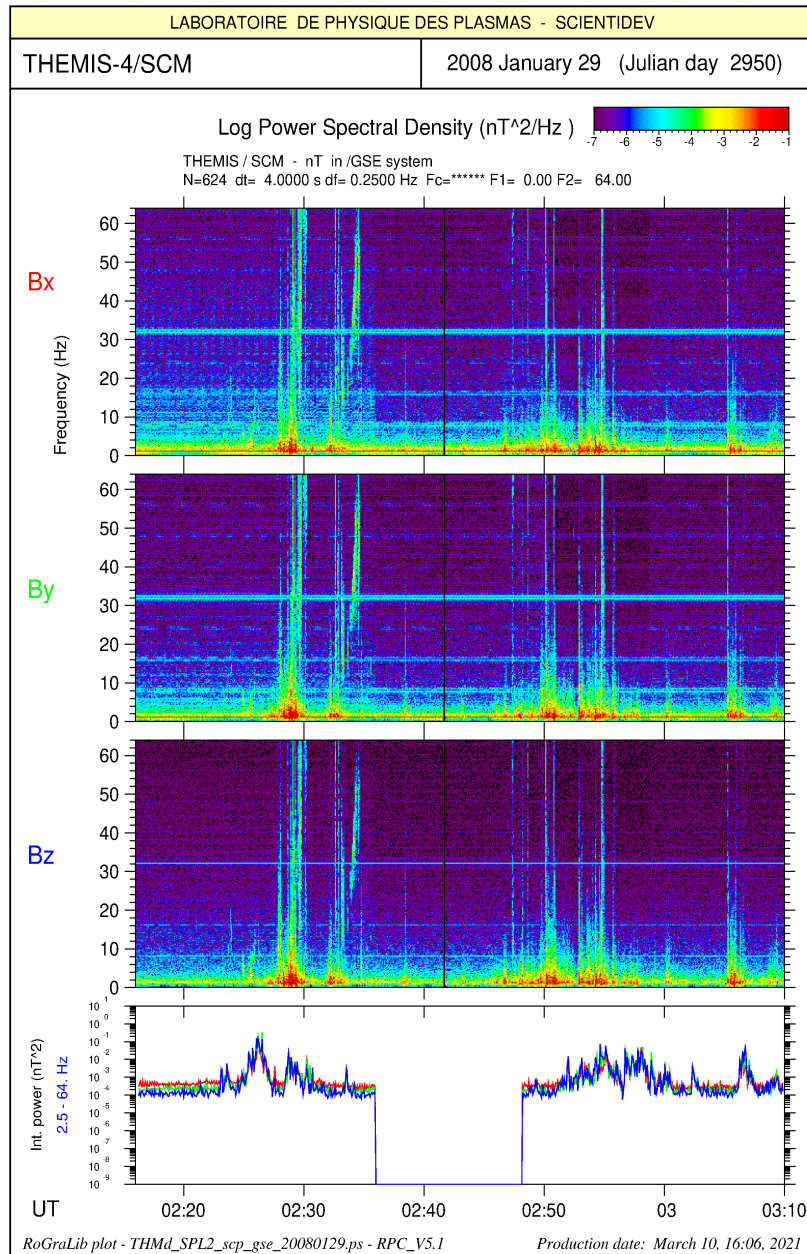


FIG. 23.5: Example of RPC application on THEMIS/SCM. A Whistler structure is detected a time $\sim 02:35$

23.3.2 Polarisation of the Whistler

The following figures show a spectrogram of the same data of fig 23.5 but zoomed on the Whistler event. In order to identify wave polarization the initial spectrogram in GSE system (fig. 23.6) has been computed in MFA system (fig 23.7), with Left, Right, Z components. instead of X,Y,Z.

A right mode is identified, but with a slight component on the Z axes. So propagation is not perfectly parallel to the DC magnetic field, as we will see with most precision in next section.

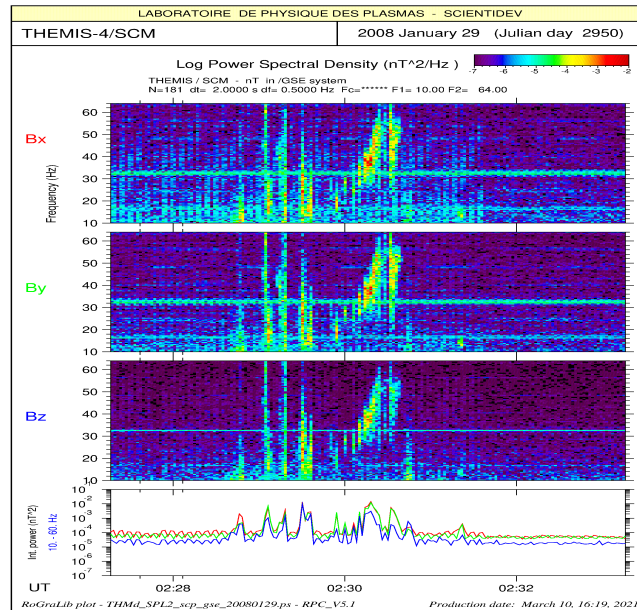


FIG. 23.6: Zoom on the Whistler structure in GSE system

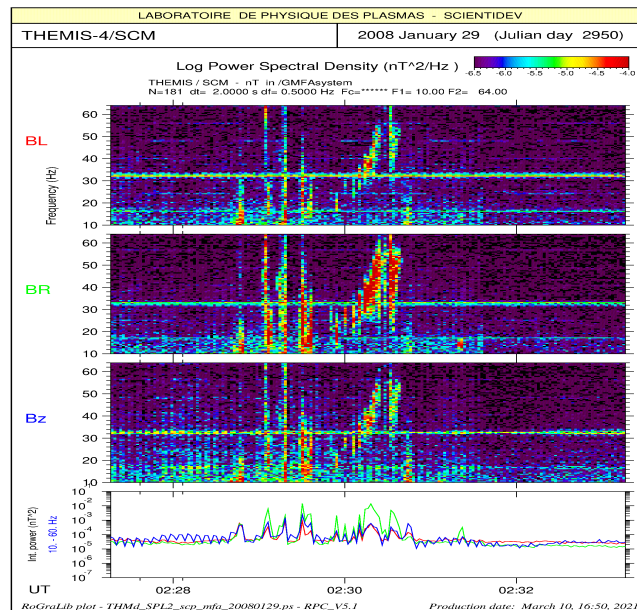


FIG. 23.7: Zoom on the Whistler structure in MFA system

23.3.3 Polarization parameters

Fig. 23.8 show the polarisation parameters of the whistler. Interpretation is following:

eccentricity : green, so ~ 0.5 , \Rightarrow circular polarization.
 theta K : dark purple, so $\sim 10^\circ$, \Rightarrow **right circular polarization with K parallel to B_0** .
 theta Major axis : green, so $\sim 90^\circ$, \Rightarrow perpendicular to B_0 , consistent with the direction of K .

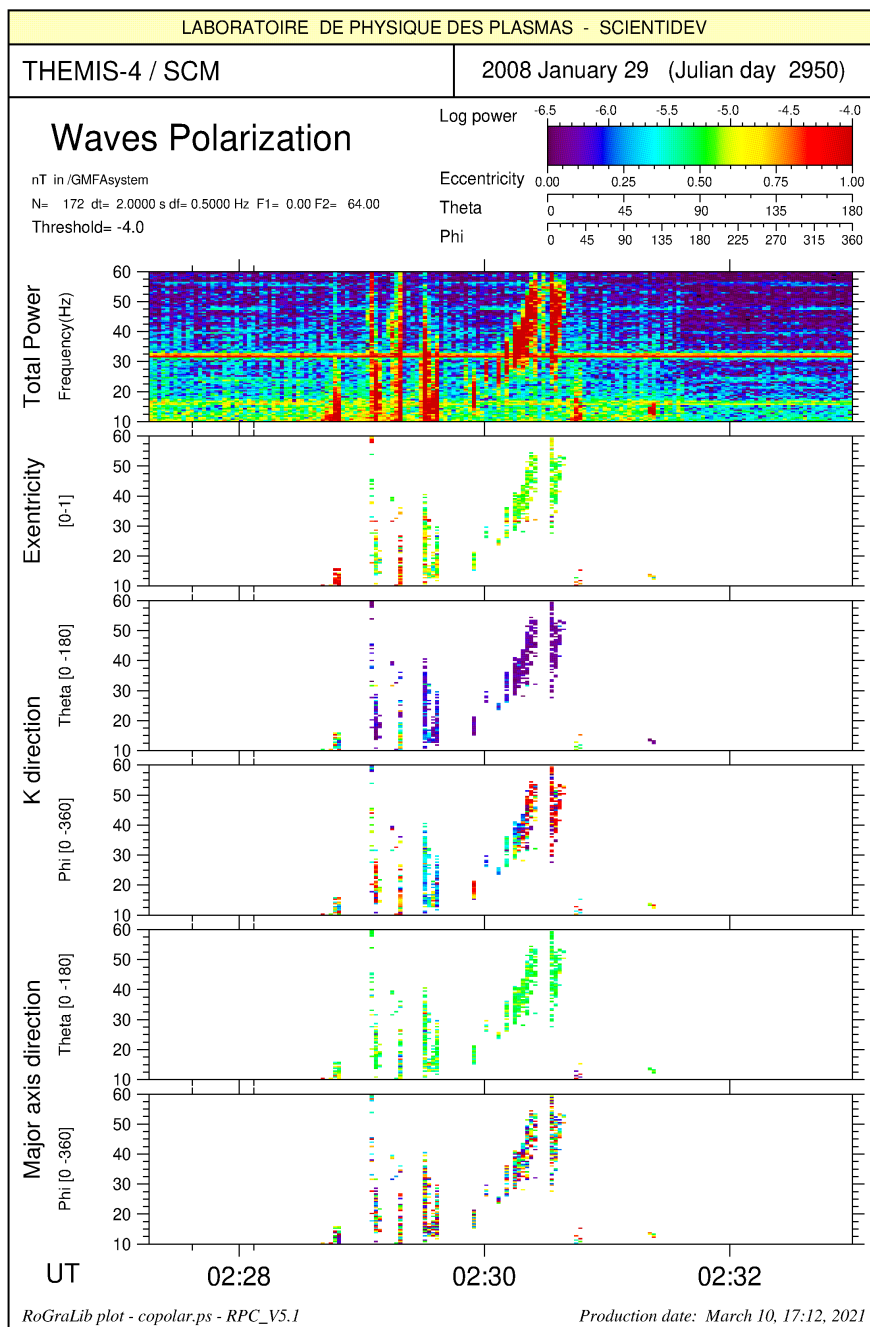


FIG. 23.8: Polarization parameters of the Whistler

23.3.4 THEMIS position position

The RFF vectime file position has been created, as all other, from a THEMIS flat file by the procedure `RPC_flat_to_rff`.

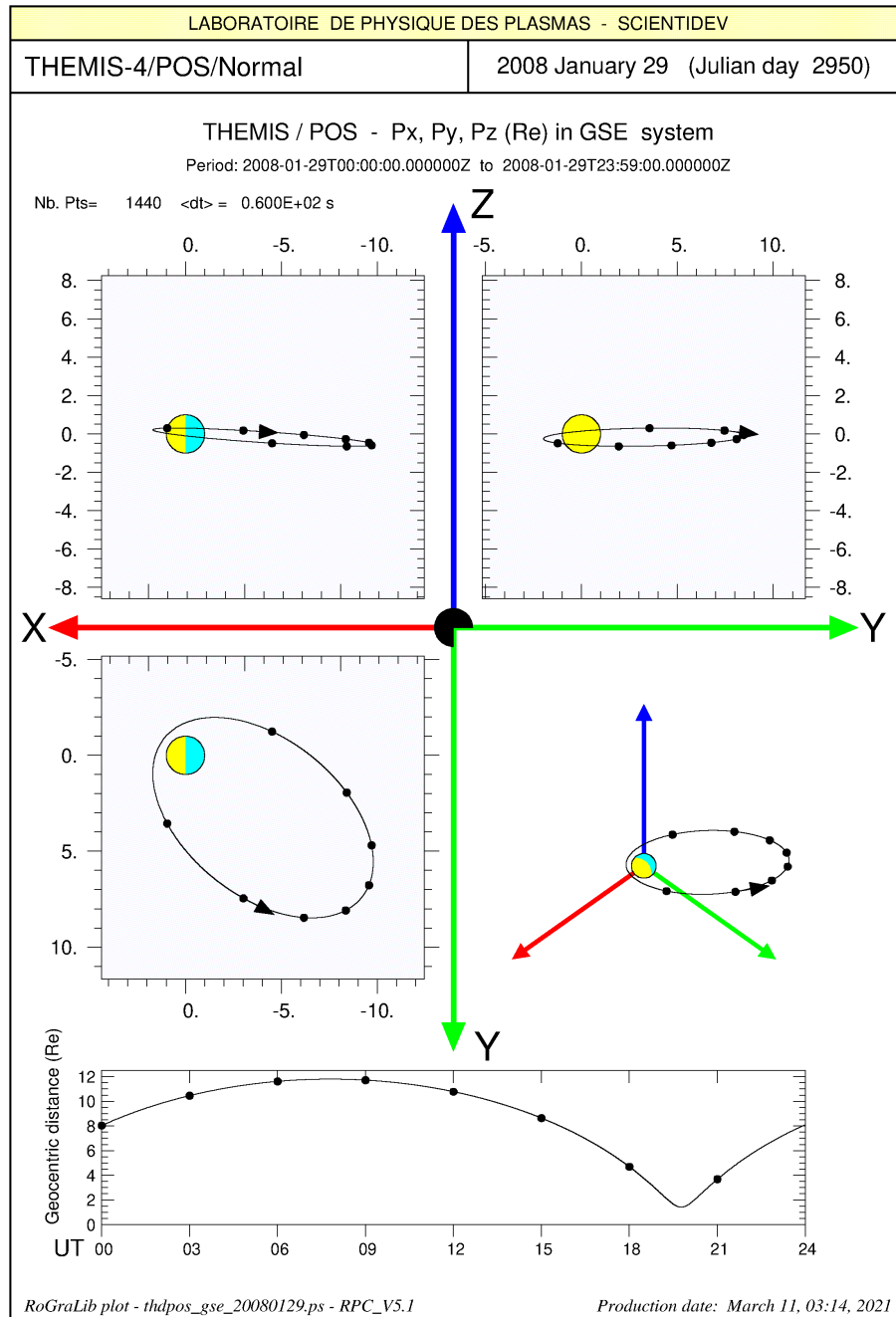


FIG. 23.9: Position of THM-d during events of fig 23.8

23.4 MMS / SCM

23.4.1 Spectrogram at large scale

RPC commands are used to process the burst mode (8192 Hz) of the MMS data of S/Cs #1.

Figure 23.10 show a spectrogram in GSE system of these data, during an interesting time period where we can observe a quasi monochromatic wave around 13:05. Since it is an interesting event, a polarization study has been made, and results displayed on next sections.

