

**Centre National de la Recherche Scientifique**  
**Laboratoire de Physique des Plasmas**

UMR 7648  
Ecole Polytechnique, route de Saclay, 91128 PALAISEAU CEDEX

**Les commandes RCL pour le traitement de masse  
des données de CLUSTER/STAFF-SC**

---

**Par**  
**Patrick ROBERT**

Version 1.0 : 2011-05-12  
Version 2.0 : 2012-01-17  
Version 3.0 : 2015-04-16



## AVANT PROPOS

Ce document a été écrit entre le mois de mai 2011 et celui de mai 2015. Il résume une grande partie des codes développés pour le « CLUSTER ACTIVE ARCHIVE », devenu en 2015 le « CLUSTER SCIENCE ARCHIVE ». A partir de ce document, un utilisateur connaissant les commandes de base d'Unix devrait pouvoir installer le logiciel correspondant et l'utiliser sans problèmes.

Prévu initialement pour traiter en exploitation de masse les données STAFF-SC, FGM et les trajectoires de CLUSTER, ce logiciel comprends un certain nombre de commandes génériques qui peuvent être appliquées à n'importe quel type de données de type séries temporelles, comme les search-coils ou les magnétomètres. Pour les commandes dédiées à CLUSTER, comme la calibration des formes d'onde, le programme correspondant peut facilement être adapté pour d'autres expériences du même type, tant que l'on connaît les fonctions de transfert de l'instrument.

Merci à Rodrigue Piberne pour sa patience à relire ce document et pour ses suggestions d'amélioration.



## Table des matières

<b>1. INTRODUCTION.....</b>	<b>15</b>
1.1. Rappel sur les procédures scientifiques Roproc .....	15
1.2. Les commandes RCL en général.....	16
1.3. Les commandes RCL pour le traitement de STAFF-SC.....	17
1.4. Quelques mots sur le format RFF.....	17
<b>2. LES CHAINES DE PRODUCTION STAFF-SC .....</b>	<b>19</b>
2.1. Les deux types de chaînes pour les RFF et CEF .....	19
2.2. Les produits RFF dans les bases locales LPP.....	19
2.2.1 Les WFL1.rff.....	19
2.2.2 Les VTL1.rff.....	20
2.2.3 Les VTL2.rff.....	20
2.2.4 Les SPL2.rff.....	20
2.3. Les produits STAFF-SC au CSA .....	21
2.3.1 Les DWF .....	21
2.3.2 Les CWF.....	21
2.3.3 Les CS .....	21
2.3.4 Les graphiques « quick looks ».....	21
2.4. Organisation des bases de données locales .....	21
2.4.1 Stratégie de conservation .....	21
2.4.2 Nomenclature des noms de fichiers .....	22
2.4.3 Arborecence des fichiers sur les bases locales.....	22
• Arborecence des fichiers RFF et PNG .....	23
• Arborecence des fichiers CEF .....	23
2.5. Description des chaînes de production .....	25
2.5.1 Production des WFL1.rff.....	25
2.5.2 Production des VTL1.rff.....	26
2.5.3 Production des SPL2.rff.....	26
2.5.4 Production des VTL2.rff.....	27
2.5.5 Production des DWF.cef.....	28
2.5.6 Production des CS.cef.....	29
2.5.7 Production des CWF.cef.....	30
2.5.8 Remarque sur les produits livrés au CSA.....	30
2.5.9 Chaîne de production des images WEB.....	30
2.5.10 Validation des CWF.....	31
2.5.11 Schéma général de la chaîne STAFF .....	33
<b>3. ORGANISATION DE LA PRODUCTION STAFF .....</b>	<b>35</b>
3.1. Contraintes de production .....	35
3.1.1 Disponibilité des fichiers Tcor .....	35
3.1.2 Production des WFL1 et des VTL1 « Daily ».....	35
3.1.3 Production des SPL2 « Daily » et des CS .....	35
3.1.4 Production des PNG.....	35
3.1.5 Production des VTL1 et des DWF.....	35
3.1.6 Production des VTL2 et des CWF .....	35

3.2. Paramétrage des commandes de production .....	36
3.2.1 RCL_product_VTL1_oneday .....	36
3.2.2 RCL_product_VTL2_oneday .....	36
3.2.3 RCL_product_SPL2_oneday .....	37
3.2.4 RCL_product_visu_SPL2_oneday .....	37
3.2.5 RCL_product_visu_orbit_oneday .....	38
• Les positions par séquence de 3 heures .....	38
• Les positions pour une orbite complète .....	38
• Les paramètres du tétraèdre .....	38
3.3. Consommation en temps CPU des différentes chaînes .....	38
3.3.1 Traitement des données brutes .....	38
3.3.2 Traitements de niveau 1 et 2 .....	39
• Ordre de grandeur du temps de production des VTL1 et DWF .....	39
• Ordre de grandeur du temps de production des SPL2, CS et PNG .....	39
• Ordre de grandeur du temps de production des VTL2 et des CWF .....	39
3.4. Volume des données traitées .....	39
3.4.1 Les WFL1 .....	40
3.4.2 Les VTL1 .....	40
3.4.3 Les VTL2 .....	41
3.4.4 Les SPL2 .....	42
3.4.5 Les plots 4Bz en PNG .....	43
3.4.6 Résumé des volumes occupés .....	44
 4. DESCRIPTION DES COMMANDES RCL .....	 45
4.1. Informations sur le logiciel .....	45
4.1.1 RCL_menu .....	45
4.1.2 RCL_list .....	45
4.1.3 RCL_version .....	46
4.1.4 RCL_make_minidoc .....	46
4.1.5 RCL_compare_versions .....	46
4.1.6 RCL_compare_versions_long .....	46
4.2. Manipulation des fichiers RFF .....	46
4.2.1 RCL_check_rff .....	46
4.2.2 RCL_check_rff_dir .....	46
4.2.3 RCL_info_rff .....	46
4.2.4 RCL_info_rff_dir .....	46
4.2.5 RCL_clean_rff .....	47
4.2.6 RCL_copy_rff .....	47
4.2.7 RCL_diff_rff .....	47
4.2.8 RCL_move_rff .....	47
4.2.9 RCL_reduce_time_rff .....	47
4.3. Commandes de traitements génériques .....	48
4.3.1 RCL_list_block_WF .....	48
4.3.2 RCL_waveform_to_vectime .....	48
4.3.3 RCL_vectime_to_spectro .....	48
4.3.4 RCL_spectro_to_polar .....	48
4.3.5 RCL_spectro_xyz_to_lrz .....	49
4.4. Accès à la base locale CLUSTER/STAFF-SC L1 .....	49
4.4.1 RCL_get_data_CLUSTA_VTL1 .....	49
4.4.2 RCL_get_data_CLUSTA_VTL2 .....	49
4.4.3 RCL_get_data_CLUSTA_WFL1_forVTL1 .....	49
4.4.4 RCL_get_data_CLUSTA_VTL1_forVTL2 .....	49
4.4.5 RCL_get_data_CLUSTA_VTL1_forSPL2 .....	49
4.5. Accès à la base locale CLUSTER/Orbit .....	50
4.5.1 RCL_get_data_CLUPOS .....	50
4.5.2 RCL_get_data_CLUGEOM .....	50
4.5.3 RCL_give_CLUORB_period .....	50

4.6. Accès à la base locale CLUSTER / FGM L2.....	51
4.6.1 RCL_get_data_CLUFGM.....	51
4.6.2 RCL_fgm_cef_to_rff.....	51
4.6.3 RCL_fgm_cef_to_bav.....	51
4.6.4 RCL_fgm_cef_gse_to_sr2.....	51
4.7. Accès à la base distante CSA / FGM L2.....	51
4.7.1 RCL_download_data_CLUFGM_oneday_t1t2.....	51
4.7.2 RCL_download_data_CLUFGM_oneday.....	51
4.7.3 RCL_download_data_CLUFGM_onemonth.....	51
4.7.4 RCL_download_data_CLUFGM_onemonth_nolog.....	52
4.7.5 RCL_download_data_CLUFGM_oneyear.....	52
4.8. Traitements spécifiques à CLUSTER/STAFF-SC.....	52
4.8.1 RCL_vectime_L1_to_spectro_L2.....	52
4.8.2 RCL_vectime_calibration_CLUSTER.....	53
4.8.3 RCL_vectime_to_mfa.....	53
4.8.4 RCL_vectime_L1_to_cef.....	53
4.8.5 RCL_vectime_L2_to_cef.....	53
4.8.6 RCL_spectro_L2_to_cef.....	53
4.8.7 RCL_give_spin_dir.....	54
4.9. Production de masse des données STAFF-SC.....	54
4.9.1 RCL_product_VTL1_oneday.....	54
4.9.2 RCL_product_VTL1_onemonth_nolog.....	54
4.9.3 RCL_product_VTL1_onemonth.....	54
4.9.4 RCL_product_VTL1_oneyear.....	54
4.9.5 RCL_product_VTL1_daily.....	55
4.9.6 RCL_product_VTL2_oneday.....	55
4.9.7 RCL_product_VTL2_onemonth_nolog.....	55
4.9.8 RCL_product_VTL2_onemonth.....	55
4.9.9 RCL_product_VTL2_onemonth_onesat.....	55
4.9.10 RCL_product_SPL2_oneday.....	55
4.9.11 RCL_product_SPL2_onemonth_nolog.....	55
4.9.12 RCL_product_SPL2_onemonth.....	56
4.9.13 RCL_product_SPL2_oneyear.....	56
4.9.14 RCL_product_SPL2_oneday_fordaily.....	56
4.9.15 RCL_product_SPL2_daily.....	56
4.9.16 RCL_product_DWF_oneday.....	56
4.9.17 RCL_product_DWF_onemonth_nolog.....	56
4.9.18 RCL_product_DWF_onemonth.....	56
4.9.19 RCL_product_DWF_oneyear.....	57
4.9.20 RCL_product_CWF_oneday.....	57
4.9.21 RCL_product_CWF_onemonth_nolog.....	57
4.9.22 RCL_product_CWF_onemonth.....	57
4.9.23 RCL_product_CWF_oneyear.....	57
4.9.24 RCL_product_CS_oneday.....	57
4.9.25 RCL_product_CS_onemonth_nolog.....	57
4.9.26 RCL_product_CS_onemonth.....	58
4.9.27 RCL_product_CS_oneyear.....	58
4.9.28 RCL_product_visu_SPL2_oneday.....	58
4.9.29 RCL_product_visu_SPL2_onemonth_nolog.....	58
4.9.30 RCL_product_visu_SPL2_onemonth.....	58
4.9.31 RCL_product_visu_SPL2_daily.....	58
4.9.32 RCL_product_PNG_16m.....	58
4.9.33 RCL_product_PNG_256.....	59
4.9.34 RCL_product_PNG_LowRes.....	59
4.10. Production de masse des visus d'orbite.....	59
4.10.1 RCL_product_visu_orbit_oneday.....	59
4.10.2 RCL_product_visu_orbit_onemonth_nolog.....	59
4.10.3 RCL_product_visu_orbit_onemonth.....	59
4.10.4 RCL_product_visu_orbit_oneyear.....	60
4.11. Utilitaires pour le temps.....	60
4.11.1 RCL_current_date.....	60
4.11.2 RCL_nday_of_month.....	60
4.11.3 RCL_next_day.....	60
4.11.4 RCL_previous_day.....	60
4.11.5 RCL_decode_datiso.....	60

4.11.6	RCL_decode_datim .....	61
4.11.7	RCL_encode_datiso .....	61
4.11.8	RCL_list_days_of_month .....	61
4.12.	Commandes de Visualisation.....	61
4.12.1	RCL_visu_spectro .....	61
4.12.2	RCL_visu_spectro_3H.....	62
4.12.3	RCL_visu_spectro_4Bz .....	62
4.12.4	RCL_visu_spectro_4Bz_3H .....	62
4.12.5	RCL_visu_ave_spectrum .....	63
4.12.6	RCL_visu_vectime .....	63
4.12.7	RCL_visu_vectime_widget.....	63
4.12.8	RCL_visu_polar.....	63
4.12.9	RCL_visu_CLUPOS.....	63
4.12.10	RCL_visu_CLUGEOM .....	63
4.13.	Utilitaires de conversion des PostScripts.....	64
4.13.1	RCL_ps_to_pdf.....	64
4.13.2	RCL_ps_to_png.....	64
4.13.3	RCL_ps_to_png_256.....	64
4.13.4	RCL_ps_to_png_16m.....	64
4.14.	Gestion des répertoires .....	64
4.14.1	RCL_system_info .....	64
4.14.2	RCL_dir_size.....	65
4.14.3	RCL_dir_size_tree .....	65
4.14.4	RCL_dir_size_pretty_tree.....	66
4.14.5	RCL_dir_properties .....	66
4.14.6	RCL_dir_properties_tree .....	67
4.14.7	RCL_dir_properties_pretty_tree .....	68
4.14.8	RCL_check_dirname_tree .....	69
4.14.9	RCL_search_duplicates .....	69
4.15.	Liste Alphabétique de toutes les commandes .....	70
5.	LES SCRIPTS RCL.....	71
5.1.	Principe .....	71
5.1.1	Exemple .....	71
5.1.2	Un peu de shell .....	72
5.1.3	Explications .....	73
5.2.	Calcul et plot d'un spectrogramme .....	73
5.2.1	Spectrogramme à partir des VTL2 .....	73
5.2.2	Spectrogramme à partir des VTL1 .....	74
5.3.	Calcul et plot d'un spectre moyen.....	76
5.4.	Plot d'une forme d'onde brute ou calibrée .....	76
5.4.1	La commande RCL_visu_vectime .....	76
5.4.2	Application batch à un fichier VTL1 de STAFF.....	76
5.4.3	Application batch à un fichier VTL2 de STAFF en ISR2 .....	77
5.4.4	Application batch à un fichier VTL2 de STAFF en GSE.....	79
5.4.5	Remarques sur les formes d'onde obtenues .....	79
	• Les données brutes VTL1 .....	79
	• Les données VTL2 calibrées dans le repère ISR2 .....	79
	• Les données VTL2 calibrées dans le repère GSE .....	79
5.5.	Calcul et plot des paramètres de polarisation.....	80
5.5.1	Historique .....	80
5.5.2	Création du fichier SPL2.....	80
5.5.3	Calcul et visualisation des paramètres de polarisation .....	80
	• Calcul pour un cas.....	80
	• Script d'automatisation à partir des VTL1 .....	81
	• Traitement en série de plusieurs cas .....	82
	• Script d'automatisation à partir des VTL2 .....	84



5.6. Spectres et spectrogrammes sur les données FGM.....	86
5.6.1 Principe.....	86
5.6.2 Exemple de résultat.....	86
<b>6. INSTALLATION DU PACKAGE RCL .....</b>	<b>89</b>
6.1. Installation du pack binaire.....	89
6.1.1 Copie du pack sur la machine cible.....	89
6.1.2 Réglage des variables d'environnement .....	89
6.2. Installation du pack source .....	91
6.2.1 Copie du pack sur la machine cible.....	91
6.2.2 Mise a jour du Makefile.....	91
• Choix du compilateur .....	91
• Option debug ou optimized.....	91
• Options de compilation.....	92
• Options du make clean .....	92
• Options du make mrproper .....	92
6.2.3 Régénération des .sav d'IDL .....	93
6.2.4 Création d'un pack binaire.....	93
6.3. Procédures de test.....	94
6.3.1 test_commands .....	94
6.3.2 test_get_data .....	94
6.3.3 test_production_oneday .....	94
6.3.4 test_production_onemonth.....	95
6.3.5 test_production_oneyear .....	95
<b>7. A L'INTENTION DES DEVELOPPEURS.....</b>	<b>97</b>
7.1. Organisation du projet .....	97
7.1.1 Fichier README.txt .....	98
7.1.2 Fichier menu.txt.....	98
7.1.3 Fichier RCL_config.bash.....	98
7.1.4 Fichier RCL_config.csh.....	98
7.1.5 Fichier Makefile.....	98
7.1.6 Fichier Make_IDL_sav.bash.....	98
7.1.7 Fichier Make_bin_pack.bash.....	98
7.1.8 Répertoire bash .....	98
7.1.9 Répertoire bin .....	98
7.1.10 Répertoire doc.....	98
7.1.11 Répertoire gainantset .....	98
7.1.12 Répertoire mod .....	98
7.1.13 Répertoire obj .....	98
7.1.14 Répertoire pro .....	99
7.1.15 Répertoire script.....	99
7.1.16 Répertoire src.....	99
7.1.17 Répertoire test.....	99
7.2. Commandes RCL utilisant des executables F90 .....	100
7.3. Commandes RCL utilisant des SAV IDL .....	101
7.4. Commandes RCL utilisant d'autres commandes RCL .....	101
7.5. Description des programmes F90.....	104
7.5.1 Opération sur les fichiers RFF .....	104
• check_rff.f90.....	104
• clean_rff.f90 .....	104
• diff_rff.f90.....	104

7.5.2	Traitements génériques .....	104
	• waveform_to_vectime.f90 .....	104
	• vectime_to_spectro.f90 .....	104
	• reduce_time_rff.f90 .....	105
	• spectro_to_polar.f90 .....	105
	• spectro_xyz_to_lrz.f90 .....	105
7.5.3	Traitements STAFF .....	105
	• vectime_calibration_CLUSTA.f90 .....	105
	• vectime_L1_to_spectro_L2.f90 .....	105
	• vectime_to_mfa.f90 .....	105
	• search_strong_dc.f90 .....	106
7.5.4	Traitements FGM .....	106
	• fgm_cef_gse_to_sr2.f90 .....	106
7.5.5	Accès aux données d'orbites .....	106
	• get_data_CLUPOS.f90 .....	106
	• get_data_CLUGEOM.f90 .....	106
7.5.6	Conversion RFF vers CEF .....	106
	• vectime_L1_to_cef.f90 .....	106
	• vectime_L2_to_cef.f90 .....	107
	• spectro_L2_to_cef.f90 .....	107
	• w_cal_caveat_header.f90 .....	107
	• w_cwf_cef_header.f90 .....	107
	• w_vectime_cef_header.f90 .....	107
7.5.7	Conversions CEF vers RFF .....	107
	• DWF_cef_to_rff.f90 .....	107
	• CWF_cef_to_rff.f90 .....	107
	• fgm_cef_to_rff.f90 .....	107
7.5.8	Divers .....	108
	• dir_properties_pretty_tree.f90 .....	108
	• dir_size_pretty_tree.f90 .....	108
	• check_efw_status.f90 .....	108
7.5.9	Bibliothèques nécessaires à chaque programme F90 .....	108
7.6.	Description des bibliothèques F90 .....	110
7.6.1	lib_deconvo.f90 .....	110
7.6.2	lib_gainant.f90 .....	110
7.6.3	lib_rw_rff.f90 .....	111
7.6.4	lib_rw_cef.f90 .....	111
7.6.5	lib_time.f90 .....	112
7.6.6	lib_CLUORB.f90 .....	113
7.6.7	lib_CLU_tools.f90 .....	113
7.6.8	lib_utility.f90 .....	114
7.6.9	Rocotlib_V2p0.f90 .....	114
	• « Compute » modules .....	114
	• « Give » modules .....	114
	• « Print » and « Read » modules .....	115
	• « Transform » modules .....	115
7.7.	Description des procédures IDL .....	116
7.7.1	Procédures de lecture .....	116
	• read_data_CLUPOS.pro .....	116
	• read_FGM_bav.pro .....	116
	• read_fgm_cef.pro .....	116
	• rff_read_SPfile_4.pro .....	116
	• rff_read_SPfile.pro .....	116
	• rff_read_VTfile.pro .....	116
	• get_FGM_freq.pro .....	116
	• r_copolarlib.pro .....	116

7.7.2	Procédures de visualisations .....	117
	• visu_spectro.pro .....	117
	• visu_spectro_4Bz.pro .....	117
	• visu_ave_spectrum.pro .....	117
	• visu_polar.pro .....	117
	• visu_vectime.pro .....	117
	• visu_vectime_widget.pro .....	117
	• visu_CLUGEOM.pro .....	117
	• visu_CLUPOS.pro .....	117
7.7.3	Utilitaires .....	118
	• rpws_bepatient.pro .....	118
	• rpws_center_widget.pro .....	118
	• t3d.pro .....	118
	• zoom_x.pro .....	118
7.7.4	Bibliothèques ou procédures nécessaires à la fabrication des sav d'IDL .....	118
7.8.	Description de LA bibliothèques IDL Roplotlib .....	120
	• Procédures .....	120
	• Fonctions .....	121

## 8. LA CALIBRATION DES SPECTRES ET DES FORMES D'ONDE.....123

8.1.	Introduction .....	123
8.2.	Calibration des spectres .....	123
8.2.1	Théorie.....	123
8.2.2	Les étapes successives de la calibration de la forme d'onde .....	124
	• Étape 1 : conversion en Volt .....	124
	• Étape 2 : despin .....	124
	• Complément à l'étape 2: calcul du champ DC dans le plan de spin.....	125
	• Autre complément à l'étape 2: calcul du misalignement angle .....	126
	• Étape 3 : calibration en repère tournant .....	127
	• Étape 4 : passage en SR2 .....	129
	• Étape 5 : rajout du champ DC sur X et Y .....	130
	• Étape 6: mode de contrôle pour l'étude du DC <sub>⊥</sub> .....	130
	• Étape 7: Calcul en ISR2 .....	130
	• Étape 8: Calcul en GSE .....	130
8.2.3	Calcul du spectre complexe .....	131
8.2.4	Paramètres de la procédure RCL_vectime_L1_to_spectro_L2 .....	132
	• Rappel .....	132
	• Nom des fichiers .....	133
	• Fréquence de « detrend » .....	133
	• Fréquence de coupure .....	134
	• Fréquences min et max .....	134
	• Les étapes de la calibration .....	134
	• La taille du noyau de calibration .....	135
	• La taille du déplacement de la fenêtre .....	135
	• Le type d'apodisation .....	135
	• La forme et la normalisation des fenêtres de pondération .....	136
	• Exemple choisi pour la production de masse .....	137
8.3.	Calibration continue des formes d'onde .....	138
8.3.1	Théorie.....	138
8.3.2	Les étapes successives de la calibration .....	138
8.3.1	Paramètres de la procédure RCL_vectime_calibration_CLUSTA .....	139
	• Rappel .....	139

<b>9. ANNEXES.....</b>	<b>141</b>
9.1. Information sur le statut dans les WFL1 et VTL1.....	143
9.1.1 Information générales .....	143
9.1.2 Signification des caractères du statut .....	143
9.1.3 Quelques mots sur l'indice de compression .....	145
9.2. Calcul du "spin phase" dans les WFL1 et les VTL1 .....	146
9.2.1 Définition .....	146
9.2.2 Méthode de calcul.....	146
9.2.3 Les fichiers SPINPHASE.....	148
9.2.4 Les fichiers des Sun Pulses interpolés :.....	148
9.3. Définition des systèmes de coordonnées utilisés .....	149
9.3.1 The Sensor Coordinate System (SCS).....	149
9.3.2 The Orthogonal Sensor System (OSS).....	149
9.3.3 The Data Sensor System (DSS) .....	150
9.3.4 The Body Build System (BBS) .....	150
9.3.5 The Spin Reference System (SRS).....	150
9.3.6 The spin reference2 system (SR2) .....	151
9.3.7 The Inverse SR2 system (ISR2).....	152
9.3.8 Simplification of the cumulative matrix products .....	152
9.3.9 The Geocentric Equatorial Inertial system (GEI).....	153
9.3.10 The Geocentric Solar Ecliptic system (GSE).....	153
9.3.11 Geocentric Solar Magnetospheric system (GSM) .....	154
9.3.12 Magnetic Field Aligned system (MFA) .....	154
9.4. Estimation du champ continu dans le plan de spin.....	155
9.4.1 Problématique.....	155
9.4.2 Amplitude et phase de $B_{\perp}$ en repère tournant SR.....	155
9.4.3 Tests sur $B_{\perp x}$ et $B_{\perp y}$ .....	156
9.4.4 Amplitude et phase de $B_{\perp}$ en repère fixe SR2.....	156
9.5. Estimation du "misalignment angle".....	157
9.5.1 Définitions .....	157
9.5.2 Signal delivered by a rotating antenna into a DC field.....	158
9.5.3 Estimate of the misalignment angle $\theta_z$ for low value .....	158
9.5.4 Estimate of the misalignment angle $\theta_z$ for high value .....	158
9.6. Calcul de la matrice de correction du dépointage .....	161
9.6.1 Passage de coordonnées non orthogonales à un système orthogonal .....	161
• Position du problème.....	161
• Passage d'un système non orthogonal à un système orthogonal.....	162
• Calcul des composantes cartésiennes XYZ du champ à partir des mesures sur PQR.....	163
• Définition de la matrice de dépointage .....	163
9.6.2 Matrice de correction du dépointage .....	163
• Calcul de la matrice de dépointage inverse .....	164
• Exemple d'application: cas des antennes électriques d'Interball.....	164
• Estimation expérimentale de certains termes .....	165
9.7. Petits rappels sur la transformée de Fourier.....	167
9.7.1 La transformée de Fourier au sens mathématique .....	167
9.7.2 La transformée de Fourier discrète.....	167
9.7.3 La « Fast Fourier Transform » .....	168
• Les fréquences négatives et positives .....	168
• Quelques relations utiles :.....	169
• Test de normalisation d'une FFT .....	169
• Effets de bords.....	170
9.8. Utilité des composantes circulaires en polarisation.....	171
9.8.1 Définition .....	171
9.8.2 Passages en composantes circulaires Gauche et Droite.....	171
• Calcul depuis les formes d'onde .....	171
• Calcul depuis les spectres .....	172
• Conséquences de la rotation sur le spectre $S_{xy}(f)$ .....	172

9.9. Exemples de fichiers RFF.....	173
9.9.1 Exemple de WFL1.rff.....	173
9.9.2 Exemple de VTL1.rff.....	176
9.9.3 Exemple de VTL2.rff.....	179
9.9.4 Exemple de SPL2.rff.....	182
9.10. Exemple de visualisations.....	185
9.10.1 Résultat de RCL_visu_spectro à partir des VTL2.....	185
9.10.2 Résultat de RCL_visu_spectro à partir des VTL1.....	186
9.10.3 Résultat de RCL_visu_spectro_4Bz_3h.....	187
9.10.4 Résultat de RCL_visu_ave_spectrum à partir des VTL1.....	188
9.10.5 Résultat de RCL_visu_vectime sur les VTL1.....	189
9.10.6 Résultat de RCL_visu_vectime sur les VTL2 en ISR2.....	190
9.10.7 Résultat de RCL_visu_vectime sur les VTL2 en GSE.....	191
9.10.8 Résultat de RCL_visu_polar sur des VTL2 en GSE.....	192
9.10.9 Résultat de RCL_visu_CLUPOS / 3 heures.....	197
9.10.10 Résultat de RCL_visu_CLUPOS / orbite complète.....	199
9.10.11 Résultat de RCL_visu_CLUGEOM.....	201
9.11. Résumé des commandes RCL.....	203
9.11.1 Liste fonctionnelle des commandes RCL.....	203
• Software Informations :.....	203
• Tools to handle RFF files (general).....	203
• Generic commands.....	203
• Access to local CLUSTER/STAFF-SC L1 data base.....	203
• Access to local CLUSTER/Orbit data base.....	203
• Access to local CLUSTER/FGM L2 data base.....	203
• Access to CSA/FGM L2 data base.....	204
• CLUSTER/STAFF-SC processing.....	204
• Mass production for STAFF-SC :.....	204
• Mass production for orbit visualization :.....	205
• Time Utilities :.....	205
• Visualization tools :.....	205
• Graphics Utilities :.....	205
• System & Directories properties :.....	205
9.11.2 Liste alphabétique des commandes avec arguments.....	206
9.12. Quelques mots sur les ROPROC.....	209
9.12.1 Historique des Roproc et lien avec les RCL.....	209
9.12.2 Introduction aux Roproc.....	210
9.12.3 Liste thématique des Roproc.....	211
9.12.4 Exemples of arguments of Roproc commands.....	217
 10. LISTE DES ACRONYMES.....	 221
 11. LISTE DES TABLES ET FIGURES.....	 223
11.1. Liste des figures.....	223
11.2. Liste des tables.....	224
 12. BIBLIOGRAPHIE.....	 227



## 1. INTRODUCTION

L'expérience STAFF embarquée sur les 4 satellites CLUSTER est une expérience «onde», dont la partie STAFF-SC délivre des séries temporelles dans deux modes de fonctionnement exclusifs : le mode NBR (Normal Bit Rate) avec une fréquence d'échantillonnage de 25 Hz, et le mode HBR (High Bit Rate) à 450 Hz [voir 6, Cornilleau, 1997]. Deux progiciels, ou ensemble de commandes permettant toutes sortes d'opérations, ont été développés pour traiter ces données : un pack à usage scientifique, les « Roproc », et un pack dédié aux traitements de masse nécessaires à l'alimentation des bases de données, locale au laboratoire les « RCL ». Ce dernier pack sert surtout pour le CAA (CLUSTER Active Archive) [voir 2, Perry, 2005], devenu maintenant le CSA (CLUSTER Science Archive) et permet de faire tourner sur plus de dix ans les chaînes de traitement pour délivrer les produits requis (voir § 2.3).

### 1.1. RAPPEL SUR LES PROCÉDURES SCIENTIFIQUES ROPROC

Les commandes Roproc (Robert's Procedures) sont un ensemble de procédures, ou de commandes, permettant le traitement et la visualisation de données délivrées sous forme de séries temporelles. Ces procédures sont avant tout à usage scientifique, mais certaines concernent l'acquisition et la calibration des données. Une partie des commandes Roproc a donc été utilisée pour former le pack RCL qui fait l'objet de ce document.

Les procédures Roproc ont été initialement développées avant les années 2000 pour les données CLUSTER de STAFF-SC et FGM, où l'auteur est CoI (Co-Investigator) et les données de trajectoire. Mais elles ont aussi été utilisées pour d'autres missions, telles que Double-Star / STAFF, les données des fusées CUSP et ICI-3, et pourraient être utilisées pour d'autres missions passées (THEMIS/SCM et FGM, GEOS, etc.) ou futur (MMS) fournissant des séries temporelles.

Les Roproc traitent deux catégories de données :

- Les données de forme d'onde, où la fréquence d'échantillonnage est constante. Des opérations du type calcul de spectre, changement de repère, calcul de polarisation, etc. sont disponibles sur ce type de données, comme CLUSTER/STAFF.
- Les données du style suite de vecteurs datés temporellement, où le taux d'échantillonnage peut être variable ou non, comme CLUSTER/FGM ou les données de position.

Des procédures de conversion d'un type à l'autre sont également fournies.