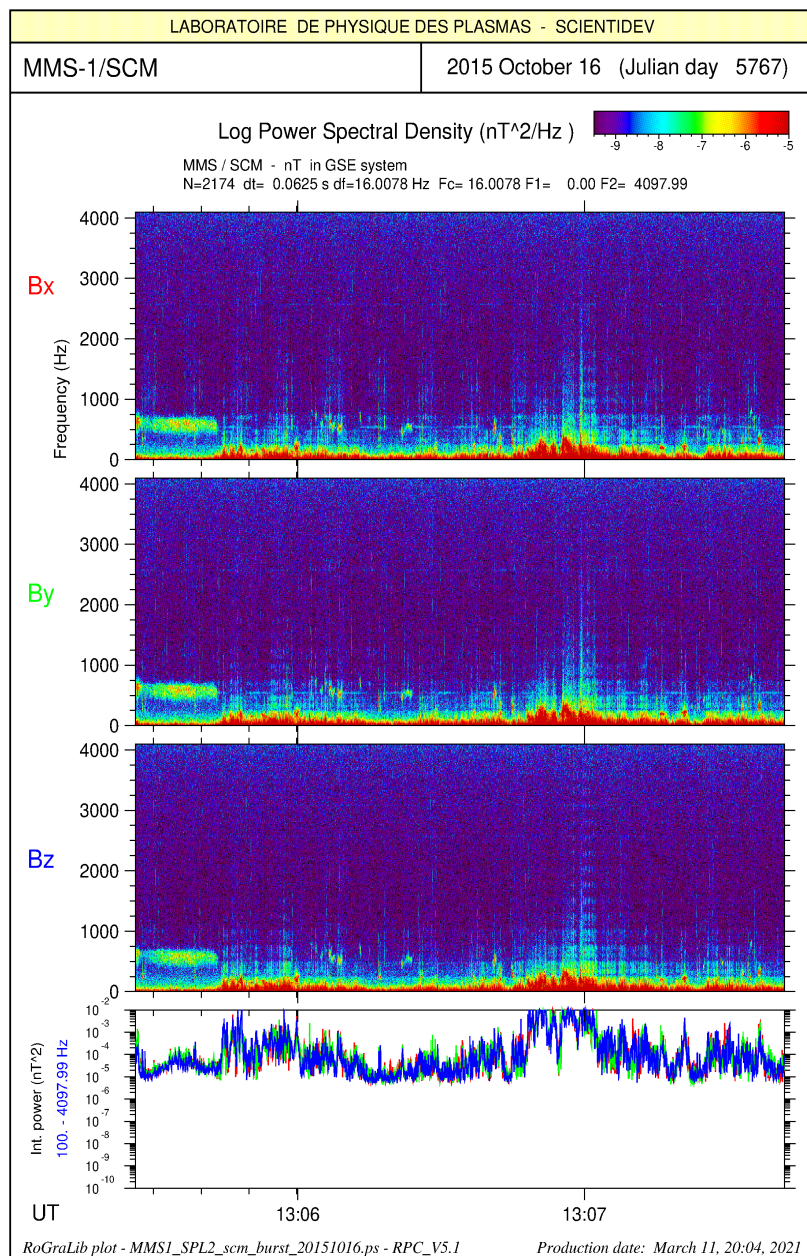


FIG. 23.10: Example of RPC application on MMS/SCM data. Aa quasi monochromatic wave is detected a time $\sim 13:05$

23.4.2 Polarisation of the event

The following figures show a spectrogram of the same data of fig 23.10 but zoomed on the Whistler event. In order to identify wave polarization the initial spectrogram in GSE system (fig. 23.11) has been computed in MFA system (fig 23.12), with Left, Right, Z components. instead of X,Y,Z.

A right mode is identified, but with an important component on the Z axes. So propagation is not perfectly parallel to the DC magnetic field, as we will see with most precision in next section.

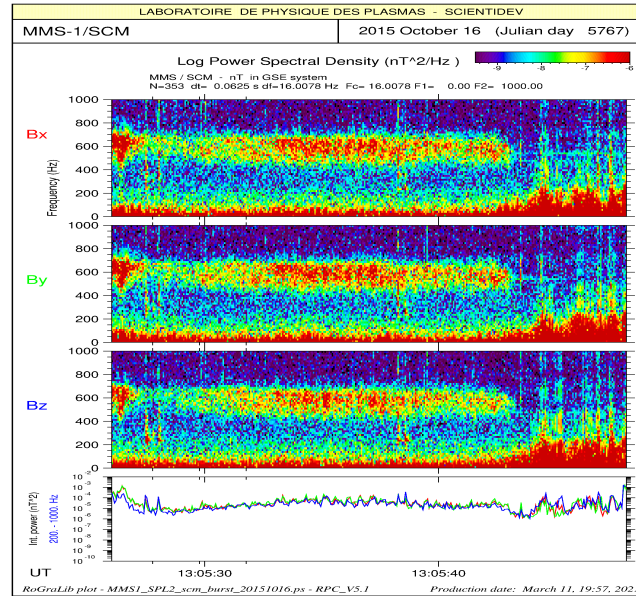
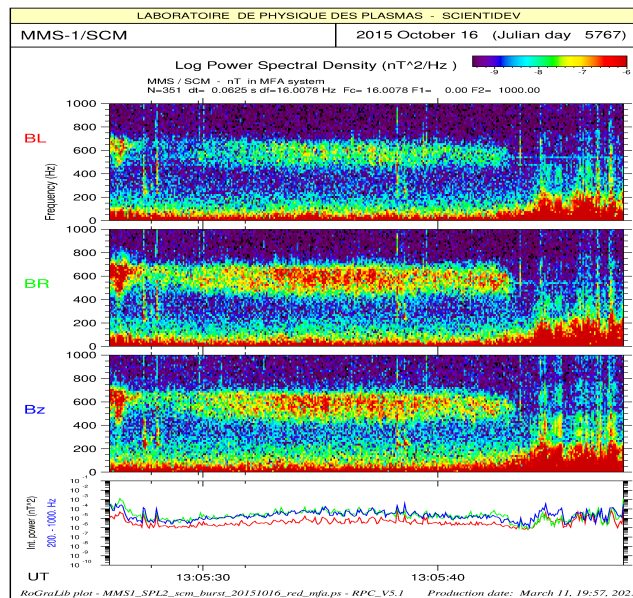


FIG. 23.11: Zoom on the Whistler structure in GSE system



cm

FIG. 23.12: Zoom on the Whistler structure in MFA system

23.4.3 Polarization parameters

Fig. 23.8 show the polarisation parameters of the whistler. Interpretation is following:

eccentricity :	green-yellow-red,	so ~ 0.7 ,	\Rightarrow circular polarization.
theta K :	cyan-blue,	so $\sim 45^\circ$,	\Rightarrow <i>right circular polarization with oblique propagation by respect to B_0.</i>
theta Major axis :	green-blue,	so $\sim 45^\circ$,	\Rightarrow perpendicular to B_0 , consistent with the direction of K.

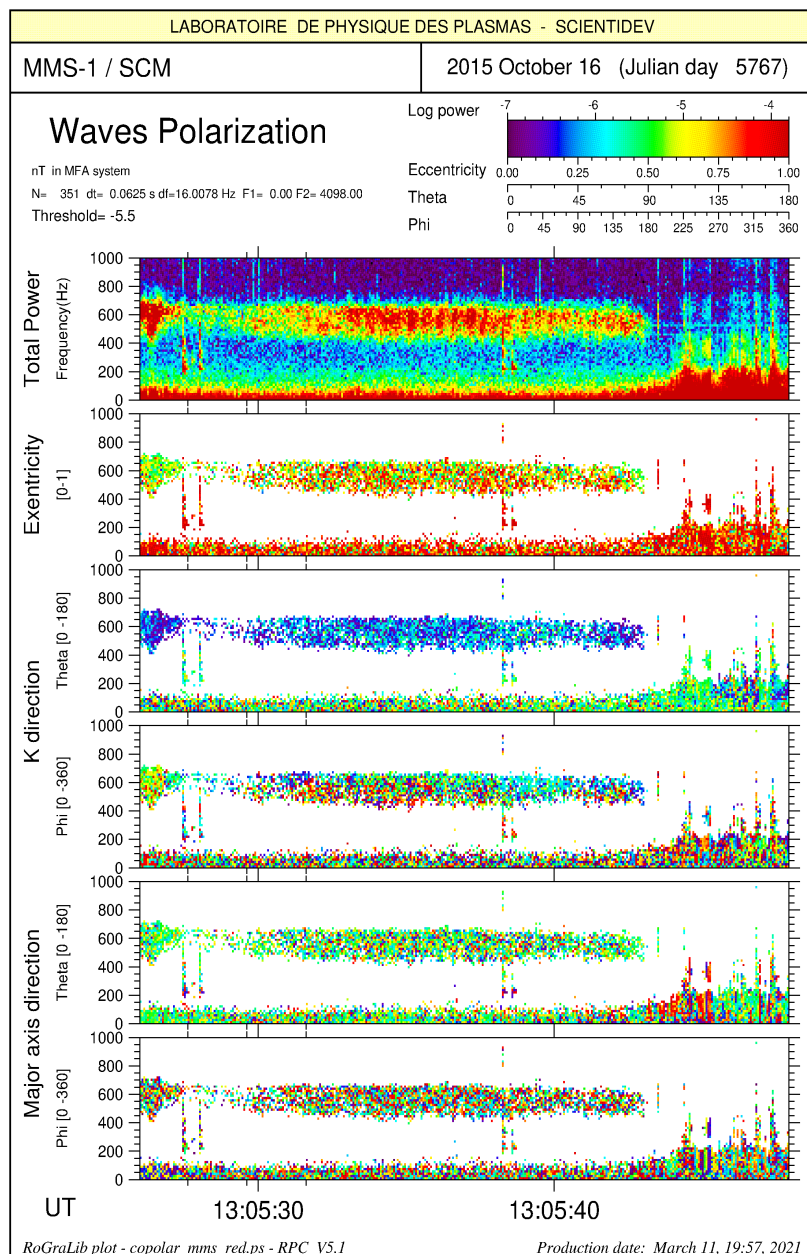


FIG. 23.13: Polarization parameters of the event

23.4.4 MMS position

The RFF vectime file position has been created, as all other, from a MMS flat file by the procedure `RPC_flat_to_rff`.

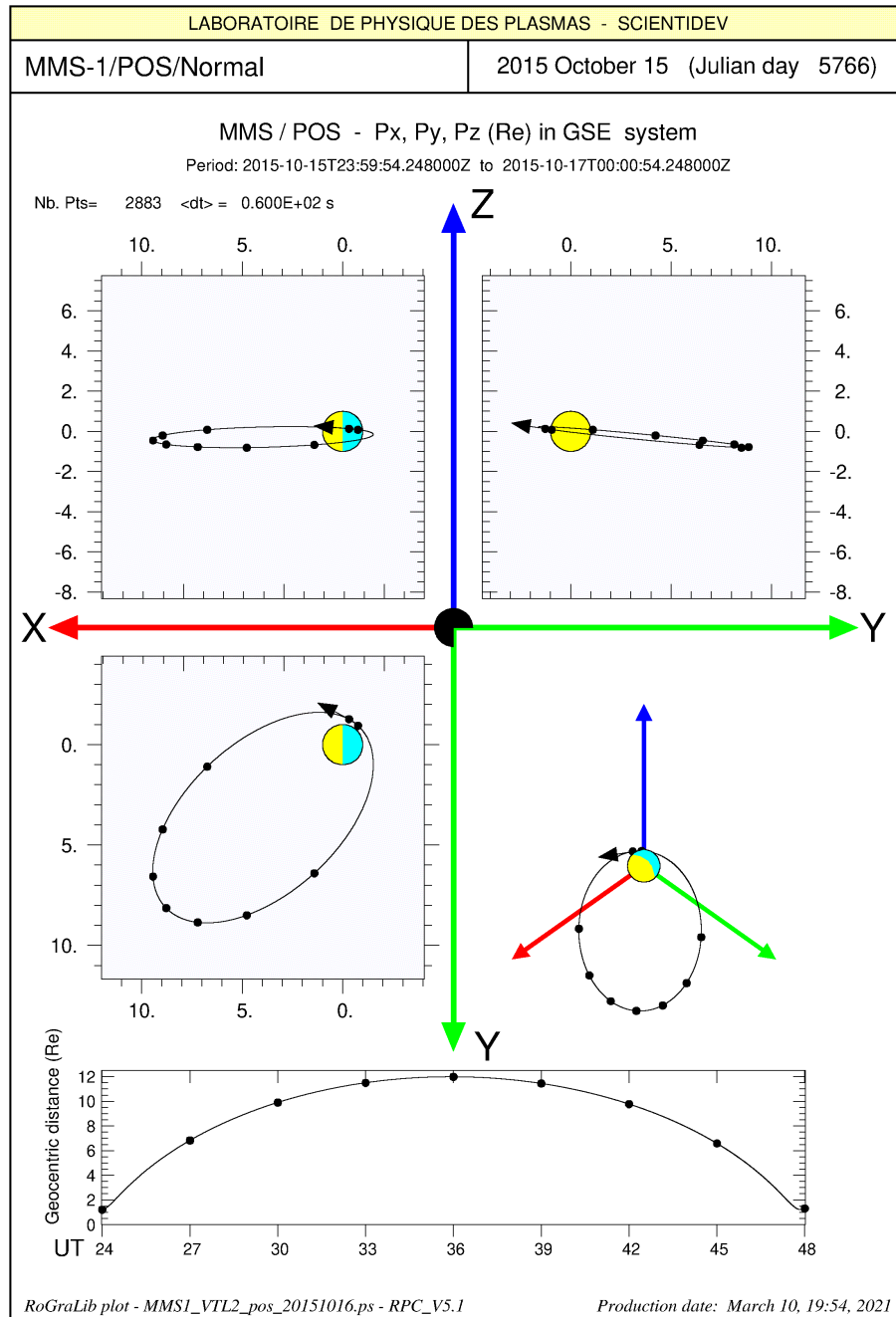


FIG. 23.14: Position of MMS-1 during events of fig 23.10

Chapter 24

SUMMARY OF F90 PROGRAMS AND LIBRARIES

24.1 Programs F90

24.1.1 Common programs

- **add_DxDy_to_BxBy.f90** nb. of lines : 187

program	add_DxDy_to_BxBy	create file 3 components with Bx+Dx, By+Dy,Bz	P. Robert, ScientiDev, Dec 2020
---------	------------------	---	---------------------------------

- **alitime_4_vectime.f90** nb. of lines : 268

program	alitime_4_vectime	aligne en temps 4 fichiers vectime	P. Robert, Scientidev, Janvier 2021
---------	-------------------	------------------------------------	-------------------------------------

- **change_coordinate_system.f90** nb. of lines : 548 nb. of subroutine: 10

program	change_coordinate_system		change coordinate system of a VT file	P. Robert, LPP, May 2011
subroutine	test	(rep,allowed)		P. Robert, CRPE, 1992
subroutine	t_sm_to_gse	(x1,y1,z1,x2,y2,z2)		P. Robert, CRPE, 1992
subroutine	t_gse_to_sm	(x1,y1,z1,x2,y2,z2)		P. Robert, CRPE, 1992
subroutine	t_mg_to_gse	(x1,y1,z1,x2,y2,z2)		P. Robert, CRPE, 1992
subroutine	t_gse_to_mag	(x1,y1,z1,x2,y2,z2)		P. Robert, CRPE, 1992
subroutine	t_sm_to_gsq	(x1,y1,z1,x2,y2,z2)		P. Robert, CRPE, 1992
subroutine	t_gsq_to_sm	(x1,y1,z1,x2,y2,z2)		P. Robert, CRPE, 1992
subroutine	t_mag_to_gsq	(x1,y1,z1,x2,y2,z2)		P. Robert, CRPE, 1992
subroutine	t_gsq_to_mag	(x1,y1,z1,x2,y2,z2)		P. Robert, CRPE, 1992

- **check_rff.f90** nb. of lines : 39

program	check_rff	check RFF WaveForm and VecTime files	P. Robert, LPP, Mar. 2011
---------	-----------	--------------------------------------	---------------------------

- **clean_rff.f90** nb. of lines : 54

program	clean_rff	Read & clean a RFF WaveForm or VecTime files	P. Robert, LPP, Mars 2011
---------	-----------	--	---------------------------

- **compute_curl_div_4sat.f90** nb. of lines : 536

program	compute_curl_div_4sat	calcul du rotationnel et de la divergence	P. Robert, CETP, Juin 2001
---------	-----------------------	---	----------------------------

- **compute_sat_orbit_param.f90** nb. of lines : 248

program	compute_sat_orbit_param	compute sat. orbital parameters	P. Robert,Scientidev, Jan 2021
---------	-------------------------	---------------------------------	--------------------------------

• **compute_sat_trajectory.f90** nb. of lines : 388 nb. of subroutine: 1

program	compute_sat_trajectory	calcul de la trajectoire d'un satellite autour de la terre.	P. Robert, CRPE, 1985
subroutine	init_metadata		P. Robert, CRPE, 1985

• **co_IGRF_field.f90** nb. of lines : 308 nb. of subroutine: 5

program	co_IGRF_field		compute_IGRF magnetic field from the 13th generation IGRF	P. Robert, ScientiDev, Nov 2021 (only this routine)
subroutine	cp_leap_year	(iyear,ileap)	compute_leap_year with ileap=1 for leap year, 0 if not	P. Robert, CRPE, 1992
subroutine	read_date	(iyear,imonth,iday,nday,ndayiny)	read_date from input and check validity	P. Robert, CRPE, 1992, rev november 2021
subroutine	read_coordinate	(x,y,z,r,tet,phi)	read coordinate values from input	P. Robert, CRPE, 2002
subroutine	t_sph_to_car	(r,teta,phi,x,y,z)	transforms_sph_to_car: SPH -> CAR system	P. Robert, CRPE, 1992
subroutine	t_car_to_sph	(x,y,z,r,teta,phi)	transforms_car_to_sph: CAR -> SPH system	P. Robert, CRPE, 1992

• **create_simulated_data.f90** nb. of lines : 417 nb. of subroutine: 1

program	create_simulated_data	create simulated data of an imaginary search-coil, mag or pos	P. Robert, ScientiDev, Feb. 2020
subroutine	set_common_param		P. Robert, ScientiDev, Feb. 2020

• **current_tube.f90** nb. of lines : 464 nb. of subroutine: 4

program	current_tube		P. Robert, ScientiDev, Feb. 2020
subroutine	cal_b_tube	(xs,ys,zs,bxs,bys,bzs,bmod)	P. Robert, ScientiDev, Feb. 2020
subroutine	vdhxyb	(x,y,b,bv,bd,bh,v,d,h)	P. Robert, ScientiDev, Feb. 2020
subroutine	xybvhd	(v,d,h,bv,bd,bh,x,y,b)	P. Robert, ScientiDev, Feb. 2020
subroutine	sincosp	(x,y,z,st,ct,sp,cp)	P. Robert, ScientiDev, Feb. 2020

• **diff_rff.f90** nb. of lines : 744 nb. of subroutine: 1

program	diff_rff	create RFF file containing difference between 2 rff files	P. Robert, LPP, MAR. 2011
subroutine	diff_meta_data		P. Robert, LPP, MAR. 2011

• **dir_pretty_tree.f90** nb. of lines : 199

program	dir_size_pretty_tree	Compute & print properly directory tree	P. Robert, LPP, 1990
---------	----------------------	---	----------------------

• **dir_properties_pretty_tree.f90** nb. of lines : 253

program	dir_properties_pretty_tree	Compute & print properly directory tree and properties	P. ROBERT, CETP, 1990
---------	----------------------------	--	-----------------------

• **dir_size_pretty_tree.f90** nb. of lines : 244

program	dir_size_pretty_tree	Compute & print properly directory tree and size of each	P. Robert, LPP, 1990
---------	----------------------	--	----------------------

• **flat_to_rff.f90** nb. of lines : 281 nb. of subroutine: 1

program	flat_to_rff	convert a flat file to RFF	P. Robert, ScientiDev, Feb. 2020
subroutine	set_common_param		P. Robert, ScientiDev, Feb. 2020

• **give_dip_dir.f** nb. of lines : 841 nb. of subroutine: 3

program	give_dip_dir		give dipole direction	P. Robert, ScientiDev, Feb. 2020
subroutine	RECALC_08	(IYEAR,IDAY,IHOUR,MIN,ISEC,VGSEX,VGSEY,VGSEZ)		N. Tsyganenko 2008
subroutine	SUN_08	(IYEAR,IDAY,IHOUR,MIN,ISEC,GST,SLONG,SRASN,SDEC)		N. Tsyganenko 2008

• **reduce_time_rff.f90** nb. of lines : 229

program	reduce_time_rff	extract a time period into a WF or VT RFF file	P. Robert, LPP, 2011 May 27
---------	-----------------	--	-----------------------------

• **reduce_time_vectime.f90** nb. of lines : 238

program	reduce_time_vectime	extract a time period into a WF or VT RFF file	P. Robert, LPP, 2011 May 27
---------	---------------------	--	-----------------------------

• **spectro_to_polar.f90** nb. of lines : 808 nb. of subroutine: 8

program	spectro_to_polar		Read a SP.rff file and compute polar.resu containing all polarisation parameters	P. Robert, LPP, 2011 Mar 07
subroutine	cpolar	(sx,sy,sz,ptot,a,b,rkx,rky,rkz,ax,ay,az,phase)		P. Robert, CETP, Nov. 2001
subroutine	CALAIFI	(SX,SY,SZ,AII,FI)		P. Robert, CRPE, 1980
subroutine	ELLIPSE	(AII,FI,PHA,AA,BB,EXC,GXI,GZI)		P. Robert, CRPE, 1980
subroutine	POLAIR	(V,VMOD,TET,PHI,ID)		P. Robert, CRPE, 1980
subroutine	strlen	(str,nc)	calcul de la longueur utile d'une chaîne de caractères	P. Robert, CETP, 1998
subroutine	wricosfilhead	(ifc,	écriture sur fichier cospectro.resu du header du fichier	P. Robert, CETP, Jan. 2001
subroutine	wri6spectro	(ifc,dtspe,dfspe,ns1,ns2,nf1,nf2,datiso,	écriture sur fichier cospectro.resu des 3 spectres	P. Robert, CETP, Jan. 2001

• **spectro_xyz_to_lrz.f90** nb. of lines : 164

program	spectro_xyz_to_lrz		Transform a SP.rff in xyz cartesian to Left Right Z components	P. Robert, LPP, Feb. 2015
---------	--------------------	--	--	---------------------------

• **test_kepler_lib.f90** nb. of lines : 136

program	test_kepler_lib		calcul de la trajectoire de la terre autour du soleil	P. Robert, CRPE, 1985,
---------	-----------------	--	---	------------------------

• **vectime_to_mva.f90** nb. of lines : 183

program	vectime_to_mva		transform a VecTime VTL2 in any system into MVA system	P. Robert, ScientiDev, Oct. 2020,
---------	----------------	--	--	-----------------------------------

• **vectime_to_spectro.f90** nb. of lines : 1277 nb. of subroutine: 3

program	vectime_to_spectro		compute spectra and create spectrogram RFF file.	P. Robert, Dec. 2012
subroutine	read_N_cont_vectors	(N_Vectors)		P. Robert, Dec. 2012
subroutine	r_rff_data_block_fit	(in_file_unit,nor_index,ext_index,fit_block)		P. Robert, Dec. 2012
subroutine	read_input	(in_file_name,ou_file_name,N_Kern,N_Shift,apod)		P. Robert, Dec. 2012

• **vectime_to_mfa.f90** nb. of lines : 366

program	vectime_to_mfa		transform a VecTime VTL2 in GSE system into mfa system	P. Robert, ScientiDev, Oct. 2020,
---------	----------------	--	--	-----------------------------------

• **visu_2_vectime_3D.f90** nb. of lines : 1053 nb. of subroutine: 4

program	visu_2_vectime_3D		Visualize on 3D planes 4 vectime.rff files	P. Robert, ScientiDev, Feb. 2021
subroutine	check_meta			P. Robert, ScientiDev, Feb. 2021
subroutine	load_data			P. Robert, ScientiDev, Feb. 2021
subroutine	plot_traj	(vectime,nbvec,i1,i2,sig1,sig2,indpo)		P. Robert, ScientiDev, Feb. 2021
subroutine	txyz_to_xycube	(x,y,z,xc,yc,zc)		P. Robert, ScientiDev, Feb. 2021

• **visu_2_vectime.f90** nb. of lines : 668 nb. of subroutine: 5

program	visu_2_vectime		Visualize 2 vectime.rff	P. Robert, ScientiDev, Feb. 2021
subroutine	check_meta			P. Robert, ScientiDev, Feb. 2021
subroutine	load_data			P. Robert, ScientiDev, Feb. 2021
subroutine	extend	(vect,nbvec)		P. Robert, ScientiDev, Feb. 2021
subroutine	read_VT_header	(ifc,VT,nbloc,mission,experi,mode,rep,dt,datiso1,dati		P. Robert, ScientiDev, Feb. 2021

• **visu_ave_spectrum.f90** nb. of lines : 615 nb. of subroutine: 5

program	visu_ave_spectrum		Visualize a mean spectrum from a SP.rff	P. Robert, ScientiDev, Feb. 2021
subroutine	co_ind			P. Robert, ScientiDev, Feb. 2021
subroutine	load_data			P. Robert, ScientiDev, Feb. 2021
subroutine	check_meta			P. Robert, ScientiDev, Feb. 2021
subroutine	plot_ave_spe			P. Robert, ScientiDev, Feb. 2021
subroutine	plot_tit_spectro_	(nbp,f1,f2,fc,fs,dt,df,sdec,srasc,titpan,titspe,tc		P. Robert, ScientiDev, Feb. 2021

• **visu_polar.f90** nb. of lines : 889 nb. of subroutine: 5

program	visu_polar		visualise un fichier copolar.resu	P. Robert, ScientiDev, Feb. 2021
subroutine	spe_plot_ima	(posfx,posfy,sizfx,sizfy,spebx,nbseq,nbfre,smi,sma,imag		P. Robert, ScientiDev, Feb. 2021
subroutine	spe_plot_grad	(valgra,titgra,valdt,nbgma,nbpma,t1,t2,f1,f2,tgra)		P. Robert, ScientiDev, Feb. 2021
subroutine	rheacopolar	(idset,fultit,projet,experi,sid,isat,fnat,		P. Robert, ScientiDev, Feb. 2021
subroutine	rdatcopolar	(tspea,tspeb,dt,df,nbseq,nbfre,datiso,spebx,speby,spebz,		P. Robert, ScientiDev, Feb. 2021
subroutine	plotpalpol	(ox,oy,sx,sy,xmin,xmax,tgra)		P. Robert, ScientiDev, Feb. 2021

• **visu_spectro.f90** nb. of lines : 963 nb. of subroutine: 5

program	visu_spectro		Visualize a SP.rff	P. Robert, ScientiDev, Feb. 2021
subroutine	co_ind			P. Robert, ScientiDev, Feb. 2021
subroutine	load_data			P. Robert, ScientiDev, Feb. 2021
subroutine	check_meta			P. Robert, ScientiDev, Feb. 2021
subroutine	plot_tit_spectro	(nbp,f1,f2,fc,fs,dtspe,dfspe,		P. Robert, ScientiDev, Feb. 2021
subroutine	plotpuiint	(puix,puiy,puiz,time,nbipes,titpui,f1,f2,smi,sma)		P. Robert, ScientiDev, Feb. 2021

• **visu_vectime_3D.f90** nb. of lines : 937 nb. of subroutine: 4

program	visu_vectime_3D		Visualize on 3D planes 4 vectime.rff files	P. Robert, ScientiDev, Feb. 2021
subroutine	check_meta			P. Robert, ScientiDev, Feb. 2021
subroutine	load_data			P. Robert, ScientiDev, Feb. 2021
subroutine	txyz_to_xycube	(x,y,z,xc,yc,zc)		P. Robert, ScientiDev, Feb. 2021
subroutine	plot_traj	(vectime,nbvec,i1,i2,sig1,sig2,indpo)		P. Robert, ScientiDev, Feb. 2021

• **visu_vectime.f90** nb. of lines : 419 nb. of subroutine: 3

program	visu_vectime		Visualize a vectime.rff	P. Robert, ScientiDev, Feb. 2021
subroutine	check_meta			P. Robert, ScientiDev, Feb. 2021
subroutine	load_data			P. Robert, ScientiDev, Feb. 2021
subroutine	extend	(vect,nbvec)		P. Robert, ScientiDev, Feb. 2021

• **waveform_to_vectime.f90** nb. of lines : 608

program	waveform_to_vectime	read C RFF WaveForm file and convert it 1568 ! into a RFF vectime file.	R. Piberne, LPP, 2011, Rev. PR 2021
---------	---------------------	---	-------------------------------------

24.1.2 CLUSTER programs

• **cef_gse_to_sr2_CLUFGM.f90** nb. of lines : 635

program	fgm_cef_gse_to_sr2	Program to convert a gse fgm.cef file into a sr2 one	R. Piberne, LPP, 2012 March 08
---------	--------------------	--	--------------------------------

• **cef_to_rff_CLUFGM.f90** nb. of lines : 470

program	cef_to_rff_CLUFGM	convert a fgm cef file to a rff file	R. Piberne, LPP, March 2012
---------	-------------------	--------------------------------------	-----------------------------

• **cef_to_rff_CLUPOS.f90** nb. of lines : 448

program	cef_to_rff_CLUPOS	convert a pos cef file to a rff file	P. Robert, ScientiDev, Nov. 2020
---------	-------------------	--------------------------------------	----------------------------------

• **cef_to_rff_cwf_CLUSTA.f90** nb. of lines : 723

program	cef_to_rff_cwf_CLUSTA	convert a CWF cef file to a VTL2 rff file	P. Robert, Nov 2020
---------	-----------------------	---	---------------------

• **cef_to_rff_dwf_CLUSTA.f90** nb. of lines : 618

program	cef_to_rff_dwf_CLUSTA	convert a DWF cef file to a VTL1 rff file	R. Piberne, LPP, 2011 Nov
---------	-----------------------	---	---------------------------

- **CLUEFW_check_status.f90** nb. of lines : 66

program	check			
---------	-------	--	--	--

- **CLUSTA_search_strong_DC.f90** nb. of lines : 138

program	search_strong_DC		search part of file having strong DC	P. Robert, LPP, 2015 Feb 07
---------	------------------	--	--------------------------------------	-----------------------------

- **CLUSTA_spectro_L2_to_cef.f90** nb. of lines : 246

program	spectro_L2_to_cef		read CLUSTER/STAFF-SC RFF spectro file and create a CS CAA cef file.	R. Piberne, LPP, Nov. 2012
---------	-------------------	--	--	----------------------------

- **CLUSTA_vectime_L1_to_cef.f90** nb. of lines : 374

program	vectime_L1_to_cef		read CLUSTER/STAFF-SC RFF Vectime file and create a header ! of a DWF CAA cef file.	R. Piberne, LPP, April 2011
---------	-------------------	--	---	-----------------------------

- **CLUSTA_vectime_L2_to_cef.f90** nb. of lines : 445

program	vectime_L2_to_cef		read CLUSTER/STAFF-SC RFF cwf file and create a CWF CAA cef file	R. Piberne, LPP, Feb. 2012
---------	-------------------	--	--	----------------------------

- **CLUSTA_w_cal_caveat_header.f90** nb. of lines : 187

program	w_cal_caveat_header		create the caveat header file associated to calibration data	R. Piberne, LPP, 2012 Jul 22
---------	---------------------	--	--	------------------------------

- **CLUSTA_w_cwf_cef_header.f90** nb. of lines : 369

program	w_cwf_cef_header		write CWF cef file	R. Piberne, LPP, 2011 Apr 05
---------	------------------	--	--------------------	------------------------------

- **CLUSTA_w_vectime_cef_header.f90** nb. of lines : 435

program	w_vectime_cef_header		write DWF cef file.	R. Piberne, LPP, 2011 April 05
---------	----------------------	--	---------------------	--------------------------------

- **get_data_CLUGEOM.f90** nb. of lines : 277

program	get_data_CLUGEOM		calcul les parametres geometriques du tetrahedre	P. Robert, CETP, Mars 2001
---------	------------------	--	--	----------------------------

- **vectime_calibration_CLUSTA.f90** nb. of lines : 2009 nb. of subroutine: 3

program	vectime_calibration_CLUSTA		read VTL1, calibrate data with continuous method, create VTL2	P. Robert, 2009
subroutine	read_N_cont_vectors	(N_Vectors)		P. Robert, 2009
subroutine	r_rff_data_block_int	(in_file_unit,nor_index,ext_index,int_block)		P. Robert, 2009
subroutine	read_input	(in_file_name,ou_file_name,cal_dir_path,		P. Robert, 2009

- **vectime_gse_to_isr2.f90** nb. of lines : 121

program	vectime_gse_to_isr2		convert vectime from GSE to ISR2 (P. Robert, ScientiDev, January 2021
---------	---------------------	--	------------------------------------	-------------------------------------