Il existe aussi un ensemble de commandes travaillant sur les données de positions, disponibles uniquement pour la mission CLUSTER, et un ensemble d'outils pour le calcul du champ magnétique d'après un modèle (dipôle, IGRF, Tsyganenko 1987 à 2004) incluant des calculs de lignes de force, de points conjugués, etc.

Pour la mission CLUSTER, le calcul de quantités incluant la dimension spatiale fournie par les 4 satellites comme la matrice des gradients, le rotationnel et la divergence du champ magnétique, ainsi que son rayon de courbure est également disponible. Pour plus d'informations on pourra se

reporter à [1, Robert, 2003] et [5, Robert, 2011]. Les commandes Roproc de la version 4.3 sont rappelées au chapitre 9.12.3.

## 1.2. LES COMMANDES RCL EN GÉNÉRAL

Le pack de commandes RCL (Roproc Command Language) avait pour but initial de reprendre l'ensemble des commandes Roproc dans un format homogène, avec des fichiers d'échanges (en entrée comme en sortie) basés sur un format unique, le format RFF (Roproc File Format), voir [4, Robert, 2004]. Les commandes peuvent être utilisées sous forme de script, un peu comme un script shell ou bash, et constituent un langage de traitement de données à part entière. Le chapitre 5 est consacré en détail à ce sujet, mais on donne ici une petite introduction à cet aspect important de ce produit.

Par exemple, le script suivant (voir Table 1) copie le fichier VTL1 de la base locale des données non calibrées de niveau 1 de STAFF-SC en ajoutant, via le `get_data`, le dernier bloc du jour d'avant et le premier bloc du jour d'après et produit ainsi un fichier journalier complet de nom `CLU3_STASC_VTL1_NBR_20010923_full.rff`. Ensuite, il réduit l'intervalle de temps de la journée aux 30 minutes auxquelles on s'intéresse (par le `reduce_time`), puis calibre ces données selon les paramètres désirés (au moyen du vectime `calibration`). Enfin il calcule le spectrogramme correspondant (par le vectime `to_spectro`), et le visualise sous la forme d'un fichier PS et PDF (via le `visu_spectro`), ainsi que le spectre moyen associé (via le `visu_ave_spectrum`), et ceci dans un autre intervalle de temps plus réduit.

```
#!/bin/bash
RCL_get_data_CLUSTA_VTL1_forVTL2 4 2001 09 23 NBR
RCL_reduce_time_rff                CLU4_STASC_VTL1_NBR_20010923_full.rff \
                                   CLU4_STASC_VTL1_NBR_20010923.rff \
                                   2001-09-23T09:15:00.000Z \
                                   2001-09-23T10:15:00.000Z
RCL_vectime_calibration_CLUSTA CLU4_STASC_VTL1_NBR_20010923.rff \
                                CLU4_STASC_VTL2_NBR_20010923.rff \
                                0. 0.1 0.5 0. 4 1024 2 g
RCL_vectime_to_spectro          CLU4_STASC_VTL2_NBR_20010923.rff \
                                CLU4_STASC_SPL2_NBR_20010923.rff \
                                512 512 t
RCL_visu_spectro                CLU4_STASC_SPL2_NBR_20010923.rff \
                                2001-09-23T09:20:00.000Z \
                                2001-09-23T10:10:00.000Z 0. 10. 0. 0. 0.2 10.
RCL_visu_ave_spectrum           CLU4_STASC_SPL2_NBR_20010923.rff \
                                2001-09-23T09:20:00.000Z \
                                2001-09-23T10:10:00.000Z 0. 10. 0. 0. XY y n
```

**Table 1 :** Exemple de script RCL pour calculer et visualiser un spectrogramme de données calibrées

Bien sûr tous les arguments peuvent être paramétrés suivant le shell utilisé (sh ou bash) pour automatiser ce type de calcul. On verra au chapitre 5 que ce script peut être ramené à une seule commande.



Néanmoins, par manque de temps, le projet RCL dans sa totalité a été abandonné. Il a été réduit uniquement à la production des bases de données locales de CLUSTER-STAFF-SC et à la production des fichiers requis par le CSA. Cette fonction aurait pu être ajoutée aux Roproc, mais ces dernières sont issues de codes d'origines variées, alors que la programmation des RCL a été faite avec plus de sécurité, plus de performance par l'utilisation du Fortran90 natif et de ses fonctions optimisées, une plus complète gestion des codes d'erreur, et la production de fichiers de log pour les commandes de production. Les procédures de commandes sont en bash, le shell par défaut sous les machines Linux Ubuntu (sh pour les Roproc). Ces précautions étaient nécessaires pour la production de masse sur plusieurs années.

L'introduction du format RFF a permis également de faciliter les échanges et transformations de fichiers d'une procédure à l'autre. Il faut noter que dans son état actuel, les RCL peuvent traiter facilement les données de toute autre expérience de type « onde », pourvu que les données d'entrée soient mises au format RFF. Pour la procédure de calibration **RCL\_vectime\_calibration\_CLUSTER**, il suffit de « cloner » cette commande et le programme correspondant et d'introduire dans ce dernier la nouvelle fonction de transfert pour obtenir des formes d'onde calibrées en continue pour n'importe quelle expérience passée ou future.

### 1.3. LES COMMANDES RCL POUR LE TRAITEMENT DE STAFF-SC

Les chaînes de production des données de CLUSTER/STAFF-SC ont deux objectifs:

- d'une part, traiter les données de niveau N0, disponibles sous la forme d'images de CD-Rom, et en faire des produits de niveau N1 et N2, qui sont archivés dans une base locale au laboratoire, sur laquelle s'appuient des logiciels de traitements scientifiques [voir 1, Robert, 2003],
- d'autre part produire des fichiers de niveaux N1 et N2 livrés au CLUSTER Active Archive (CAA) de l'ESA [voir 2, Perry, 2005] sous un format dédié, le format CEF [3, Allen, 2009]. Ces données sont ensuite disponibles pour l'ensemble de la communauté scientifique [7, Cornilleau, 2011]. Notons que depuis la fin de 2014 le CAA a été abandonné pour le CSA (Cluster Science Archive).

Le passage des Raw data de niveau N0 vers les données décommutées et datées de niveau N1 est assuré par des programmes écrits en langage C (sous la responsabilité de V. Bouzid). Le pack RCL travaille uniquement à partir des données N1. La partie calcul et traitement est écrite en F90, la partie visualisation est écrite en IDL. Toutes les commandes sont des scripts bash, qui appellent des exécutables FTN ou des .sav d'IDL.

Ces chaînes de traitement ont été développées d'abord au CETP, puis au LPP, qui en assure à présent la maintenance et l'exploitation pour toute la durée de vie de CLUSTER.

### 1.4. QUELQUES MOTS SUR LE FORMAT RFF

Le format RFF a été conçu initialement pour traiter des données de type vectoriel (scalaire, vecteur, matrice, tenseur), pouvant varier en fonction d'un paramètre lui-même également de type vectoriel. Une application simple et souvent rencontrée est un vecteur ou une matrice, dépendant d'un scalaire qui est le temps, comme une forme d'onde ou un spectrogramme. Mais ce format peut également décrire un ensemble de lignes de force, ou un champ vectoriel dans un volume, dont l'index est la position x,y,z et la donnée le vecteur en ce point  $V_x, V_y, V_z$ . Il peut également décrire des images, où l'index peut être le temps, et la donnée une matrice correspondant aux valeurs de l'image (i.e. la couleur de chaque pixel).

Ainsi les données de niveau 1 en sortie du programme de décommutation «**decommute\_SC**» (voir § 2.5.1) sont des blocs de vecteurs codés en entiers, correspondant aux valeurs de télémétrie (25 ou 45 selon le mode NBR ou HBR), chacun datés en temps. Au sens RFF, ces blocs sont donc

des matrices dépendant du temps, dont la classe est appelé **MatTime**, mais plus communément appelées **WaveForm**, dans le sens ou la plupart des séries temporelles issues des magnétomètres sont à l'origine sous ce format. Les fichiers correspondants sont donc de type **WFL1** (*Wave Forms Level 1*).

L'application **RCL\_waveform\_to\_vectime** effectue l'interpolation en temps de ces blocs de données, et produit à présent une suite de vecteurs tous datés en temps, dont la classe est appelé **VecTime** (voir Table 2). Quand les données correspondantes sont de niveau 1 (Level 1), les fichiers sont de classe **VTL1.rff**. Quand ce sont des données calibrées, ce sont des **VTL2.rff**.

Un spectrogramme est, quand à lui, constitué d'une suite de valeurs dépendant à la fois du temps et de la fréquence. Si le spectrogramme est réalisé sur une grille de temps régulière, il peut être considéré, au sens RFF, comme une image constituée de pixels réguliers, donc une unique matrice pour chaque composante. C'est donc un fichier de classe **MatSca** dont l'index scalaire se rapporte à une des composantes Bx, By et Bz. Le temps de départ est une constante de ce fichier.

Néanmoins, il se peut qu'après un trou de données, le temps ne reparte pas à un multiple entier de la largeur d'un pixel temporel. Les largeurs en temps sont toujours de largeur fixe, mais l'espacement, ou la datation de chaque spectre, n'est plus proportionnel à la durée d'un pixel. Il faut alors considérer le spectrogramme comme une suite de spectres datés, donc de vecteurs datés (dont les composantes sont fréquentielles). Le format RFF y reconnaît donc une classe **VecTime**. Mais si on veut faire figurer 3 composantes, on a alors affaire à une classe **MatTime** (une dimension pour les fréquences, une dimension pour les composantes). C'est le cas pour les fichiers appelés SPL2 (spectrogrammes de niveau 2, donc calibrés) de CLUSTER/STAFF-SC, qui contiennent donc une succession de spectres complexes datés à 6 composantes codés comme des matrices réelles de dimension (6,Nf) ou Nf est le nombre de pas en fréquence, chaque spectre en fréquence étant représenté par un vecteur (BxR, BxI, ByR, ByI, BzR, BzI). Par commodité de syntaxe, cette classe particulière de fichier **MatTime** est appelé « **Spectrogram** ».

Ainsi on peut voir que les différentes classes de fichiers RFF décrivent comment les données sont organisées. Pour plus de détails, on pourra se référer à [4, Robert, 2004].

START INDEXED DATA			
2001-09-23T00:00:00.904328Z	000000000001	17.35	2001-09-23T00:00:00.904328Z, 00000000000110, 17.35, 34047, 33190, 32795
34047 33190 32795 0			2001-09-23T00:00:00.944327Z, 00000000000100, 20.93, 34078, 33107, 32796
34078 33107 32796 0			2001-09-23T00:00:00.984326Z, 00000000000100, 24.51, 34098, 33028, 32796
34098 33028 32796 0			2001-09-23T00:00:01.024325Z, 00000000000100, 28.09, 34107, 32941, 32801
34107 32941 32801 0			2001-09-23T00:00:01.064324Z, 00000000000100, 31.67, 34110, 32857, 32795
34110 32857 32795 0			2001-09-23T00:00:01.104323Z, 00000000000100, 35.25, 34109, 32776, 32799
34109 32776 32799 0			2001-09-23T00:00:01.144322Z, 00000000000100, 38.83, 34105, 32693, 32797
34105 32693 32797 0			2001-09-23T00:00:01.184321Z, 00000000000100, 42.40, 34101, 32611, 32799
34101 32611 32799 0			2001-09-23T00:00:01.224320Z, 00000000000100, 45.98, 34090, 32528, 32796
34090 32528 32796 0			2001-09-23T00:00:01.264319Z, 00000000000100, 49.56, 34064, 32445, 32795
34064 32445 32795 0			2001-09-23T00:00:01.304318Z, 00000000000100, 53.14, 34041, 32365, 32794
34041 32365 32794 0			2001-09-23T00:00:01.344317Z, 00000000000100, 56.72, 34009, 32290, 32790
34009 32290 32790 0			2001-09-23T00:00:01.384316Z, 00000000000100, 60.30, 33979, 32217, 32792
33979 32217 32792 0			2001-09-23T00:00:01.424315Z, 00000000000100, 63.88, 33937, 32139, 32796
33937 32139 32796 0			2001-09-23T00:00:01.464314Z, 00000000000100, 67.46, 33898, 32071, 32794
33898 32071 32794 0			2001-09-23T00:00:01.504314Z, 00000000000100, 71.04, 33848, 32000, 32792
33848 32000 32792 0			2001-09-23T00:00:01.544313Z, 00000000000100, 74.62, 33797, 31932, 32790
33797 31932 32790 0			2001-09-23T00:00:01.584312Z, 00000000000100, 78.20, 33746, 31873, 32793
33746 31873 32793 0			2001-09-23T00:00:01.624311Z, 00000000000100, 81.78, 33681, 31812, 32791
33681 31812 32791 0			2001-09-23T00:00:01.664310Z, 00000000000100, 85.36, 33614, 31757, 32790
33614 31757 32790 0			2001-09-23T00:00:01.704309Z, 00000000000100, 88.93, 33552, 31704, 32790
33552 31704 32790 0			2001-09-23T00:00:01.744308Z, 00000000000100, 92.51, 33478, 31663, 32791
33478 31663 32791 0			2001-09-23T00:00:01.784307Z, 00000000000100, 96.09, 33406, 31617, 32787
33406 31617 32787 0			2001-09-23T00:00:01.824306Z, 00000000000100, 99.67, 33330, 31584, 32789
33330 31584 32789 0			2001-09-23T00:00:01.864305Z, 00000000000100, 103.25, 33255, 31551, 32786
33255 31551 32786 0			2001-09-23T00:00:01.904304Z, 00000000000110, 106.83, 33175, 31522, 32787
2001-09-23T00:00:01.904304Z	000000000001	106.83	2001-09-23T00:00:01.944303Z, 00000000000100, 110.41, 33092, 31498, 32783
33175 31522 32787 0			2001-09-23T00:00:01.984302Z, 00000000000100, 113.99, 33012, 31477, 32786
33092 31498 32783 0			2001-09-23T00:00:02.024301Z, 00000000000100, 117.57, 32929, 31466, 32785
33012 31477 32786 0			2001-09-23T00:00:02.064300Z, 00000000000100, 121.15, 32852, 31456, 32781
32929 31466 32785 0			2001-09-23T00:00:02.104299Z, 00000000000100, 124.73, 32770, 31455, 32780
32852 31456 32781 0			2001-09-23T00:00:02.144298Z, 00000000000100, 128.31, 32687, 31455, 32778

Table 2: Exemple de passage d'un format WF (WaveForm) à un format VT (VecTime) par l'application RCL\_waveform\_to\_vectime

## 2. LES CHAINES DE PRODUCTION STAFF-SC

### 2.1. LES DEUX TYPES DE CHÂÎNES POUR LES RFF ET CEF

On peut distinguer deux catégories dans les chaînes de production:

- les chaînes destinées à l'alimentation des bases de données locales,
- les chaînes destinées à la livraison des produits CSA, qui s'appuient sur les bases de données locales au LPP, et qui délivrent donc des produits archivés au CSA.

Les chaînes de traitement et de production au delà du niveau 1 sont toutes assurées par des commandes RCL.

Les bases locales sont au format RFF [voir 4, Robert, 2004], format maintenant utilisées par les Roproc [1,5, Robert, 2003, 2011] dans la dernière version 4.5. Les bases du CSA sont au format CEF (Cluster Exchange Format) [3, Allen, 2009], qui comporte des métas-data plus volumineuses. Néanmoins on s'est assuré que toutes les informations importantes contenues dans les méta-data des fichiers RFF (les « Mandatory Parameters ») se retrouvent dans celles des CEF. Ainsi, on peut disposer d'applications qui convertissent les CEF en RFF, sans perte d'information, et donc retrouver les fichiers RFF initiaux qui ont servi à la fabrication des CEF. Cette possibilité est importante pour les utilisateurs des Roproc en dehors du laboratoire car ils peuvent télécharger les données du CSA, pour ensuite utiliser les Roproc dessus.

### 2.2. LES PRODUITS RFF DANS LES BASES LOCALES LPP

#### 2.2.1 *Les WFL1.rff*

Ce sont le résultat de la première étape de production, à partir des CD-ROM, des Tcor files et du logiciel Ted. Ces fichiers contiennent les données de télémessure, sous forme de nombre entiers dans la gamme 0-65535. Ces données sont groupées par blocs de 25 (en NBR) ou de 125 (en HBR) vecteurs à 3 composantes, à laquelle est rajoutée une valeur entière définissant le statut de la compression. Chaque bloc est daté et comprend un statut ainsi que la valeur du «spin phase angle» indispensable pour pouvoir effectuer le passage du repère tournant des antennes à un repère fixe. La signification du statut et du caractère de compression peut être trouvée dans [7, Cornilleau, 2011] et est rappelée au chapitre 9.1.2.

Une entête, formant le méta-data, contient bon nombre d'informations indispensables, comme la fréquence d'échantillonnage exacte (propre à chaque satellite), la fréquence de spin, la direction de l'axe de spin dans le repère inertiel GEI, etc.

Ces fichiers contiennent toute l'information disponible pour les traitements scientifiques, ainsi que pour la production des DWF livrés au CSA (voir chapitre 2.3). L'information contenue dans les méta-data est dupliquée dans l'étape suivante que sont les fichiers VTL1.rff. Un exemple de fichier WFL1.rff est donné au chapitre 9.9.1.

### 2.2.2 Les VTL1.rff

Ces fichiers sont produits à partir des WFL1.rff et la chaîne de traitement correspondante utilise la commande **RCL\_waveform\_to\_vectime** (voir chapitre 2.2.2 et la Table 1).

Les fichiers VTL1 contiennent des données de niveau N1 (Level 1 en anglais), non calibrées, dont tous les points sont datés avec le maximum de précision. C'est la base de référence, car ces fichiers contiennent toute l'information disponible sous une forme claire, et chaque vecteur est daté avec le maximum de précision. Mais les données n'étant pas calibrées, elles ne peuvent être utilisables directement pour des applications scientifiques, d'où la nécessité de livrer des données de forme d'onde calibrées de type VTL2. Un exemple de VTL1 est donné au chapitre 9.9.2.

Ce sont ces VTL1.rff qui seront transformés en DWF.cef (Decommuted WaveForm) et livrés au CSA. Cette transformation se fait par un filtre très rapide en temps CPU (commande **RCL\_vectime\_L1\_to\_cef**), le champ data restant le même entre les deux formats, seul le contenu des métas-data étant différent. Dans la mesure où ils sont reproductibles facilement à partir de la base des VTL1, si le besoin s'en faisait sentir, ces fichiers CEF pourraient ne pas être sauvegardés localement durablement après livraison.

### 2.2.3 Les VTL2.rff

Ces fichiers sont produits à partir des VTL1.rff et la chaîne de traitement correspondante utilise la commande **RCL\_vectime\_calibration\_CLUSTA**. Les fichiers VTL2 sont issus des VTL1 suite à un traitement de calibration continue assez complexe et coûteux en temps CPU. C'est pourquoi ces fichiers sont également archivés localement. Un exemple de VTL2 est donné au chapitre 9.9.3.

Ce sont ces fichiers qui seront transformés en CWF.cef (Calibrated WaveForm) et livrés au CSA. Comme pour les DWF, cette transformation se fait très rapidement (par la commande **RCL\_vectime\_L2\_to\_cef**) et donc ces fichiers pourraient ne pas être sauvegardés localement durablement.

Il faut noter que la production des VTL2 est faite avec un jeu de paramètres de calibration standard, établi pour calibrer correctement la majorité des événements rencontrés. Néanmoins, dans certains cas particuliers (traversée de magnétopause, fort champ DC perpendiculaire, zones en fréquence à filtrer, etc.), il peut être bon d'ajuster au mieux l'ensemble de ces paramètres. La conservation et l'archivage des VTL1 est donc absolument nécessaire.

### 2.2.4 Les SPL2.rff

Ces fichiers sont des spectrogrammes au sens RFF (voir 2.1.1), dont chaque colonne est constituée par un spectre complexe calibré. Il y a deux manières de calculer des spectrogrammes calibrés :

- L'application **RCL\_vectime\_to\_spectro** produit les SPL2.rff à partir des formes d'onde calibrées VTL2.rff. Cette commande générique prends un fichier d'origine quelconque mais de la classe VecTime et en calcule le spectrogramme. Elle n'effectue donc aucune calibration supplémentaire, et peut être utilisée pour n'importe quelle expérience de type « onde ». Dans la mesure où cette commande est appliquée sur un fichier VTL2 calibré, le résultat est un spectrogramme calibré SPL2 de classe « Spectrogram ».
- L'application **RCL\_vectime\_L1\_to\_spectro\_L2** produit les SPL2.rff à partir des formes d'onde brutes VTL1.rff en appliquant une calibration classique sur chaque spectre, et en effectuant le changement de repère SR (spin référence) vers SR2 ou ISR2, ce repère fixe étant proche du GSE, voir [8, Robert, 2003].

La deuxième méthode présente un léger avantage sur les trous de télémétrie : chaque fenêtre de donnée L1 est donc calibrée, et on passe directement à la suivante. En cas de trou de télémétrie, si la fenêtre est incomplète, le spectre est bien sûr abandonné, et le spectre suivant démarre à l'instant même ou la télémétrie reprend. En cas de télémétrie hachurée, cette méthode permet aussi de produire des spectrogramme, hachurés aussi, certes, mais présentant des données que la première méthode ne saurait construire.

La première méthode travaille de la même manière, cependant, pour produire les VTL2, il a fallu utiliser une fenêtre « élargie » pour effectuer la calibration. Les trous de télémétrie sont donc

légèrement plus grands dans les fichiers VTL2 que dans les fichiers VTL1, d'où la remarque ci-dessus pour les spectres hachurés..

C'est pourquoi c'est la deuxième méthode qui a été retenue pour la fabrication des **CS.cef**.

De plus on peut, de cette manière, produire rapidement les **SPL2**, et donc les **CS**, avant les **CWF** qui demandent plus de ressources CPU. Un exemple de fichier **SPL2.rff** est donné chapitre 9.9.4.

## 2.3. LES PRODUITS STAFF-SC AU CSA

Comme on l'a vu lors de la définition de la nomenclature des fichiers, le LPP livre au CSA les produits ci-dessous, qui sont donc disponibles pour l'ensemble de la communauté scientifique.

### 2.3.1 Les DWF

Les fichiers **DWF.cef** (Decommuted WaveForms) sont le résultat de la conversion de format des fichiers **VTL1.rff**, faite par l'application **RCL\_vectime\_L1\_to\_cef**. Comme on l'a vu au chapitre 2.1 toutes les informations contenues dans le méta-data des fichiers VTL1 se retrouvent dans celles des DWF, lesquelles sont complétées par d'autres informations propres au fichier CEF.

### 2.3.2 Les CWF

Les **CWF.cef** (Calibrated WaveForms) sont le résultat de la conversion de format des fichiers **VTL2.rff**, faite par l'application **RCL\_vectime\_L2\_to\_cef**. Comme pour les DWF, toutes les informations contenues dans les méta-data des fichiers VTL2 se retrouvent dans celles des CWF, lesquelles sont complétées par d'autres informations propres au fichier CEF.

### 2.3.3 Les CS

Les **CS.cef** sont le résultat de la conversion de format des fichiers **SPL2.rff**, faite par l'application **RCL\_spectro\_L2\_to\_cef**. Les méta-data obéissent aux mêmes règles que celles des DWF et CWF.

### 2.3.4 Les graphiques « quick looks »

Ce sont des images au format PNG représentant les spectrogrammes de la composante Bz pour les 4 satellites. Leur production est décrite au chapitre 2.5.9. Un exemple est donné au chapitre 9.10.3.

## 2.4. ORGANISATION DES BASES DE DONNÉES LOCALES

### 2.4.1 Stratégie de conservation

Les **WFL1.rff** doivent être conservés le temps de la validation des produits VTL1. Dans la pratique, ils sont zippés et archivés. Ils sont les points d'entrée pour les Roproc dans la version antérieure à V4.5, notamment la V4.4d, toujours en cours d'utilisation. Les fichiers **VTL1.rff** et **VTL2.rff** sont gardés dans l'archive locale, pour les traitements scientifique d'une part (voir [5, Robert, 2011]) et d'autre part pour produire les **DWF.cef** et **CWF.cef**.

Ils servent aussi à produire les spectrogrammes calibrés **SPL2.rff** qui produiront eux-mêmes les **CS.cef**. Contrairement aux VTL1 et VTL2, les **SPL2** ne sont pas conservés durablement mais les **SPL2** en GSE servent à produire les **CS**, tandis que les **SPL2** en ISR2 servent à produire les plots à 4 Bz, dont un exemple est donné chapitre 9.10.3, et qui sont livrés au CSA.

D'une manière générale, les produit CEF (DWF, CWF et CS) sont livrés au CSA, mais pourraient ne pas être sauvegardés localement durablement.

### 2.4.2 Nomenclature des noms de fichiers

Suite à une exigence tardive du CAA (avant le passage au CSA) de concaténer en un seul fichier les modes NBR et HBR pour les CWF, la nomenclature des fichiers a été quelque peu bousculée, ce qui fait que la nomenclature n'est plus homogène, contrairement à celle qui avait été définie au début du projet. En effet, il a été choisi de conserver les noms initiaux des fichiers plutôt que d'avoir à tous les renommer et d'avoir à changer les shells de production.

Chaque jour débute avec le premier enregistrement du jour et se termine avec le dernier enregistrement avant le jour suivant. Dans le cas des WFL1, il faut avoir recours au dernier bloc du jour précédent si on veut fabriquer des VTL1 complet sur la journée. Pour les VTL2, il faut avoir recours à une portion de données de VTL1 de la fin du jour d'avant ET du début du jour d'après.

La nomenclature des noms de fichier à ce jour est celle indiquée dans la Table 3 et la table Table 4 ci-dessous. L'exemple pris est le 21 mars 2002, pour le satellite 1. Il y a un fichier par jour.

Classe RFF	Exemple de nom de fichier
WFL1	CLU1_STASC_NBR_WFL1_20020321.rff CLU1_STASC_HBR_WFL1_20020321.rff
VTL1	CLU1_STASC_VTL1_NBR_20020321.rff CLU1_STASC_VTL1_HBR_20020321.rff
VTL2	CLU1_STASC_VTL2_GSE_NBR_20020321.rff CLU1_STASC_VTL2_GSE_HBR_20020321.rff CLU1_STASC_VTL2_ISR2_NBR_20020321.rff CLU1_STASC_VTL2_ISR2_HBR_20020321.rff
SPL2	CLU1_STASC_SPL2_GSE_NBR_20020321.rff CLU1_STASC_SPL2_GSE_HBR_20020321.rff CLU1_STASC_SPL2_ISR2_NBR_20020321.rff CLU1_STASC_SPL2_ISR2_HBR_20020321.rff

**Table 3 : Nomenclature des noms de fichiers RFF**

Type de CEF	Exemple de nom de fichier
DWF	C1_CP_STA_DWF_NBR_20020321_V05.cef C1_CP_STA_DWF_HBR_20020321_V05.cef
CS	C1_CP_STA_CS_NBR_20020321_V05.cef C1_CP_STA_CS_HBR_20020321_V05.cef
CWF	C1_CP_STA_CWF_GSE_20020321_V01.cef C1_CP_STA_CWF_ISR2_20020321_V01.cef

**Table 4: Nomenclature des noms de fichiers RFF**

### 2.4.3 Arborescence des fichiers sur les bases locales

Leur arborescence sur la base locale est décrite ci-dessous pour le mode NBR. Elle est la même pour le mode HBR.

Il faut noter que pour les CWF, sur la demande du CAA, les modes NBR et HBR sont fusionnés au sein d'un même fichier. Cette requête tardive a nécessité un changement de l'arborescence et a donc rendu la nomenclature des noms de fichiers plus très homogène. En effet, initialement, on avait choisi le mode au même niveau que STAFF-SC, au dessus de L1, comme une expérience indépendante, d'où l'ordre NBR/WFL1 qui se traduisait par NBR\_WFL1 dans les noms de fichier.

• **Arborescence des fichiers RFF et PNG**

À ce jour le point d'entrée est: /NFS/tamaya/data2/staff-op/STAFF-SC à l'exception des WFL1 qui sont sur /NFS/tamaya/CLUSTERII/STAFF-SC/L1/RFF/WF. L'arborescence est la suivante:



Table 5 : Arborescence des fichiers RFF et PNG

• **Arborescence des fichiers CEF**

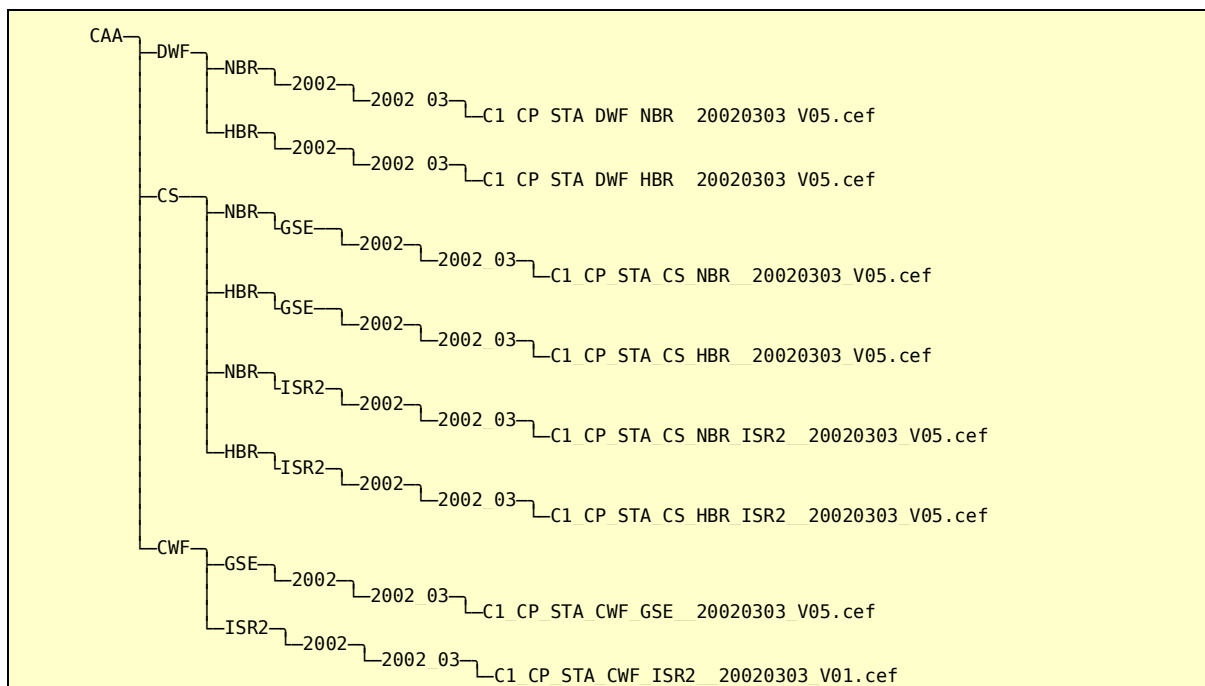
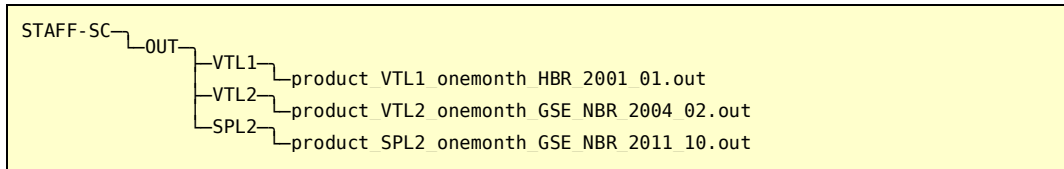


Table 6: Arborescence des fichiers CEF



On peut aussi noter l'existence d'un répertoire OUT, situé au même niveau que le L1 et L2 et qui contient l'output des programmes, que l'on peut consulter en cas de problèmes. Ces fichiers sont effacés régulièrement, à mesure que l'on a vérifié la bonne exécution de la chaîne de traitement.