- **vectime_L1_to_spectro_L2_CLUSTA.f90** nb. of lines : 2070 nb. of subroutine: 3

program	vectime_L1_to_spectro_L2_CLUSTA		read VTL1, compute calibrated spectra, create SPL2	P. Robert, May 2012
subroutine	read_N_cont_vectors	(N_Vectors)		P. Robert, May 2012
subroutine	r_rff_data_block_int	(in_file_unit,nor_index,ext_index,int_block)		P. Robert, May 2012
subroutine	read_input	(in_file_name,ou_file_name,cal_dir_path,		

- **vectime_to_mfa_CLUSTA.f90** nb. of lines : 399

program	vectime_to_mfa_CLUSTA		transform a VecTime VTL2 in ISR2 or GSE system into mfav system	P. Robert, ScientiDev, Oct. 2020
---------	-----------------------	--	---	----------------------------------

- **visu_CLUGEOM.f90** nb. of lines : 580 nb. of subroutine: 2

program	visu_CLUGEOM		Lit un fichier clugeom.resu et cre le PS visu_CLUGEOM.ps	P. Robert, CETP, Juin 2005
subroutine	read_header_CLUGEOM	(rep,nrec)		P. Robert, CETP, Juin 2005
subroutine	read_data_CLUGEOM	(jul001,iyear1,imon1,iday1,time,posgse,nrec,		P. Robert, CETP, Juin 2005

• **visu_curl_div_4sat.f90** nb. of lines : 985 nb. of subroutine: 1

program	visu_curl_div_4sat	Lit un fichier compute_curl_div_4sat.resu et cre visu_curl_div_4sat.ps	P. Robert, CETP, Juin 2005
subroutine	lect_cocurldiv		P. Robert, CETP, Juin 2005

• **visu_spectro_4Bz.f90** nb. of lines : 1368 nb. of subroutine: 5

program	visu_spectro_4Bz		Visualize BZ of 4 SP.rff	P. Robert, ScientiDev, Feb. 2021
subroutine	co_ind			P. Robert, ScientiDev, Feb. 2021
subroutine	load_data			P. Robert, ScientiDev, Feb. 2021
subroutine	check_meta			P. Robert, ScientiDev, Feb. 2021
subroutine	plot_tit_spectro_	(nbp,f1,f2,fc,fs,dtspe,dfspe,		P. Robert, ScientiDev, Feb. 2021
subroutine	plotpuiint	(time1,pui1,time2,pui2,time3,pui3,time4,pui4,		P. Robert, ScientiDev, Feb. 2021

• **visu_vectime_3D_4sat.f90** nb. of lines : 1214 nb. of subroutine: 4

program	visu_vectime_3D_4sat		Visualize on 3D planes 4 vectime.rff files	P. Robert, ScientiDev, Feb. 2021
subroutine	check_meta			P. Robert, ScientiDev, Feb. 2021
subroutine	load_data			P. Robert, ScientiDev, Feb. 2021
subroutine	plot_traj	(vectime,nbvec,i1,i2,sig1,sig2,indpo)		P. Robert, ScientiDev, Feb. 2021
subroutine	txyz_to_xycube	(x,y,z,xc,yc,zc)		P. Robert, ScientiDev, Feb. 2021

• **visu_vectime_4sat.f90** nb. of lines : 636 nb. of subroutine: 3

program	visu_vectime_4sat		Visualize 4 vectime.rff	P. Robert, ScientiDev, Feb. 2021
subroutine	check_meta			P. Robert, ScientiDev, Feb. 2021
subroutine	load_data			P. Robert, ScientiDev, Feb. 2021
subroutine	extend	(vect,nbvec)		P. Robert, ScientiDev, Feb. 2021

24.1.3 GEOS programs

• **vectime_calibration_GEOULF.f90** nb. of lines : 2148 nb. of subroutine: 6

program	vectime_calibration_GEOULF		read VTL1, calibrate data with continuous method, cre VTL2	P. Robert, ScientiDev, Feb. 2020
subroutine	read_N_cont_vectors	(N_Vectors)		P. Robert, LPP, 2009-2012
subroutine	r_rff_data_block_int	(in_file_unit,nor_index,ext_index,int_block)		P. Robert, LPP, 2009-2012
subroutine	read_input	(in_file_name,ou_file_name,		P. Robert, ScientiDev, Feb. 2020
subroutine	correch	(s,nbp,fe,i)		P. Robert, LPP, 2009-2012
subroutine	VDH_to_xyz	(v,d,h,a1,a2,a3,x,y,z)		P. Robert, ScientiDev, Feb. 2020
subroutine	xyz_to_VDH	(x,y,z,a1,a2,a3,v,d,h)		P. Robert, ScientiDev, Feb. 2020

• **vectime_L1_to_spectro_L2_GEOULF.f90** nb. of lines : 1998 nb. of subroutine: 3

program	vectime_L1_to_spectro_L2_GEOULF		read VTL1, compute calibrated spectra (classical method), make SPL2	P. Robert, ScientiDev, Feb. 2020
subroutine	read_N_cont_vectors	(N_Vectors)		P. Robert, LPP, 2009-2012
subroutine	r_rff_data_block_int	(in_file_unit,nor_index,ext_index,int_block)		P. Robert, LPP, 2009-2012
subroutine	read_input	(in_file_name,ou_file_name,		P. Robert, ScientiDev, Feb. 2020

• **vectime_to_mfav_GEOULF.f90** nb. of lines : 412 nb. of subroutine: 1

program	vectime_to_mfav_GEOULF		transform a VecTime VTL2 in SRV system into mfav system	P. Robert, ScientiDev, Oct. 2020
subroutine	t_srv_to_mfav	(xsrv,ysrv,zsrv,bx,by,bz,xfav,yfav,zfav)	transforms_srv_to_mfa: srv -> MFA system	P. Robert, ScientiDev, 2020

• **vectime_vdh_to_srv_GEOMAG.f90** nb. of lines : 202 nb. of subroutine: 1

program	vectime_vdh_to_srv_GEOMAG		create RFF file of GEOS/S331 in SRV system (as S300)	P. Robert, ScientiDev, Oct 2020
subroutine	vdh_to_srv	(x,y,z,E2,E3,s,r,v)		P. Robert, ScientiDev, Oct. 2020

• **vectime_vdh_to_xyz_GEOMAG.f90** nb. of lines : 151

program	vectime_vdh_to_xyz_GEOMAG	create RFF file of GEOS/S331 in XYZ system (as S300 L1)	P. Robert, ScientiDev, Oct 2020
---------	---------------------------	---	---------------------------------

24.1.4 GEOS/GROUND data programs

• **vectime_calibration_GEOGRD.f90** nb. of lines : 1572 nb. of subroutine: 3

program	vectime_calibration_GEOGRD		read VTL1, calibrate data with continuous method, create VTL2	P. Robert, ScientiDev, January 2019
subroutine	read_N_cont_vectors	(N_Vectors)		P. Robert, LPP, 2009-2012
subroutine	r_rff_data_block_int	(in_file_unit,nor_index,ext_index,int_block)		P. Robert, LPP, 2009-2012
subroutine	read_input	(in_file_name,ou_file_name,		P. Robert, LPP, 2009-2012

• **vectime_L1_to_spectro_L2_GEOGRD.f90** nb. of lines : 1544 nb. of subroutine: 3

program	vectime_L1_to_spectro_L2_GEOGRD		read VTL1, compute calibrated spectra (classical method), create SPL2	P. Robert, ScientiDev, January 2021
subroutine	read_N_cont_vectors	(N_Vectors)		P. Robert, LPP, 2009-2012
subroutine	r_rff_data_block_int	(in_file_unit,nor_index,ext_index,int_block)		P. Robert, LPP, 2009-2012
subroutine	read_input	(in_file_name,ou_file_name,		P. Robert, LPP, 2009-2012

• **vectime_to_mfav_GEOGRD.f90** nb. of lines : 348 nb. of subroutine: 1

program	vectime_to_mfav_GEOGRD		transform a VecTime VTL2 in NWV system into mfav system	P. Robert, ScientiDev, January 2021
subroutine	t_nwv_to_mfav	(xnwv,ynwv,znwv,bx,bz,xfav,yfav,zfav)	transforms_nwv_to_mfa: NWV -> MFA system	P. Robert, ScientiDev, 2020

24.2 Libraries

24.2.1 Common libraries

24.2.1.1 lib_deconvo.f90

nb. of lines : 1305 nb. of subroutine: 17

subroutine	CAPODI	(X,NBP,IAPOD,NOM)	CALCUL UNE FENETRE D'APODISISATION. IAPOD=1,8	P. ROBERT, 1977-1985 ;Rev. 2015
subroutine	csurf	(x,nbp,nom,surf)	calcule la surface du tableau X sur nbp points	P. ROBERT, 1977-1985
subroutine	test_blk_cont	(nor_index,imsres,ierr)	test sur la continuite des blocks	P. Robert, LPP , 2010
subroutine	test_blk_cont_no_cal	(nor_index,ext_index,imsres,ierr)	test sur la continuite des blocks/saut des calibrations	P. Robert, LPP , 2010
subroutine	w_com	(pr_out,com)	write on stdo comment only if pr_out is true	P. Robert, LPP , 2010
subroutine	subtrend	(xfo,work,nbp,nlis,iano)	lissage d'une courbe et soustraction a l'original	P. Robert, CETP, 2003
subroutine	desinus	(xio,n,fe,fs,amp,pha)	despinage d'une forme d'onde xio (algorithme demod)	P. Robert, CRPE, 1977-1984
subroutine	desinus_D	(xio,n,fe,fs,amp,pha)	despinage d'une forme d'onde xio (algorithme demod)	P. Robert, CRPE, 1977-1984
subroutine	deconvo_R	(xio,tra,TFcor,nbp,ix,DSS,DCS,M_Kern)	deconvolution des formes d'onde x y z	P. Robert, CRPE, 1977-1984
subroutine	modpha	(s,rm,rp)	calcule le module et la phase (degre) d un nb. complexe	P. Robert, CRPE, 1977-1984
subroutine	trotfix	(wave,nbp,fe,fs,spin,rotdeg)	passage d'un repere en rotation a un repere fixe	P. Robert, CRPE, 1977-2008
subroutine	casinus	(sig,n,fe,fs,as,phi,com)	calcul d'une sinusoide contenue dans un signal reel	P. Robert, CRPE, 1977-1984
subroutine	casinus_D	(sig,n,fe,fs,as,phi,com)	calcul d'une sinusoide contenue dans un signal reel	P. Robert, CRPE, 1977-1984
subroutine	deconvo_C3	(Z,TFcor,N,DSS,DCS,M)	deconvolution optimisee par D-FFT et I-FFT incluses	P. Robert, LPP , 2012
subroutine	deconvo_C3_sd	(Z,TFcor,N,DSS,DCS,M)	deconvolution optimisee par D-FFT et I-FFT incluses	P. Robert, LPP , 2012
subroutine	fftpat_XY	(X,Y,N,DSS,DCS,M,IND)	fft directe ou inverse sans table de sinus	P. Robert, CRPE, 1984, C/T
subroutine	lissage	(x,nbp,nlis,iano)	lissage d'un tableau sur nlis points	P. Robert, CETP, 2001

24.2.1.2 lib_time.f90

nb. of lines : 1314 nb. of subroutine: 27

subroutine	check_datiso	(datiso)			P. Robert, LPP, 2011 Mar.
subroutine	gdatiso	(datiso)		donne la date ISO au moment du call	P. Robert, LPP, 2011 Mar.
subroutine	print_date_time	(com)		print le CPU time ecoule depuis le dernier appel	P. Robert, CRPE, 1993
subroutine	read_date	(prompt,iday,imonth,year)		lit la date depuis l'input et test sa validite	P. Robert, CRPE, 1993
subroutine	read_time	(prompt,ih,im,is)		lit l'heure depuis l'input et test sa validite	J. Ahmad, CETP, 2005
subroutine	coforstr	(str, format)		Renvoie le format associe aux chiffres d'une string	P. Robert, CRPE, 1992
subroutine	cdatdoty	(idoy,iyear,imonth,iday)		compute_date_from_day_of_year and for a given year	P. Robert, CRPE, 1992
subroutine	cdatjd50	(jd50,iyear,imonth,iday)		compute_date_from_julian_day_1950 with jd50=0 for jan. 1	P. Robert, CRPE, 1992
subroutine	coleapyear	(iyear,ileap)		compute_leap_year with ileap=1 for leap year, 0 if not	P. Robert, CRPE, 1992
subroutine	codoty	(imonth,iday,iyear,idoty)		compute_day_of_the_year with idoty=1 for january 1	P. Robert, CETP, 2000 Jul.
subroutine	cojd00	(iyear,imonth,iday,jd00)		compute_julian_day_2000 with jd00=0 for january 1, 2000	P. Robert, CETP, 2000 Jul.
subroutine	cojd50	(iyear,imonth,iday,jd50)		compute_julian_day_1950 with jd50=0 for january 1, 1950	J. Ahmad, CETP, 2006
subroutine	gdatime	(iyear,imon,iday,ih,im,is,ims)		donne la date et le temps au moment du call	P. Robert, CETP, 2002
subroutine	addsec_datiso	(datiso1,sec,datiso2)		additionne a la date ISO un temps en sec.	P. Robert, CETP, 2002
subroutine	addsecdbl_datiso	(datiso1,secdbl,datiso2)		additionne a la date ISO un temps en sec. double	P. Robert, CETP, 2002 Sep.
subroutine	clean_datiso	(datiso,datap)		converti la date iso ex: '2000-09-13 12:09:56.868055'	P. Robert, ScientiDev, Jan 2021
subroutine	diff_datiso	(date1,date2,secdif)		calcule la difference entre 2 dates ISO date2-date1	P. Robert, CETP, 2001
subroutine	compare_datiso	(datiso1,datiso2,icomp)		compare 2 dates ISO ; if date2 > date1 icomp=1 else -1	P. Robert, CETP, 2001
subroutine	decode_datiso	(datiso,iyear,imon,iday,ih,im,is,ims,imc)		decodes la date ISO en date et heure	P. Robert, CETP, 2002 Sep.
subroutine	encode_datiso	(iyear,imon,iday,ih,im,is,ims,imc,datiso)		encode la date ISO en date et heure	P. Robert, CETP, 2001 Jan.
subroutine	encode_datpat	(iyear,imon,iday,ih,im,is,ims,imus,datap)		cre la 'date P.' ex: '2000-09-13 12:09:56.868055'	P. Robert, CETP, 2001 Jan.
subroutine	codecssec	(nbday,ih,im,is,ims,imc,decsec)		calcule la seconde decimale d'un temps	P. Robert, CETP, 2001 Jan.
subroutine	codecssecdbl	(nbday,ih,im,is,ims,imc,secdbl)		calcule la seconde decimale dble d'un temps	P. Robert, CETP, 2001 Jan.
subroutine	codecssecinv	(decsec,nbday,ih,im,is,ims,imc)		convertie la seconde decimale dble en nbday,ih,im,is,ims	P. Robert, CETP, 2001 Jan.
subroutine	codecssecdblinv	(secdbl,nbday,ih,im,is,ims,imc)		convertie la seconde decimale dble en nbday,ih,im,is,ims	P. Robert, CETP, 2001 Jan.
subroutine	comilday	(ih,im,is,ims,milday)		calcule la milliseconde du jour	P. Robert, CETP, 2001 Jan.
subroutine	comildayinv	(milday,ih,im,is,ims)		convertie la milliseconde du jour en ih,im,is,ims	P. Robert, CETP, 2001 Jan.