L'arborescence se présente comme suit, avec les modes NBR et HBR et les coordonnées GSE et ISR2 :



*Table 7: Arborescence des fichiers CEF*



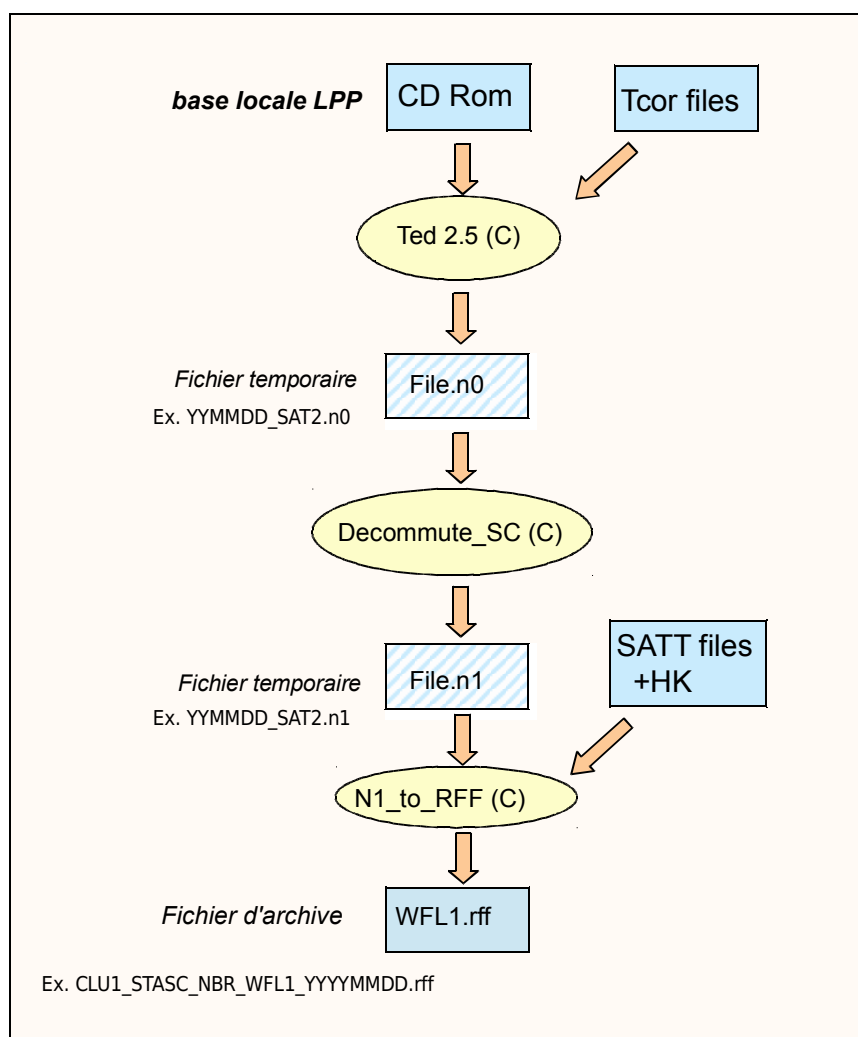
## 2.5. DESCRIPTION DES CHAÎNES DE PRODUCTION

### 2.5.1 Production des WFL1.rff

Les WFL1.rff sont donc la première étape de production, à partir des CD-ROM, des Tcor files et du logiciel Ted. La Figure 1 ci-dessous illustre les différentes étapes de cette chaîne.

Cette chaîne, écrite en langage C sous la direction de V. Bouzid, ne fait pas partie du pack RCL, qui ne travaille qu'à partir des données de niveau 1 complètes (avec des données des House Keeping indispensables), c'est à dire justement ces WFL1.

Un des points délicats dans cette chaîne concerne le calcul du « spin phase », c'est-à-dire l'angle entre le demi-plan défini par les axes +ZSR et +XSR du système de référence SR depuis le moment où la direction du soleil était contenue dans ce plan. Il se fait à partir du temps du « sun pulse » et son calcul est détaillé au chapitre 9.2.



**Figure 1 :** Chaîne de production des WFL1.rff. Les CD sont préalablement concaténés, ainsi que les SATT files et les HK satellites associés.

### 2.5.2 Production des VTL1.rff

Les VTL1.rff sont donc produits à partir des WFL1.rff par la chaîne suivante :

L'application **RCL\_get\_data\_CLUSTA\_WFL1** (écrite en bash) prend le fichier WFL1.rff du jour, ainsi que le premier bloc du fichier du jour précédent, et produit un fichier complet WFL1\_c.rff temporaire incluant, et dépassant même, une journée complète.

L'application **RCL\_waveform\_to\_vectime** calcule le temps de chaque vecteur de données, et transforme le format *WaveForm* en un format *Vectime*.

La Figure 2 ci-dessous résume cette chaîne. Les commandes de production correspondantes, détaillées au chapitre 3, sont :

```
RCL_product_VTL1_oneday 1 2009 12 14 NBR
RCL_product_VTL1_onemonth 2009 12 NBR
RCL_product_VTL1_oneyear 2009 NBR
```

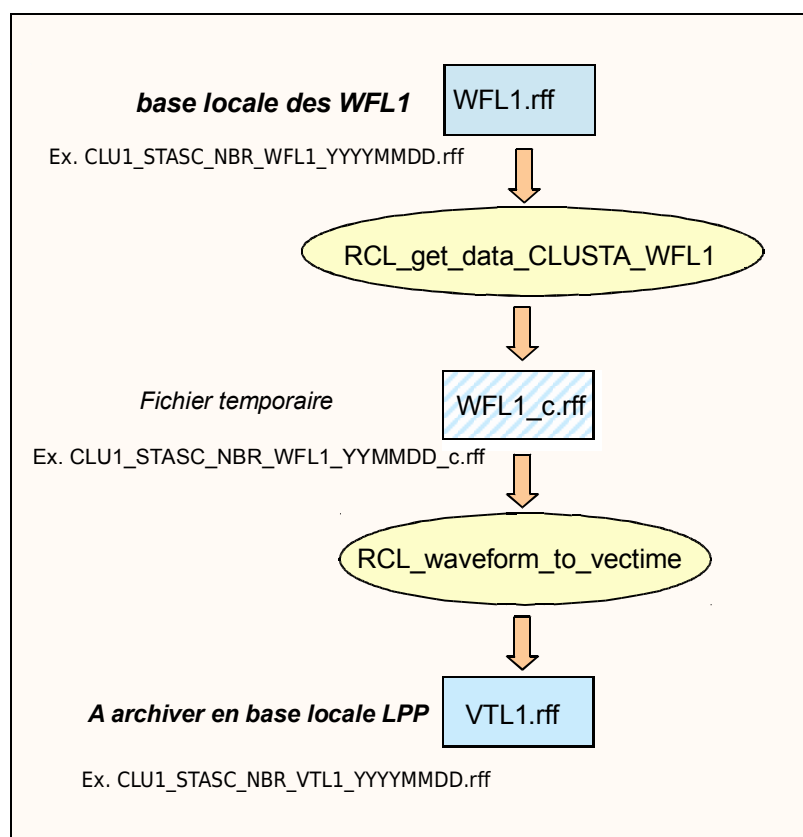


Figure 2 : Chaîne de production des VTL1.rff. Les fichiers WFL1.rff du jour courant ET du jour précédent sont requis.

### 2.5.3 Production des SPL2.rff

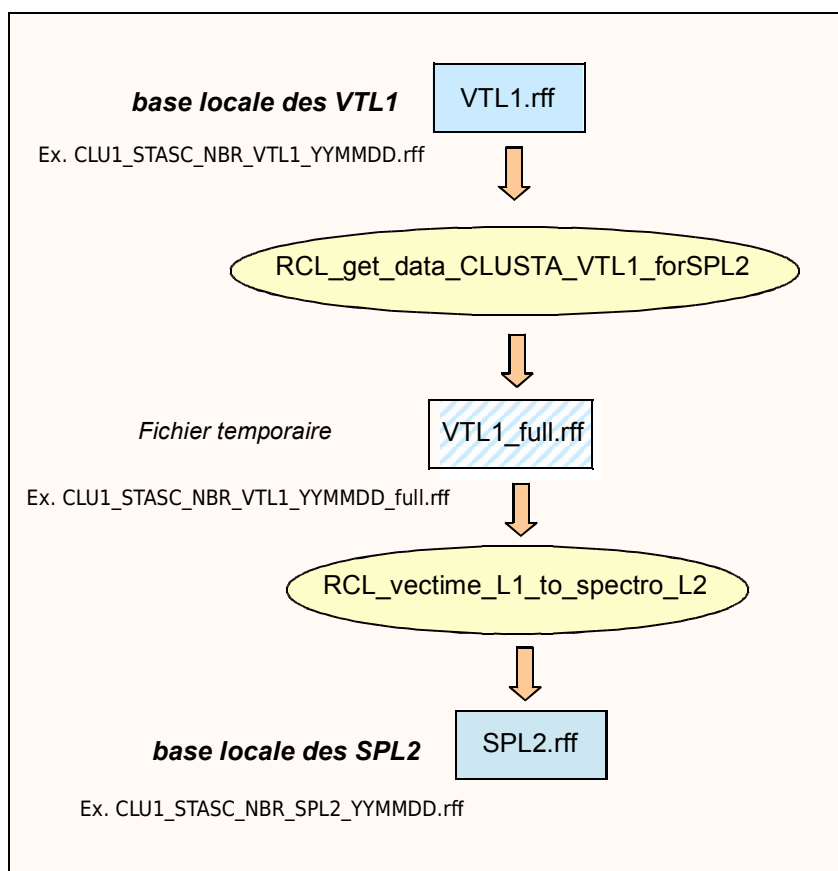
Les spectrogrammes **SPL2.rff** sont calculés à partir des données brutes **VTL1.rff** par l'application **RCL\_vectime\_L1\_to\_spectro\_L2** (voir chapitre 2.2.4) qui effectue le calcul des spectres et leur calibration par la méthode classique, fenêtres après fenêtres, sans chevauchement. Ce programme effectue aussi le changement de repère tournant (SR) vers le repère fixe ISR2 ou GSE.

L'application **RCL\_get\_data\_CLUSTA\_VTL1\_for\_SPL2** prend le fichier VTL1.rff du jour, ainsi que début du VTL1 du jour d'après, et produit un fichier complet VTL1\_full.rff temporaire incluant une journée complète, avec une partie du jour d'après, afin que le dernier spectre du jour puisse commencer juste avant minuit.

L'application **RCL\_vectime\_L1\_to\_spectro\_L2** effectue le calcul des spectres, leur calibration et le changement éventuel de coordonnées.

La Figure 3 ci-dessous résume cette chaîne. Les commandes de production correspondantes, détaillées au chapitre 3, sont :

```
RCL_product_SPL2_oneday    1 2009 12 14 NBR
RCL_product_SPL2_onemonth 2009 12      NBR
RCL_product_SPL2_oneyear   2009      NBR
```



**Figure 3:** Chaîne de production des SPL2.rff. Les fichiers VTL1.rff du jour courant ET du jour suivant sont requis.

#### 2.5.4 Production des VTL2.rff

Les **VTL2.rff** sont donc produits à partir des VTL1.rff par la chaîne suivante :

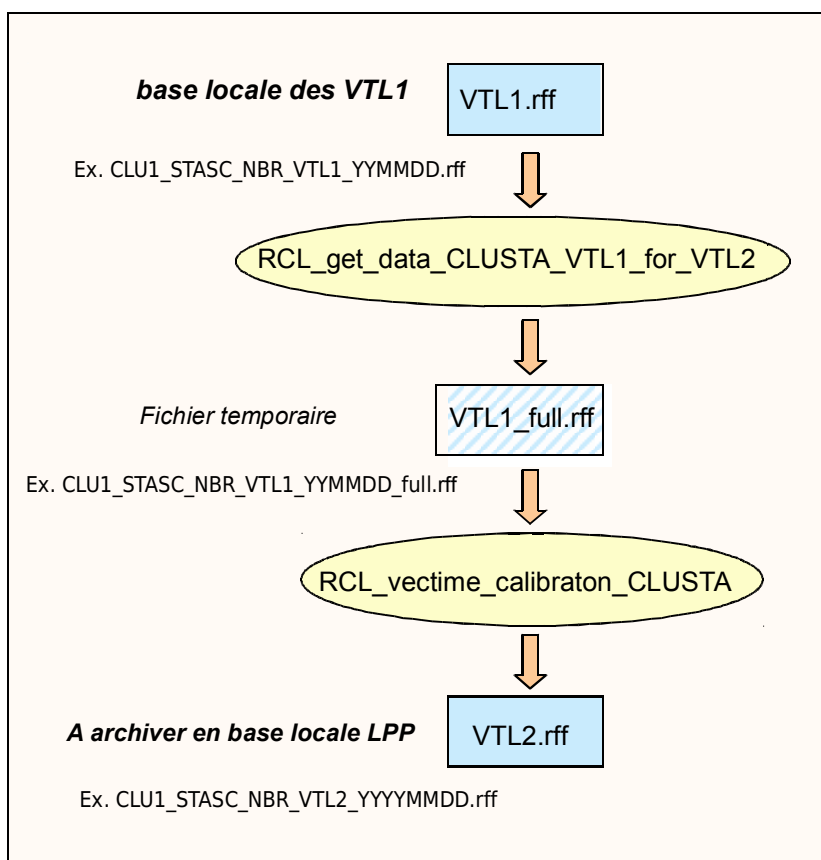
L'application **RCL\_get\_data\_CLUSTA\_VTL1\_for\_VTL2** (écrite en bash) prend le fichier VTL1.rff du jour, ainsi que la fin du fichier VTL1 du jour d'avant ET le début du VTL1 du jour d'après. Il produit ainsi un fichier VTL1\_full.rff temporaire incluant une journée complète, avec une partie de la fin du jour d'avant et une du début du jour d'après, car la calibration continue requière, pour le

calcul d'un point de temps, une fenêtre de données d'environ 40s (en NBR). Il faut donc, pour pouvoir obtenir un fichier démarrant à 0H et finissant à minuit 10 secondes avant, et 10 secondes après. Le raisonnement est le même en HBR.

L'application **RCL\_vectime\_calibration\_CLUSTERA** effectue la calibration continue du fichier VTL1\_full.rff, et produit un fichier VTL2.rff débarrassé des points en dehors de la journée.

La Figure 4 ci-dessous résume cette chaîne. Les commandes de production correspondantes, détaillées au chapitre 3, sont :

```
RCL_product_VTL2_oneday      1 2009 12 14 NBR
RCL_product_VTL2_onemonth_onesat 1 2009 12      NBR
RCL_product_VTL2_onemonth    2009 12      NBR
```

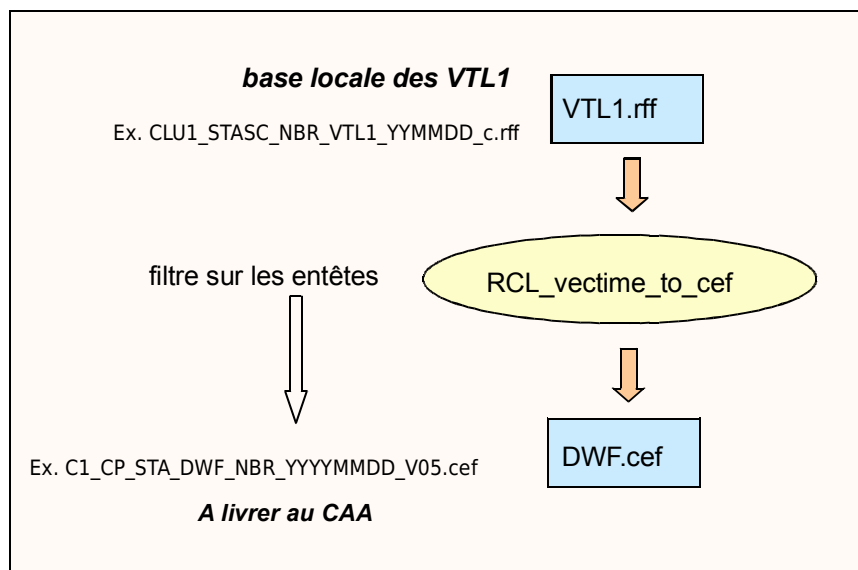


**Figure 4:** Chaîne de production des VTL2.rff. Les fichiers WFL1.rff du jour courant, du jour précédent ET du jour d'après sont requis.

### 2.5.5 Production des DWF.cef

Les **DWF.cef** sont donc le résultat de la conversion de format des fichiers VTL1.rff, faite par l'application **RCL\_vectime\_L1\_to\_cef** (voir Figure 5). Les commandes de production disponibles sont :

```
RCL_product_DWF_oneday      1 2009 12 14 NBR
RCL_product_DWF_onemonth    2009 12      NBR
RCL_product_DWF_oneyear     2009          NBR
```

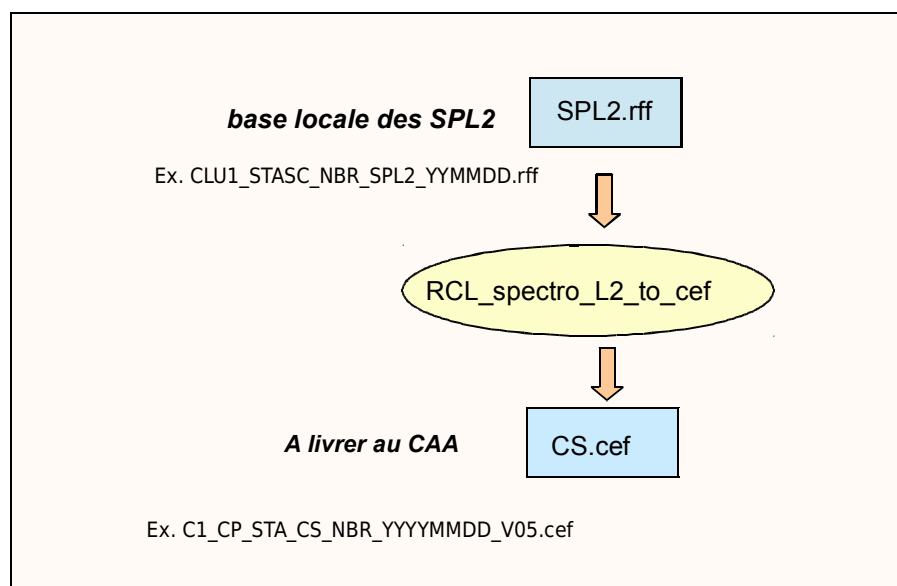


**Figure 5:** Chaîne de production des DWF.cef. Celle-ci se résume à un simple filtre de changement de format, ne concernant que la partie méta data.

### 2.5.6 Production des CS.cef

Les en CS.cef sont donc le résultat de la conversion de format des fichiers SPL2.rff, faite par l'application **RCL\_spectro\_L2\_to\_cef** (Figure 6). Les commandes de production disponibles sont :

RCL_product_SPL2_oneday	1	2009	12	14	NBR	GSE
RCL_product_SPL2_onemonth		2009	12		NBR	GSE
RCL_product_SPL2_oneyear		2009			NBR	GSE

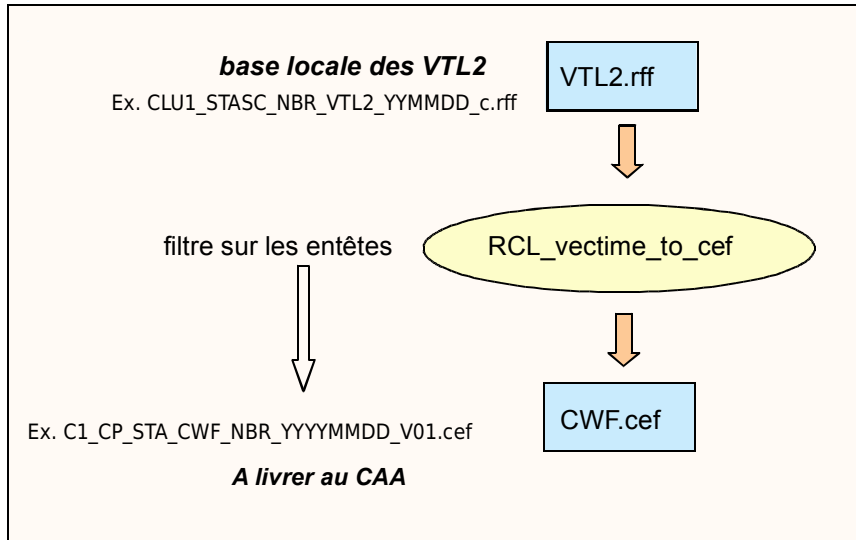


**Figure 6:** Chaîne de production des CS.cef. Celle-ci se résume à un simple filtre de changement de format, ne concernant que la partie méta data.

### 2.5.7 Production des CWF.cef

Les **CWF.cef** sont le résultat de la conversion de format des fichiers VTL2.rff, faite par l'application **RCL\_vectime\_L2\_to\_cef** (Figure 7). Les commandes de production disponibles sont :

```
RCL_product_CWF_oneday 1 2009 12 14 NBR
RCL_product_CWF_onemonth 2009 12 NBR
```



**Figure 7:** Chaîne de production des CWF.cef. Celle-ci se résume à un simple filtre de changement de format, ne concernant que la partie méta data.

### 2.5.8 Remarque sur les produits livrés au CSA

Ces produits pourraient ne pas être conservés durablement au LPP, dans la mesure où leur régénération à partir des VTL1.rff ou VTL2.rff est rapide (voir paragraphe 2.4.3). De plus, pour les DWF et les CWF, le champ «data» est identique aux fichiers VTL1.rff et VTL2.rff (cela a été voulu, voir chapitre 2.1). Seule la partie méta-data est différente pour ces deux produits entre le format RFF et le format CEF. Les CS, si le besoin s'en faisait sentir, ne sont pas trop longs non plus à régénérer.

### 2.5.9 Chaîne de production des images WEB

Les images web pour STAFF-SC sont constituées de spectrogrammes de la composante Z pour les 4 satellites. Il pourrait être produit de deux manières :

- A partir des SPL2.rff de la chaîne précédente
- Directement à partir des VTL2

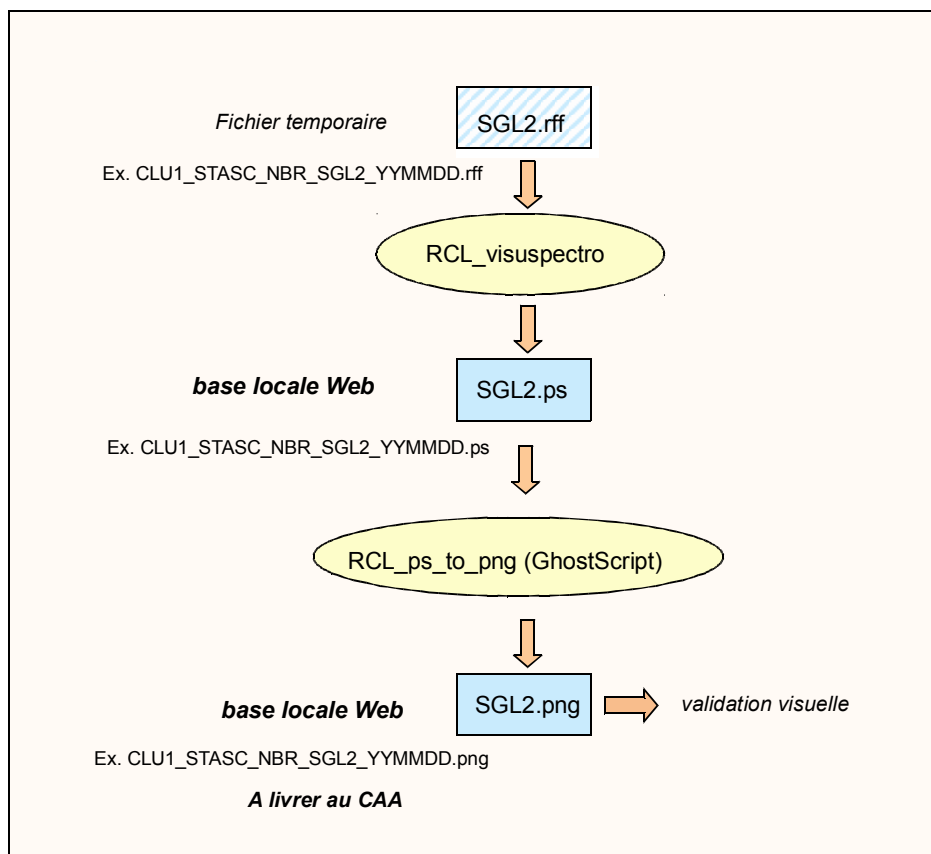
Néanmoins la méthode de calibration utilisée pour produire les VTL2 nécessite de connaître une demi-fenêtre de mesure avant et après le point de télémétrie que l'on calibre. Cela veut dire que, en cas de trous de données, les VTL2 «élargissent» les trous de données (voir chapitre 2.2.4) d'une demi fenêtre de calibration (1024 points, soit 40.95 sec en NBR et 8192 points, soit 18.20 s en HBR).

Pour cette raison les spectrogrammes du web sont produits à partir de la première méthode, qui correspond le mieux aux données initiales, et permettent ainsi de valider le fonctionnement de l'instrument ainsi que la production des CS.cef. Le détail de cette chaîne est résumé par la Figure 8 ci-dessous. Les SPL2 utilisés sont ceux qui sont produits en ISR2. En effet, en GSE, les 3

composantes sont mélangées et filtrées des basses fréquences (en dessous de 0.6 Hz), et donc moins utilisables pour contrôler le fonctionnement de l'expérience.

Les commandes de production, portant sur la base des SPL2.rff existante, sont :

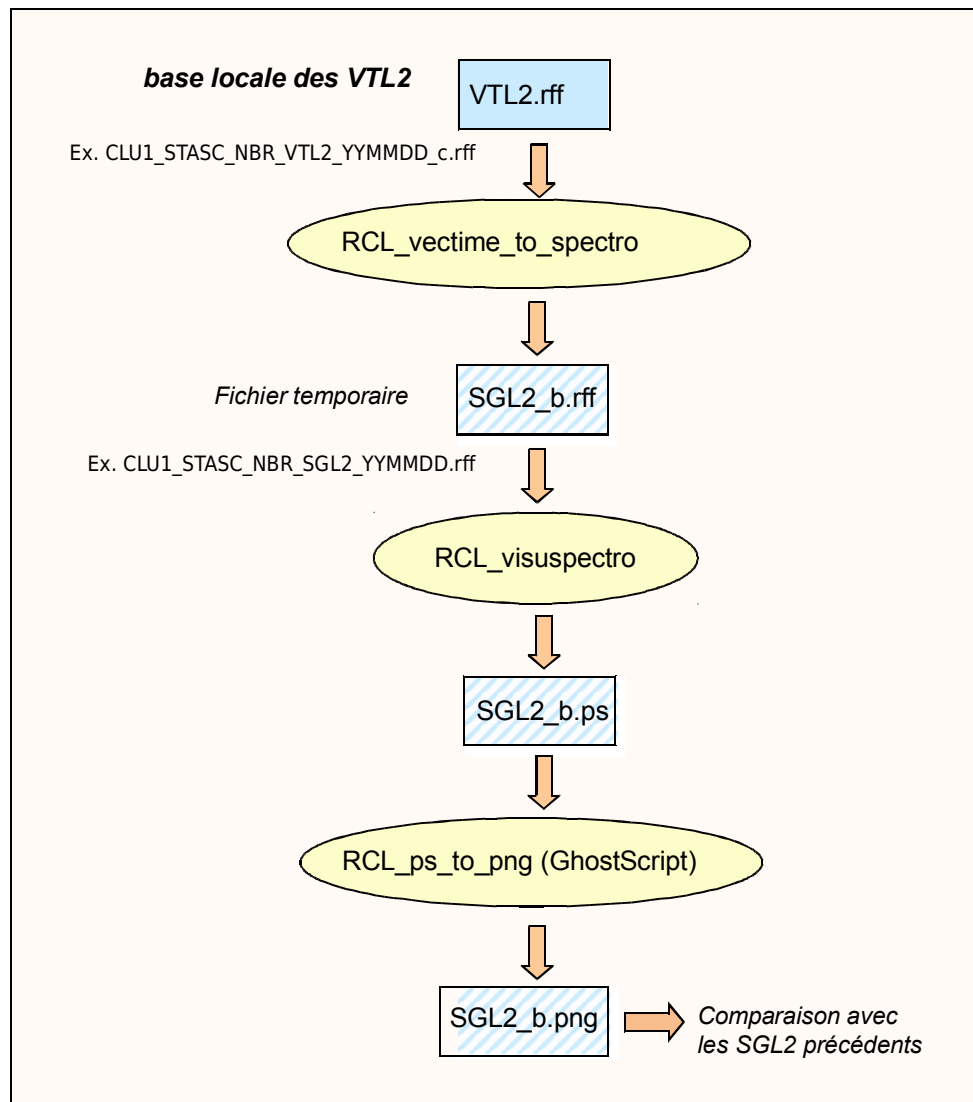
RCL_product_visu_SPL2_oneday	1	2009	12	14	NBR
RCL_product_visu_SPL2_onemonth		2009	12		NBR
RCL_product_visu_SPL2_oneyear		2009			NBR



**Figure 8:** Chaîne de production des images à mettre sur le web. Celles-ci servent aussi à la validation des données et des chaînes de traitement.

### 2.5.10 Validation des CWF

Pour ce faire, on peut créer des spectrogrammes, à 3 composante cette fois, selon les deux méthodes précédentes. On a ainsi deux classes de spectrogrammes, l'un dérivé directement des VTL1, et l'autre des VTL2. Les spectrogrammes dérivés des VTL2 sont produits suivant la chaîne ci-dessous (Figure 9).



**Figure 9:** Chaîne de production des spectrogrammes pour la validation des CWF.

Une manière de valider les CWF, et donc les VTL2, consiste alors à comparer les spectrogrammes calculés directement à partir des VTL1 et ceux issue des VTL2. Les résultats montrent un degré de cohérence élevé. Les seules différences observées sont bien sûr l'élargissement léger des trous de télémétrie dans le deuxième cas et éventuellement un meilleur despin pour ce dernier. En effet, lors de la production des VTL2, si la calibration est faite continuellement, le despin l'est également, et est donc de meilleure qualité. Mais les puissances observées restent les mêmes dans les deux cas.



### 2.5.11 Schéma général de la chaîne STAFF

Le Figure 10 ci-dessous résume les différentes chaînes de production utilisées et une estimation de leur temps d'exécution. Les chiffres sont données pour un mois de traitement et 4 satellites, dans les deux modes NBR et HBR, sur la machine cactus où elles ont été développées (Ubuntu x86\_64 GNU/Linux). Dans la pratique, la production de masse s'effectue sur la machine Tamaya de même type, mais plus puissante et sur 6 processeurs. Le détail des temps calculs et du volume des données traitées est indiqué dans les chapitres 3.3 et 3.4.

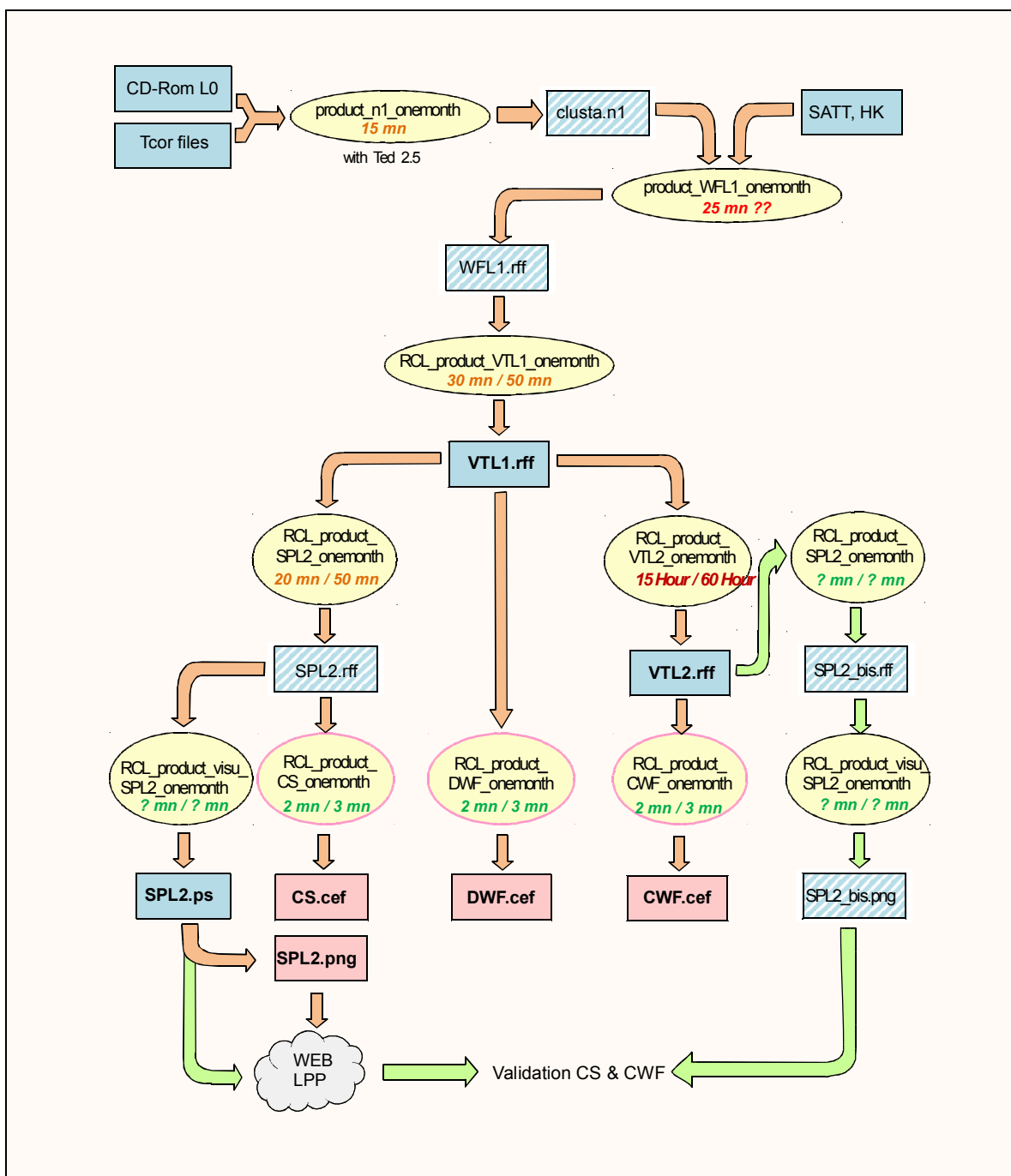


Figure 10: Schéma général des chaînes de production STAFF-SC



## 3. ORGANISATION DE LA PRODUCTION STAFF

### 3.1. CONTRAINTES DE PRODUCTION

#### 3.1.1 Disponibilité des fichiers Tcor

Les fichiers TCOR sont des fichiers délivrés par l'ESA environ 6 mois après les fichiers de télémétrie. Ils permettent de dater très précisément les blocs des fichiers WFL1. Ils sont donc indispensables pour fabriquer les VTL1 et leur pendant coté CSA que sont les DWF, ainsi que les VTL2 et les CWF pour les formes d'onde calibrées.

Cependant on ne peut pas attendre 6 mois pour produire les spectres et regarder les données par la visualisation des spectrogrammes SPL2. C'est pourquoi on produit, d'une manière provisoire, des WFL1 et des VTL1 quotidiennement, sans attendre la disponibilité des TCOR. La correction en temps, qui affecte de l'ordre de la dizaine de microseconde le temps initial, n'est donc pas faite, mais cette correction est de toutes manières invisible à l'œil nu sur les plots des spectrogrammes.

#### 3.1.2 Production des WFL1 et des VTL1 « Daily »

Les chaines de production sont exactement les mêmes que celles décrites aux chapitres 2.2.1 et 2.2.2, sauf que les TCOR ne sont pas utilisés dans la production des WFL1. Ils sont donc rangés dans une arborescence provisoire et vont servir pour la production des SPL2 et des CS.

#### 3.1.3 Production des SPL2 « Daily » et des CS

Ils sont donc produits à partir des VTL1 « daily » ce qui implique que la datation des CS délivrés au CSA *n'est pas corrigés des TCOR*. Les CS sont livrés en GSE.

#### 3.1.4 Production des PNG

Les images produites par la chaîne décrite au chapitre 2.5.9 (avec  $\Delta T \sim 10$  s.) à partir des WFL1 « daily » sont délivrées au CSA en temps que « Quick Looks », en même temps que les CS en ISR2. Seul les PNG sont livrés. Cette chaîne est également utilisée, mais avec des paramètres différents ( $\Delta T \sim 20$  s.), pour produire les PostScript et PNG qui servent à alimenter le web STAFF, ce qui permet de « browser » rapidement les spectrogrammes jour après jour et mode après mode, ce qui permet de vérifier le bon fonctionnement de l'expérience.

#### 3.1.5 Production des VTL1 et des DWF

Pour l'archivage au CSA des VTL1 au format DWF, les fichiers TCOR sont bien sûr requis, afin d'avoir une datation aussi précise que possible des vecteurs. Ils sont donc produits environ 6 mois après la réception des fichiers de télémétrie.

#### 3.1.6 Production des VTL2 et des CWF

Ils ne peuvent être faits bien sûr qu'après la production des VTL1 correctement datés. Rappelons que la production des VTL2 est assez lourde coté temps calcul (voir chapitre 3.3.2) et qu'une journée de VTL2 requiert non seulement le fichier VTL1 du jour correspondant, mais également la fin de celui du jour d'avant et le début de celui du jour d'après.

## 3.2. PARAMÉTRAGE DES COMMANDES DE PRODUCTION

### 3.2.1 RCL\_product\_VTL1\_oneday

La production des VTL1, « daily » ou définitive avec les fichiers TCOR, ne nécessite aucun paramétrage particulier, puisque l'application correspondante ne fait qu'interpoler les blocs de vecteur des WFL1. Il faut noter cependant que la fréquence d'échantillonnage précise varie d'un satellite à l'autre, comme l'indique la Table 8 ci-dessous. Ces valeurs sont bien sûr prises en compte dans les codes.

	NBR	HBR
C1	25.0005833	450.0105
C2	25.0003611	450.0065
C3	25.0001667	450.0030
C4	25.0001667	450.0030

Table 8: Cluster WEC (DWP) exact sampling frequency derived from the DWP clock

### 3.2.2 RCL\_product\_VTL2\_oneday

La production des VTL2 utilise la commande **RCL\_vectime\_calibration\_CLUSTA** décrite au chapitre 4.8.2 qui lance le programme de calibration continue des formes d'onde. La méthode de calibration utilisée est décrite dans la publication [13, Robert, 2013].

Comme il est expliqué dans ce papier, les formes d'onde enregistrées à bord par les search-coils sont « convoluées » non-linéairement par la fonction de transfert de l'instrument. Il en résulte que la déconvolution, et donc la calibration, *n'as pas de solution unique*.

Par conséquent il a été établi des paramètres de calibration « moyens », qui sont adaptés au mieux à la majorité des situations rencontrées, mais qui peuvent être très différentes. En effet, on ne choisit pas de la même manière les paramètres de calibration pour des événements stationnaires, comme des ondes pseudo monochromatiques, ou au contraire des événements très brefs ou abrupts, comme des traversées de magnétopause par exemple.

De même, suivant que le champ stationnaire soit fort ou non, aligné ou perpendiculaire à l'axe de spin, rapidement variable ou grossièrement constant, tout cela rentre en compte dans le choix des paramètres disponible dans la procédure **RCL\_vectime\_calibration\_CLUSTA** décrite au chapitre 4.8.2. Ils sont codés en dur dans la procédure **RCL\_product\_VTL2\_oneday** (voir chapitre 4.9.6) avec les valeurs données dans la Table 9 ci-dessous.

		NBR		HBR		Remarques
		ISR2	GSE	ISR2	GSE	
Fdet	detrend frequency	0.	0.	0.	0.	0. for classic despin
Fc	Frequency cut-off	0.1	0.6	0.1	0.6	To avoid spin tone with 3 mixed components
Fmin	frequency min for filtering	0.	0.	0.	0.	0. means Fc assumed
Fmax	frequency max for filtering	0.	0.	0.	0.	0. means Nyquist frequency assumed
Step	Processing steps	7	8	7	8	Reminder :
						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
N_Kern	Kernel size	1024	1024	4096	4096	~ 40 sec NBR, ~ 9s in HBR > ~ 2 spin period
N_shift	for sliding window	2	2	2	2	Best value
Apod	Wheiting function	g	g	g	g	Gaussian

Table 9 : Valeurs des paramètres de la commande RCL\_product\_VTL2\_oneday

### 3.2.3 RCL\_product\_SPL2\_oneday

La résolution temporelle demandée pour les spectres devait être la plus courte possible, à condition bien sûr de conserver une résolution en fréquence raisonnable (rappelons que  $\Delta f \cdot \Delta t = 1$ ).

Compte tenu qu'il faut pouvoir despinner correctement les formes d'onde avant de faire le spectre, la résolution temporelle minimum doit être d'au moins deux périodes de spin, ce qui conduit au choix des paramètres donnés dans la Table 10 ci-dessous:

		NBR		HBR		Remarques
		ISR2	GSE	ISR2	GSE	
Fdet	detrend frequency	0.	0.	0.	0.	0. for classic despin
Fc	Frequency cut-off	0.1	0.6	0.1	0.6	To avoid spin tone whit 3 mixed components
Fmin	frequency min for filtering	0.	0.	0.	0.	0. means Fc assumed
Fmax	frequency max for filtering	0.	0.	0.	0.	0. means Nyquist frequency assumed
Step	Processing steps	7	8	7	8	Reminder :
N_Kern	Kernel size	256	256	4096	4096	~ 10 sec NBR, ~ 9s in HBR > ~ 2 spin period
N_shift	for sliding window	256	256	4096	4096	No overlap
Apod	Wheiting function	t	t	t	t	Trapeze

**Table 10:** Valeurs des paramètres de la commande *RCL\_product\_SPL2\_oneday*

### 3.2.4 RCL\_product\_visu\_SPL2\_oneday

Le programme de visualisation destiné à produire les Quick-Looks utilise la commande **RCL\_visu\_spectro\_4Bz\_3h** décrite au chapitre 4.12.4. Cette commande visualise la composante Bz pour chacun des 4 satellites. Elle est appliquée sur les SPL2 en repère ISR2. Les paramètres de productions sont donnés dans la Table 11 ci-dessous.

		ISR2	Remarques
F1	frequency min for plot	0.	
F2	frequency max for plot	0.	0. means Nyquist frequency assumed
Pmin	Log min power for dynamic colors	0.	0. means automatic computation
Pmax	Log max power for dynamic colors	0.	0. means automatic computation
Fi1	frequency min for integrated power	0.6	To avoid large power in low frequencies
Fi2	frequency max for integrated power	0.	0. means Nyquist frequency assumed

**Table 11:** Valeurs des paramètres de la commande *RCL\_product\_visu\_SPL2\_oneday*

Les fichiers PostScript sont convertis automatiquement en PNG par la commande **RCL\_ps\_to\_png** ; la résolution des PNG est définie dans le *RCL\_config.bash*. Pour la production, elle est fixée à 300 ppp (or dpi). Un exemple des ces spectrogrammes est donné au chapitre 9.10.3.

### 3.2.5 RCL\_product\_visu\_orbit\_oneday

Rappelons qu'il y a plusieurs visualisations des trajectoires des satellites et des paramètres du tétraèdre. La commande lance la production de tous les graphiques décrits ci-dessous.

- **Les positions par séquence de 3 heures**

Elles sont produites en GSE *et* en GSM. Un exemple est donné au chapitre 9.10.9.

La procédure utilise la commande **RCL\_get\_data\_CLUPOS** pour obtenir les fichiers de position. Elle est lancée sur des séquences de 190.1 minutes, avec une résolution de 10 min, puis on crée les plots avec la commande **RCL\_visu\_CLUPOS**.

- **Les positions pour une orbite complète**

De la même manière, elles sont produites en GSE *et* en GSM. Un exemple est donné chapitre 9.10.10.

Comme précédemment on utilise la commande **RCL\_get\_data\_CLUPOS** pour obtenir les fichiers de position, puis on crée les plots avec la commande **RCL\_visu\_CLUPOS**.

- Pour les années antérieures à 2009, la période prise pour une orbite complète est de 3070 min (soit 51 heures et 10 min).
- Pour décembre 2009, la période prise est de 3070 min (soit 51 heures et 10 min).
- Pour les années supérieures ou égales à 2009, la période prise pour une orbite complète est de 3250 mn (soit 54 heures et 10 min).

Ces valeurs ont été définies grâce à la commande **RCL\_give\_CLUORB\_period**.

- **Les paramètres du tétraèdre**

Deux graphiques sont produits :

- Un graphique 2-D pour la position et la vitesse de chaque satellite, en repère GSE, avec en plus les positions en polaire, pour une période de 24 heures.
- Un second graphique donnant d'une part les positions dans le centre de masse, les inter-distances et les surface de chaque face, et d'autre part les paramètres d'élongation et de planarité du tétraèdre définie dans [14, Robert, 1998]. L'amplitude et la direction des axes de l'ellipsoïde sont également fournies, toujours pour une période de 24 heures.

Les paramètres de la géométrie du tétraèdre sont obtenus avec la commande **RCL\_get\_data\_CLUGEOM** sur une période de 20 min, en GSE, avec l'option « long ». La visualisation est faite par la commande **RCL\_visu\_CLUGEOM**.

Des exemples sont donnés au chapitre 9.10.11.

## 3.3. CONSOMMATION EN TEMPS CPU DES DIFFÉRENTES CHÂÎNES

### 3.3.1 Traitement des données brutes

Dans la chaîne de production des VTL1.rff, il faut distinguer de tout ce qui suit la partie « decom », qui travaille depuis les données compressées de niveau 0 sur les CD-ROM distribués par l'ESA, ou accessible en ligne, et qui sont ensuite entrées dans la chaîne de production des fichiers WFL1.rff. En effet, cette partie permet d'obtenir des formes d'onde décommutées et datées par blocs, sur lesquels on a ajouté le calcul de la phase du Sun pulse, qui permet ensuite d'effectuer des changements de coordonnées (voir chapitre 9.2). Tous les codes correspondants ont été écrits en C. Ces fichiers sont les fichiers d'entrée de l'exploitation scientifique faite par les Roproc pour

les versions  $\leq 4.4d$  [voir 5, Robert, 2011], et les programmes qui s'appliquent ensuite à ces fichiers ont été écrits en Fortran 90.

### 3.3.2 Traitements de niveau 1 et 2

Les traitements de niveau 1 et 2 (données calibrées) sont le point d'entrée des traitements scientifiques, et se traduit par le projet RCL, initialement dérivé des Roproc. Les codes ont été entièrement repris et réécrits en vue de simplifier les codes déjà existants, de faciliter leur maintenance et leur évolution, et aussi d'optimiser considérablement la vitesse d'exécution. En effet, à l'origine, le projet RCL avait des ambitions plus larges que la simple production des bases de données locales et des produits CSA, mais la version initiale était trop lourde pour être utilisée correctement en production, et difficilement maintenable ou modifiable (version antérieures ou égales à V0.7.13).

#### • *Ordre de grandeur du temps de production des VTL1 et DWF*

Le passage des WFL1 aux VTL1 (interpolation des blocs) prend environ 30 min en NBR et 50 min en HBR pour un mois et 4 satellites. La conversion en DWF est très rapide (2 à 3 min).

L'ensemble (VTL1, DWF) y compris la vérification des DWF par le CEFPass (logiciel ESA) prend environ 1 à 2 jours pour traiter un an de données.

#### • *Ordre de grandeur du temps de production des SPL2, CS et PNG*

Le passage des VTL1 aux SPL2 prend environ 20 mn en NBR et 50 mn en HBR pour un mois et 4 satellites. La conversion en CS est très rapide (2 à 3 mn).

L'ensemble (SPL2, CS) y compris la vérification des CS par le CEFPass et leur visualisation qui produit PS et PNG prend également environ 1 à 2 jours pour traiter un an de données.

#### • *Ordre de grandeur du temps de production des VTL2 et des CWF*

Le passage des VTL1 aux VTL2, qui correspond à la calibration continue des formes d'onde, est la partie la plus gourmande en temps CPU. Le programme correspondant a nécessité un gros travail d'optimisation avant de pouvoir être exploité. Il pourrait être encore optimisé davantage, mais les contraintes de calendrier de livraison au CSA ont fait que l'on s'est arrêté dans l'optimisation dès lors que le temps d'exécution est devenu raisonnable.

Ainsi la production des VTL2 prend environ 15 heures en NBR et 60 heures en HBR pour un mois et 4 satellites. La conversion en CWF est toujours très rapide (2 à 3 mn).

L'ensemble (VTL2, CWF) y compris la vérification des CWF par le CEFPass prends environ une semaine pour traiter un an de données.

## 3.4. VOLUME DES DONNÉES TRAITÉES

Si les temps de production paraissent élevés, il faut être conscient du volume des données traitées. Le lancement de la procédure **RCL\_dir\_properties\_pretty\_tree** sur les bases de données permet d'afficher ces résultats. On trouvera ci-dessous les résultats de la commande :

```
RCL_dir_properties_pretty_tree DIR
```

Où DIR est par exemple /NFS/tamaya/data2/staff-op/STAFF-SC/L2/RFF si on veut le volume occupé par la base des fichiers L2 (on a enlevé les informations sur les mois pour plus de clarté).

On peut rajouter le paramètre « Mo » si on veut le résultat des volumes dans cette unité (c'est en octets par défaut).

### 3.4.1 Les WFL1

On peut voir sur la Table 12 ci-dessous que les WFL1 « zippés » occupent de l'ordre de 200 Go en NBR et 115 Go en HBR, soit **315 Go** au total. L'arborescence comprend **340** répertoires et plus de **38000** fichiers. Le répertoire utilisé a été /NFS/tamaya/CLUSTERII/DAILY/STAFF-SC/L1/RFF au 28 octobre 2014.

Moctets, nb files, nb dir				
318996.054	37984	341.		
318996.054	37984	340	_WF	
203793.236	18992	169	_NBR	
8075.829	1460	12		_2001
13673.170	1460	12		_2002
18246.613	1460	12		_2003
17482.346	1464	12		_2004
17496.031	1460	12		_2005
17526.297	1460	12		_2006
17205.991	1460	12		_2007
16792.565	1464	12		_2008
16085.501	1460	12		_2009
15855.594	1460	12		_2010
14886.331	1460	12		_2011
14779.765	1464	12		_2012
15687.201	1460	12		_2013
115202.802	18992	169	_HBR	
8526.413	1460	12		_2001
8790.128	1460	12		_2002
8604.542	1460	12		_2003
10829.577	1464	12		_2004
10067.280	1460	12		_2005
8934.137	1460	12		_2006
8414.817	1460	12		_2007
8668.233	1464	12		_2008
8333.990	1460	12		_2009
7981.583	1460	12		_2010
8217.482	1460	12		_2011
10087.830	1464	12		_2012
7746.794	1460	12		_2013

Table 12: Taille et nombre de fichiers de l'arborescence des fichiers WFL1 « zippés »

### 3.4.2 Les VTL1

Le répertoire exploré est /NFS/tamaya/data2/staff-op/STAFF-SC/L1/RFF. Ainsi on peut voir que les VTL1 occupent 2.4 To en NBR et 1.4 To en HBR, soit **3.8 To** au total. L'arborescence comprend **355** répertoires et plus de **39000** fichiers. Ces fichiers ne sont pas « zippés ».

Moctets, nb files, nb dir				
3864000.725	39436	355.		
3864000.725	39436	354	_VT	
1428708.524	19716	176	_HBR	
102771.597	1460	12		_2001
102669.910	1460	12		_2002
97071.030	1460	12		_2003
125344.162	1464	12		_2004
117985.567	1460	12		_2005
108804.334	1460	12		_2006
104511.947	1460	12		_2007
106100.236	1464	12		_2008
107661.918	1460	12		_2009
100842.676	1460	12		_2010
92509.348	1460	12		_2011
118336.659	1464	12		_2012
86512.484	1460	12		_2013
57586.627	724	6		_2014
2435292.332	19720	176	_NBR	
90370.654	1460	12		_2001
151058.465	1460	12		_2002
201485.222	1460	12		_2003
196222.517	1464	12		_2004
197711.708	1460	12		_2005
199703.708	1460	12		_2006
199199.015	1460	12		_2007
197253.579	1464	12		_2008
194939.257	1460	12		_2009
188235.186	1460	12		_2010
174645.674	1464	12		_2011
172518.883	1464	12		_2012



182064.628	1460	12	_2013
89883.705	724	6	_2014

Table 13: Taille et nombre de fichiers de l'arborescence des fichiers VTL1

### 3.4.3 Les VTL2

La même opération faite sur les RFF des VTL2 est donnée Table 14. Mais les VTL2 sont « zippés » (.gz) car beaucoup plus volumineux que les VTL1.

Ainsi les VTL2 « zippés » occupent 3.6 To en NBR et 2. To en HBR, soit **5.6 To** au total. L'arborescence compte deux fois plus de fichiers et de répertoires que les VTL1, car ces données sont produites dans deux repères, ISR2 et GSE. On a donc **710** répertoires et plus de **78000** fichiers.

On peut aussi remarquer que les données en ISR2 sont plus volumineuses que celle en GSE : en effet, en ISR2 on a rajouté les deux composantes Dx et Dy du champ continu dans le plan de spin.

Moctets, nb files, nb dir			
6992655.024	153406	1384.	
<b>5617136.697</b>	78864	710	_VT
<b>1989336.629</b>	39432	354	_HBR
651465.654	19716	176	_GSE
36460.061	1460	12	_2001
35239.272	1460	12	_2002
32128.494	1460	12	_2003
42926.191	1464	12	_2004
42231.206	1460	12	_2005
39795.900	1460	12	_2006
38132.511	1460	12	_2007
38667.784	1464	12	_2008
38921.290	1460	12	_2009
36804.080	1460	12	_2010
33851.900	1460	12	_2011
43411.325	1464	12	_2012
115517.260	1460	12	_2013
77378.363	724	6	_2014
1337870.909	19716	176	_ISR2
42610.508	1460	12	_2001
40727.560	1460	12	_2002
37340.619	1460	12	_2003
50054.038	1464	12	_2004
49252.307	1460	12	_2005
46491.390	1460	12	_2006
44548.841	1460	12	_2007
45156.635	1464	12	_2008
186896.400	1460	12	_2009
176141.648	1460	12	_2010
161192.526	1460	12	_2011
206493.073	1464	12	_2012
150292.414	1460	12	_2013
100672.979	724	6	_2014
<b>3627799.806</b>	39432	354	_NBR
1214791.418	19716	176	_GSE
35060.670	1460	12	_2001
59506.663	1460	12	_2002
79414.215	1460	12	_2003
77417.071	1464	12	_2004
78097.064	1460	12	_2005
78838.718	1460	12	_2006
78574.174	1460	12	_2007
77737.009	1464	12	_2008
76843.532	1460	12	_2009
73764.962	1460	12	_2010
68808.958	1460	12	_2011
67733.545	1464	12	_2012
242975.195	1460	12	_2013
120019.698	724	6	_2014
2413008.519	19716	176	_ISR2
41821.327	1460	12	_2001
70857.056	1460	12	_2002
94944.166	1460	12	_2003
92525.781	1464	12	_2004
93323.248	1460	12	_2005
94198.866	1460	12	_2006
93906.633	1460	12	_2007
93064.487	1464	12	_2008
338249.351	1460	12	_2009
326365.118	1460	12	_2010
302757.577	1460	12	_2011
298717.479	1464	12	_2012
316124.791	1460	12	_2013
156152.562	724	6	_2014

Table 14: Taille et nombre de fichiers de l'arborescence des fichiers VTL2 « zippés »

### 3.4.4 Les SPL2

On peut voir sur la Table 15 que les SPL2 « zippés » occupent **900 Mo** en NBR et **500 Mo** en HBR, soit **1.4 To** au total. Comme ce sont des spectres, et à cause des trous de télémesure, l'arborescence compte un peu moins de fichiers et de répertoires que les VTL2, soit 74542 fichiers répartis sur 672 répertoires.

Moctets, nb files, nb dir							
-----							
1375518.458	74542	672	_SP				
500988.346	37267	335		_HBR			
245689.025	18272	164			_GSE		
19371.465	1460	12				2001	
18712.396	1460	12				2002	
17099.440	1460	12				2003	
22617.530	1464	12				2004	
21954.951	1460	12				2005	
20534.348	1460	12				2006	
19697.404	1460	12				2007	
19941.413	1464	12				2008	
20171.678	1460	12				2009	
19059.575	1460	12				2010	
17434.405	1460	12				2011	
22303.977	1464	12				2012	
6790.436	740	7				2013	
255299.322	18995	169			_ISR2		
19376.951	1460	12				2001	
18722.161	1460	12				2002	
17104.051	1460	12				2003	
22623.576	1464	12				2004	
21960.866	1460	12				2005	
20539.357	1460	12				2006	
19709.463	1460	12				2007	
19960.293	1464	12				2008	
20210.139	1460	12				2009	
19068.451	1460	12				2010	
17440.354	1460	12				2011	
22327.908	1464	12				2012	
16255.749	1463	12				2013	
874530.144	37275	335		_NBR			
427787.059	18280	164			_GSE		
17045.467	1460	12				2001	
28625.101	1460	12				2002	
38246.109	1460	12				2003	
37261.562	1464	12				2004	
37512.810	1460	12				2005	
37887.095	1460	12				2006	
37801.640	1460	12				2007	
37468.709	1464	12				2008	
36938.813	1460	12				2009	
35691.155	1460	12				2010	
33100.055	1462	12				2011	
32519.010	1470	12				2012	
17689.522	740	7				2013	
446743.085	18995	169			_ISR2		
17119.602	1460	12				2001	
28751.329	1460	12				2002	
38411.870	1460	12				2003	
37419.459	1464	12				2004	
37669.954	1460	12				2005	
38049.403	1460	12				2006	
37960.909	1460	12				2007	
37622.641	1464	12				2008	
37102.264	1460	12				2009	
35825.332	1460	12				2010	
33229.756	1460	12				2011	
32741.673	1464	12				2012	
34838.884	1463	12				2013	

Table 15: Taille et nombre de fichiers de l'arborescence des fichiers SPL2 « zippés »

### 3.4.5 Les plots 4Bz en PNG

Quand aux PNG livrés au CSA, leur taille est beaucoup moindre, comme on peut le voir sur la Table 16. Quelques tests en GSE on été effectués pour la validation des CS, mais ils ne sont généralement pas gardés. L'ensemble fait environ **50 Mo**.

Moctets, nb files, nb dir			
50564.497	44103	415	_PNG_
50564.497	44103	414	_SP_
45471.842	40980	208	_NBR_
4186.101	3941	28	_GSE_
1538.010	1629	12	_2001_
295.452	237	1	_2004_
261.393	232	1	_2012_
7.068	8	1	_2013_
2084.177	1835	8	_2014_
41285.743	37039	178	_ISR2_
1486.826	1629	12	_2001_
2403.720	2310	12	_2002_
3294.633	2864	12	_2003_
3244.050	2835	12	_2004_
3264.213	2847	12	_2005_
3328.334	2883	12	_2006_
3305.040	2876	12	_2007_
3300.310	2877	12	_2008_
3271.127	2887	12	_2009_
3216.709	2887	12	_2010_
3046.599	2847	12	_2011_
2978.573	2687	12	_2012_
3121.880	2775	12	_2013_
2023.728	1835	8	_2014_
5092.654	3120	204	_HBR_
598.132	324	24	_GSE_
318.557	158	12	_2001_
35.631	19	1	_2004_
243.945	147	8	_2014_
4494.522	2796	178	_ISR2_
312.194	158	12	_2001_
302.498	163	12	_2002_
280.915	148	12	_2003_
377.060	220	12	_2004_
364.785	228	12	_2005_
334.152	182	12	_2006_
316.663	166	12	_2007_
326.948	187	12	_2008_
331.717	194	12	_2009_
319.764	217	12	_2010_
318.642	231	12	_2011_
380.951	287	12	_2012_
289.188	268	12	_2013_
239.045	147	8	_2014_

Table 16: Taille et nombre de fichiers de l'arborescence des fichiers PNG»

### 3.4.6 Résumé des volumes occupés

La table Table 17 ci-dessous résume les volumes occupés au 1<sup>er</sup> avril 2015, et sont donnés à titre indicatif, la mission n'étant pas terminée.