24.2.1.3 lib_alitime.f90

nb. of lines : 573 nb. of subroutine: 5

subroutine	alitime_data_4sat	(datiso1,datiso2,datiso3,datiso4,	aligne en temps les 4 tableaux de donnees vectim par inte	P. Robert, Scientidev, Jan. 2021
subroutine	get_interpol_value	(datiso,datiso_arr,vectim_arr,nbloc,j1,val_xy	donne la valeur interpolée de vectim_arr au temps datiso	P. Robert, ScientiDev, Jan. 2021
subroutine	interpo	(tab1,mils1,n1,tab2,mils2,n2,mdtmax,iano,fill_val)	interpolation de tab1, resultat dans tab2	P. Robert, Oct. 2001
subroutine	xyz_vec	(xini,yini,zini,vecti,nbp)		P. Robert, Scientidev, Jan. 2021
subroutine	vec_xyz	(vecti,xini,yini,zini,nbp)		P. Robert, Scientidev, Jan. 2021

24.2.1.4 lib_utility.f90

nb. of lines : 527 nb. of subroutine: 12

subroutine	int_to_char	(ival,cval,nc)	converti un entier en character, en le cadrant a gauche)	P. Robert, CETP, 2003
subroutine	cbestfor	(x,format)	compute_best_format as '(f3.1)' for x=2.5	P. Robert, CETP, 1995, 2003
subroutine	cbestfori	(ix,format)	compute_best_format_integer as '(i3)' for ix=124	P. Robert, CETP, 1995, 2003
subroutine	real_to_string	(rx,sx,nc)	met un reel en string, avec le meilleurs format	P. Robert, CETP, 2003
subroutine	ringbel		ring bell by printing '007' octal character	P. Robert, CETP, 2000
subroutine	upper_case	(string,nc)	met la string en majuscule si elle ne l'est pas	P. Robert, CETP, 2000
subroutine	lower_case	(string,nc)	met la string en minuscule si elle ne l'est pas	P. Robert, CETP, 2000
subroutine	uencfor	(format,ifg,n,nd)	encode un format a partir de ses caracteristiques	P. Robert, CETP, 2000
subroutine	udecfor	(format,ifg,n,nd)	decode un format en fournissant ses caracteristiques	P. Robert, CETP, 2000
subroutine	basename	(path,dir,nd,name,nf)	calcul le nom d'un fichier/directory a partir du path	P. Robert, CETP, 2002
subroutine	crandom01	(ranval,size)	genere une valeur aleatoire entre 0 et size	P. Robert, CETP, 1994
subroutine	randgen71	(ranval,idum)	utilitaire de crandom01	P.W.DALY, sept. 1994

24.2.1.5 lib_rw_rff.f90

nb. of lines : 2698 nb. of function : 18 nb. of subroutine: 27 nb. of module : 4

module	param_def		modules for RFF Parameter definition	P. Robert, LPP, Jan. 2011
module	data_def		modules for RFF indexed data definition	R. Pièrre, LPP, Feb. 2011
subroutine	rff_allocate_data_arrays		Allocate data arrays from mandatory parameters	R. Pièrre, LPP, Feb. 2011
subroutine	rff_get_current_date	(dateiso)	donne la date ISO au moment du call	P. Robert, LPP, Mars 2011
subroutine	rff_set_default_init		set all parameters to zero or 'undefined'(default)	R. Pièrre, LPP, Mars 2012 Rev. P.R.Jan 2021
subroutine	rff_set_default_manda_param		Set default mandatory parameters	P. Robert, ScientiDev, Jan. 2021
subroutine	rff_set_default_optio_param		Set default optional parameters	P. Robert, ScientiDev, Jan. 2021
subroutine	rff_set_default_const_data		Set default constant data	P. Robert, ScientiDev, Jan. 2021
subroutine	rff_set_default_DATA_DESCRIPTION		Set DATA_DESCRIPTION parameter value to default	P. Robert, LPP, Mars 2012
subroutine	rff_set_default_BLOCK_DESCRIPTION		Set BLOCK_DESCRIPTION parameter value to default	P. Robert, LPP, Mars 2012
subroutine	rff_set_default_INDEX_DESCRIPTION		Set INDEX_DESCRIPTION parameter value to default	P. Robert, LPP, 2012 Mars 28
subroutine	rff_set_default_INDEX_EXTENSION_DESCRIP		Set INDEX_EXTENSION_DESCRIP parameter value to default	P. Robert, LPP, Mar. 2012
subroutine	rff_update_history	(create,com)	Update HISTORY parameter in optional_parameters	P. Robert, Mars LPP, 2012
subroutine	rff_R_file	(ifc,file)	Read rff file, metadata & data, WaveForm or VecTime	R. Pièrre, LPP, Feb. 2011
subroutine	rff_R_manda_param	(ifc)	Read mandatory parameter in the rff file	P. Robert, Jan. LPP, 2011
subroutine	rff_R_optio_param	(ifc)	Read optional parameters in the rff file	P. Robert, LPP, Jan. 2011
subroutine	rff_R_const_data	(ifc)	Read constant data in the rff file	P. Robert, LPP, Jan. 2011 Rev. Nov 2020
subroutine	rff_R_metadata	(ifc,file)	Open RFF file, read header, metadata & constant data	R. Pièrre, LPP, Feb. 2011
subroutine	rff_R_indexed_data	(ifc)	Read indexed data for waveform and vectime RFF files	R. Pièrre, LPP, Feb. 2011
subroutine	rff_R_tail	(ifc)	Read tail of a RFF file	P. Robert, LPP, Mars 2011
subroutine	rff_W_file	(ifc,file)	Write rff file, metadata & data, WaveForm or VecTime	P. Robert, LPP, Mars 2011
subroutine	rff_W_manda_param	(ifc)	Write mandatory parameters	P. Robert, LPP, Jan. 2011
subroutine	rff_W_optio_param	(ifc)	Write optional parameters	P. Robert, LPP, Jan. 2011
subroutine	rff_W_const_data	(ifc)	Write constant data	P. Robert, LPP, Jan. 2011
subroutine	rff_W_metadata	(ifc,file)	Open RFF file, Write header, metadata & constant data	P. Robert, LPP
subroutine	rff_W_indexed_data	(ifc)	Write indexed data for waveform and vectime RFF files	P. Robert, LPP, Mars 2011
subroutine	rff_W_tail	(ifc)	Write tail of a RFF file	P. Robert, LPP, Mar. 2011
subroutine	rff_format_W_to_R	(format_W,format_R)	convert a Write format in a Read format	P. Robert, LPP, Oct. 2011
subroutine	rff_format_R_to_W	(format_R,format_W)	convert a Read format in a Write format	P. Robert, LPP, Aug. 2012
function	get_paramT	(param,ifc,nbli)		P. Robert, LPP, Jan. 2011
function	get_pos	(param,ifc)	Read rff file until searched Character parameter	P. Robert, LPP, Jan. 2011
function	get_paramC	(param,ifc)	Read rff file until searched Character parameter & return it	P. Robert, LPP, Jan. 2011
function	get_paramI	(param,ifc)	Read rff file until searched Integer parameter & return it	P. Robert, LPP, Jan. 2011
function	get_paramI2	(param,ifc)	Read rff file until searched 2nd Int. parameter & return it	P. Robert, LPP, Jan. 2011
function	get_paramR	(param,ifc)	Read rff file until searched Real SINGL parameter & return it	P. Robert, LPP, Jan. 2011
function	get_paramD	(param,ifc)	Read rff file until searched Real DBLE parameter & return it	P. Robert, LPP, Jan. 2011
function	get_paramT	(param,ifc,nbli)	Read rff file until searched Character parameter & return it	P. Robert, LPP, Jan. 2011
function	Fbasename	(path)	Return file name of a path	P. Robert, LPP, Mar. 2011
function	Fdirname	(path)	Return directory name of a path	P. Robert, LPP, Mar. 2011
function	give_RPC_version		Return RPC_version	P. Robert, LPP, Jan. 2014
module	String_Functions		pack of string functions	David Frank

24.2.1.6 lib_signal.f90

nb. of lines : 279 nb. of subroutine: 6

subroutine	co_plane_wave_xy	(a,b,f,dt,n,akdotr,bx,by,bz)	computation of a single plane elliptic-polarized waves	P. Robert, CETP, May 1998
subroutine	co_plane_wave	(akx,aky,akz,a,b,f,dt,n,akdotr,bx,by,bz)	computation of a single plane elliptic-polarized waves	P. Robert, CETP, May 1998
subroutine	co_plane_wave_fv	(akx,aky,akz,a,b,f,t2,dtv,n,dt,akdr,bx,by,bz)	computation of a single plane elliptic-polarized waves	P. Robert, CETP, May 1998 -Rev.2021
subroutine	co_BO_coord	(x,y,z,b0x,b0y,b0z,xs,ys,zs)	compute x,y,z in Bo coordinate system where Z'=Bo	P. Robert, CETP, May 1998
subroutine	ret_BO_coord	(xs,ys,zs,b0x,b0y,b0z,x,y,z)	retrieve x,y,z from Bo coordinate system where Z'=Bo	P. Robert, CETP, May 1998
subroutine	rot_xy_BO	(x,y,z,b0x,b0y,b0z,xs,ys,zs)	make a rotation in the XY plane of the elliptical wave	P. Robert, CETP, May 1998

24.2.1.7 rocotlib_V3p2.f90

nb. of lines : 5458 nb. of subroutine: 189

subroutine	cp_angle_and_ratio	(ux,uy,vx,vy,vz,angle,ratio)	compute_angle_and_ratio between U and V vectors	P. Robert, CRPE, 1992
subroutine	cp_Euler_interp	(a1,b1,c1,a2,b2,c2,ti,dt,ai,bi,ci)	compute_Euler_angles_interpolation	P. Robert, SDev, 2020
subroutine	cp_geo_dipole_dir	(iyear,idoy,d1,d2,d3)	compute_dipole_direction in GEO system	P. Robert, LPP, 2016
subroutine	cp_gei_sun_dir	(iyear,idoy,ih,im,is)	compute_sun_direction in GEI system	CT.Russel, 1971, rev. PR, 1992, 2002
subroutine	cp_sunrise_sunset	(iyear,imon,iday,rlat,rlon,	compute_sunset_time and others	P. Robert, CRPE, 2001 Revised Dec. 2011
subroutine	cp_time_param	(iyear,imonth,iday,ih,im,is)	compute_time_parameters and time-dependent matrix	P. Robert, CRPE, 1992
subroutine	cp_time_param2	(jd1950,houday)	compute_time_parameters and time-dependent matrix	P. Robert, CRPE, 2001
subroutine	cp_time_param3	(jd2000,houday)	compute_time_parameters and time-dependent matrix	P. Robert, CRPE, 2001
subroutine	cp_tpn_param	(xo,yo,zo,xs, Tx,Ty,Tz, Px,Py,Pz, Nx,Ny,Nz)	compute_TPN_system	P. Robert, CETP, 2004
subroutine	cp_nbday_in_month	(iyear,imonth,nbday)	compute_number_of_day_of_the_month	P. Robert, CRPE, 2001
subroutine	cp_en_day_name	(iday,cday,nbcha)	compute_english_day_name, ex: 'Monday' for iday=1	P. Robert, CRPE, 2001
subroutine	cp_en_month_name	(imonth,cmonth,nchar)	compute_english_month_name	P. Robert, CRPE, 2001
subroutine	cp_fr_day_name	(iday,cday,nbcha)	compute_french_day_name, ex: 'Lundi' for iday=1	P. Robert, CRPE, 2001
subroutine	cp_fr_month_name	(imonth,cmonth,nchar)	compute_french_month_name	P. Robert, CRPE, 2001
subroutine	cp_leap_year	(iyear,ileap)	compute_leap_year with ileap=1 for leap year, 0 if not	P. Robert, CRPE, 1992
subroutine	cp_seasons	(iyear,id_sso,id_wso,id_seq,id_feq,	compute_seasons, i.e. solstice & equinox	P. Robert, SDev, 2017
subroutine	cv_doty_to_date	(idoy,iyear,imonth,iday)	convert_day_of_year for a given year in date	P. Robert, CRPE, 1992
subroutine	cv_jul2000_to_date	(jd00,iyear,imonth,iday)	convert_julian_day_2000 in date	P. Robert, CRPE, 1992
subroutine	cv_jul1950_to_date	(jd50,iyear,imonth,iday)	convert_julian_day_1950 in date	P. Robert, CRPE, 1992
subroutine	cv_weekn_to_date	(iweek,iyear,imonth,iday)	convert_first_day_of_week_number in date	P. Robert, CRPE, 2001
subroutine	cv_date_to_dow	(iyear,imonth,iday,idow)	convert_date_in_day_of_the_week	P. Robert, CRPE, 2001
subroutine	cv_date_to_doty	(iyear,imonth,iday,idoy)	convert_date_in_day_of_year with idoy=1 for january 1	P. Robert, CRPE, 1992
subroutine	cv_hms_to_dech	(ih,im,is,houday)	convert_hours_minutes_seconds in decimal hour of the day	P. Robert, CRPE, 1992
subroutine	cv_date_to_jul1950	(iyear,imonth,iday,jd50)	convert_date_in_julian_day_1950 with jd50=0 for jan 1	P. Robert, CRPE, 1992
subroutine	cv_date_to_jul2000	(iyear,imonth,iday,jd00)	convert_date_in_julian_day_2000 with jd00=0 for january 1	P. Robert, CRPE, 1992
subroutine	cv_dhms_to_msotd	(ih,im,is,ims,milday)	convert_hours_minutes_seconds_ms in millisecc_of_day	P. Robert, CRPE, 2001
subroutine	cv_dech_to_hms	(houday,ih,im,is)	convert_decimal hour of the day in time	P. Robert, CRPE, 1992
subroutine	cv_msotd_to_hmsms	(milday,ih,im,is,ims)	convert_millicsec. of the day in time	P. Robert, CRPE, 2001
subroutine	cv_date_to_weekn	(iyear,imonth,iday,iweek)	convert_date_in_week_of_the_year	P. Robert, CRPE, 2001, rev. 2017
subroutine	g_gei_geo_dipole_dir	(dxgei,dygei,dzgei,	give_dipole_direction in GEI and GEO system	P. Robert, CRPE, 1992
subroutine	g_gsm_dipole_tilt_angle	(diptan)	give_dipole_tilt_angle in radians	P. Robert, CRPE, 1992
subroutine	g_gei_geo_ecliptic_dir	(exgei,eygei,ezgei,	give_ecliptic_direction in GEI and GEO system	P. Robert, CRPE, 1992
subroutine	g_gei_geo_sun_rot	(rxgei,rygei,rzgei,rxgeo,rygeo,rzgeo)	give_sun_rotation_direction in GEI and GEO system	P. Robert, CRPE, 1992
subroutine	g_gei_geo_sun_dir	(sxgei,sygei,szgei,sxgeo,sygeo,szgeo)	give_sun_direction in GEI and GEO system	P. Robert, CRPE, 1992
subroutine	g_gei_sun_param	(gmst,slon,sras,sdec,obli)	give_sun_parameter dependant of time in GEI system	P. Robert, CRPE, 1992
subroutine	g_rocot_version_number	(vernum,verdat)	give_version_number and modification date of the library	P. Robert, CRPE, 1992
subroutine	mat_cp_varmin	(ifc,Vx,Vy,Vz,n,irep,covar,lambda,eigvec)	compute variance minimum coordinate of a signal Vx,Vy,Vz	P. Robert, CETP, 2001, rev. PR 2016
subroutine	mat_cp_covariance	(Vx,Vy,Vz,n,covar)	compute covariance matrix for a vector series V(n)	P. Robert, CETP, 2001, rev. PR 2016
subroutine	mat_diagonalise	(mat,lambda,eigvec)	diagonalise the given matrix mat(3,3)	P. Robert, CETP, 2001, rev. PR 2016
subroutine	mat_check_ortho	(ifc,mat)	check orthogonality of matrix components	P. Robert, CETP, 2001, rev. PR 2016
subroutine	mat_cp_determin	(mat,det)	compute determinant of the given matrix	P. Robert, CETP, 2001, rev. PR 2016
subroutine	mat_cp_eigen_vec	(mat,lambda,eigvec)	compute eigen vectors and eigen values of real mat(3,3)	unknown, CETP, 2001, rev. PR 2016
subroutine	mat_cp_pythag_func	(a,b,fpyth)	Pythagore function of two real (used by mat_cp_eigen_vec)	unknown, CETP, 2001, rev. PR 2016
subroutine	mat_normalize_vec	(mat)	normalize to 1. the vectors of the input matrix	P. Robert, CETP, 2001, rev. PR 2016
1.0 subroutine	mat_product	(mat1,mat2,mat3)	matrix product of two given matrix of dim. 3	P. Robert, CETP, 2001, rev. PR 2016
subroutine	mat_somme	(mat1,mat2,mat3)	matrix somme of two given matrix of dim. 3	P. Robert, LPP, 2016
subroutine	mat_diff	(mat1,mat2,mat3)	matrix difference of two given matrix	P. Robert, LPP, 2016
subroutine	mat_transpose	(mat)	transpose input matrix	P. Robert, CETP, 2001, rev. PR 2016
subroutine	mat_change_coord	(mat,Vx,Vy,Vz,n)	change coordinate of a vector serie with a given matrix	P. Robert, CETP, 2001, rev. PR 2016
subroutine	mat_write	(ifc,com,mat)	print on ifc unit mat(3,3) with a comment	P. Robert, CETP, 2001, rev. PR 2016
subroutine	mat_write_eigen_vec	(ifc,lambda,mat)	print on ifc unit eigen values & vectors of mat(3,3)	P. Robert, CETP, 2001, rev. PR 2016
subroutine	print_rocot_info		print_library informations	P. Robert, CRPE, 1992
subroutine	r_coordinate_values	(x,y,z,cs)	read coordinate values from input	P. Robert, CRPE, 2002
subroutine	r_coordinate_system	(csys)	read coordinate system from input and check validity	P. Robert, CRPE, 2002
subroutine	r_date	(iyear,imonth,iday)	read_date from input and check validity	P. Robert, CRPE, 1992
subroutine	r_time	(ih,im,is)	read_time from input and check validity	P. Robert, CRPE, 1992