RFF	compression	Volume en Go	Nombre de fichiers	Nombre de répertoires
WFL1	zippés	319	37984	340
VTL1	*aucune*	3875	39898	360
VTL2 GSE+ISR2	Zippés partiellement	5683	79788	722
SPL2 GSE+ISR2	zippés	1387	75463	683
<b>Total</b>		<b>11 264</b>	<b>233 133</b>	<b>2 105</b>

CEF	compression	Volume en Go	Nombre de fichiers	Nombre de répertoires
DWF	zippés	1674	39448	357
CWF GSE+ISR2	zippés	3040	39512	354
CS GSE seuls	zippés	561	39964	363
PNG ISR2 seuls		50	44799	426
<b>Total</b>		<b>5325</b>	<b>163723</b>	<b>1500</b>
<b>TOTAL RFF + CEF</b>		<b>16 589</b>	<b>396 856</b>	<b>3 605</b>

**Table 17:** *Résumé des volumes occupés par les bases de données locales (Go)*

L'ensemble des VTL1, VTL2, SPL2 fait près de **11 To**. Dans ce chiffre, les WFL1, VTL2 et SPL2 sont « zippés ». A titre indicatif, un fichier VTL1 de 140 Mo « zippé » voit sa taille réduite à 30 Mo, soit un facteur de compression de 4.7 et donc une réduction de volume de **21 %**.

Ce qui veut dire que sans compresser la base des VTL2 le volume de données sur 13 ans atteindrais de l'ordre de **32 To**.

Pour les fichiers CEF, il faut noter que les CWF sont données en ISR2 et en GSE, mais que les deux modes NBR et HBR sont fusionnés. Les CS ne sont donnés que en GSE.

## 4. DESCRIPTION DES COMMANDES RCL

Dans ce chapitre, la description des commandes s'est faite en partie par un programme qui utilise l'aide en ligne de chaque commande, qui est en anglais. La description de chaque commande est restée dans cette langue.

### 4.1. INFORMATIONS SUR LE LOGICIEL

#### 4.1.1 *RCL\_menu*

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

Usage : `RCL_menu`

#### 4.1.2 *RCL\_list*

Return alphabetic list of all RCL commands; create file `RCL_list.txt`

Usage : `RCL_list`

Result will be as:

```
=====
create list of all available RCL commands
=====

Content of RCL_list.txt :
-----

RCL_check_rff
RCL_check_rff_dir
RCL_clean_rff
RCL_compare_versions
RCL_compare_versions_long
RCL_copy_rff
RCL_current_date
RCL_decode_datiso
RCL_diff_rff
-----

RCL_visu_CLUSTER0M
RCL_visu_CLUSTER0S
RCL_visu_polar
RCL_visu_spectro
RCL_visu_spectro_3H
RCL_visu_spectro_4Bz
RCL_visu_spectro_4Bz_3H
RCL_visu_vectime
RCL_visu_vectime_widget
RCL_waveform_to_vectime

Nb of RCL commands : 100
```

**Table 18:** *Exemple de résultat de la commande RCL\_list*

**4.1.3 RCL\_version**

Return current RCL version used, ex: RCL\_V2.2

Usage : RCL\_version

**4.1.4 RCL\_make\_minidoc**

Mini\_doc.txt creates a file containing the list of RCL commands, and a brief description and list of parameters for each command. Served notably for writing this chapter.

Usage : RCL\_make\_minidoc

**4.1.5 RCL\_compare\_versions**

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

Usage : RCL\_compare\_versions RCL\_previous\_dir

ex : RCL\_compare\_versions /NFS/tamaya/data2/staff-op/RCL\_V2p0

**4.1.6 RCL\_compare\_versions\_long**

Compare two versions of RCL pack. Same as RCL\_compare\_versions, but comparison is deeper.

Usage : RCL\_compare\_versions\_long RCL\_previous\_dir

ex : RCL\_compare\_versions\_long /NFS/tamaya/data2/staff-op/RCL\_V2p0

## 4.2. MANIPULATION DES FICHIERS RFF

**4.2.1 RCL\_check\_rff**

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

Usage : RCL\_check\_rff toto.rff  
toto.rff : name of a RFF file

**4.2.2 RCL\_check\_rff\_dir**

Check of all RFF files of a given directory

Usage : RCL\_check\_rff\_dir toto  
toto : name of a directory containing RFF files

**4.2.3 RCL\_info\_rff**

Return main info of given RFF file

Usage : RCL\_info\_rff toto.rff option  
toto.rff : name of a RFF file  
option : l (long) s (short)

**4.2.4 RCL\_info\_rff\_dir**

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

Usage : RCL\_info\_rff\_dir toto  
toto : name of a directory containing RFF files  
This command uses RCL\_info\_rff

**4.2.5 RCL\_clean\_rff**

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

Usage : RCL\_clean\_rff toto.rff  
 toto.rff : name of a RFF file

**4.2.6 RCL\_copy\_rff**

To copy a file by updating the field FILE\_NAME.

Usage : RCL\_copy\_rff toto1.rff toto2.rff  
 toto1.rff : RFF source file  
 toto2.rff : RFF target file

**4.2.7 RCL\_diff\_rff**

Compute difference between 2 RFF files

Usage : RCL\_diff\_rff toto1.rff toto2.rff  
 toto1.rff : name of a first RFF file  
 toto2.rff : name of a second RFF file

**4.2.8 RCL\_move\_rff**

Move a RFF file with update of file name inside

Usage : RCL\_move\_rff toto1.rff toto2.rff  
 toto1.rff : RFF file  
 toto2.rff : RFF new name

This command uses RCL\_copy\_rff

**4.2.9 RCL\_reduce\_time\_rff**

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

Usage : RCL\_reduce\_time\_rff toto1\_VT.rff toto2\_VT.rff datiso1 datiso2  
 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  
 ex: 2001-09-23T09:00:39.504Z

ex :RCL\_reduce\_time\_rff toto1\_VT.rff toto2\_VT.rff 2001-09-23T09:10:00.000Z \  
 2001-09-23T09:30:00.000Z

### 4.3. COMMANDES DE TRAITEMENTS GÉNÉRIQUES

Ces commandes ont été développées pour le traitement de CLUSTER/SATFF-SC, mais peuvent être utilisées pour n'importe quel type de données de type *Waveform* ou *Vectime*, pour n'importe quelle expérience de type « Onde ».

#### 4.3.1 RCL\_list\_block\_WF

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

Usage : RCL\_list\_block\_WFL1.rff

ex : RCL\_list\_block\_WF CLU4\_STASC\_NBR\_WFL1\_20050323.rff  
will give :

```
=====
Run of RCL_list_block_WF.exe :
Please wait...

2005-03-23T00:00:00.965544Z 000000000001 102.70
2005-03-23T00:00:01.965538Z 000000000001 189.71
2005-03-23T00:00:02.965531Z 000000000001 276.72
2005-03-23T00:00:03.965524Z 000000000001 3.74
2005-03-23T00:00:04.965517Z 000000000001 90.75
2005-03-23T00:00:05.965510Z 000000000001 177.76
etc...
```

Table 19: Exemple de résultat de la commande RCL\_list\_block\_WF

#### 4.3.2 RCL\_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. The inverse transformation is possible. All vectors are now dated to the time of the block thank to the precise sampling frequency for each satellite..

Usage : RCL\_waveform\_to\_vectime toto\_WF.rff toto\_VT.rff

toto\_WF.rff : input file name of class WaveForm RFF.

toto\_VT.rff : output file name of class VecTime RFF.

#### 4.3.3 RCL\_vectime\_to\_spectro

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

Usage : RCL\_vectime\_to\_spectro VT.rff SP.rff N-Kern N\_shift Apod

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)

ex: RCL\_vectime\_to\_spectro CLU1\_STASC\_NBR\_VT\_20091213.rff\  
CLU1\_STASC\_NBR\_SP\_20091213.rff 512 512 t

#### 4.3.4 RCL\_spectro\_to\_polar

Produce a spectrogram from a VecTime file, according Roproc .resu file format.

Usage : RCL\_spectro\_to\_polar toto\_VT.rff

toto\_VT.rff : name of an input RFF VTL1 file



**4.3.5 RCL\_spectro\_xyz\_to\_lrz**

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

Usage : RCL\_spectro\_xyz\_to\_lrz toto\_SP.rff  
 toto\_SP.rff : name of an input RFF SP file

**4.4. ACCÈS À LA BASE LOCALE CLUSTER/STAFF-SC L1****4.4.1 RCL\_get\_data\_CLUSTA\_VTL1**

Get entire day of CLUSTER/STA VTL1 data

Usage : RCL\_get\_data\_CLUSTA\_VTL1 SatNum year month day BitRate

ex: RCL\_get\_data\_CLUSTA\_VTL1 3 2001 09 23 NBR

Note : if data base file is zipped, gunzip it

**4.4.2 RCL\_get\_data\_CLUSTA\_VTL2**

Get entire day of CLUSTER/STA VTL2 data

Usage : RCL\_get\_data\_CLUSTA\_VTL2 SatNum year month day BitRate Coord

ex: RCL\_get\_data\_CLUSTA\_VTL2 3 2001 09 23 NBR ISR2

Note : if data base file is zipped, gunzip it

**4.4.3 RCL\_get\_data\_CLUSTA\_WFL1\_forVTL1**

Get entire day of CLUSTER/STAFF WFL1 + last bloc of previous day.

Usage : RCL\_get\_data\_CLUSTA\_WFL1\_forVTL1 SatNum year month day BitRate

ex: RCL\_get\_data\_CLUSTA\_WFL1\_forVTL1 3 2001 09 23 NBR

This command uses RCL\_previous\_day  
 RCL\_version

**4.4.4 RCL\_get\_data\_CLUSTA\_VTL1\_forVTL2**

Get entire day of CLUSTER/STA VTL1 +a bit from previous and next day .

Usage : RCL\_get\_data\_CLUSTA\_VTL1\_forVTL2 SatNum year month day BitRate

ex: RCL\_get\_data\_CLUSTA\_VTL1\_forVTL2 3 2001 09 23 NBR

**4.4.5 RCL\_get\_data\_CLUSTA\_VTL1\_forSPL2**

Get entire day of CLUSTER/STA VTL1 +a bit from next day .

Usage : RCL\_get\_data\_CLUSTA\_VTL1\_forSPL2 SatNum year month day BitRate

ex: RCL\_get\_data\_CLUSTA\_VTL1\_forSPL2 3 2001 09 23 NBR

This command uses RCL\_next\_day  
 RCL\_previous\_day  
 RCL\_version

#### 4.5.1 RCL\_get\_data\_CLUPOS

```
Usage : RCL_get_data_CLUPOS year month day hh mm ss duration dt coord
                                year month day : date
                                hh mm ss       : start time
                                duration         : duration of period in minutes
                                dt              : time resolution in minutes
                                coord           : coordinates (only gel, gse or gsm)
```

#### 4.5.2 RCL get data CLUGEOM

[illegible]

#### 4.5.3 RCL give CLUORB period

will give :

**Table 20:** *Exemple de résultat de la commande RCL give CLUORB period*

## 4.6. ACCÈS À LA BASE LOCALE CLUSTER / FGM L2

### 4.6.1 ***RCL\_get\_data\_CLUFGM***

Get CLUSTER/FGM data from cef data base ; read the day file.cef in the local database and create a .bav ; Mode : SPIN or 5VPS

Usage : RCL\_get\_data\_CLUFGM SatNum year month day Mode

ex: RCL\_get\_data\_CLUFGM 3 2001 09 23 SPIN

### 4.6.2 ***RCL\_fgm\_cef\_to\_rff***

Convert a fgm.cef file into a fgm\_VTL2.rff file

Usage : RCL\_fgm\_cef\_to\_rff toto.cef

toto.cef : name of an input CEF file

### 4.6.3 ***RCL\_fgm\_cef\_to\_bav***

Convert a fgm.cef file into a fgm bav file

Usage : RCL\_fgm\_cef\_to\_bav toto.cef

toto.cef : name of an input CEF file

### 4.6.4 ***RCL\_fgm\_cef\_gse\_to\_sr2***

Convert a fgm.cef file with data in GSE to SR2 system

Usage : RCL\_fgm\_cef\_gse\_to\_sr2 toto.cef rasc dec

toto.cef : name of an input CEF file

rasc,dec: right ascension and declination of the spin axis in GEI system

## 4.7. ACCÈS À LA BASE DISTANTE CSA / FGM L2

### 4.7.1 ***RCL\_download\_data\_CLUFGM\_oneday\_t1t2***

Get time period of FGM data from CSA (SPIN, 5VPS or FULL).

Usage : RCL\_download\_data\_CLUFGM\_oneday SatNum year month day H1 M1 H2 M2 Mode

ex: RCL\_download\_data\_CLUFGM\_oneday 3 2001 09 23 09 20 10 30 SPIN

### 4.7.2 ***RCL\_download\_data\_CLUFGM\_oneday***

Get entire day of FGM data from CSA (SPIN, 5VPS or FULL).

Usage : RCL\_download\_data\_CLUFGM\_oneday SatNum year month day Mode

ex: RCL\_download\_data\_CLUFGM\_oneday 3 2001 09 23 SPIN

This command use RCL\_next\_day

### 4.7.3 ***RCL\_download\_data\_CLUFGM\_onemonth***

Download cef files 1 month 4 sat

Usage: RCL\_download\_data\_CLUFGM\_onemonth year month Mode

ex: RCL\_download\_data\_CLUFGM\_onemonth 2009 12 SPIN

This command use RCL\_download\_data\_CLUFGM\_onemonth\_nolog

**4.7.4 RCL\_download\_data\_CLUFGM\_onemonth\_nolog**

Get entire month of FGM data CSA , with creation of local directories tree.

Usage : RCL\_download\_data\_CLUFGM\_onemonth\_nolog year month Mode

ex: RCL\_download\_data\_CLUFGM\_onemonth\_nolog 2001 09 SPIN

This command use RCL\_list\_days\_of\_month et  
RCL\_download\_data\_CLUFGM\_oneday

**4.7.5 RCL\_download\_data\_CLUFGM\_oneyear**

Get entire year of FGM data from CSA with creation of directories tree.

Usage : RCL\_download\_data\_CLUFGM\_oneyear year Mode

ex: RCL\_download\_data\_CLUFGM\_oneyear 2009 SPIN

This command use RCL\_download\_data\_CLUFGM\_onemonth

**4.8. TRAITEMENTS SPÉCIFIQUES À CLUSTER/STAFF-SC****4.8.1 RCL\_vectime\_L1\_to\_spectro\_L2**

Calibrate a L1 STAFF-SC RFF Vectime file and produce calibrated spectrogram .

**The way to do this calibration is explained in details in section 8.3.1**

Usage : 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)

ex: RCL\_vectime\_L1\_to\_spectro\_L2 CLU1\_STASC\_NBR\_VTL1\_20091213.rff\  
CLU1\_STASC\_NBR\_SPL2\_20091213.rff\  
0.1 0.5 0. 4 1024 2 g

#### 4.8.2 **RCL\_vectime\_calibration\_CLUSTA**

Calibrate a L1 STAFF-SC RFF Vectime file . This is a very important procedure.

**The way to do this calibration is explained in details in section 8.3.2**

Full calibration method is detailed in [13, Robert, 2013].

Usage : RCL\_vectime\_calibration\_CLUSTA VTL1.rff VTL2.rff Fdet Fc F1 F2 \ Step N-Kern N\_shift Apod

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)
	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
N_Kern	: Kernel size
N_shift	: for sliding window
Apod	: trapezium (t) or Gaussian (g)

ex: RCL\_vectime\_calibration\_CLUSTA CLU1\_STASC\_NBR\_VTL1\_20091213.rff\ CLU1\_STASC\_NBR\_VTL2\_20091213.rff\ 0. 0.1 0.5 0. 4 1024 2 g

#### 4.8.3 **RCL\_vectime\_to\_mfa**

Tranform a RFF VTL2 file, given in SR2 or ISR2 coordinate system, into MFA system.

FGM SPIN resolution data base is used.

Usage : RCL\_vectime\_to\_mfa toto1\_VT.rff toto2\_VT.rff  
toto1\_VT.rff : name of an input RFF VTL2 file  
toto2\_VT.rff : name of an output RFF VTL2 file

#### 4.8.4 **RCL\_vectime\_L1\_to\_cef**

Tranform a RFF VTL1 into a cef DWF file.

Usage : RCL\_vectime\_L1\_to\_cef toto\_VT.rff  
toto\_VT.rff : name of an input RFF VTL1 file

#### 4.8.5 **RCL\_vectime\_L2\_to\_cef**

Tranform a RFF VTL2 into a cef CWF file .

Usage : RCL\_vectime\_L2\_to\_cef toto\_VTL2.rff  
toto\_VTL2.rff : name of an input RFF VTL2 file

#### 4.8.6 **RCL\_spectro\_L2\_to\_cef**

Transform a SPL2.rff into a CS.cef

Usage : RCL\_spectro\_L2\_to\_cef toto\_VT.rff  
toto\_VT.rff : name of an input RFF VTL1 file



**4.9.5 RCL\_product\_VTL1\_daily**

Product daily VTL1.rff files 1 day 4 sat

Usage : RCL\_product\_VTL1\_daily year month day BitRate

ex: RCL\_product\_VTL1\_daily 2013 01 02 NBR

This command uses RCL\_product\_VTL1\_oneday

**4.9.6 RCL\_product\_VTL2\_oneday**

Product VTL2.rff files 1 day 1 sat

Usage : RCL\_product\_VTL2\_oneday SatNum year month day BitRate Coord

ex: RCL\_product\_VTL2\_oneday 1 2009 12 14 NBR GSE

This command uses RCL\_get\_data\_CLUSTA\_VTL1\_forVTL2  
RCL\_vectime\_calibration\_CLUSTA

**4.9.7 RCL\_product\_VTL2\_onemonth\_nolog**

Product VTL2.rff files 1 month 4 sat

Usage : RCL\_product\_VTL2\_onemonth\_nolog year month BitRate Coord

ex: RCL\_product\_VTL2\_onemonth\_nolog 2009 12 NBR GSE

This command uses RCL\_list\_days\_of\_month  
RCL\_product\_VTL2\_oneday

**4.9.8 RCL\_product\_VTL2\_onemonth**

Product VTL2.rff files 1 month 4 sat + log\_file

Usage : RCL\_product\_VTL2\_onemonth year month BitRate Coord

ex: RCL\_product\_VTL2\_onemonth 2009 12 NBR GSE

This command uses RCL\_product\_VTL2\_onemonth\_nolog

**4.9.9 RCL\_product\_VTL2\_onemonth\_onesat**

Product VTL2.rff files 1 month 4 sat

Usage : RCL\_product\_VTL2\_onemonth\_onesat year month BitRate Coord

ex: RCL\_product\_VTL2\_onemonth\_onesat 2009 12 NBR GSE

This command uses RCL\_nday\_of\_month  
RCL\_product\_VTL2\_oneday

**4.9.10 RCL\_product\_SPL2\_oneday**

Product SPL2.rff files 1 day 1 sat

Usage : RCL\_product\_SPL2\_oneday SatNum year month day BitRate Coord

ex: RCL\_product\_SPL2\_oneday 1 2009 12 14 NBR GSE

This command uses RCL\_get\_data\_CLUSTA\_VTL1\_forSPL2  
RCL\_vectime\_L1\_to\_spectro\_L2

**4.9.11 RCL\_product\_SPL2\_onemonth\_nolog**

Product SPL2.rff files 1 month 4 sat

Usage : RCL\_product\_SPL2\_onemonth\_nolog year month BitRate Coord

ex: RCL\_product\_SPL2\_onemonth\_nolog 2009 12 NBR GSE

This command uses RCL\_product\_SPL2\_oneday  
RCL\_list\_days\_of\_month

**4.9.12 RCL\_product\_SPL2\_onemonth**

Product SPL2.rff files 1 month 4 sat + log\_file

Usage : RCL\_product\_SPL2\_onemonth year month BitRate Coord

ex: RCL\_product\_SPL2\_onemonth 2009 12 NBR GSE

This command uses RCL\_product\_SPL2\_onemonth\_nolog

**4.9.13 RCL\_product\_SPL2\_oneyear**

Product SPL2.rff files 1 year 4 sat

Usage : RCL\_product\_SPL2\_oneyear year BitRate Coord

ex: RCL\_product\_SPL2\_oneyear 2009 NBR GSE

This command uses RCL\_product\_SPL2\_onemonth

**4.9.14 RCL\_product\_SPL2\_oneday\_fordaily**

Product SPL2 files 1 day 1 sat without TCOR

Usage : RCL\_product\_SPL2\_oneday\_fordaily SatNum year month day \ BitRate Coord

ex: RCL\_product\_SPL2\_oneday\_fordaily 1 2009 12 14 NBR GSE

This command uses RCL\_get\_data\_CLUSTA\_VTL1\_forSPL2  
RCL\_vectime\_L1\_to\_spectro\_L2

**4.9.15 RCL\_product\_SPL2\_daily**

Product daily SPL2.rff files 1 day 4 sat 2 modes

Usage : RCL\_product\_SPL2\_daily year month day BitRate Coord

ex: RCL\_product\_SPL2\_daily 2013 01 02 NBR ISR2

This command uses RCL\_product\_SPL2\_oneday\_fordaily

**4.9.16 RCL\_product\_DWF\_oneday**

Product DWF.cef files 1 day 1 sat

Usage : RCL\_product\_DWF\_oneday SatNum year month day BitRate

ex: RCL\_product\_DWF\_oneday 1 2009 12 14 NBR

This command uses RCL\_vectime\_L1\_to\_cef

**4.9.17 RCL\_product\_DWF\_onemonth\_nolog**

Product DWF.cef files 1 month 4 sat

Usage : RCL\_product\_DWF\_onemonth\_nolog year month BitRate

ex: RCL\_product\_DWF\_onemonth\_nolog 2009 12 NBR

This command uses RCL\_product\_DWF\_oneday  
RCL\_list\_days\_of\_month

**4.9.18 RCL\_product\_DWF\_onemonth**

Product DWF.cef files 1 month 4 sat + log\_file

Usage : RCL\_product\_DWF\_onemonth year month BitRate

ex: RCL\_product\_DWF\_onemonth 2009 12 NBR

This command uses RCL\_product\_DWF\_onemonth\_nolog



**4.9.19 RCL\_product\_DWF\_oneyear**

Product DWF.cef files 1 year 4 sat

Usage : RCL\_product\_DWF\_oneyear year BitRate

ex: RCL\_product\_DWF\_oneyear 2009 NBR

This command uses RCL\_product\_DWF\_onemonth

**4.9.20 RCL\_product\_CWF\_oneday**

Product CWF.cef file 1 day 1 sat

Usage : RCL\_product\_CWF\_oneday SatNum year month day Coord

ex: RCL\_product\_CWF\_oneday 1 2009 12 14 GSE

This command uses RCL\_vectime\_L2\_to\_cef

**4.9.21 RCL\_product\_CWF\_onemonth\_nolog**

Product CWF.cef files 1 month 4 sat

Usage : RCL\_product\_CWF\_onemonth\_nolog year month Coord

ex: RCL\_product\_CWF\_onemonth\_nolog 2009 12 GSE

This command uses RCL\_product\_CWF\_oneday

**4.9.22 RCL\_product\_CWF\_onemonth**

Product CWF.cef files 1 month 4 sat + log\_file

Usage : RCL\_product\_CWF\_onemonth year month Coord

ex: RCL\_product\_CWF\_onemonth 2009 12 GSE

This command uses RCL\_product\_CWF\_onemonth\_nolog

**4.9.23 RCL\_product\_CWF\_oneyear**

Product CWF.cef files 1 year 4 sat

Usage : RCL\_product\_CWF\_oneyear year Coord

ex: RCL\_product\_CWF\_oneyear 2009 GSE

This command uses RCL\_product\_CWF\_onemonth

**4.9.24 RCL\_product\_CS\_oneday**

Product CS.cef files 1 day 1 sat

Usage : RCL\_product\_CS\_oneday SatNum year month day BitRate Coord

ex: RCL\_product\_CS\_oneday 1 2009 12 14 NBR GSE

This command uses RCL\_spectro\_L2\_to\_cef

**4.9.25 RCL\_product\_CS\_onemonth\_nolog**

Product CS.cef files 1 month 4 sat

Usage : RCL\_product\_CS\_onemonth\_nolog year month Coord

ex: RCL\_product\_CS\_onemonth\_nolog 2009 12 NBR GSE

This command uses RCL\_product\_CS\_oneday

**4.9.26 RCL\_product\_CS\_onemonth**

Product CS.cef files 1 month 4 sat + log\_file

Usage : RCL\_product\_CS\_onemonth year month BitRate Coord

ex: RCL\_product\_CS\_onemonth 2009 12 NBR GSE

This command uses RCL\_product\_CS\_onemonth\_nolog

**4.9.27 RCL\_product\_CS\_oneyear**

Product CS.cef files 1 year 4 sat

Usage : RCL\_product\_CS\_oneyear year BitRate Coord

ex: RCL\_product\_CS\_oneyear 2009 NBR GSE

This command uses RCL\_product\_CS\_onemonth

**4.9.28 RCL\_product\_visu\_SPL2\_oneday**

Product SPL2.ps files 1 day 4 sat 4Bz

Usage : RCL\_product\_visu\_SPL2\_oneday year month day BitRate

ex : RCL\_product\_visu\_SPL2\_oneday 2009 12 14 NBR ISR2

RCL\_product\_visu\_SPL2\_oneday 2009 12 14 NBR GSE

:This command uses RCL\_visu\_spectro\_4Bz\_3H and RCL\_ps\_to\_png

**4.9.29 RCL\_product\_visu\_SPL2\_onemonth\_nolog**

Product SPL2.ps files 1 month 4 sat 4Bz

Usage : RCL\_product\_visu\_SPL2\_onemonth\_nolog year month BitRate Coord

ex: RCL\_product\_visu\_SPL2\_onemonth\_nolog 2009 12 NBR ISR2

This command uses RCL\_product\_visu\_SPL2\_oneday  
RCL\_list\_days\_of\_month

**4.9.30 RCL\_product\_visu\_SPL2\_onemonth**

Product SPL2.ps files 1 month 4 sat 4Bz + log\_file

Usage : RCL\_product\_visu\_SPL2\_onemonth year month BitRate

ex : RCL\_product\_visu\_SPL2\_onemonth 2009 12 NBR ISR2

This command uses RCL\_product\_visu\_SPL2\_onemonth\_nolog

**4.9.31 RCL\_product\_visu\_SPL2\_daily**

Product SPL2.PS files 1 day 4 Bz

Usage : RCL\_product\_visu\_SPL2\_daily year month day BitRate Coord

ex : RCL\_product\_visu\_SPL2\_daily 2013 01 03 NBR ISR2

This command uses RCL\_product\_visu\_SPL2\_oneday

**4.9.32 RCL\_product\_PNG\_16m**

From a given directory named "PS", find all .ps files and produce corresponding .png file.  
Produced png files have 16 millions of colors and 300 dpi resolution.  
A tree of png files is created, identical to the one of the .ps files.

Usage : RCL\_product\_PNG\_16m PS\_path

ex : RCL\_product\_PNG\_16m /data/CLUSTER/Product\_visu\_SPL2\_4Bz/L2/PS

This command uses RCL\_ps\_to\_png\_16m

**4.9.33 RCL\_product\_PNG\_256**

From a given directory named "PS", find all .ps files and produce corresponding .png file.  
Produced png files have 256 colors and 300 dpi resolution.  
A tree of png files is created, identical to the one of the .ps files.

Usage : RCL\_product\_PNG\_256 PS\_path

ex : RCL\_product\_PNG\_256 /data/CLUSTER/Product\_visu\_SPL2\_4Bz/L2/PS

This command uses RCL\_ps\_to\_png\_256

**4.9.34 RCL\_product\_PNG\_LowRes**

From a given directory named "PS", find all .ps files and produce corresponding .png file.  
Produced png files have 256 colors and 75 dpi resolution.Used to create thumbnail images  
for web browser.

A tree of png files is created, identical to the one of the .ps files.

Usage : RCL\_product\_PNG\_LowRes PS\_path

ex : RCL\_product\_PNG\_LowRes /data/CLUSTER/Product\_visu\_SPL2\_4Bz/L2/PS

This command uses RCL\_ps\_to\_png\_256

**4.10. PRODUCTION DE MASSE DES VISUS D'ORBITE****4.10.1 RCL\_product\_visu\_orbit\_oneday**

Product orbit graphical files 1 day 4 sat

Usage : RCL\_product\_visu\_orbit\_oneday year month day

ex: RCL\_product\_visu\_orbit\_oneday 2009 12 14

This command uses

- RCL\_get\_data\_CLUGEOM
- RCL\_visu\_CLUGEOM
- RCL\_get\_data\_CLUPOS
- RCL\_visu\_CLUPOS
- RCL\_ps\_to\_png

**4.10.2 RCL\_product\_visu\_orbit\_onemonth\_nolog**

Product orbit plots for web 1 month 4 sat

Usage : RCL\_product\_visu\_orbit\_onemonth\_nolog year month

ex: RCL\_product\_visu\_orbit\_onemonth\_nolog 2009 12

This command uses

- RCL\_product\_visu\_orbit\_oneday
- RCL\_list\_days\_of\_month

**4.10.3 RCL\_product\_visu\_orbit\_onemonth**

Product visu\_orbit files 1 month 4 sat

Usage : RCL\_product\_visu\_orbit\_onemonth year month

ex: RCL\_product\_visu\_orbit\_onemonth 2009 12

This command uses RCL\_product\_visu\_orbit\_onemonth\_nolog

**4.10.4 RCL\_product\_visu\_orbit\_oneyear**

Product orbit plots for web

Usage : RCL\_product\_visu\_orbit\_oneyear year

ex: RCL\_product\_visu\_orbit\_oneyear 2009

This command uses RCL\_product\_visu\_orbit\_onemonth

**4.11. UTILITAIRES POUR LE TEMPS****4.11.1 RCL\_current\_date**

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

Usage : RCL\_current\_date

Could return : 2014-11-20 15:32:33 day 324 Julsec 1416493953

**4.11.2 RCL\_nday\_of\_month**

Return number of day in a month

Usage : RCL\_nday\_of\_month year month

ex: RCL\_nday\_of\_month 1997 11

Retourne dans ce cas 30. Valide depuis 1972 jusqu'à 2058.

**4.11.3 RCL\_next\_day**

Return date of next day, as '20070925'

Usage : RCL\_next\_day date

ex: RCL\_next\_day 20070925

Retourne dans ce cas 20070926. Valide depuis 1972 jusqu'à 2058.

This command uses RCL\_nday\_of\_month

**4.11.4 RCL\_previous\_day**

Return date one day before, as '20070925'

Usage: RCL\_previous\_day date

ex: RCL\_previous\_day 20070925

Retourne dans ce cas 20070924. Valide depuis 1972 jusqu'à 2058.

This command uses RCL\_nday\_of\_month

**4.11.5 RCL\_decode\_datiso**

Return elements of an ISO date .

Usage : RCL\_decode\_datiso datiso

ex: RCL\_decode\_datiso 2001-09-23T09:17:03.000Z

will return 2001 09 23 09 17 03

**4.11.6 RCL\_decode\_datim**

Return elements of a date\_time variable.

Usage : RCL\_decode\_datim datim

ex: RCL\_decode\_datim 20010923\_091703

will return 2001 09 23 09 17 03

**4.11.7 RCL\_encode\_datiso**

Return ISO\_date from year month day hour min sec.

Usage : RCL\_encode\_datiso month day hour min sec

ex: RCL\_encode\_datiso 2001 09 23 09 17 03

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

**4.11.8 RCL\_list\_days\_of\_month**

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

Usage : RCL\_list\_days\_of\_month year month

ex: RCL\_list\_days\_of\_month 1997 11

will 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

This command uses RCL\_nday\_of\_month

**4.12. COMMANDES DE VISUALISATION****4.12.1 RCL\_visu\_spectro**

Visualization of a Spectrogram RFF file

Usage : RCL\_visu\_spectro SP.rff datiso1 datiso2 f1 f2 pmin pmax

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

fi1, fi2 : frequency bounds for integrated power

ex1 : RCL\_visu\_spectro toto\_SP.rff 2001-09-23T09:00:00.000Z \ 2001-09-23T11:00:00.000Z 0. 5. 0. 0. 0.2 10.

ex2 : RCL\_visu\_spectro toto\_SP.rff 0 0 0. 0. 0. 0.2 0.

This command uses RCL\_version  
RCL\_decode\_datiso  
RCL\_get\_data\_CLUPOS  
RCL\_get\_data\_CLUFGM

#### 4.12.2 RCL\_visu\_spectro\_3H

Visualization of a Spectrogram RFF file (8 x 3H files)

Usage : RCL\_visu\_spectro\_3H SP.rff f1 f2 puimin puimax f\_int1 f\_int2  
                                   SP.rff                  : name of a spectro RFF file  
                                   f1 f2                  : frequency bounds to plot  
                                   puimin puimax         : min max power for dynamic colors  
                                   f\_int1, f\_int2         : frequency bounds for integrated power

ex1 : RCL\_visu\_spectro\_3H toto\_SP.rff 0. 5. 0. 0. 0.2 10.

ex2 : RCL\_visu\_spectro\_3H toto\_SP.rff 0. 0. 0. 0. 0.2 0.

This command uses RCL\_version  
                           RCL\_decode\_datiso  
                           RCL\_get\_data\_CLUPOS  
                           RCL\_get\_data\_CLUFGM

#### 4.12.3 RCL\_visu\_spectro\_4Bz

Visualization of 4 Spectrogram RFF file Bz only

Usage : RCL\_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 : RCL\_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 : RCL\_visu\_spectro\_4Bz SP1.rff SP2.rff SP3.rff SP4.rff 0 0 0. 0. 0. 0.2 0.

This command uses RCL\_version  
                           RCL\_decode\_datiso  
                           RCL\_get\_data\_CLUPOS  
                           RCL\_get\_data\_CLUFGM

#### 4.12.4 RCL\_visu\_spectro\_4Bz\_3H

Visualization of 4 Spectrogram RFF file Bz only (8 x 3H files)

Usage : RCL\_visu\_spectro\_4Bz\_3H SP1.rff SP2.rff SP3.rff SP4.rff f1 f2 \  
   pmin pmax fi1 fi2  
                           SP1-4.rff                  : names of 4 spectro RFF files  
                           f1 f2                      : frequency bounds to plot  
                           pmin pmax                 : min max power for dynamic colors  
                           fi1, fi2                  : frequency bounds for integrated power

ex1 : RCL\_visu\_spectro\_4Bz\_3H SP1.rff SP2.rff SP3.rff SP4.rff 0. 5. 0. 0. 0.2 10.

ex2 : RCL\_visu\_spectro\_4Bz\_3H SP1.rff SP2.rff SP3.rff SP4.rff 0. 0. 0. 0. 0.2 0.

This command uses RCL\_version  
                           RCL\_decode\_datiso  
                           RCL\_get\_data\_CLUPOS  
                           RCL\_get\_data\_CLUFGM



## 4.13. UTILITAIRES DE CONVERSION DES POSTSCRIPTS

### 4.13.1 *RCL\_ps\_to\_pdf*

Create PDF file with/without image compression

Produce .pdf file from .ps one

Usage : `RCL_ps_to_pdf.sh toto.ps`

`RCL_ps_to_pdf.sh toto.ps compress` (active image compression)

### 4.13.2 *RCL\_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 : `RCL_ps_to_png.sh toto.ps 80 256` (80 ppi or dpi, and 250 colors)

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

### 4.13.3 *RCL\_ps\_to\_png\_256*

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

Produce .png file from .ps one

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

### 4.13.4 *RCL\_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 : `RCL_ps_to_png.sh toto.ps 300` (300 ppi or dpi)

## 4.14. GESTION DES RÉPERTOIRES

### 4.14.1 *RCL\_system\_info*

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

Usage : `RCL_system_info`

exemple de résultat :

```
-----
network node hostname : lx-robert
kernel name           : Linux
kernel release        : 3.5.0-49-generic
kernel version        : #73-precise1-Ubuntu SMP Wed Apr 2 18:35:12 UTC 2014
operating system      : GNU/Linux
machine hardware name  : x86_64
processor type         : x86_64
hardware platform      : x86_64
-----
```

**Table 22:** *Exemple de résultat de la commande RCL\_system\_info*



**4.14.2 RCL\_dir\_size**

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

Usage: RCL\_dir\_size  
 RCL\_dir\_size Mo  
 RCL\_dir\_size DIR  
 RCL\_dir\_size DIR Mo

ex : RCL\_dir\_size

will give :

```
1002521653 RCL_V2p2
```

RCL\_dir\_size Mo

will give :

```
1002 RCL_V2p2
```

**4.14.3 RCL\_dir\_size\_tree**

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

Usage: RCL\_dir\_size\_tree  
 RCL\_dir\_size\_tree Mo  
 RCL\_dir\_size\_tree DIR  
 RCL\_dir\_size\_tree DIR Mo

ex1 : RCL\_dir\_size\_tree

will give :

```
1002522800 .
 1905745 ./src
   25668 ./mod
 14254432 ./obj
 933793776 ./test
 784820891 ./test/test_doc
   495215 ./pro
 46799751 ./bin
 409114 ./bash
 1379857 ./gainantset
 3252477 ./doc
 2715017 ./doc/doc_user
 349501 ./doc/doc_dev
```

**Table 23:** Exemple de résultat de la commande RCL\_dir\_size\_tree

ex2 : RCL\_dir\_size\_tree Mo

will give :

```
1002 .
 1 ./src
 0 ./mod
 14 ./obj
 933 ./test
 784 ./test/test_doc
 0 ./pro
 46 ./bin
 0 ./bash
 1 ./gainantset
 3 ./doc
 2 ./doc/doc_user
 0 ./doc/doc_dev
```

**Table 24:** Exemple de résultat de la commande RCL\_dir\_size\_tree Mo

This command uses RCL\_dir\_size

4.14.4 RCL\_dir\_size\_pretty\_tree

Same as above, but with more pretty appearance. Uses RCL\_dir\_size\_tree  
Usage : RCL\_dir\_size\_pretty\_tree  
will give :

```
-----
Directories tree
-----

Starting point: /data/CACTUS_HOME/RCL/RCL_V2p2
Please wait ...

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

Starting date      : 2014-10-22 17:51:27
host name         : lx-robert
kernel name/release : Linux 3.5.0-49-generic
Starting directory :

Total number of directories : 13
Total number of files      : 495
-----
---
nb. of directories considered: 13
Maximum number of levels    : 2

Moctets
-----
1002.848. |
  1.906   | _src
  0.026   | _mod
 14.254   | _obj
933.794   | _test
784.821   |      | _test_doc
  0.495   | _pro
 46.800   | _bin
  0.407   | _bash
  1.380   | _gainantset
  3.580   | _doc
  3.043   |      | _doc_user
  0.350   |      | _doc_dev
-----
```

Table 25: Exemple de résultat de la commande RCL\_dir\_size\_pretty\_tree

4.14.5 RCL\_dir\_properties

Give directory size (octets or Mo), number of files & directories  
Usage: RCL\_dir\_properties  
RCL\_dir\_properties Mo  
RCL\_dir\_properties DIR  
RCL\_dir\_properties DIR Mo

ex : RCL\_dir\_properties  
will give :

1002524337	497	12	RCL_V2p2
------------	-----	----	----------

RCL\_dir\_properties Mo  
will give :

1002	497	12	RCL_V2p2
------	-----	----	----------

#### 4.14.6 *RCL\_dir\_properties\_tree*

Give directory properties for all the tree

Usage: `RCL_dir_properties_tree`  
`RCL_dir_properties_tree Mo`

ex : `RCL_dir_properties_tree`

will give :

```
-----
system/host: Linux lx-robot
directory: /data/CACTUS_HOME/RCL/RCL_V2p2/.
size in octets, Nb files, Nb Dir.
-----

1002524630      497      12  .
    1905745      74      0  ./src
      25668       3      0  ./mod
    14254432     32      0  ./obj
    933793776    54      1  ./test
    784820891    50      0  ./test/test_doc
      495215     28      0  ./pro
    46799751     31      0  ./bin
      409114    141      0  ./bash
    1379857      34      0  ./gainantset
    3252834      74      2  ./doc
    2715374      22      0  ./doc/doc_user
      349501     45      0  ./doc/doc_dev
```

ex : `RCL_dir_properties_tree Mo`

will give :

```
-----
system/host: Linux lx-robot
directory: /data/CACTUS_HOME/RCL/RCL_V2p2/.
size in Mo, Nb files, Nb Dir.
-----

1002      497      12  .
   1       74      0  ./src
  0         3      0  ./mod
   14       32      0  ./obj
   933      54      1  ./test
   784      50      0  ./test/test_doc
   0        28      0  ./pro
   46       31      0  ./bin
   0       141      0  ./bash
   1        34      0  ./gainantset
   3        74      2  ./doc
   2        22      0  ./doc/doc_user
   0        45      0  ./doc/doc_dev
```

**Table 26:** *Exemples de résultats de la commande `RCL_dir_properties_tree`*

This command uses `RCL_dir_properties`

#### 4.14.7 RCL\_dir\_properties\_pretty\_tree

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

This command uses RCL\_dir\_properties\_tree

RCL\_dir\_properties\_pretty\_tree

will give :

```

-----
Directories tree
-----

Starting point: /data/CACTUS_HOME/RCL/RCL_V2p2
Please wait ...

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

Starting date      : 2014-10-21 18:12:14
host name         : lx-robert
kernel name/release : Linux 3.5.0-49-generic
Starting directory : /data/CACTUS_HOME/RCL/RCL_V2p2

Total number of directories :    13
Total number of files      :   498
-----

nb. of directories considered:    13
Maximum number of levels      :    2

Moctets, nb files, nb dir
-----

1002.526  498  12. _
  1.906   74   0  _src
  0.026    3   0  _mod
 14.254   32   0  _obj
 933.794   54   1  _test
 784.821   50   0  _test_doc
  0.495   28   0  _pro
 46.800   31   0  _bin
  0.409  141   0  _bash
  1.380   34   0  _gainantset
  3.253   74   2  _doc
  2.716   22   0  _doc_user
  0.350   45   0  _doc_dev
-----

Used disk space (ko)
-----

Sys. de fichiers  1K-blocs  Utilisé Disponible Uti% Monté sur
/dev/sda2          96121612  14985404   76253420   17% /
udev              16428612     4   16428608    1% /dev
tmpfs             3287592    1304   3286288    1% /run
none              5120      0     5120     0% /run/lock
none             16437956    432   16437524    1% /run/shm
/dev/sda1         193784     128    193656    1% /boot/efi

/dev/sda3         119250172  44927916   68264632   40% /home
/dev/sdb1        5813205592  3726274024 1793963020   68% /data

-----
copy of results are in dir_properties_tree.txt
-----

```

**Table 27:** Exemple de résultat de la commande RCL\_dir\_properties\_pretty\_tree

#### 4.14.8 ***RCL\_check\_dirname\_tree***

Search in a tree directories having a character blank in the name, and propose to replace blank by “\_”

Usage : `RCL_check_dirname_tree`

#### 4.14.9 ***RCL\_search\_duplicates***

Search and list duplicate files in a tree .

Usage : `RCL_search_duplicates mode (I or B)`

I : interactive run (stop at each duplicate found)

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

#### 4.15. LISTE ALPHABÉTIQUE DE TOUTES LES COMMANDES

La liste alphabétique des commandes RFF disponibles est donnée dans la Table 28 ci-dessous pour la version V2.2. Elle peut être obtenue par la commande **RCL\_list**. Cette commande génère un fichier RCL\_list.txt. Les commandes en rouge ont été ajoutées depuis la version précédente.

RCL_check_dirname_tree RCL_check_rff RCL_check_rff_dir RCL_clean_rff RCL_compare_versions RCL_compare_versions_long RCL_copy_rff RCL_current_date  RCL_decode_datim RCL_decode_datiso RCL_diff_rff RCL_dir_properties RCL_dir_properties_pretty_tree RCL_dir_properties_tree RCL_dir_size RCL_dir_size_pretty_tree RCL_dir_size_tree RCL_download_data_CLUFGM_oneday RCL_download_data_CLUFGM_oneday_t1t2 RCL_download_data_CLUFGM_onemonth RCL_download_data_CLUFGM_onemonth_nolog RCL_download_data_CLUFGM_oneyear RCL_encode_datiso  RCL_fgm_cef_gse_to_sr2 RCL_fgm_cef_to_bav RCL_fgm_cef_to_rff  RCL_get_data_CLUFGM RCL_get_data_CLUGOM RCL_get_data_CLUPOS RCL_get_data_CLUSTA_VTL1 RCL_get_data_CLUSTA_VTL2 RCL_get_data_CLUSTA_VTL1_forSPL2 RCL_get_data_CLUSTA_VTL1_forVTL2 RCL_get_data_CLUSTA_WFL1_forVTL1 RCL_give_CLUORB_period RCL_give_spin_dir  RCL_info_rff RCL_info_rff_dir RCL_list RCL_list_block_WF RCL_list_days_of_month  RCL_make_minidoc RCL_menu RCL_move_rff RCL_nday_of_month RCL_next_day RCL_previous_day  RCL_product_CS_oneday RCL_product_CS_onemonth RCL_product_CS_onemonth_nolog RCL_product_CS_oneyear  RCL_product_CWF_oneday RCL_product_CWF_onemonth RCL_product_CWF_onemonth_nolog RCL_product_CWF_oneyear  RCL_product_DWF_oneday RCL_product_DWF_onemonth RCL_product_DWF_onemonth_nolog RCL_product_DWF_oneyear	RCL_product_PNG_16m RCL_product_PNG_256 RCL_product_PNG_LowRes  RCL_product_SPL2_daily RCL_product_SPL2_oneday RCL_product_SPL2_oneday_fordaily RCL_product_SPL2_onemonth RCL_product_SPL2_onemonth_nolog RCL_product_SPL2_oneyear  RCL_product_visu_orbit_oneday RCL_product_visu_orbit_onemonth RCL_product_visu_orbit_onemonth_nolog RCL_product_visu_orbit_oneyear  RCL_product_visu_SPL2_daily RCL_product_visu_SPL2_oneday RCL_product_visu_SPL2_onemonth RCL_product_visu_SPL2_onemonth_nolog  RCL_product_VTL1_daily RCL_product_VTL1_oneday RCL_product_VTL1_onemonth RCL_product_VTL1_onemonth_nolog RCL_product_VTL1_oneyear  RCL_product_VTL2_oneday RCL_product_VTL2_onemonth RCL_product_VTL2_onemonth_nolog RCL_product_VTL2_onemonth_onesat  RCL_ps_to_pdf RCL_ps_to_png RCL_ps_to_png_16m RCL_ps_to_png_256  RCL_reduce_time_rff RCL_search_duplicates RCL_spectro_L2_to_cef RCL_spectro_to_polar RCL_spectro_xyz_to_lrz RCL_system_info  RCL_vectime_calibration_CLUSTA RCL_vectime_L1_to_cef RCL_vectime_L1_to_spectro_L2 RCL_vectime_L2_to_cef RCL_vectime_to_mfa RCL_vectime_to_spectro RCL_version  RCL_visu_ave_spectrum RCL_visu_CLUGOM RCL_visu_CLUPOS RCL_visu_polar RCL_visu_spectro RCL_visu_spectro_3H RCL_visu_spectro_4Bz RCL_visu_spectro_4Bz_3H RCL_visu_vectime RCL_visu_vectime_widget RCL_waveform_to_vectime  Nb of RCL commands : 113
--	---

Table 28: Exemple de résultat de la commande RCL\_list

## 5. LES SCRIPTS RCL

### 5.1. PRINCIPE

Les commandes RCL sont considérées dans l'environnement Linux comme des commandes de base du shell. Cette possibilité est l'apanage des commandes RCL et en fait toute l'utilité, car on peut fabriquer des scripts permettant l'enchaînement de commandes RCL successives, agrémentées par des commandes shell, et permettant donc d'automatiser des traitements fastidieux. Cette possibilité est largement exploitée par les commandes dites « de production ».

#### 5.1.1 Exemple

Ainsi, lorsque l'on veut par exemple calculer et tracer un spectrogramme des données VTL2 de STAFF-SC, il faudrait théoriquement enchaîner à la main les commandes suivantes :

- Indiquer où sont les données :

```
RCS_DATA=/NFS/tamaya/data2/staff-op/STAFF-SC
```

(En principe c'est déjà fait dans le RCL\_config.bash à l'installation du logiciel, voir chapitre 6).

- Copier le fichier journalier de la base locale dans le répertoire courant :

```
RCL_get_data_CLUSTA_VTL2 4 2001 09 23 NBR ISR2
```

- Réduire l'intervalle de temps :

```
file1=CLU4_STASC_VTL2_ISR2_NBR_20010923.rff
```

```
file2=CLU4_STASC_VTL2_ISR2_NBR_20010923_red.rff
```

```
RCL_reduce_time_rff $file1 $file2 2001-09-23T09:20:00.000Z 2001-09-23T10:10:00.000Z
```

- Calculer le spectrogramme :

```
file3=CLU4_STASC_SPL2_ISR2_NBR_20010923_red.rff
```

```
RCL_vectime_to_spectro $file2 $file3 512 512 t
```

- Produire le ps ou le pdf :

```
RCL_visu_spectro $file3 0 0 0. 0. 0. 0. 0.2 0.
```

Enchaîner ces commandes à la main serait particulièrement fastidieux. On pourrait bien sûr copier ces lignes dans un fichier qu'on appellerait par exemple `make_staff_spectro_from_VTL2.sh` et exécuter ce fichier. Mais il faudrait changer à la main les dates à chaque utilisation. C'est pourquoi on a développé le script détaillé dans le chapitre suivant, qui a l'avantage de prendre en arguments les paramètres requis. Ainsi on va voir que tout le travail ci-dessus est alors effectué par la seule commande :

```
PR_make_staff_spectro_from_VTL2 4 NBR ISR2 20010923_092000 20010923_101000 512 t
```

### 5.1.2 Un peu de shell

Le script de la commande mentionné au paragraphe précédent est donné ci-dessous dans la Table 29. Pour ceux qui connaissent un peu Unix et le shell, il est aisé à comprendre. Pour les autres, des explications sont données ci-après. Dans tous les cas il peut être facilement cloné pour fabriquer d'autres commandes du même type, comme celles décrites dans les paragraphes suivants.

```
#!/bin/sh

# Script: PR_make_staff_spectro_from_VTL2
# Usage : PR_make_staff_spectro_from_VTL2 4 SatNum BitRate Coord datim1 datim2 Nkern Apod
# Exemple: PR_make_staff_spectro_from_VTL2 4 NBR ISR2 20010923_092000 20010923_101000 512 t

SatNum=$1
BitRate=$2
Coord=$3
datim1=$4
datim2=$5
Nkern=$6
Apod=$7

# if need, change default path for STAFF data base
#RCS_DATA=/NFS/tamaya/data2/staff-op/STAFF-SC

echo
echo "-----"
echo "Asked parameters:"
echo
echo "SatNum, BitRate, Coord= $SatNum $BitRate $Coord"

# extract year month day
set `RCL_decode_datim $datim1`
year=$1
month=$2
day=$3
date_file=$1$2$3

# compute ISO dates
date_iso1=`RCL_encode_datiso $1 $2 $3 $4 $5 $6 `
set `RCL_decode_datim $datim2`
date_iso2=`RCL_encode_datiso $1 $2 $3 $4 $5 $6 `

echo
echo "datim1= $datim1"
echo "datim2= $datim2"
echo
echo "date_iso1= $date_iso1"
echo "date_iso2= $date_iso2"
echo "Nkern, Apod= $Nkern $Apod"
echo "-----"
echo

# define files names

file1=CLU$SatNum" "STASC_VTL2_$Coord"_"$BitRate" "$date_file.rff
file2=CLU$SatNum" "STASC" "VTL2" "$Coord"_"$BitRate"_"$date_file"_"red.rff
file3=CLU$SatNum" "STASC" "SPL2"_"$Coord"_"$BitRate"_"$date_file"_"red.rff

# RCL launch

RCL_get_data_CLUSTA_VTL2 $SatNum $year $month $day $BitRate $Coord

RCL_reduce_time_rff $file1 $file2 $date_iso1 $date_iso2

RCL_vectime_to_spectro $file2 $file3 $Nkern $Nkern $Apod

RCL_visu_spectro $file3 0 0 0. 0. 0. 0. 0.2 0.

# END

echo "-----END of PR_make_staff_spectro_from_VTL2 -----"
```

Table 29: Contenu du script «PR\_make\_staff\_spectro\_from\_VTL2 »



### 5.1.3 Explications

Chaque ligne commençant par un # est un commentaire.

\$1 à \$8 représentent les arguments du shell-script. A partir de ces arguments, on définit des variables plus explicites (SatNum à Apod). A partir de ces variables, on effectue un certain nombre d'opérations pour extraire des champs (year, month, day, hour, min, sec) au moyen de la commande « set » et des commandes **RCL\_decode\_datim** et **RCL\_encode\_datiso**. On écrit sur l'écran les valeurs calculées au moyen de la commande « echo ».

Ensuite on définit les chemins et les noms de fichier utilisés, et enfin on lance les commandes RCL nécessaire à l'extraction de la période de temps désirée, le calcul et la visualisation du spectre.

On peut alors lancer la commande :

```
PR_make_staff_spectro_from_VTL2 4 NBR ISR2 20010923_092000 20010923_101000 512 t
```

Qui fera tout le travail, et produira le fichier CLU4\_STASC\_SPL2\_ISR2\_NBR\_20010923\_red.ps ainsi que le pdf, et éventuellement le png, suivant les choix de visualisation définis dans le fichier de configuration RCL\_config.bash (voir chapitre 6.1.2). Si la variable R\_VISU\_SPLASH a été mise à « yes », le fichier pdf sera automatiquement ouvert à l'écran. Le fichier pdf produit par cette commande est donné en annexe 9.10.

Notons qu'on a ainsi créé à la demande *une nouvelle commande RCL*, que l'on peut nommer par exemple **PR** (comme **P**ersonnal **RCL**), et que l'on peut mettre dans le directory « script » du pack RCL afin qu'elles soient connues à partir de n'importe quel répertoire sur la machine utilisée. Ne pas oublier de donner une permission d'exécution à toutes les commandes PR\* par la commande unix : **chmod +x PR\***

Il est préférable de nommer les scripts que l'on met dans le directory « script » différemment des commandes RCL\* natives, afin de les différencier. En effet, une commande RCL a pour principe de faire des tests sur les arguments, et de fournir un mini-help en cas de mauvais arguments, ce que ne fait pas le script proposé ci-dessus. Par la suite, quand ces commandes ont été bien testées et éprouvées, on pourra en faire de nouvelles commandes RCL, en s'inspirant des tests effectués dans les commandes natives, et ainsi mettre à disposition des utilisateurs une version N+1 du package. La possibilité de construire et d'apporter de nouvelles commandes RCL fait que l'ensemble du produit RCL peut traiter de grandes variétés de données et fournir des outils nouveaux dans un même cadre de produit.

## 5.2. CALCUL ET PLOT D'UN SPECTROGRAMME

### 5.2.1 Spectrogramme à partir des VTL2

C'est l'exemple donné ci-dessus.

Il faut noter que cette commande peut être réutilisée pour d'autres données que STAFF, comme FGM par exemple (voir chapitre 5.6 et la Figure 11) et peut également être utilisée pour d'autres missions autres que CLUSTER, du moment que les données de forme d'onde soit au format Vectime RFF comme les VTL2.

Le résultat de la commande :

```
PR_make_staff_spectro_from_VTL2 4 NBR ISR2 20010923_092000 20010923_101000 512 t
```

est donné en annexe 9.10.1.

### 5.2.2 Spectrogramme à partir des VTL1

Comme on l'a vu au chapitre 2.2.4, il y a deux manières de calculer des spectrogrammes : la première, ci-dessus, par simple FFT de la forme d'onde calibrée, et la seconde, exposée ici, ou on fait le spectre des VTL1 et on calibre le spectre.

Comme dans l'exemple précédent, lorsque l'on veut calculer et tracer un spectrogramme calibré à partir des données VTL1 de STAFF-SC, il faudrait théoriquement enchaîner à la main les commandes suivantes :

- Indiquer où sont les données :

```
RCS_DATA=/NFS/tamaya/data2/staff-op/STAFF-SC
```

(En principe c'est déjà fait dans le RCL\_config.bash à l'installation du logiciel, voir chapitre 6).

- Copier le fichier journalier de la base locale des VTL1 dans le répertoire courant :

```
RCL_get_data_CLUSTA_VTL1 4 2001 09 23 NBR ISR2
```

- Réduire l'intervalle de temps :

```
file1=CLU4_STASC_VTL1_NBR_20010923.rff
```

```
file2=CLU4_STASC_VTL1_NBR_20010923_red.rff
```

```
RCL_reduce_time_rff $file1 $file2 2001-09-23T09:20:00.000Z 2001-09-23T10:10:00.000Z
```

- Calculer et calibrer le spectrogramme :

```
file3=CLU4_STASC_SPL2_ISR2_NBR_20010923_red.rff
```

```
RCL_vectime_L1_to_spectro_L2 $file2 $file3 0.06 0.0 0.7 512 512 t
```

- Produire le ps ou le pdf :

```
RCL_visu_spectro $file3 0 0 0.0 0.0 0.0 0.2 0.
```

Pour éviter de taper à chaque fois l'ensemble de ces commandes, on a développé le script détaillé ci-dessous, qui a l'avantage de prendre en argument les paramètres requis. Tout le travail ci-dessus est alors effectué par la seule commande :

```
PR_make_staff_spectro_from_VTL1 4 NBR ISR2 20010923_092000 20010923_101000 0.06 512 t
```

Le contenu de cette procédure est donné dans la Table 30 ci-dessous, et comme précédemment peut servir de base pour développer d'autres scripts du même type.

Le fichier pdf produit par cette commande est donné en annexe 9.10.2.

```
#!/bin/sh

# Script: PR_make_staff_spectro_from_VTL1
# Usage : PR_make_staff_spectro_from_VTL1 4 SatNum BitRate Coord datim1 datim2 Fdet Fc Nkern Apod
# Example: PR_make_staff_spectro_from_VTL1 4 NBR ISR2 20010923_092000 20010923_101000 0. 0.1 512 t

SatNum=$1
BitRate=$2
Coord=$3
datim1=$4
datim2=$5
Fdet=$6
Fc=$7
Nkern=$8
Apod=$9

# if need, change default path for STAFF data base
#RCS_DATA=/NFS/tamaya/data2/staff-op/STAFF-SC

echo
echo "-----"
echo "Asked parameters:"
echo
echo "SatNum, BitRate, Coord= $SatNum $BitRate $Coord"

# extract year month day
set `RCL_decode_datim $datim1`
year=$1
month=$2
day=$3
date_file=$1$2$3

# compute ISO dates
date_iso1=`RCL_encode_datiso $1 $2 $3 $4 $5 $6 `

set `RCL_decode_datim $datim2`
date_iso2=`RCL_encode_datiso $1 $2 $3 $4 $5 $6 `

# compute calibration Step according desired Coordinates
Step=0
if test $Coord = ISR2 ; then Step=7 ; fi
if test $Coord = GSE ; then Step=8 ; fi

if test $Step = 0
then echo "**** Bad Coord, shell aborted !"
exit 1
fi

echo
echo "datim1= $datim1"
echo "datim2= $datim2"
echo
echo "date_iso1= $date_iso1"
echo "date_iso2= $date_iso2"
echo "Fdet, Fc= $Fdet $Fc"
echo "Nkern, Apod= $Nkern $Apod"
echo "-----"
echo

# define files names

file1=CLU$SatNum_"STASC_VTL1"_"$BitRate"_"$date_file.rff
file2=CLU$SatNum_"STASC_VTL1"_"$BitRate"_"$date_file" "red.rff
file3=CLU$SatNum_"STASC_" "SPL2"_"$Coord"_"$BitRate"_"$date_file"_"red.rff

# RCL launch

RCL_get_data_CLUSTER_VTL1 $SatNum $year $month $day $BitRate

RCL_reduce_time_rff $file1 $file2 $date_iso1 $date_iso2

RCL_vectime_L1_to_spectro_L2 $file2 $file3 $Fdet $Fc 0. 0. $Step $Nkern $Nkern $Apod

RCL_visu_spectro $file3 0 0 0. 0. 0. 0. 0.2 0.

# END

echo "-----END of PR_make_staff_spectro_from_VTL1 -----"
```

**Table 30:**      *Contenu du script «PR\_make\_staff\_spectro\_from\_VTL1 »*

### 5.3. CALCUL ET PLOT D'UN SPECTRE MOYEN

Pour cette opération, il suffit de calculer le spectrogramme par l'un des deux scripts ci-dessus, et d'enchaîner la commande suivante :

```
RCL_visu_ave_spectrum CLU4_STASC_SPL2_ISR2_NBR_20010923_red.rff 0 0 0.5 0.0 XY y n
```

On peut aussi ajouter cette commande à la fin de l'un de ces deux scripts, en remplaçant simplement le nom « CLU4\_STASC\_SPL2\_ISR2\_NBR\_20010923\_red.rff » par la variable \$file3.