subroutine	t_car_to_sph	(x,y,z,r,teta,phi)	transforms_car_to_sph: CAR -> SPH system	P. Robert, CRPE, 1992
subroutine	t_dm_to_geo	(xdme,ydme,zdme,rlat,rlong,xgeo,ygeo,zgeo)	transforms_dme_to_geo: DM -> GEO system	P. Robert, CRPE, 1992
subroutine	t_gei_to_geo	(xgei,ygei,zgei,xgeo,ygeo,zgeo)	transforms_gei_to_geo: GEI -> GEO system	P. Robert, CRPE, 1992
subroutine	t_gei_to_gse	(xgei,ygei,zgei,xgse,ygse,zgse)	transforms_gei_to_gse: GEI -> GSE system	P. Robert, CRPE, 1992
subroutine	t_gei_to_gsm	(xgei,ygei,zgei,xgsm,ygsm,zgsm)	transforms_gei_to_gsm: GEI -> GSM system	P. Robert, CRPE, 1992
subroutine	t_gei_to_gseq	(xgei,ygei,zgei,xgsq,ygsq,zgsq)	transforms_gei_to_gsq: GEI -> GSEQ system	P. Robert, CRPE, 1992
subroutine	t_gei_to_mag	(xgei,ygei,zgei,xmag,ymag,zmag)	transforms_gei_to_mag: GEI -> MAG system	P. Robert, CRPE, 1992
subroutine	t_gei_to_sm	(xgei,ygei,zgei,xsma,ysma,zsma)	transforms_gei_to_sma: GEI -> SM system	P. Robert, CRPE, 1992
subroutine	t_geo_to_dm	(xgeo,ygeo,zgeo,rlat,rlong,xdme,ydme,zdme)	transforms_geo_to_dme: GEO -> DM system	P. Robert, CRPE, 1992
subroutine	t_geo_to_gei	(xgeo,ygeo,zgeo,xgei,ygei,zgei)	transforms_geo_to_gei: GEO -> GEI system	P. Robert, CRPE, 1992
subroutine	t_geo_to_gse	(xgeo,ygeo,zgeo,xgse,ygse,zgse)	transforms_geo_to_gse: GEO -> GSE system	P. Robert, CRPE, 1992
subroutine	t_geo_to_gsm	(xgeo,ygeo,zgeo,xgsm,ygsm,zgsm)	transforms_geo_to_gsm: GEO -> GSM system	P. Robert, CRPE, 1992
subroutine	t_geo_to_gseq	(xgeo,ygeo,zgeo,xgsq,ygsq,zgsq)	transforms_geo_to_gsq: GEO -> GSEQ system	P. Robert, CRPE, 1992
subroutine	t_geo_to_mag	(xgeo,ygeo,zgeo,xmag,ymag,zmag)	transforms_geo_to_mag: GEO -> MAG system	P. Robert, CRPE, 1992
subroutine	t_geo_to_sm	(xgeo,ygeo,zgeo,xsma,ysma,zsma)	transforms_geo_to_sma: GEO -> SM system	P. Robert, CRPE, 1992
subroutine	t_geo_to_vdh	(xgeo,ygeo,zgeo,rlat,rlong,xvdh,yvdh,zvdh)	transforms_geo_to_vdh: GEO -> VDH system	P. Robert, CRPE, 1992
subroutine	t_gse_to_gei	(xgse,ygse,zgse,xgei,ygei,zgei)	transforms_gse_to_gei: GSE -> GEI system	P. Robert, CRPE, 1992
subroutine	t_gse_to_geo	(xgse,ygse,zgse,xgeo,ygeo,zgeo)	transforms_gse_to_geo: GSE -> GEO system	P. Robert, CRPE, 1992
subroutine	t_gse_to_gsm	(xgse,ygse,zgse,xgsm,ygsm,zgsm)	transforms_gse_to_gsm: GSE -> GSM system	P. Robert, CRPE, 1992
subroutine	t_gse_to_gseq	(xgse,ygse,zgse,xgsq,ygsq,zgsq)	transforms_gse_to_gsq: GSE -> GSEQ system	P. Robert, CRPE, 1992
subroutine	t_gse_to_mfa	(xgse,ygse,zgse,bx,by,bz,xfma,yfma,zfma)	transforms_gse_to_mfa: GSE -> MFA system	P. Robert, LPP , 2016
subroutine	t_gse_to_sr2	(xgse,ygse,zgse,rotx,roty,rotz,	transforms_gse_to_sr2: GSE -> SR2 system	P. Robert, CETP, 2001
subroutine	t_gse_to_tpn	(xgse,ygse,zgse,xo,yo,zo,xtpn,ytpn,ztpn)	transforms_gse_to_tpn: GSE -> TPN system	P. Robert, LPP , 2016
subroutine	t_gsm_to_gei	(xgsm,ygsm,zgsm,xgei,ygei,zgei)	transforms_gsm_to_gei: GSM -> GEI system	P. Robert, CRPE, 1992
subroutine	t_gsm_to_geo	(xgsm,ygsm,zgsm,xgeo,ygeo,zgeo)	transforms_gsm_to_geo: GSM -> GEO system	P. Robert, CRPE, 1992
subroutine	t_gsm_to_gse	(xgsm,ygsm,zgsm,xgse,ygse,zgse)	transforms_gsm_to_gse: GSM -> GSE system	P. Robert, CRPE, 1992
subroutine	t_gsm_to_gseq	(xgsm,ygsm,zgsm,xgsq,ygsq,zgsq)	transforms_gsm_to_gsq: GSM -> GSEQ system	P. Robert, CRPE, 2002
subroutine	t_gsm_to_mag	(xgsm,ygsm,zgsm,xmag,ymag,zmag)	transforms_gsm_to_mag: GSM -> MAG system	P. Robert, CRPE, 2002
subroutine	t_gsm_to_sm	(xgsm,ygsm,zgsm,xsma,ysma,zsma)	transforms_gsm_to_sma: GSM -> SM system	P. Robert, CRPE, 1992
subroutine	t_gsm_to_tpn	(xgsm,ygsm,zgsm,xo,yo,zo,xtpn,ytpn,ztpn)	transforms_gsm_to_tpn: GSM -> TPN system	P. Robert, LPP , 2016
subroutine	t_gseq_to_gei	(xgsq,ygsq,zgsq,xgei,ygei,zgei)	transforms_gsq_to_gei: GSEQ-> GEI system	P. Robert, CRPE, 1992
subroutine	t_gseq_to_geo	(xgsq,ygsq,zgsq,xgeo,ygeo,zgeo)	transforms_gsq_to_geo: GSEQ-> GEO system	P. Robert, CRPE, 1992
subroutine	t_gseq_to_gse	(xgsq,ygsq,zgsq,xgse,ygse,zgse)	transforms_gsq_to_gse: GSEQ-> GSE system	P. Robert, CRPE, 1992
subroutine	t_gseq_to_gsm	(xgsq,ygsq,zgsq,xgsm,ygsm,zgsm)	transforms_gsq_to_gsm: GSEQ-> GSM system	P. Robert, CRPE, 2002
subroutine	t_mag_to_gei	(xmag,ymag,zmag,xgei,ygei,zgei)	transforms_mag_to_gei: MAG -> GEI system	P. Robert, CRPE, 1992
subroutine	t_mag_to_geo	(xmag,ymag,zmag,xgeo,ygeo,zgeo)	transforms_mag_to_geo: MAG -> GEO system	P. Robert, CRPE, 1992
subroutine	t_mag_to_gsm	(xmag,ymag,zmag,xgsm,ygsm,zgsm)	transforms_mag_to_gsm: MAG -> GSM system	P. Robert, CRPE, 2002
subroutine	t_mag_to_sm	(xmag,ymag,zmag,xsma,ysma,zsma)	transforms_mag_to_sma: MAG -> SM system	P. Robert, CRPE, 1992
subroutine	t_sm_to_gei	(xsma,ysma,zsma,xgei,ygei,zgei)	transforms_sma_to_gei: SM -> GEI system	P. Robert, CRPE, 1992
subroutine	t_sm_to_geo	(xsma,ysma,zsma,xgeo,ygeo,zgeo)	transforms_sma_to_geo: SM -> GEO system	P. Robert, CRPE, 1992
subroutine	t_sm_to_gsm	(xsma,ysma,zsma,xgsm,ygsm,zgsm)	transforms_sma_to_gsm: SM -> GSM system	P. Robert, CRPE, 1992
subroutine	t_sm_to_mag	(xsma,ysma,zsma,xmag,ymag,zmag)	transforms_sma_to_mag: SM -> MAG system	P. Robert, CRPE, 1992
subroutine	t_sph_to_car	(r,teta,phi,x,y,z)	transforms_sph_to_car: SPH -> CAR system	P. Robert, CRPE, 1992
subroutine	t_sr2_to_gse	(xsr2,ysr2,zsr2,rotx,roty,rotz,	transforms_sr2_to_gse: SR2 -> GSE system	P. Robert, CETP, 2001
subroutine	t_sr2_to_mfa	(xsr2,ysr2,zsr2,bx,by,bz,rox,roy,roz,	transforms_sr2_to_mfa: SR2 -> MFA system	P. Robert, CETP, 2001
subroutine	t_sr2_to_sr	(xsr2,ysr2,spifre,spipha,deltaT,xsre,ysre)	transforms_sr2_to_sre: SR2 -> SRef system	P. Robert, CRPE, 2001
subroutine	t_sr_to_sr2	(xsre,ysre,spifre,spipha,deltaT,xsr2,ysr2)	transforms_sre_to_sr2: SRef-> SR2 system	P. Robert, CRPE, 2001
subroutine	t_vdh_to_geo	(xvdh,yvdh,zvdh,rlat,rlong,xgeo,ygeo,zgeo)	transforms_vdh_to_geo: VDH -> GEO system	P. Robert, CRPE, 1992
subroutine	t_xyz_to_vdh	(x,y,z,a1,a2,a3,v,d,h)	transforms_xyz_to_vdh: xyz spinning -> VDH	P. Robert, SDev, 2020
subroutine	t_vdh_to_xyz	(v,d,h,a1,a2,a3,x,y,z)	transforms_vdh_to_xyz: VDH -> xyz spinning	P. Robert, SDev, 2020

24.2.1.8 lib_gainant.f90

nb. of lines : 278 nb. of subroutine: 2

subroutine	initial	(ifcod,dirfic)	initialisation du file code et du directory des f.calib	P. Robert, CETP, Sept 2000
subroutine	r_califile	(pathfil,ncal,freq,Preal,Pimag)	lecture du fichier des gains complexes	P. Robert, CETP, Sept 2000

24.2.1.9 lib_kepler.f90

nb. of lines : 261 nb. of subroutine: 7

subroutine	co_axes	(apo,per,a,b,c,e)	calcule grand axe et excentricite de l'ellipse	P. Robert, LPP, Novembre 2010
subroutine	co_period	(a, GM, GT)	calcul de la periode d'un satellite autour d'un corps cen	P. Robert, LPP, Novembre 2010
subroutine	co_apex_time	(GT,e, Tapex)	calcul du temp de passage au premier sommet secondaire (o	P. Robert, LPP, Novembre 2010
subroutine	co_anom_excen	(e,time,GT,AE)	calcul de l'anomalie excentrique par resolution numerique	P. Robert, LPP, Novembre 2010
subroutine	co_anom_vraie	(AE,e,a,AV,r)	calcul de l'anomalie vraie donnant la position angulaire	P. Robert, LPP, Novembre 2010
subroutine	posgei	(r,AV,a1,a2,a3,b1,b2,b3,sfac,x,y,z)	calcul la position dans le GEI depuis l'anomalie vraie	P. Robert, LPP, Novembre 2010
subroutine	U_cross_V	(Ux,Uy,Uz,Vx,Vy,Vz,Wx,Wy,Wz,sina,cosa)	calcule le produit vectoriel de U et V en double precisio	P. Robert, LPP, Novembre 2010

24.2.1.10 lib_rw_copolar.f90

nb. of lines : 673 nb. of subroutine: 6

subroutine	r_copolar	(idset,fultit,projet,experi,sid,isat,fnat,		
subroutine	rheacopolar	(idset,fultit,projet,experi,sid,isat,fnat,		
subroutine	rdatcopolar	(tspea,tspeb,dt,df,nt,nf,		
subroutine	w_copolar	(fich2,idset,fultit,projet,experi,sid		
subroutine	wrdatcopolar	(tspea,tspeb,dt,df,nt,nf,spebx,speby,		

24.2.1.11 lib_igrf13.f90

nb. of lines : 768 nb. of subroutine: 2

subroutine	igrf13syn	(isv,date,itype,alt,colat,elong,x,y,z,f)	compute_IGRF magnetic field from the 13th generation IGRF	IAGA Working Group V-MOD. f90: P. R. Nov 2021
subroutine	co_igrf	(year,r,theta,phi,Bx,By,Bz,B0)	compute_IGRF magnetic field from the 13th generation IGRF	P. Robert, ScientiDev, Nov 2021 (only this routine)