La signification précise de tous les arguments est donnée au chapitre 4.

Le résultat de cette commande est affiché dans l'annexe 9.10.4. Notons que deux plots sont produits, l'un en échelle linéaire pour la fréquence, l'autre en échelle logarithmique.

### 5.4. PLOT D'UNE FORME D'ONDE BRUTE OU CALBRÉE

#### 5.4.1 *La commande RCL\_visu\_vectime*

La procédure de visualisation **RCL\_visu\_vectime** visualise n'importe quel fichier de type vectime, que ce soit des données calbrées ou non, issue de CLUSTER ou non, pourvu que ce soit un fichier de type Vectime avec des vecteurs à 3 composantes. Si davantage de composantes sont présentes, seules les 3 premières seront prises en compte et seront supposées être les composantes cartésiennes X, Y, Z à tracer.

Les codes couleurs des satellites de CLUSTER seront appliqués (noir-rouge-vert-bleu pour 1-2-3-4), de même que pour THEMIS (magenta, rouge, vert, cyan, bleu pour 1-2-3-4-5). Pour les autres projets, les courbes seront en noir.

#### 5.4.2 *Application batch à un fichier VTL1 de STAFF*

Si le fichier vectime existe, par exemple CLU4\_STASC\_VTL1\_NBR\_20010923\_red.rff, il suffit de lancer la commande :

```
RCL_visu_vectime CLU4_STASC_VTL1_NBR_20010923_red.rff
```

Si on part de zéro, il faut comme dans l'exemple du chapitre 5.1.1 enchaîner les commandes :

```
RCL_get_data_CLUSTA_VTL1 4 2001 09 23 NBR
file1=CLU4_STASC_VTL1_NBR_20010923.rff
file2=CLU4_STASC_VTL1_NBR_20010923_red.rff
RCL_reduce_time_rff $file1 $file2 2001-09-23T09:28:00.000Z 2001-09-23T09:32:00.000Z
RCL_visu_vectime $file2
```

Ou utiliser le script suivant

```
PR_visu_waveform_VTL1 4 NBR 20010923_092800 20010923_093200
```

Le contenu de ce script est donné dans la Table 31 ci-dessous ; on peut noter les analogies au niveau de l'acquisition des données avec les scripts précédents.

La signification précise de tous les arguments est donnée au chapitre 4.12.6. Le résultat de cette commande est affiché dans l'annexe 9.10.4. Notons que l'on trace deux panels : le panel supérieur, en coordonnées cartésiennes, et le panel inférieur, en coordonnées polaires.

```
#!/bin/sh

# Script: PR_visu_staff_VTL1
# Usage : PR_visu_staff_VTL1 4 SatNum BitRate datim1 datim2
# Exemple: PR_visu_staff_VTL1 4 NBR 20010923_092950 20010923_093300

SatNum=$1
BitRate=$2
datim1=$3
datim2=$4

# if need, change default path for STAFF data base
#RCS_DATA=/NFS/tamaya/data2/staff-op/STAFF-SC

echo
echo "-----"
echo "Asked parameters:"
echo
echo "SatNum, BitRate, Coord= $SatNum $BitRate $Coord"

# extract year month day
set `RCL_decode_datim $datim1`
year=$1
month=$2
day=$3
date_file=$1$2$3

# compute ISO dates
date_iso1=`RCL_encode_datiso $1 $2 $3 $4 $5 $6 `

set `RCL_decode_datim $datim2`
date_iso2=`RCL_encode_datiso $1 $2 $3 $4 $5 $6 `

echo
echo "datim1= $datim1"
echo "datim2= $datim2"
echo
echo "date_iso1= $date_iso1"
echo "date_iso2= $date_iso2"
echo "-----"
echo

# define files names
file1=CLU$SatNum" "STASC_VTL1" "$BitRate" "$date_file.rff
file2=CLU$SatNum"_"STASC_VTL1"_"$BitRate"_"$date_file"_"red.rff

# RCL launch

RCL_get_data_CLUSTA_VTL1 $SatNum $year $month $day $BitRate

RCL_reduce_time_rff $file1 $file2 $date_iso1 $date_iso2

RCL_visu_vectime $file2

# END

echo "-----END of PR_visu_staff_VTL1 -----"
```

**Table 31:**      *Contenu du script «PR\_visu\_staff\_VTL1 »*

### 5.4.3 Application batch à un fichier VTL2 de STAFF en ISR2

Si le fichier vectime existe, par exemple CLU4\_STASC\_VTL1\_ISR2\_NBR\_20010923\_red.rff, il suffit de lancer la commande :

**RCL\_visu\_vectime** CLU4\_STASC\_VTL1\_ISR2\_NBR\_20010923\_red.rff

Si on part de zéro, il faut comme dans l'exemple du chapitre 5.1.1 enchaîner les commandes :

```

RCL_get_data_CLUSTA_VTL2 4 2001 09 23 NBR ISR2
file1=CLU4_STASC_VTL1_ISR2_NBR_20010923.rff
file2=CLU4_STASC_VTL1_ISR2_NBR_20010923_red.rff
RCL_reduce_time_rff $file1 $file2 2001-09-23T09:28:00.000Z 2001-09-23T09:32:00.000Z
RCL_visu_vectime $file2

```

Ou utiliser le script suivant

```
PR_visu_waveform_VTL2 4 NBR ISR2 20010923_092800 20010923_093200
```

Le contenu de ce script est donné dans la Table 32 ci-dessous et le résultat de cette commande est affiché dans l'annexe 9.10.6.

```

#!/bin/sh

# Script: PR_visu_staff_VTL2
# Usage : PR_visu_staff_VTL2 4 SatNum BitRate Coord datim1 datim2
# Exemple: PR_visu_staff_VTL2 4 NBR ISR2 20010923_092800 20010923_101000

SatNum=$1
BitRate=$2
Coord=$3
datim1=$4
datim2=$5

# if need, change default path for STAFF data base
#RCS_DATA=/NFS/tamaya/data2/staff-op/STAFF-SC

echo
echo "-----"
echo "Asked parameters:"
echo
echo "SatNum, BitRate, Coord= $SatNum $BitRate $Coord"

# extract year month day
set `RCL_decode_datim $datim1`
year=$1
month=$2
day=$3
date_file=$1$2$3

# compute ISO dates
date_iso1=`RCL_encode_datiso $1 $2 $3 $4 $5 $6 `

set `RCL_decode_datim $datim2`
date_iso2=`RCL_encode_datiso $1 $2 $3 $4 $5 $6 `

echo
echo "datim1= $datim1"
echo "datim2= $datim2"
echo
echo "date_iso1= $date_iso1"
echo "date_iso2= $date_iso2"
echo "-----"
echo

# define files names
file1=CLU$SatNum_STASC_VTL2_"$Coord"_"$BitRate"_"$date_file".rff
file2=CLU$SatNum_STASC_VTL2_"$Coord"_"$BitRate"_"$date_file"_red".rff

# RCL launch

RCL_get_data_CLUSTA_VTL2 $SatNum $year $month $day $BitRate $Coord
RCL_reduce_time_rff $file1 $file2 $date_iso1 $date_iso2

RCL_visu_vectime $file2

# END
echo "-----END of PR_visu_staff_VTL2 -----"

```

**Table 32:**      *Contenu du script «PR\_visu\_staff\_VTL2 »*

#### 5.4.4 Application batch à un fichier VTL2 de STAFF en GSE

Si le fichier vectime existe, par exemple CLU4\_STASC\_VTL1\_ISR2\_NBR\_20010923\_red.rff, il suffit comme précédemment de lancer la commande :

```
RCL_visu_vectime CLU4_STASC_VTL1_GSE_NBR_20010923_red.rff
```

Si on part de zéro, il suffit simplement de lancer le script précédent, mais de choisir GSE pour le système de coordonnées, soit :

```
PR_visu_waveform_VTL2 4 NBR GSE 20010923_092800 20010923_093200
```

Le résultat est donné dans l'annexe 9.10.7.

#### 5.4.5 Remarques sur les formes d'onde obtenues

##### • Les données brutes VTL1

Les composantes X et Y dans le plan de spin sont largement dominées par la large composante sinusoïdale à la fréquence de spin générée par la rotation du satellite dans le champ continue terrestre, bien plus important que le champ des ondes à mesurer. C'est une des difficultés prise en compte par le programme de calibration [voir 13, Robert, 2013]. Par contre, sur la composante Z, qui n'est pas affectée par la rotation (ou faiblement, l'antenne Z étant proche de l'axe de spin, à moins de 1°) on peut voir le commencement des ondes bien visible sur le spectrogramme.

##### • Les données VTL2 calibrées dans le repère ISR2

Sur ce graphique, on peut voir maintenant que l'amplitude des ondes est sensiblement la même que sur les 3 composantes. Néanmoins, ces données contiennent des basses fréquences d'amplitude très importantes, principalement sur les composantes X et Y pour lesquelles la sensibilité est meilleure que sur Z.

##### • Les données VTL2 calibrées dans le repère GSE

Pour ces données, les basses fréquences ont été filtrées en dessous de 0.6 HZ, et donc tous les problèmes liés au spin ont été « gommés ». Les amplitudes sont maintenant les mêmes sur les 3 composantes. Ces exemples illustrent l'importance des paramètres de calibration si on veut faire de la bonne science avec les données.

## 5.5. CALCUL ET PLOT DES PARAMÈTRES DE POLARISATION

### 5.5.1 Historique

Les commandes décrites ci-dessous sont issues des Roproc. Historiquement, la procédure `rpc_copolar` des Roproc prenait en entrée un fichier de type WFL2 et contenant des données STAFF-SC calibrées dans le système SR2 (voir chapitre 8). Elles sont passées dans le repère MFA (lié au champ magnétique) par la commande `rpc_wave_to_mfa_rcs`.

La commande `rpc_copolar` calcule alors tous les paramètres de polarisation des ondes et les range dans un fichier `copolar.resu`, qui peut alors être visualisé par la commande `rpc_visupolar`.

Ces commandes fonctionnent bien sûr les données STAFF-SC, et pourraient être appliquées sur des données en provenance d'autres expériences. Néanmoins la procédure `rpc_copolar` fait intrinsèquement une transformée de fourrier pour obtenir le spectre complexe des ondes, et c'est à partir de ce spectre que l'on calcule ensuite les paramètres de polarisation par une méthode particulière [voir 18, Robert, 2008].

Si l'utilisateur veut travailler à partir d'un spectre obtenu par une autre méthode qu'une FFT classique, ceci n'était pas possible actuellement.

Les RCL ayant vocation de travailler sur de nombreux type de données, il est apparu utile d'isoler la partie « calcul des paramètres de polarisation » de la partie « calcul du spectre », et de créer la procédure `RCL_spectro_to_polar`, qui prend en entrée un fichier de type « spectrogramme », calibré bien sûr, donc du type `SPL2.rff`.

### 5.5.2 Création du fichier `SPL2`

Ce fichier peut tout aussi bien venir de la procédure `RCL_waveform_to_spectro` (FFT classique) que d'une procédure « client » particulière, du moment que le fichier d'entrée respecte le format RRF ayant pour type « spectrogramme ».

### 5.5.3 Calcul et visualisation des paramètres de polarisation

#### • Calcul pour un cas

Le fichier `SPL2.rff` étant disponible, il suffit de lancer la commande décrite au § 4.3.4 soit par exemple :

```
RCL_spectro_to_polar SPL2.rff
```

Il est à noter que le fichier créé `copolar.resu` n'est pas un fichier `rff`. Il est au même format que celui créé par la Roproc `rpc_copolar`. Il pourrait être mis au format RFF dans une version future des RCL, mais ce n'est pas le cas aujourd'hui.

Pour avoir un PS ou un PDF des résultats, il suffit de lancer la commande `RCL_visupolar` décrite au § 4.12.8, soit par exemple :

```
RCL_visupolar copolar.resu -5.5 50. 225.
```

Un exemple d'application est donné au chapitre 9.10.8 et montre notamment la polarisation circulaire droite, perpendiculaire à  $B_0$ , de type « Whistler »



### • Script d'automatisation à partir des VTL1

Le script de la Table 33 ci-dessous permet de faire en une seule commande la calibration des formes d'onde, leur passage dans le repère MFA, le calcul et le tracé des spectres calibrés dans le repère SR2 et dans le repère MFA, visualisés en composantes circulaires gauche et droite, ainsi que le graphique résumant tous les paramètres de polarisation. Il a été utilisé pour produire les exemples du chapitre 9.10.8 .

```
#!/bin/sh

# -----
# Script: PR_compute_staff_polarisation_from_VTL1
# Usage : PR_compute_staff_polarisation_from_VTL1 arguments
# arguments: sat yyyymmdd h1m1s1 h2m2s2 nbp fdet fc f1 f2 level mode P1 P2
#           2 20040130_164700 20040130_164800 128 20. 5.10 50.0 0. 4 HBR -9.6 -3.0
# -----

if test $# != 12
then
    echo "this procedure requires 12 arguments. Example:"
    echo "PR_compute_staff_polarisation_from_VTL1 2 20040130_164700 \
        20040130_164800 \
        128 20. 5.10 50.0 0. 4 HBR -9.6 -3.0"
    exit 1
fi

SatNum=$1
datim1=$2
datim2=$3
nbp=$4
fdet=$5
fc=$6
f1=$7
f2=$8

shift 8

level=$1
BitRate=$2
p1=$3
p2=$4

if test $BitRate = 'NBR'
then
    Nkern=1024
else
    Nkern=4096
fi

# if need, change default path for STAFF data base
# RCS_DATA=/NFS/tamaya/data2/staff-op/STAFF-SC
# RCS_DATA=/data/CACTUS_DATA/CLUSTER/STAFF-SC

# extract year month day
set `RCL_decode_datim $datim1`
year=$1
month=$2
day=$3
date_file=$1$2$3

# compute ISO dates
date_iso1=`RCL_encode_datiso $1 $2 $3 $4 $5 $6 `

set `RCL_decode_datim $datim2`
date_iso2=`RCL_encode_datiso $1 $2 $3 $4 $5 $6 `

echo
echo "datim1= $datim1"
echo "datim2= $datim2"
echo
echo "date_iso1= $date_iso1"
echo "date_iso2= $date_iso2"
echo "-----"
echo
```

```

# define files names
file1=CLU$SatNum_STASC_VTL1 "$BitRate" "$date_file".rff
file2=CLU$SatNum_STASC_VTL1 "$BitRate" "$date_file_red".rff
file3=CLU$SatNum_STASC_VTL2 "$BitRate" "$date_file_red".rff
file4=CLU$SatNum_STASC_VTL2 "$BitRate" "$date_file_red_mfa".rff
file5=CLU$SatNum_STASC_SPL2 "$BitRate" "$date_file_red_mfa".rff
file6=CLU$SatNum_STASC_SPL2 "$BitRate" "$date_file_red".rff

# Run successive RCL commands

RCL_get_data_CLUSTA_VTL1 $SatNum $year $month $day $BitRate
if test $? != 0 ; then echo "***get_data : abnormal termination - procedure aborted" ; exit 1 ; fi

RCL_reduce_time_rff $file1 $file2 $date_iso1 $date_iso2
if test $? != 0 ; then echo "***reduce_time : abnormal termination - procedure aborted" ; exit 2 ; fi

RCL_vectime_calibration_CLUSTA $file2 $file3 $fdet $fc 0.1 0.4 $Nkern 2 g
if test $? != 0 ; then echo "***vectime_calibration_CLUSTA: abnormal termination - procedure aborted" ; exit 3 ; fi

RCL_vectime_to_mfa $file3 $file4
if test $? != 0 ; then echo "***vectime_to_mfa : abnormal termination - procedure aborted" ; exit 4 ; fi

RCL_vectime_to_spectro $file4 $file5 $nbp $nbp t
if test $? != 0 ; then echo "***vectime_to_spectro : abnormal termination - procedure aborted" ; exit 5 ; fi

RCL_vectime_to_spectro $file3 $file6 $nbp $nbp t
if test $? != 0 ; then echo "***vectime_to_spectro : abnormal termination - procedure aborted" ; exit 5 ; fi

RCL_spectro_to_polar $file5
if test $? != 0 ; then echo "***spectro_to_polar : abnormal termination - procedure aborted" ; exit 6 ; fi

RCL_visu_spectro $file5 0 0 $f1 $f2 $p1 $p2 LR $f1 $f2
if test $? != 0 ; then echo "***visu_spectro : abnormal termination - procedure aborted" ; exit 7 ; fi

RCL_visu_spectro $file6 0 0 $f1 $f2 $p1 $p2 LR $f1 $f2
if test $? != 0 ; then echo "***visu_spectro : abnormal termination - procedure aborted" ; exit 7 ; fi

RCL_visu_ave_spectrum $file5 0 0 $f1 $f2 $p1 $p2 LR y y
if test $? != 0 ; then echo "***visu_ave_spectrum : abnormal termination - procedure aborted" ; exit 7 ; fi

RCL_visu_ave_spectrum $file6 0 0 $f1 $f2 $p1 $p2 LR y y
if test $? != 0 ; then echo "***visu_ave_spectrum : abnormal termination - procedure aborted" ; exit 7 ; fi

RCL_visu_polar_copolar.resu -5.5 $f1 $f2
if test $? != 0 ; then echo "***visu_polar : abnormal termination - procedure aborted" ; exit 8 ; fi

echo "-----END of PR_compute_staff_polarisation_from_VTL1 -----"

```

**Table 33:** *Contenu du script « PR\_compute\_staff\_polarisation\_from\_VTL1 »*

#### • Traitement en série de plusieurs cas

Lorsqu'on veut traiter plusieurs cas à la suite, on peut faire un script qui lance en série le script précédent. On obtient alors tous les résultats sous forme de fichier PS, PDF et PNG. Ce script devenant un traitement lourd, il est préférable de le lancer dans une fenêtre, et de faire autre chose pendant qu'il s'exécute...

Le fichier « bigrun.bash » donné dans la Table 34 ci-après montre un exemple de traitement en série. Comme dans cet exemple, il est recommandé aussi de mettre d'initialiser les variables de contrôle des plots comme suit :

```

R_VISU_SPLASH=no
R_VISU_PNG=yes_300_256

```

Il est aussi préférable de le lancer dans un répertoire vide, ou de le nettoyer avant comme indiqué dans l'exemple par un « rm \*in \*out \*old \*resu \*rff \*ps \*pdf \*png \*~ \*cef \*bav »

```
#!/bin/bash

R_VISU_SPLASH=no
R_VISU_PNG=yes_300_256

rm *in *out *old *resu *rff *ps *pdf *png *~ *cef *bav

PROG=PR_compute_staff_polarisation_from_VTL1

#      sat yyyymmdd h1m1s1      h2m2s2  nbp fdet  fc   f1   f2 level mode P1   P2
$PROG 3  20010613_014000 20010613_022000 256  0. 0.50  1.0 12.5 4  NBR -6.4 -2.8
$PROG 1  20010923_092000 20010923_100000 256  0. 0.10  0.5 12.5 4  NBR -6.4  1.
$PROG 1  20020823_123000 20020823_125000 128  0. 0.50  2.0 12.5 4  NBR -6.4 -2.4
$PROG 3  20021009_015000 20021009_032000 512  0. 0.10  1.0  8.  4  NBR -6.4 -1.
$PROG 3  20030601_153000 20030601_160000 256  0. 0.50  4.0 12.5 4  NBR -6.4 -1.6
$PROG 4  20030726_101000 20030726_110000 512  0. 0.50  1.5 11.  4  NBR -6.4 -2.3
$PROG 2  20031105_164000 20031105_172000 256  0. 0.10  4.0 12.5 4  NBR -6.4 -2.
$PROG 2  20031124_172000 20031124_175500 256  0. 0.10  4.0 12.5 4  NBR -6.4 -1.
$PROG 2  20040130_164700 20040130_164800 128 20. 5.10 50.0 225.0 4  HBR -9.6 -3.0
$PROG 2  20040227_040000 20040227_050000 512  0. 0.10  0.5  6.  4  NBR -6.4  1.
$PROG 1  20040504_065000 20040504_074000 512  0. 0.50  0.5 12.5 4  NBR -6.4 -1.
$PROG 1  20040520_121000 20040520_124500 256  0. 0.50  2.0 12.5 4  NBR -6.4 -0.7
$PROG 2  20040610_212600 20040610_221600 512  0. 0.50  1.5 12.5 4  NBR -6.4 -3.2
$PROG 3  20040715_140000 20040715_142000 256  0. 0.50  2.0 11.  4  NBR -6.4 -2.7
$PROG 2  20040902_034000 20040902_045000 512  0. 0.10  0.7 10.  4  NBR -6.4 -0.7
$PROG 2  20080724_183000 20080724_191000 256  0. 0.50  1.0 12.5 4  NBR -6.4 -2.9
$PROG 1  20090116_120000 20090116_134000 512  0. 0.10  1.0 10.  4  NBR -6.4 -3.0
```

**Table 34:** Exemple de script de traitement de masse pour l'étude de la polarisation

Dans cet exemple, de nombreux fichiers seront créés. Tous ne sont pas listés, mais obtiendras les fichiers du genre:

#### Les CEF :

CLU1\_FGM\_SPIN\_20021009.cef CLU2\_FGM\_SPIN\_20021009.cef CLU4\_FGM\_SPIN\_20021009.cef  
 CLU1\_FGM\_SPIN\_20040130.cef CLU3\_FGM\_SPIN\_20040130.cef CLU4\_FGM\_SPIN\_20040130.cef

#### Les RFF :

CLU2\_STASC\_SPL2\_HBR\_20040130\_red\_mfa.rff CLU3\_STASC\_SPL2\_NBR\_20021009\_red.rff  
 CLU2\_STASC\_SPL2\_HBR\_20040130\_red.rff CLU3\_STASC\_VTL1\_NBR\_20021009\_red.rff  
 CLU2\_STASC\_VTL1\_HBR\_20040130\_red.rff CLU3\_STASC\_VTL1\_NBR\_20021009.rff  
 CLU2\_STASC\_VTL1\_HBR\_20040130.rff CLU3\_STASC\_VTL2\_ISR2\_NBR\_20021009.rff  
 CLU2\_STASC\_VTL2\_HBR\_20040130\_red\_mfa.rff CLU3\_STASC\_VTL2\_NBR\_20021009\_red\_mfa.rff  
 CLU2\_STASC\_VTL2\_HBR\_20040130\_red.rff CLU3\_STASC\_VTL2\_NBR\_20021009\_red.rff  
 CLU3\_STASC\_SPL2\_NBR\_20021009\_red\_mfa.rff

#### Les PNG (et de meme les PS et PDF)

CLU2\_STASC\_SPL2\_HBR\_20040130\_red\_ave\_spe.png  
 CLU2\_STASC\_SPL2\_HBR\_20040130\_red\_mfa\_ave\_spe.png  
 CLU2\_STASC\_SPL2\_HBR\_20040130\_red\_mfa.png  
 CLU2\_STASC\_SPL2\_HBR\_20040130\_red.png  
 CLU3\_STASC\_SPL2\_NBR\_20021009\_red\_ave\_spe.png  
 CLU3\_STASC\_SPL2\_NBR\_20021009\_red\_mfa\_ave\_spe.png  
 CLU3\_STASC\_SPL2\_NBR\_20021009\_red\_mfa.png  
 CLU3\_STASC\_SPL2\_NBR\_20021009\_red.png  
 copolar.png

### • Script d'automatisation à partir des VTL2

Le script de la ci-dessous permet de faire la même chose que le précédent, mais directement à partir des formes d'onde calibrées. On effectue ensuite leur passage dans le repère MFA, le calcul et le tracé des spectres calibrés dans le repère SR2 et dans le repère MFA, visualisés en composantes circulaires gauche et droite, ainsi que le graphique résumant tous les paramètres de polarisation.

```
#!/bin/sh

# -----
# Script: PR_compute_staff_polarisation_from_VTL2
# Usage : PR_compute_staff_polarisation_from_VTL2 arguments
# arguments: sat yyyyymmdd h1m1s1 h2m2s2 nbp fdet fc f1 f2 level mode P1 P2
#           2 20040130_164700 20040130_164800 128 20. 5.10 50.0 0. 4 HBR -9.6 -3.0
# -----

if test $# != 12
then
    echo "this procedure requires 12 arguments. Example:"
    echo "PR_compute_staff_polarisation_from_VTL1 2 20040130_164700 \
20040130_164800 \
128 20. 5.10 50.0 0. 4 HBR -9.6 -3.0"
    exit 1
fi

SatNum=$1
datim1=$2
datim2=$3
nbp=$4
fdet=$5
fc=$6
f1=$7
f2=$8

shift 8

level=$1
BitRate=$2
p1=$3
p2=$4

if test $BitRate = 'NBR'
then
    Nkern=1024
else
    Nkern=4096
fi

# if need, change default path for STAFF data base
# RCS_DATA=/NFS/tamaya/data2/staff-op/STAFF-SC
# RCS_DATA=/data/CACTUS_DATA/CLUSTER/STAFF-SC

# extract year month day
set `RCL_decode_datim $datim1`
year=$1
month=$2
day=$3
date_file=$1$2$3

# compute ISO dates
date_iso1=`RCL_encode_datiso $1 $2 $3 $4 $5 $6 `
set `RCL_decode_datim $datim2`
date_iso2=`RCL_encode_datiso $1 $2 $3 $4 $5 $6 `

echo
echo "datim1= $datim1"
echo "datim2= $datim2"
echo
echo "date_iso1= $date_iso1"
echo "date_iso2= $date_iso2"
echo "-----"
echo
```

```

# define files names

file1=CLU$SatNum"_STASC_VTL2_ISR2_"$BitRate" "$date_file".rff
file2=CLU$SatNum"_STASC_VTL2_"$BitRate" "$date_file"_red".rff
file3=CLU$SatNum"_STASC_VTL2_"$BitRate"_"$date_file"_red".rff
file4=CLU$SatNum"_STASC_VTL2_"$BitRate" "$date_file"_red_mfa".rff
file5=CLU$SatNum"_STASC_SPL2_"$BitRate" "$date_file"_red_mfa".rff
file6=CLU$SatNum"_STASC_SPL2_"$BitRate"_"$date_file"_red".rff

# RCL launch

RCL_get_data CLUSTA_VTL2 $SatNum $year $month $day $BitRate ISR2
if test $? != 0 ; then echo "***get_data : abnormal termination - procedure aborted"; exit 1 ; fi

RCL_reduce_time_rff $file1 $file2 $date_iso1 $date_iso2
if test $? != 0 ; then echo "***reduce_time : abnormal termination - procedure aborted"; exit 2 ; fi

RCL_vectime_to_mfa $file3 $file4
if test $? != 0 ; then echo "***vectime_to_mfa : abnormal termination - procedure aborted"; exit 4 ; fi

RCL_vectime_to_spectro $file4 $file5 $nbp $nbp t
if test $? != 0 ; then echo "***vectime_to_spectro: abnormal termination - procedure aborted"; exit 5 ; fi

RCL_vectime_to_spectro $file3 $file6 $nbp $nbp t
if test $? != 0 ; then echo "***vectime_to_spectro: abnormal termination - procedure aborted"; exit 5 ; fi

RCL_spectro_to_polar $file5
if test $? != 0 ; then echo "***spectro_to_polar : abnormal termination - procedure aborted"; exit 6 ; fi

RCL_visu_spectro $file5 0 0 $f1 $f2 $p1 $p2 LR $f1 $f2
if test $? != 0 ; then echo "***visu_spectro : abnormal termination - procedure aborted"; exit 7 ; fi

RCL_visu_spectro $file6 0 0 $f1 $f2 $p1 $p2 LR $f1 $f2
if test $? != 0 ; then echo "***visu_spectro : abnormal termination - procedure aborted"; exit 7 ; fi

RCL_visu_ave_spectrum $file5 0 0 $f1 $f2 $p1 $p2 LR y y
if test $? != 0 ; then echo "***visu_ave_spectrum : abnormal termination - procedure aborted"; exit 7 ; fi

RCL_visu_ave_spectrum $file6 0 0 $f1 $f2 $p1 $p2 LR y y
if test $? != 0 ; then echo "***visu_ave_spectrum : abnormal termination - procedure aborted"; exit 7 ; fi

RCL_visu_polar copolar.resu -5.5 $f1 $f2
if test $? != 0 ; then echo "***visu_polar : abnormal termination - procedure aborted"; exit 8 ; fi

echo "-----END of PR_compute_staff_polarisation_from_VTL2 -----"

```

**Table 35:**      *Contenu du script « PR\_compute\_staff\_polarisation\_from\_VTL2 »*

Comme pour le script équivalent sur les VTL1, de nombreux fichiers RFF, PNG etc. seront créés, il est donc recommandé de lancer ce script dans un répertoire vide.

## 5.6. SPECTRES ET SPECTROGRAMMES SUR LES DONNÉES FGM

### 5.6.1 Principe

Pour faire des spectres ou spectrogramme de *n'importe quelle expérience « onde »*, il suffit de pouvoir disposer des données de forme d'onde correspondantes au format VTL2 (forme d'onde calibrée).

Après, il suffit d'utiliser les commandes RCL génériques, comme RCL\_vectime\_to\_spectro et RCL\_visu\_spectro.

Le script donné Table 36 ci-dessous permet :

- De télécharger les données FGM FULL résolution depuis le CSA,
- De convertir le fichier CEF obtenu en un fichier au format RFF et de valider son format,
- De calculer et visualiser le spectrogramme correspondant.

```
#!/bin/bash

# download FGM/FULL data from CSA
RCL_download_data_CLUFGM_oneday_t1t2 1 2001 09 23 09 20 10 30 FULL

# conversion in RFF
RCL_fgm_cef_to_rff CLU1_FGM_FULL_20010923_0920_1030.cef

# check valid RFF format
RCL_check_rff CLU1_FGM_FULL_20010923_0920_1030.rff
RCL_info_rff CLU1_FGM_FULL_20010923_0920_1030.rff l

# compute spectrogram
RCL_vectime_to_spectro CLU1_FGM_FULL_20010923_0920_1030.rff \
                      CLU1_FGM_FULL_20010923_0920_1030_SP.rff 512 512 t

# visualize spectrogram (PS, PDF and PNG file)
RCL_visu_spectro CLU1_FGM_FULL_20010923_0920_1030_SP.rff 0 0 0. 4. -4. 0. XY
0.5 4.
```

**Table 36:** Script permettant de calculer et visualiser un spectrogramme de FGM

### 5.6.2 Exemple de résultat

La

Figure 11 ci-après montre le résultat obtenu par l'exécution de ce script. On a borné volontairement la fréquence à 4 Hz afin de zoomer sur la partie basse fréquence. En effet, si on regarde la même période de temps obtenue avec les données STAFF (voir Figure 35 page 187), seule la composante Bz est propre à basse fréquence, les composantes X et Y étant perturbées par le spin. Avec FGM, qui mesure les basses fréquences jusqu'au continu, les 3 composantes sont propres. Naturellement, le paquet d'onde monochromatique est aussi visible sur la composante Bz des données STAFF.