24.2.2 CLUSTER libraries**24.2.2.1 lib_rw_cef.f90**

nb. of lines : 1554 nb. of function : 3 nb. of subroutine: 18 nb. of module : 1

module	_def_cef		modules for CEF type definitions	L. Mirioni, LPP, 2008
function	get_cef_paramC	(param,ifc,data_reach,end_of_search)	Read cef file until searched Char. param. & return it	P. Robert, LPP, 2011 Jan.
function	get_cef_paramC_new	(param,ifc,data_reach,end_of_search)	Read cef file until searched Char. param. & return it	R. Pièrre, LPP, 2012 Feb.
subroutine	go_back_to_the_beginning	(ifc, param_name)	Rewind a file until START_VARIABLE=param_name is found	R. Pièrre, LPP, 2012 Feb.
subroutine	go_to_keyword	(param, ifc)	Read cef file until searched parameter is found	R. Pièrre, LPP, 2011 Nov.
subroutine	cef_find_n_include	(ifc,n_include)	Find number of include files in the cef header	R. Pièrre, LPP, 2011 Nov.
subroutine	cef_find_n_metadata	(ifc,n_metadata)	Find number of metadata in the cef header	R. Pièrre, LPP, 2011 Nov.
subroutine	cef_find_n_variables	(ifc,n_variables)	Find number of variables in the cef header	R. Pièrre, LPP, 2011 Nov.
subroutine	cef_R_include	(ifc,n_include,include_names,include_dirname)	Read include filenames in the cef file	R. Pièrre, LPP, 2011 Nov.
subroutine	cef_R_metadata	(ifc,n_metadata, metadata_names)	Read mandatory parameter in the cef file	R. Pièrre, LPP, 2011 Nov.
subroutine	cef_R_entry	(ifc,cef_entry,n_metadata, metadata_names)	Read metadata entry in the cef file	R. Pièrre, LPP, 2012 Jan.
subroutine	cef_R_variables	(ifc,n_variables,variable_names)	Read variable parameter in the cef file	R. Pièrre, LPP, 2011 Nov.
subroutine	cef_R_parameter	(ifc,cef_param,n_variables,variable_names)	Read parameters inside a variable of a cef file	R. Pièrre, LPP, 2011 Nov.
subroutine	cef_R_indexed_data	(ifc,n_of_vec_in_file,n_block,indexed_data)	Read indexed data for cef files	R. Pièrre, LPP, 2012 March
subroutine	concatenate_include_files	(ifc,include_dirname,cef_file_name,n_blo	Concatenate ceh files to CEF files if necessary	R. Pièrre, LPP, 2012 Jan.
subroutine	w_cef_meta	(output_file_unit,dummy_cef_meta)	Writes Meta Data dummy_cef_meta in output_file_unit	L. Mirioni, LPP, 2005, Rev RP 2011
subroutine	w_cef_parameter	(output_file_unit,dummy_var)	Writes var Data dummy_var in file output_file_unit.	L. Mirioni, LPP, 2005, Rev RP 2011
subroutine	u_cef_clear_meta	(dummy_cef_meta)	Fills each field of dummy_cef_meta with blanks.	L. Mirioni, LPP, 2005, Rev RP 2011
subroutine	u_cef_clear_var	(dummy_var)	Fills each field of dummy_var with blanks	L. Mirioni, LPP, 2005, Rev RP 2011
function	u_present_iso_time	()	Present time in ISO format	L. Mirioni, LPP, 2008
subroutine	suppress_blank	(arg_c)	supprime les blancs d'une chaine de caractere	R. Pièrre, 2011
subroutine	suppress_tab	(arg_c)	supprime les tabulations d'une chaine de caractere	R. Pièrre, LPP, 2011

24.2.2.2 lib_CLUORB.f90

nb. of lines : 980 nb. of subroutine: 20

subroutine	openorbi	(dirpath,iyear,imon,iday,ifc,ierr)	open CLUSTER orbit files	P. Robert, CRPE, 1996
subroutine	cluposvit	(iyear,imon,iday,ih,im,is,ifc,pos,vit,rev)	compute CLUSTER position & velocity of 4 S/C in GEI	P. Robert, CRPE, 1996
subroutine	cluposvit2	(iyear,imon,iday,ih,im,is,ifc,pos,vit,rev)	compute CLUSTER position & velocity of 4 S/C DBLE P.	P. Robert, CRPE, 2010
subroutine	closeorbi	(ifc)	close CLUSTER orbit files	P. Robert, CRPE, 1996
subroutine	cosatpos	(dmjdc20,ic1,ic2,ic3,ic4,satpos,satvit,satrev)	compute CLUSTER position & velocity in GEI	P. Robert, CRPE, 1996
subroutine	corevmoy	(satrev,revmoy)	compute CLUSTER mean revolution number	P. Robert, CRPE, 1996
subroutine	cvsatpos	(satpos)	compute CLUSTER position in Earth radii	P. Robert, CRPE, 1996
subroutine	cbaryclus	(s,gx,gy,gz)	compute CLUSTER position of barycentre	P. Robert, CRPE, 1996
subroutine	transclus	(s,gx,gy,gz)	translate CLUSTER position of a Gi vector	P. Robert, CRPE, 1996
subroutine	longmot	(mot,nc)	calcul la longueur utile d'une chaine de caracteres	P. Robert, CRPE, 1998
subroutine	ORBIT	(DAY,KODE,LFILE,IERROR,NSAT,X,REVNUM)	compute CLUSTER position & velocity from .orb files	ESA, 1996
subroutine	PR2000	(DAY,P)	computes precession matrix p(3,3) for conv. vector	ESA, 1996
subroutine	DJ2000	(DAY,I,J,K,JHR,MI,SEC)	computes calender date from mod. julian day 2000	ESA, 1996
subroutine	JD2000	(DAY,JEAR,MONTH,KDAY,JHR,MI,SEC)	gives the new mod. julian day	ESA, 1996
subroutine	cospingse	(iday,imon,iyear,ih,im,is,srasc,sdec,rx,ry,rz)	compute spin direction in GSE from GEI R. Asc and Dec.	P. Robert, CRPE, 1996
subroutine	gei_to_gse	(pos)	convert GEI pos. in GSE system	P. Robert, CRPE, 1996
subroutine	gei_to_gse_d6	(posvit)	convert GEI pos. in GSE system in DBLE	P. Robert, CRPE, 2001
subroutine	gei_to_gsm	(pos)	convert GEI pos. in GSM system	P. Robert, CRPE, 1996
subroutine	magtosr2	(x,y,z,nbp,rx,ry,rz)	convert GSE mag waveform in SR2 system	P. Robert, CRPE, 1996
subroutine	pantomfa	(xfo,yfo,zfo,xma,yma,zma,nbp,rx,ry,rz,	convert SR2 waveform in MFA coordinates	P. Robert, CRPE, 2001

24.2.2.3 lib_tools_CLUSTER.f90

nb. of lines : 1914 nb. of subroutine: 30

subroutine	cgeompara	(s)	compute geometric parameters of a given tetrahedron	P. Robert, CRPE, 1994
subroutine	cgeomcrit	(s, valcrit, titcrit, n)	compute geometric criterions and give their title	P. Robert, CRPE, 1994
subroutine	cdistance	(s)	compute all the distances of the tetrahedron	P. Robert, CRPE, 1994
subroutine	csurfaces		compute all the surfaces of the tetrahedron	P. Robert, CRPE, 1994
subroutine	cnormales		compute all the normales of the tetrahedron	P. Robert, CRPE, 1994
subroutine	canglefac		compute all the angles of the tetrahedron	P. Robert, CRPE, 1994
subroutine	cvolumeto	(s)	compute the volume of the tetrahedron	P. Robert, CRPE, 1994
subroutine	cspherins	(s)	compute radius & volume of sphere including tetrahedron	P. Robert, CRPE, 1994
subroutine	csomdebdl	(s, bmag)	computation of J vector inside the 4 S/C tetrahedron	P. Robert, CRPE, 1994
subroutine	ccurlinco	(iface, denjx, denjy, denjz)	compute J from a given face of the tetrahedron	P. Robert, CRPE, 1994
subroutine	csomdebds	(bmag)	computation of dib(B)	P. Robert, ScientiDev, 2020
subroutine	cdivlinco	(iface, divB)	compute div(B) from a given face of the tetrahedron	P. Robert, Scientidev 2020
subroutine	setmatrix	(tax,tay,taz, tbx,tby,tbz, tcx,tcy,tcz)	set matrices for further linear resolution system	P. Robert, CRPE, 1994
subroutine	detmatrix	(det)	compute determinant of the matrix defined by setmat	P. Robert, CRPE, 1994
subroutine	invmatrix	(gx,gy,gz,px,py,pz)	compute inverse matrix determined by setmat and detmat	P. Robert, CRPE, 1994
subroutine	cellicrit	(s,a,b,c,dirax,gl,gf)	Compute criterions defined by J. Schoenmaker	P. Robert, CRPE, 1994
subroutine	cellipara	(s,scalax,vectax)	computation of the inertial ellipsoide	P. Robert, CRPE, 1994
subroutine	cbaryclus	(s,gx,gy,gz)	compute barycentre of 4 points of cluster	P. Robert, CRPE, 1994
subroutine	transclus	(s,gx,gy,gz)	tranlate position of the 4 points of cluster of gx,gy,gz	P. Robert, CRPE, 1994
subroutine	quaellips	(POS,SAX,DIR)	compute the properties of the ellipse for a tetrahedron	J. Schoenmaekers, ESOC, 1993
subroutine	CEIGENVAL	(A,R,N,MV)	computes eigenvalues & eigenvectors of real symm. matrix	J. Schoenmaekers, ESOC, 1993
subroutine	inipobary	(s)	set the initial position of 4 S/C for barycent. coord.	G. Chanteur, P. Robert, CRPE, 1994
subroutine	inimabary	(bmag)	set the the magnetic field values for barycent. coord.	G. Chanteur, P. Robert, CRPE, 1994
subroutine	calbabary	(qfbary)	compute barycentric coordinates	G. Chanteur, P. Robert, CRPE, 1994
subroutine	estibbary		Estimate vectorial quantities of B by barycent. coord.	G.Chanteur, CETP, 1994
subroutine	ccurlbary	(curx,cury,curz)	compute curl(B) from barycentric method	G. Chanteur, P. Robert, CRPE, 1994
subroutine	ccurvabary	(mormax,mormay,mormaz,rcurv)	compute radius of curvature of field lines	G.Chanteur, CETP, 2003
subroutine	cgrabary	(gra1x,gra1y,gra1z, gra2x,gra2y,gra2z,	compute grad(B) matrix from barycentric metho	G.Chanteur, CETP, 1994
subroutine	cdivbary	(divdeb)	compute div(B) from barycentric method	G. Chanteur, P. Robert, CRPE, 1994
subroutine	ctestbary	(iok)	compute test on gradient sign from barycentric method	G. Chanteur, P. Robert, CRPE, 1994

24.3 Total code size

The **src** directory contains 68 programs used by RPC commands, 14 libraries, approximately 750 subroutines, and 14 functions. All code is written in f90 doing around 70,000 lines long.

The **bash** directory contains about 150 RPC commands, written in bash shell script, and doing about 15,000 shell lines. About half of these commands launch an executable program made from f90 source code.

The **script** directory contains about 30 RPC scripts (shell using several RPC commands, constituting a kind of "RPC macro command"). These scripts do approximately 3400 lines of shell scripts.

The **tests** directory contains about 45 RPC scripts doing approximately 1100 lines of shell scripts.

Between f90 and shell, the whole code do around 90,000 lines.

Chapter 25

LIST OF RPC COMMANDS

25.1 Alphabetical list of all RPC commands

RPC_add_DxDy_to_BxBy	RPC_download_data_CLUFGM
RPC_alitime_4_vectime	RPC_download_data_CLUPOS
RPC_cef_to_rff_CLUFGM	RPC_download_data_oneday_CLUFGM
RPC_cef_to_rff_CLUPOS	RPC_download_data_oneday_CLUPOS
RPC_cef_to_rff_cwf_CLUSTA	RPC_download_data_oneday_t1t2_CLUFGM
RPC_cef_to_rff_dwf_CLUSTA	RPC_download_data_oneday_t1t2_CLUPOS
RPC_change_coordinate_system	RPC_download_data_oneday_t1t2_CWF_CLUSTA
RPC_check_dirname_tree	RPC_download_data_oneday_t1t2_DWF_CLUSTA
RPC_check_rff	RPC_download_data_onemonth_CLUFGM
RPC_check_rff_dir	RPC_download_data_onemonth_CLUFGM_nolog
RPC_chmod_tree	RPC_download_data_onemonth_CLUPOS
RPC_clean_dir	RPC_download_data_onemonth_CLUPOS_nolog
RPC_clean_rff	RPC_download_data_oneyear_CLUFGM
RPC_compare_versions	RPC_download_data_oneyear_CLUPOS
RPC_compare_versions_long	RPC_encode_datiso
RPC_compute_curl_div_4sat	RPC_flat_to_rff
RPC_compute_sat_orbit_param	RPC_get_data_CLUFGM
RPC_compute_sat_trajectory	RPC_get_data_CLUFGM_4sat
RPC_convert_names	RPC_get_data_CLUGEOM
RPC_copy_rff	RPC_get_data_CLUPOS
RPC_copy_rff_database	RPC_get_data_CLUPOS_4sat
RPC_create_simulated_data	RPC_get_data_CLUSTA_VTL1
RPC_current_date	RPC_get_data_CLUSTA_VTL2
RPC_current_tube	RPC_get_data_CLUVIT
RPC_date_time_to_datiso	RPC_get_data_CLUVIT_4sat
RPC_datiso_to_date_time	RPC_get_data_GEOGRD
RPC_decode_datim	RPC_get_data_GEOMAG
RPC_decode_datiso	RPC_get_data_GEOPOS
RPC_diff_rff	RPC_get_data_GEOULF
RPC_dir_diff	RPC_get_indexed_data_CLUFGM
RPC_dir_pretty_tree	RPC_get_indexed_data_CLUPOS
RPC_dir_properties	RPC_get_indexed_data_GEOMAG
RPC_dir_properties_pretty_tree	RPC_get_indexed_data_GEOPOS
RPC_dir_properties_tree	RPC_give_rff_param
RPC_dir_save	RPC_give_spin_dir_CLUSTER
RPC_dir_size	RPC_info_rff
RPC_dir_size_pretty_tree	RPC_info_rff_dir
RPC_dir_size_tree	RPC_join_vectime
RPC_doc	RPC_list
RPC_dos_to_unix_all_files	RPC_list_block_WF

RPC_list_cef_database_CLUSTER
 RPC_list_days_of_month
 RPC_list_rff_database
 RPC_make_minidoc
 RPC_menu
 RPC_menu_CLUSTER
 RPC_menu_GEOGRD
 RPC_menu_GEOS
 RPC_move_rff
 RPC_nday_of_month
 RPC_next_day
 RPC_previous_day
 RPC_ps_to_pdf
 RPC_ps_to_png
 RPC_ps_to_png_16m
 RPC_ps_to_png_256
 RPC_purge_cef_database_CLUSTER
 RPC_put_cef_database_CLUSTER
 RPC_put_rff_database
 RPC_reduce_time_rff
 RPC_reduce_time_vectime
 RPC_search_duplicates
 RPC_spectro_L2_to_cef_CLUSTER
 RPC_spectro_to_polar
 RPC_spectro_xyz_to_lrz
 RPC_system_info
 RPC_test_kepler_lib
 RPC_unix_to_dos_all_files
 RPC_vectime_calibration_CLUSTER
 RPC_vectime_calibration_GEOGRD
 RPC_vectime_calibration_GEOULF

RPC_vectime_gse_to_isr2
 RPC_vectime_L1_to_cef_CLUSTER
 RPC_vectime_L1_to_spectro_L2_CLUSTER
 RPC_vectime_L1_to_spectro_L2_GEOGRD
 RPC_vectime_L1_to_spectro_L2_GEOULF
 RPC_vectime_L2_to_cef_CLUSTER
 RPC_vectime_to_indexed_data
 RPC_vectime_to_mfa
 RPC_vectime_to_mfa_CLUSTER
 RPC_vectime_to_mfav_GEOGRD
 RPC_vectime_to_mfav_GEOULF
 RPC_vectime_to_mva
 RPC_vectime_to_spectro
 RPC_vectime_vdh_to_srv_GEOMAG
 RPC_vectime_vdh_to_xyz_GEOMAG
 RPC_version
 RPC_visu_2_vectime
 RPC_visu_2_vectime_3D
 RPC_visu_ave_spectrum
 RPC_visu_CLUSTERGEOM
 RPC_visu_curl_div_4sat
 RPC_visu_polar
 RPC_visu_spectro
 RPC_visu_spectro_4Bz
 RPC_visu_vectime
 RPC_visu_vectime_3D
 RPC_visu_vectime_3D_4sat
 RPC_visu_vectime_4sat
 RPC_waveform_to_vectime
 RPC_zip_cef_database_CLUSTER

25.2 Functional list of RPC commands

25.2.1 Generic commands

• Software Informations

RPC_menu	: display the fonctionnal list of RPC commands in html.
RPC_doc	: display the documentation in HTML
RPC_list	: return list of all RPC commands; create RPC_list.txt
RPC_version	: return current RPC version used, ex: RPC_V5p2
RPC_compare_versions	: compare two different versions of RPC package
RPC_compare_versions_long	: same as previous but more deeper
RPC_make_minidoc	: create file mini_doc.txt

• RFF file handling

RPC_check_rff	: check content of a RFF file
RPC_check_rff_dir	: check of all RFF files of a given directory
RPC_info_rff	: return main info of given RFF file
RPC_info_rff_dir	: return main info of all RFF files of a given directory
RPC_clean_rff	: clean content of a RFF file
RPC_copy_rff	: copy a RRF file into another, name updated
RPC_diff_rff	: compute difference between 2 RFF files
RPC_rename_rff	: rename a RFF file with update of file name inside
RPC_give_rff_param	: give header a RFF file
RPC_put_rff_database	: move given file in the right data base
RPC_copy_rff_database	: copy given file in the right data base
RPC_list_rff_database	: according mission / manip
RPC_join_vectime	: join 2 rff vectime files of same experiment in a single file

• RFF file processing

RPC_add_DxDy_to_BxBy	: add Dx and Dy to Bx and By
RPC_alitime_4_vectime	: time alignment of 4 rff vectime files
RPC_compute_curl_div_4sat	: compute curl & div from MAG & POS data
RPC_change_coordinate_system	: on VecTime files only
RPC_compute_sat_orbit_param	: compute orbital parameters of a Earth satellite
RPC_compute_sat_trajectory	: compute elliptical trajectory of a Earth satellite
RPC_create_simulated_data	: create ULF, MAG or POS simulated vectime files
RPC_current_tube	: create a simulated file compute_curl_div_4sat.resu
RPC_reduce_time_rff	: reduce the time period of a RFF file
RPC_reduce_time_vectime	: reduce the time period of a VT RFF file (more fast)
RPC_spectro_to_polar	: produce a polar file from a Spectrogram file
RPC_spectro_xyz_to_lrz	: convert a Spectrogram in XYZ into a Left-Right-Z one
RPC_vectime_gse_to_isr2	: change coordinate system from GSE to ISR2
RPC_vectime_to_mfa	: change coordinate system to MFA
RPC_vectime_to_mva	: change coordinate system to MVA
RPC_vectime_to_spectro	: produce a Spectrogram file from a VecTime file
RPC_vectime_to_indexed_data	: create indexed data file from a rff VT file
RPC_waveform_to_vectime	: convert a RFF ofWaveform class to one of Vectime class
RPC_test_kepler_lib	: test components of the lib_Kepler
RPC_list_block_WF	: return the list of all blocks in a WF file

• Visualization tools

RPC_visu_vectime	: visualization of a VecTime RFF file (2D)
RPC_visu_vectime_3D	: visualization of a VecTime RFF file (3D, for orbit data)
RPC_visu_2_vectime	: visualization of 2 Vectime RFF files
RPC_visu_2_vectime_3D	: 3-D visualization of 2 Vectime RFF files
RPC_visu_vectime_4sat	: visualization of 4 VT files for 4 S/C
RPC_visu_vectime_3D_4sat	: 3D visualization of 4 VT files for 4 S/C
RPC_visu_spectro	: visualization of a Spectrogram RFF file
RPC_visu_spectro_4Bz	: visualization of 4 spectrogrammes Bz of 4 S/S
RPC_visu_ave_spectrum	: visualization of an average spectrum from Spectrogram file
RPC_visu_polar	: visualization of a copolar.resu file

• Date conversion

RPC_current_date	: return 2012-08-28 14:08:16 day 241 Julsec 1346155696
RPC_next_day	: return date of next day, as '20070925'
RPC_previous_day	: return date one day before, as '20070925'
RPC_nday_of_month	: return number of day in a month
RPC_list_days_of_month	: return a list of day for a given year/month as 01 02 03 30
RPC_date_time_to_datiso	: return ISO_date from yymmdd hhmmssc
RPC_datiso_to_date_time	: return yymmdd hhmmss from ISO_date
RPC_decode_datim	: for 20010923_091703 return 2001 09 23 09 17 03
RPC_decode_datiso	: for 2001-09-23T09:17:03.000Z return 2001 09 23 09 17 03
RPC_encode_datiso	: return ISO_date from year month day hour min sec

• PostScript conversion

RPC_ps_to_pdf	: create PDF file with or without image compression
RPC_ps_to_png	: create PNG file for a given resolution and color number
RPC_ps_to_png_256	: create PNG file for given dpi and 256 colors
RPC_ps_to_png_16m	: create PNG file for given dpi and 16M colors

• System and Directories

RPC_clean_dir	: remove *.in, *.out, *.old, *.resu from current dir.
RPC_chmod_tree	: set mode to all files in the tree, from current directory
RPC_convert_names	: convert recursively files name having blank or special character
RPC_check_dirname_tree	: check dir name in the tree having blank in the name
RPC_dir_pretty_tree	: give directories for all the tree
RPC_dir_save	: save recursively a directory in another one
RPC_dir_size	: give directory size (octets or Mo)
RPC_dir_size_tree	: give directory size (octets or Mo) for all the tree
RPC_dir_size_pretty_tree	: same with pretty presentation
RPC_dir_properties	: give directory size (octets or Mo), files #/directories #
RPC_dir_properties_tree	: give directory properties for all the tree
RPC_dir_properties_pretty_tree	: same with pretty presentation
RPC_dir_diff	: compare recursively two directories
RPC_search_duplicates	: search and list duplicate files in a tree
RPC_system_info	: return system information (host, system, OS etc.)
RPC_dos_to_unix_all_files	: convert all files in tree from DOS to UNIX format
RPC_unix_to_dos_all_files	: convert all files in tree from UNIX to DOS format

• Files conversion

RPC_flat_to_rff	: conversion of a flat file to a rff file
-----------------	---

25.2.2 Special GEOS commands

- Access to local database

RPC_get_data_GEOULF	: get GEOS/ULF VTL2
RPC_get_data_GEOMAG	: get GEOS/MAG data
RPC_get_data_GEOPOS	: get GEOS/POS data
RPC_get_indexed_data_GEOMAG	: get GEOS/MAG indexed data
RPC_get_indexed_data_GEOPOS	: get GEOS/POS indexed data

- Special commands

RPC_menu_GEOS	: display GEOS menu in html.
RPC_vectime_calibration_GEOULF	: calibrate a L1 GEOS/ULF RFF Vectime file
RPC_vectime_L1_to_spectro_L2_GEOULF	: compute L2 spectrogram from VTL1 file
RPC_vectime_to_mfav_GEOULF	: change coordinate of GEOS/ULF from SRV to MFAV
RPC_VDH_to_srv_GEOMAG	: change coordinate of GEOS/S331 from VDH to SRV
RPC_VDH_to_xyz_GEOMAG	: change coordinate of GEOS/S331 from VDH to XYZ

25.2.3 Special GEOS/GROUND commands

- Access to local database

RPC_get_data_GEOGRD	: get VTL1 or VTL2 ULF ground data
---------------------	------------------------------------

- Special commands

RPC_menu_GEOGRD	: display GEOGRD menu in html.
RPC_vectime_calibration_GEOGRD	: calibrate a L1 GEOGRD/ULF RFF Vectime file
RPC_vectime_L1_to_spectro_L2_GEOGRD	: compute L2 spectrogram from VTL1 file
RPC_vectime_to_mfav_GEOGRD	: change coordinate of GEOGRD/ULF from NWV to MFAV

25.2.4 Special CLUSTER commands

• CLUSTER general

RPC_menu_CLUSTER	: display CLUSTER menu in html.
RPC_give_spin_dir_CLUSTER	: give spin direction in GEI system
RPC_visu_spectro_4Bz	: visualization of 4 spectrogrammes Bz of 4 S/S
RPC_visu_vectime_4sat	: visualization of 4 VT files for 4 S/C
RPC_list_cef_database_CLUSTER	: list of rff files present in the datafile
RPC_purge_cef_database_CLUSTER	: remove all cef files of cef database
RPC_put_cef_database	: update cef database by the given file
RPC_zip_cef_database_CLUSTER	: compress all cef of the cef database

•STAFF-SC commands

RPC_download_data_oneday_t1t2_DWF_CLUSTER	: download DCW data from CAA a day between t1-t2
RPC_download_data_oneday_t1t2_CWF_CLUSTER	: download CWF data from CAA a day between t1-t2
RPC_cef_to_rff_CLUSTER_dwf	: convert a STAFF DWF.cef file into a VTL1.rff
RPC_cef_to_rff_CLUSTER_cwf	: convert a STAFF CWF.cef file into a VTL2.rff
RPC_get_data_CLUSTER_VTL1	: get CLUSTER/STA VTL1
RPC_get_data_CLUSTER_VTL2	: get CLUSTER/STA VTL2
RPC_vectime_calibration_CLUSTER	: calibrate a VTL1 STAFF-SC file
RPC_vectime_L1_to_spectro_L2_CLUSTER	: calibrate a VTL1 STA file and make spectrogram
RPC_vectime_to_mfa_CLUSTER	: transform CLUSTER VTL2 in GSE or ISR2 into MFA
RPC_add_DxDy_to_BxBy	: add Dx and Dy to Bx and By

• FGM commands

RPC_download_data_oneday_t1t2_CLUFGM	: download FGM data from CAA a day between t1-t2
RPC_download_data_oneday_CLUFGM	: download entire day of FGM data from CAA
RPC_download_data_onemonth_CLUFGM	: download CLUFGM.rff files 1 month 4 sat
RPC_download_data_oneyear_CLUFGM	: download entire year of FGM data from CAA
RPC_download_data_CLUFGM	: download FGM data from CAA for several days
RPC_cef_to_rff_CLUFGM	: convert a fgm.cef file into a fgm_VT.rff file
RPC_get_data_CLUFGM	: get CLUSTER/FGM from local database
RPC_get_data_CLUFGM_4sat	: get CLUSTER/FGM from local database
RPC_get_indexed_data_CLUFGM	: get CLUSTER/POS indexed data from data base
RPC_vectime_gse_to_isr2	: change coordinate system from GSE to ISR2
RPC_alitime_4_vectime	: time alignment of 4 rff vectime files
RPC_compute_curl_div_4sat	: compute curl & div from MAG & POS data
RPC_visu_curl_div_4sat	: visualization of compute_curl_div_4sat.resu

•Special POS commands

RPC_download_data_oneday_t1t2_CLUPOS	: download Pos. and Vel. from CAA a day between t1-t2
RPC_download_data_oneday_CLUPOS	: download entire day of POS data from CAA
RPC_download_data_onemonth_CLUPOS	: download CLUPOS.rff files 1 month 4 sat
RPC_download_data_oneyear_CLUPOS	: download entire year of POS data from CAA
RPC_download_data_CLUPOS	: download POS data from CAA for several days
RPC_cef_to_rff_CLUPOS	: convert a POS.cef file into a POS_VT.rff file
RPC_get_data_CLUPOS	: get CLUSTER/POS from local data base
RPC_get_data_CLUPOS_4sat	: get CLUSTER/POS
RPC_get_data_CLUVIT	: get CLUSTER/VIT
RPC_get_data_CLUVIT_4sat	: get CLUSTER/VIT
RPC_get_data_CLUGEOM	: get CLUSTER/GEOMETRY of 4 S/S
RPC_get_indexed_data_CLUPOS	: get CLUSTER/POS indexed data from data base
RPC_visu_vectime_3D_4sat	: 3D visualization of 4 VT files for 4 S/C
RPC_visu_CLUGEOM	: visualization of Cluster geometry over 24h

Chapter 26

BIBLIOGRAPHY

[1] **Patrick Robert**, "Les commandes RCL pour le traitement de masse des données de GEOS/STAFF-SC", Version 3.0, Avril 2015, CNRS/LPP.

ftp://ftp.lpp.polytechnique.fr/robert/keep/HERITAGE/Les_commandes_RCL_V3p0/Les_commandes_RCL_V3p0.pdf

[2] **A. Allen, S.J.Schwartz, C. Harvey, C. Perry, C. Huc, P. Robert** "CLUSTER Exchange Format - Data File Syntax", CSDS Archive Task Group, October 19, 2009.

<http://www.sp.ph.ic.ac.uk/csc-web/DOCS/DS-QMW-TN-0010.pdf>

[3] **Patrick Robert**, "The Roproc File Format, A dedicated file format for vectorial data processing", Version 2.2, February 2011, First Version 1.0, November 2004, CNRS/LPP.

ftp://ftp.lpp.polytechnique.fr/robert/keep/ROPROC/The_Roproc_File_Format_20110216.pdf

[4] **Patrick Robert**, "ROCOTLIB: a Coordinate Transformation Library for SolarTerrestrial studies", , update of CETP Internal report n° RI-CETP/01/2003, version 1.8, Novembre 2003, revised version 3.0, December 2017

ftp://ftp.lpp.polytechnique.fr/robert/keep/HERITAGE/Rocotlib_V3p0/doc_ROCOTLIB_V3p0.pdf

[5] **Patrick Robert and C. de Villedary** , "Transformation of a STAFF waveform into a Magnetic Field Aligned coordinate system", Rapport interne n° RI-CETP/6/2000, Octobre 2000.

ftp://ftp.lpp.polytechnique.fr/robert/keep/Biblio_et_CV/Working_documents/2000_Robert_RICETP_Cluster_data_processing_STAFF_to_MFA.pdf

[6] **Patrick Robert**, "An easy method to compute plane waves polarisation from a three components waveform", Document de travail CRPE, 1980. Révision et application à GEOS, Aout 2008.

ftp://ftp.lpp.polytechnique.fr/robert/keep/HERITAGE/Easy_method_to_compute_polarisation/Easy_method_to_compute_polarisation.pdf

[7] **P. Robert, A. Roux, C.C. Harvey, M.W. Dunlop, P.W. Daly, and K.H. Glassmeier** "Tetrahedron geometric Factors", Analysis Method for Multi-Spacecraft Data, Eds. G. Paschmann and P.W. Daly, International Space Science Institute, Bern, Switzerland, pp. 323-348, 1998. (ESA Publication Division, SR-001, July 1998).

ftp://ftp.lpp.polytechnique.fr/robert/keep/Biblio_et_CV/Publications/1998_Robert_ISSI-ESA_Tetrahedron_geometric_factors.pdf

[8] **P. Robert et al.**, CLUSTER/STAFF search coils magnetometer calibration - Comparisons with FGM", in Special Issue: Calibration methods and results of the in-situ experiments on GEOS and Double Star, Research Article, gi-2013-15

<http://www.geosci-instrum-method-data-syst.net/3/153/2014/>

[9] **P. Robert**, "Intensité et polarisation des ondes UBF détectées à bord de GEOS-1, Méthode d'analyse numérique du signal et production en routine de sommaires expérimentateurs, Problèmes rencontres et solutions pratiques", Note Technique CRPE/ETE/71, May 1979.

ftp://ftp.lpp.polytechnique.fr/robert/keep/Biblio_et_CV/Working_documents/1979_Robert_NTCRPE71_Intensite_et_Polarisation_des_ondes_UBF.pdf

[10] **P. Robert**, "Measurement by the S300 experiment of two components of the DC magnetic field", DTCRPE, CRPE, CNET/ETE, 1979

ftp://ftp.lpp.polytechnique.fr/robert/keep/Biblio_et_CV/Working_documents/1979_Robert_DTCRPE_Measurement_by_S300_of_2comp_DC_field.pdf

[11] **P. Robert**, "Cross calibration meeting"

fftp://ftp.lpp.polytechnique.fr/robert/keep/Biblio_et_CV/ESA_CrossCal_Meetings/

[12] **P. Robert, C. Burlaud and M. Maksimovic**, "Calibration Report of the STAFF Measurements in the Cluster Active Archive (CAA) ", CAA-STA-CR-002, 2012

ftp://ftp.lpp.polytechnique.fr/robert/keep/Biblio_et_CV/Working_documents/2012_Robert_CAA_Calibration_Report_V3p0.pdf

[13] **P. Robert et al**, "CLUSTER–STAFF search coil magnetometer calibration –comparisons with FGM", Geosci. Instrum. Method. Data Syst., 3, 153–177, 2014

ftp://ftp.lpp.polytechnique.fr/robert/keep/Biblio_et_CV/Publications/2014_Robert_Cluster-Staff_calibration_gi-3-153-2014.pdf

[14] **P. Robert and M. W. Dunlop**, " Use of twenty years CLUSTER/FGM data to observe the mean behavior of the magnetic field and current density of Earth's magnetosphere", Journal of Geophysical Research: Space Physics, 127, e2021JA029837,(2022). <https://doi.org/10.1029/2021JA029837>

- NOTES -