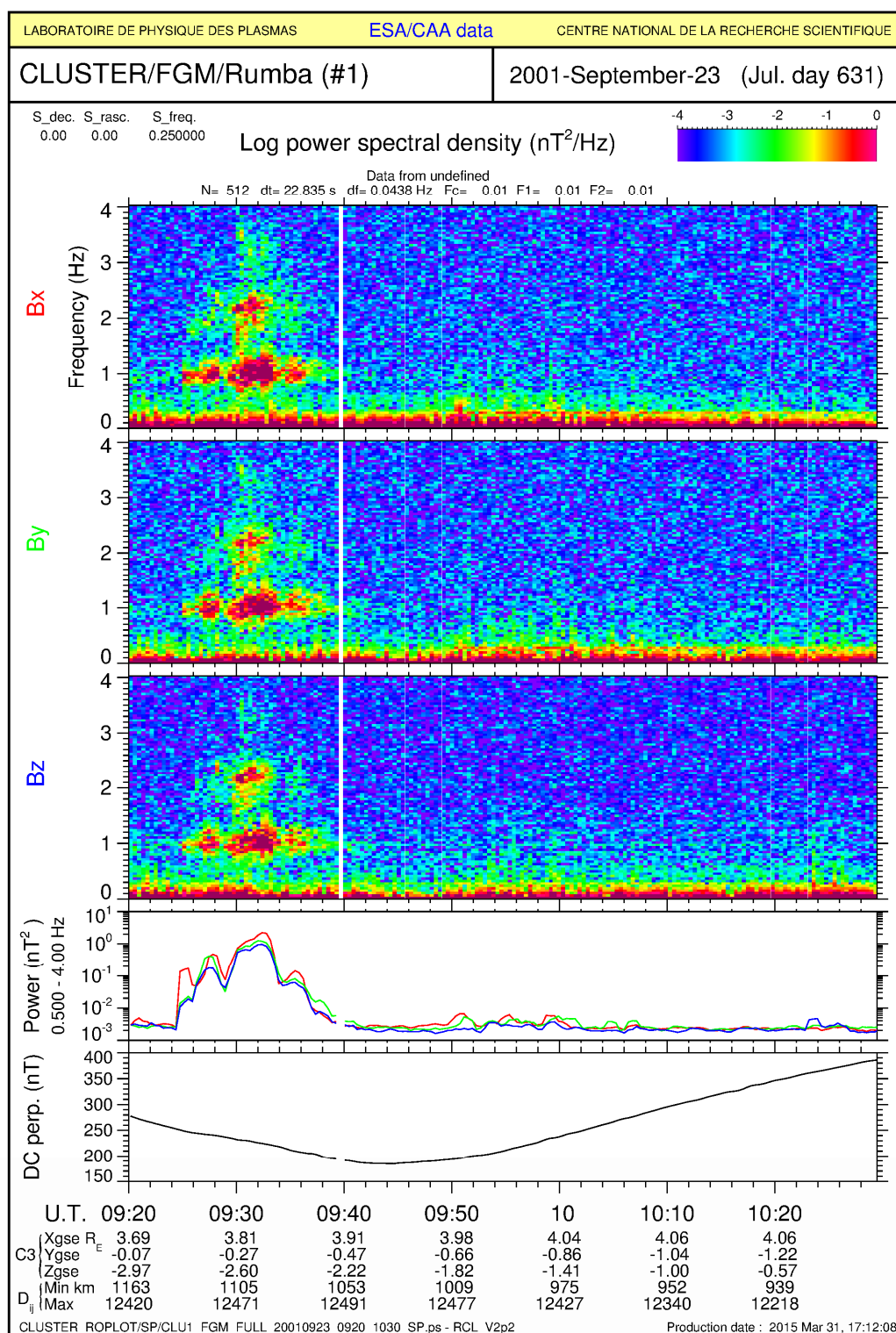


Figure 11: Exemple de spectrogramme des données FGM produit à partir des commandes RCL





## 6. INSTALLATION DU PACKAGE RCL

### 6.1. [INSTALLATION DU PACK BINAIRE](#)

#### 6.1.1 *Copie du pack sur la machine cible*

Il est nécessaire de disposer du pack correspondant à la machine cible. Par exemple, pour une machine linux 64 bits, il suffit de recopier le directory `RCL_V2p2_Linux_x86_64` n'importe où sur votre machine, puis de mettre à jour les variables d'environnement dans le `RCL_config.csh` ou le `RCL_config.bash` suivant le shell utilisé (voir § suivant).

Généralement, le shell utilisé sur les machines linux est le bash, le csh ou encore le tcsh, alors que sous windows, via l'émulateur Msys, le shell est le bash. L'initialisation des variables d'environnement se fera en bash par la commande « `. RCL_config.bash` » et en csh ou tcsh par « `source RCL_config.csh` ».

Il est pratique de mettre cette commande dans le `.bashrc`, le `.cshrc`, ou le `.tcshrc` afin qu'il soit exécuté dès la connexion ou lors de l'ouverture d'une nouvelle fenêtre de commande. Par exemple, en bash, il faut rajouter dans le fichier `$HOME/.bashrc` une ligne du type :

```
./data/CACTUS_HOME/RCL/RCL_V2p2/RCL_config.bash
```

Bien sûr le chemin indiqué correspond au répertoire où a été installé le pack. Après cette initialisation, à l'ouverture de chaque fenêtre de commande, toutes les commandes RCL sont accessibles depuis n'importe quel répertoire, comme par exemple « `RCL_menu` » que l'on peut taper pour vérifier que les commandes sont bien accessibles.

#### 6.1.2 *Réglage des variables d'environnement*

Suivant que la machine est en csh ou en bash, il faut éditer le fichier `RCL_config.csh` ou `RCL_config.bash`. Pour savoir quel shell est utilisé, il suffit de taper « `echo $SHELL` ». Un exemple de `RCL_config.bash` est donné dans la Table 37 ci-dessous. On peut voir qu'on définit les variables d'environnement suivantes :

**RCL\_DIR** : le chemin du répertoire où est installé le logiciel. Permet l'extension de la variable `$PATH`, pour que les commandes RCL puissent être trouvées

**R\_IDL** : le chemin où est définie la commande IDL sur la machine ou le réseau.

**R\_VISU\_PDF**=yes pour produire un PDF, en même temps que les PostScripts après l'exécution d'une commande `RCL_visu`.

**R\_VISU\_PNG**=yes\_300 pour produire un PNG de résolution 300 ppp, en même temps que les PostScripts après l'exécution d'une commande `RCL_visu`.

**R\_VISU\_SPLASH**=yes si on veut que le pdf soit affiché automatiquement à l'écran après l'exécution d'une commande `RCL_visu`. Doit être mis à « no » pour une exploitation. L'affichage à

l'écran se fait par le logiciel « acroread » ou « evince ». sous Linux, et par la commande « start » sous Windows

**RCS\_DATA**=/NFS/tamaya/data2/staff-op/STAFF-SC pour la base des fichiers RFF VTL1, VTL2 et SPL2

**RCO\_DATA**=/NFS/tamaya/CLUSTERII/ORBITO/DATA pour la base des .orb

**RCF\_DATA**=/data/CACTUS\_DATA/CLUSTER/FGM pour les données FGM existant sur la machine lx-robert ; seulement FGM\_SPIN et FGM\_5VPS sont disponibles pour l'instant.

```
#!/bin/bash
# =====
#
# Object:
# set env. variables to use RCL commands everywhere.
#
# usage :
# Linux sh or bash : . RCL config.bash
#
# Patrick ROBERT, CNRS/LPP, 2011-03-21
#
# =====
#
# =====
# Software path
# =====

# Manual set of RCL_config.bash directory

#linux:
export RCL_DIR=/data/CACTUS_HOME/RCL/RCL_V2p2

#windows
#export RCL DIR=d:/SOFTS a installer/RCL_V2p2

echo "RCL software is located in $RCL DIR"

# =====
# Path extension
# =====

# This is required to find the RCL commands

export PATH="$PATH:$RCL_DIR/bash"

# =====
# Environment variables for Data for visualisation options
# =====

#export R IDL=/NFS/casa/soft/rsi/idl 8.01/idl/idl80/bin/idl
export R_IDL=/usr/local/bin/idl

export R_VISU_PDF=yes
export R_VISU_PNG=yes_300
export R_VISU_SPLASH=yes

uname -s | cut -c1-5 > toto.tmp
Syst=`cat toto.tmp`; rm -f toto.tmp

if (test $Syst = 'MINGW')
then export R_SOFT_SPLASH=start
else export R_SOFT_SPLASH=acroread
fi

# =====
# Environment variables for Data
# =====

# CLUSTER DATA:

export RCS_DATA=/NFS/tamaya/data2/staff-op/STAFF-SC
export RCO_DATA=/NFS/tamaya/CLUSTERII/ORBITO/DATA
export RCF_DATA=/data/CACTUS_DATA/CLUSTER/FGM

echo "RCS data are located in $RCS_DATA"
echo "RCO data are located in $RCO_DATA"
echo "RCF data are located in $RCF_DATA"

# =====
```

*Table 37: Exemple de fichier RCL\_config.bash*

Un exemple pour le csh est également fourni dans le pack. Seule la syntaxe du shell change, par exemple « export R\_VISU\_PDF=yes » devient « setenv R\_VISU\_PDF yes »

## 6.2. INSTALLATION DU PACK SOURCE

### 6.2.1 Copie du pack sur la machine cible

Le pack source RCL\_V2p2 peut être recopié n'importe où sur votre machine, pourvu qu'elle dispose d'un compilateur Fortran90. Il faut alors éditer le Makefile, et choisir le compilateur disponible. Si le compilateur ne figure pas dans la liste proposée, il faut l'introduire, avec les options du mode « Debug » ou du mode « Optimized » (voir § suivant). Pour Windows, il faut au préalable avoir installé Msys (Mingw) afin de disposer d'un environnement bash.

Ensuite, lancer la commande « make clean » qui effacera tous les exécutables présents compilés sur l'ancienne machine (si ce n'a pas déjà été fait avant), puis « make all » qui produira tous les exécutables pour le(s) processeur(s) de la machine cible.

Après compilation, mettre à jour les variables d'environnement dans le fichier RCL\_config.csh ou .bash comme ci-dessus.

### 6.2.2 Mise à jour du Makefile

Pour transporter le pack RCL sur une autre machine, en vue d'une recompilation générale des codes sources, ou lors de modifications des programmes, il faut mettre à jour le fichier Makefile, et en particulier les champs suivants :

#### • Choix du compilateur

Les principaux compilateurs et leurs options sont déjà prédéfinis dans le makefile ; il suffit de commenter ou non le champ correspondant :

```
# =====
# Choix du compilateur
# =====

FC = ifort
#FC = f90
#FC = f95
#FC = gfortran
#FC = gfortran-4.4
#FC = g95
```

Si le compilateur désiré n'existe pas, il faut le rajouter, ainsi que ses options d'utilisation (en mode « debug » ou « optimized » comme il est indiqué plus loin).

#### • Option debug ou optimized

Pour chacun des compilateurs, on peut choisir une compilation avec l'option « debug », qui sera bavarde et fera plein de vérifications, et surtout qui sortira un diagnostic plus complet en cas de plantage, permettant ainsi de trouver rapidement l'erreur. Cette option est recommandée lors du développement ou lors de modifications des codes. Quand on passe en production, il faut choisir l'option « optimisée », qui est beaucoup plus rapide à l'exécution :

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

MODE=O
#MODE=D
```

### • Options de compilation

Si besoin est, on peut modifier les options prédéfinies de compilation, pour chacun des compilateurs, ou rajouter un nouveau compilateur avec ses options spécifiques. Par exemple, pour le compilateur Intel et gfortran, les options prédéfinies sont les suivantes (Table 38) :

```
# =====
# Choix des options de compilation
# =====

# Compilateur Intel
ifeq ($(FC),ifort)
    ifeq ($(MODE),D)
# remplacement du check all par tous les checks sauf le check arg_temp_created
        FCOMP = -c -CB -g -check format -check unit -check output_conversion -traceback
        FLINK = -CB -g -check format -check unit -check output_conversion -traceback
    else
        FCOMP = -c
        FLINK =
    endif
endif

# Compilateur gfortran linux et Windows/MinGW gfortran
ifeq ($(FC),gfortran)
    ifeq ($(MODE),D)
        FCOMP = -c -g -static -fbounds-check -Wconversion -Wall -Wunderflow -W -fno-automatic
        FLINK = -g -static -fbounds-check -Wconversion -Wall -Wunderflow -W -fno-automatic
    else
        FCOMP = -c
        FLINK =
    endif
endif
```

**Table 38:** Options de compilation du fichier Makefile

### • Options du make clean

Pour le make clean, les actions sont classiques. Il faut lancer cette commande lorsque l'on change de machine : tous les binaires, objets et exécutables, sont effacés. On peut alors faire le make all, qui recompile tout. Si on désire que le make clean efface d'autres fichiers, on peut l'indiquer dans cette section (Table 39).

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

# Efface les .exe et les .o de ./bin et ./obj
# Efface les .tmp, *.core, *~
# Garde les .in, .out, .resu et .old

clean :
    @echo "**** Removing *.exe *.o, .mod, .tmp, *~, core files..."

    @find ./bin -name "*.exe" -exec rm -f {} \;
    @find ./obj -name "*.o" -exec rm -f {} \;
    @find . -name ".tmp" -exec rm -f {} \;
    @find . -name "*core*" -exec rm -f {} \;
    @find . -name "*~" -exec rm -f {} \;
    @echo "**** Done."
```

**Table 39 :** Options « make clean » du fichier Makefile

### • Options du make mrproper

Le make mrproper fait un clean et efface en plus les .in, .out, .rff, .cef, .old et toto\* dans le répertoire test (on conserve ceux du répertoire data, voir Table 40).

```
# =====
# make mrproper
# =====
# Fait un clean et efface en plus les .in, .out, .rff, .cef, .old et toto*
# dans le directory test (on conserve ceux dans le data)

mrproper : clean
@echo "*** Removing .in, .out, rff., .cef, .old and toto* "

@find ./test -name "*.in" -exec rm -f {} \;
@find ./test -name "*.out" -exec rm -f {} \;
@find ./test -name "*.rff" -exec rm -f {} \;
@find ./test -name "*.cef" -exec rm -f {} \;
@find ./test -name "*.old" -exec rm -f {} \;
@find ./test -name "toto*" -exec rm -f {} \;
@echo "*** Done."
```

Table 40: Options « make mrproper » du fichier Makefile

### 6.2.3 Régénération des .sav d'IDL

Le fichier .sav d'IDL servent aux commandes de type **RCL\_visu\_\***, et sont *en principe* portables d'une machine à l'autre, et d'une version d'IDL à une autre. Néanmoins il doit être possible de les régénérer en cas de problèmes, ou de manque de compatibilité entre la version d'IDL qui les a produites et celle qui les utilise.

Cette régénération est faite simplement en lançant la commande `Make_IDL_sav.bash`. Cette commande enchaîne sur tous les programmes de visualisation un jeu de commandes du type décrit dans la Table 41 ci-dessous. Après création du `visu_vectime.sav`, celui-ci est transféré dans le répertoire des exécutables `../bin`

```
echo
echo "-----"
echo "create visu_vectime.sav"

$R IDL << !!
.FULL RESET_SESSION
.COMPILE roploplib_201301.pro
.COMPILE julday.pro
.COMPILE rff_read_VTfile.pro
.COMPILE visu_vectime.pro
save, /routines, filename='visu_vectime.sav'
!!

echo 'move visu_vectime.sav in bin directory'
mv visu_vectime.sav ../bin
echo "done..."
```

Table 41: Compilation des IDL.pro dans le fichier `Make_IDL_sav.bash`

### 6.2.4 Création d'un pack binaire

Pour créer un pack binaire sur la machine où on dispose des codes sources, il suffit de lancer le shell `Make_bin_pack.bash`. Celui-ci créera dans le répertoire `RCL_V2p2` un directory `RCL_V2p2_Linux_x86_64` si la machine est un `Linux_x86_64`, ainsi qu'un fichier tar correspondant, que l'on pourra transporter sur une machine de même type. Cette procédure utilise la commande Unix `uname -sm` pour identifier la plateforme.

Préalablement à la création du pack binaire, cette procédure nettoie le répertoire en effectuant un `make mrproper`, qui aura pour effet d'effacer les résultats éventuels dans le répertoire de test, puis un `make all` qui recompilera tout. Il faudra vérifier avant de lancer cette procédure que le Makefile est configuré en mode « optimized » et non en mode « debug » afin que le pack binaire soit optimisé pour la production.

### 6.3. PROCÉDURES DE TEST

Après un changement de machine, ou de compilateurs, ces procédures permettent de tester le bon fonctionnement des commandes. Elles sont particulièrement utiles avant de lancer une production de masse.

Tous les répertoires de test contiennent une procédure `clean.bash` permettant de nettoyer le répertoire des tests précédents.

#### 6.3.1 *test\_commands*

La procédure `test_commands.bash` teste les commandes usuelles, à l'exception des commandes d'extraction de la base de données et des commandes de production (voir les paragraphes suivants). Ce shell lance les commandes suivantes :

```
RCL_version
RCL_list
RCL_nday_of_month 1997 11
RCL_previous_day 20070925
RCL_check_file $RCL_DIR/data/CLU1_STASC_NBR_WFL1_20091213.rff
RCL_clean_file $RCL_DIR/data/CLU1_STASC_NBR_WFL1_20091213.rff
RCL_waveform_to_vectime $RCL_DIR/data/CLU1_STASC_NBR_WFL1_20091213.rff \
                    CLU1_STASC_NBR_VTL1_20091213.rff
RCL_check_file CLU1_STASC_NBR_VTL1_20091213.rff
RCL_vectime_to_cef CLU1_STASC_NBR_VTL1_20091213.rff
```

#### 6.3.2 *test\_get\_data*

La procédure `test_get_data.bash` lance la procédure `RCL_get_data_CLUSTA` par la commande `RCL_get_data_CLUSTA 1 2009 12 14 NBR` sur la mini base de données du répertoire `data`, effectue un `RCL_clean_file` sur le fichier initial et le fichier résultat, et lance un diff entre les deux fichiers. Cette différence permet de vérifier qu'on a bien récupéré un bloc supplémentaire, de la fin du jour précédent, dans le fichier résultat, et que les autres champs (comme la date de création) ont bien été mis à jour.

Contrairement aux tests de production, ces deux procédures de test ne nécessitent pas un accès réseau à la base de données.

#### 6.3.3 *test\_production\_oneday*

La procédure `test_production_oneday_NBR.bash` lance la commande :

```
RCL_product_DWF_oneday 1 2009 12 14 NBR
```

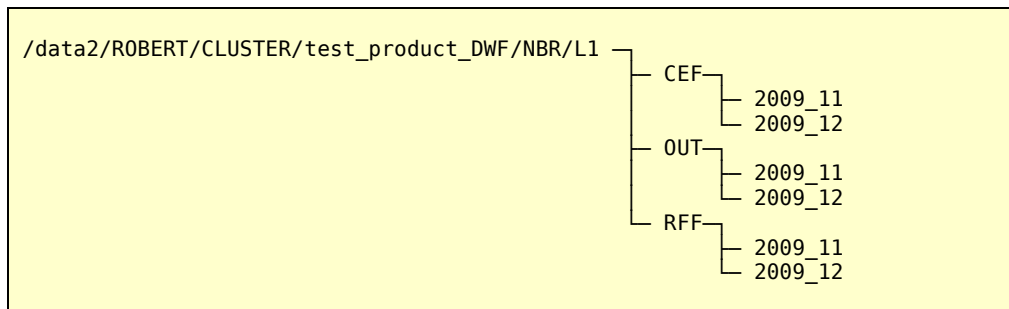
Et crée donc le fichier `CLU1_STASC_NBR_VTL1_20091214.rff`, sur lequel est lancé un `RCL_check_file`

La procédure `test_production_oneday_HBR.bash` effectue la même chose, mais sur le 12 décembre, en mode HBR.

#### 6.3.4 test\_production\_onemonth

Même chose que la commande précédente, mais produit un mois entier de données (pour décembre 2009), pour les 4 satellites. Comme le résultat est volumineux, on donne en début de procédure le chemin d'un répertoire data pour accueillir les résultats. Cette procédure prend environ 2 heures en NBR sur une machine Linux\_x86\_64 comme cactus.

Il faut noter que ces résultats seront créés dans une arborescence identique à celle de la base de données, à savoir par exemple (Table 42):



**Table 42:**      *Arborescence des répertoires créés par le test de production sur un mois.*

Le répertoire `OUT` contient les rapports d'exécution de la commande de conversion de fichier `RCL_waveform_to_vectime` sous la forme d'une suite de fichiers de type `waveform_to_vectime_20090122.out`, afin de pouvoir diagnostiquer, en cas de mauvais fonctionnement, le type de problème rencontré. Il est à détruire par la suite.

#### 6.3.5 test\_production\_oneyear

Même chose que la commande précédente, mais sur une année complète (2009). Attention, cette commande prend environ 24 heures en NBR sur une machine Linux\_x86\_64 comme cactus.



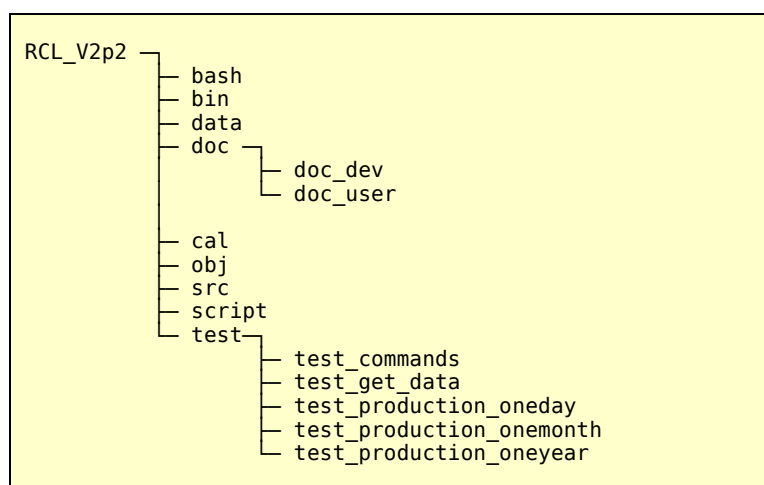


## 7. A L'INTENTION DES DEVELOPPEURS

Cette partie est destinée à ceux qui voudront maintenir et faire évoluer ce projet. Il décrit son organisation, les programmes développés, ainsi que les bibliothèques spécifiquement développées pour que le projet puisse être appliqué facilement à n'importe quelle autre mission spatiale du même type.

### 7.1. ORGANISATION DU PROJET

Au niveau de l'arborescence, le projet RCL pour le CSA est organisé classiquement (Table 43):



**Table 43:** *Arborescence des répertoires du projet RCL.*

Le répertoire RCL lui-même comprend les fichiers suivants (Table 44):

```

README.txt
menu.txt
RCL_config.bash
RCL_config.csh
Makefile
Make_IDL_sav.bash
Make_bin_pack.bash
  
```

**Table 44:** *Fichiers à la racine du projet RCL.*

### **7.1.1 Fichier README.txt**

Fichier contenant un résumé de la procédure d'installation

### **7.1.2 Fichier menu.txt**

Fichier utilisé par la commande **RCL\_menu**, et qui donne la liste classée par catégorie des commandes RCL.

### **7.1.3 Fichier RCL\_config.bash**

Fichier de configuration, à mettre à jours lors de l'installation (voir chapitre 6.1.2) afin d'initialiser les variables d'environnement nécessaires, et notamment le path des bases de données.

### **7.1.4 Fichier RCL\_config.csh**

Même chose que le précédent, mais pour les machines dont le shell est un csh.

### **7.1.5 Fichier Makefile**

Pour la compilation des sources F90 (voir chapitre 6.2.2)

### **7.1.6 Fichier Make\_IDL\_sav.bash**

Pour la génération des .sav d'IDL (nécessite d'avoir IDL d'installé sur la machine).

### **7.1.7 Fichier Make\_bin\_pack.bash**

Permet de créer un pack binaire pour l'installation sur une machine de même type.

### **7.1.8 Répertoire bash**

Répertoire contenant toutes les commandes RCL, toutes écrites en bash. C'est ce répertoire qui est ajouté dans la variable PATH quand on met à jour le fichier RCL\_config, et qui permet au système de trouver toutes les commandes RCL lorsque l'utilisateur en tape une à partir de n'importe quel endroit.

### **7.1.9 Répertoire bin**

Répertoire contenant tous les exécutable de type « .exe » qui sont générés à partir des sources F90 lors de l'exécution du fichier Makefile, ainsi que les « .sav » générés par IDL à partir des souces .pro.

### **7.1.10 Répertoire doc**

Répertoire contenant toute la documentation du projet, y compris celle-ci

### **7.1.11 Répertoire gainantset**

Répertoire contenant les tables de calibrations pour l'expérience STAFF-SC, tables anciennement appelées « gains d'antenne ».

### **7.1.12 Répertoire mod**

Répertoire contenant les modules générés lors de l'exécution du fichier Makefile.

### **7.1.13 Répertoire obj**

Répertoire contenant tous les objets de type « \*.o » générés lors de la compilation. En théorie, ils pourraient être effacés dans la mesure où les « .exe » ont été correctement générés.

#### **7.1.14 Répertoire pro**

Répertoire contenant les sources des programmes IDL, sous forme de «\* .pro »

#### **7.1.15 Répertoire script**

Répertoire contenant tous les scripts développés par l'utilisateur, et qui peuvent être considérés comme des commandes RCL en cours de validation (voir § 5.1.3). Avec le répertoire « test », c'est le seul répertoire que l'utilisateur devrait être autorisé à modifier.

#### **7.1.16 Répertoire src**

Répertoire contenant les sources des programmes F90, sous la forme de « \*.f90 ».

#### **7.1.17 Répertoire test**

Répertoire contenant les procédures de test, permettant de vérifier que les codes ont été portés correctement sur une autre machine, et qu'ils donnent les mêmes résultats. L'utilisateur peut éventuellement compléter ce répertoire par ses propres tests, comme par exemple des tests sur les scripts qu'il est amené à développer. Tous les tests répertoriés sont décrits en détail au chapitre 6.3.

## 7.2. COMMANDES RCL UTILISANT DES EXECUTABLES F90

Les commandes RCL sont écrites en bash, et certaines d'entre elles utilisent les programmes exécutables écrits en F90. Ces programmes effectuent des traitements complexes, et sont décrits au chapitre 7.5. Ils utilisent des bibliothèques spécifiques développées également en F90, et décrites au chapitre 7.6. La Table 45 ci-dessous donne la liste des commandes RCL qui utilisent ces exécutables.

Exécutable F90	Commande RCL utilisatrice
check_efw_status.exe	<i>Usage ponctuel indépendant des RCL</i>
check_rff.exe	RCL_check_rff
clean_rff.exe	RCL_clean_rff
CWF_cef_to_rff.exe	RCL_CWF_cef_to_rff
diff_rff.exe	RCL_diff_rff
dir_properties_pretty_tree.exe	RCL_dir_properties_pretty_tree
dir_size_pretty_tree.exe	RCL_dir_size_pretty_tree
DWF_cef_to_rff.exe	RCL_DWF_cef_to_rff
fgm_cef_gse_to_sr2.exe	RCL_fgm_cef_gse_to_sr2
fgm_cef_to_rff.exe	RCL_fgm_cef_to_rff
get_data_CLUGEOM.exe	RCL_get_data_CLUGEOM
get_data_CLUP0S.exe	RCL_get_data_CLUP0S
reduce_time_rff.exe	RCL_reduce_time_rff
search_strong_DC.exe	RCL_search_strong_DC
spectro_L2_to_cef.exe	RCL_spectro_L2_to_cef
spectro_to_polar.exe	RCL_spectro_to_polar
spectro_xyz_to_lrz.exe	RCL_spectro_xyz_to_lrz
vectime_calibration_CLUSTA.exe	RCL_vectime_calibration_CLUSTA
vectime_L1_to_cef.exe	RCL_vectime_L1_to_cef
vectime_L1_to_spectro_L2.exe	RCL_vectime_L1_to_spectro_L2
vectime_L2_to_cef.exe	RCL_vectime_L2_to_cef
vectime_to_mfa.exe	RCL_vectime_to_mfa
vectime_to_spectro.exe	RCL_vectime_to_spectro
waveform_to_vectime.exe	RCL_waveform_to_vectime
w_cal_caveat_header.exe	RCL_vectime_L2_to_cef
w_cwf_cef_header.exe	<i>Pour archive. N'est plus utilisé</i>
w_vectime_cef_header.exe	<i>Pour archive. N'est plus utilisé</i>

Table 45: Liste des commandes RCL utilisant les exécutable du répertoire bin

### 7.3. COMMANDES RCL UTILISANT DES SAV IDL

Les commandes RCL sont écrites en bash, et certaines d'entre elles utilisent des procédures IDL. Si les fichiers sources sont donnés dans le répertoire « pro », les procédures RCL utilisent des fichiers précompilés au format SAV d'IDL afin d'être exécutées plus rapidement.

Il faut noter qu'une procédure créée en IDL 6.0 ou avec une version ultérieure qui n'utilise pas la certaines fonctions comme EXECUTE peut être enregistrée dans un ou plusieurs fichiers de type .sav qui peuvent être exécutés dans la machine virtuelle IDL disponible gratuitement. Il est alors possible d'utiliser le logiciel RCL sans avoir de licence IDL payante.

Pour information, dans la version RCL\_V2.2, les .sav ont été générés avec IDL Version 8.2.3 (linux x86\_64 m64). Si on utilise une version plus récente, il faudra recompiler les sources (voir chapitre 6.2.3 pour la re-génération des .sav). La Table 46 ci-dessous donne la liste des commandes utilisant IDL utilisant une version avec ou sans License.

SAV IDL	Commande RCL utilisatrice
visu_ave_spectrum.sav	RCL_visu_ave_spectrum
visu_spectro_4Bz.sav	RCL_visu_spectro_4Bz
visu_spectro.sav	RCL_visu_spectro
visu_vectime.sav	RCL_visu_vectime
visu_vectime_widget.sav	RCL_visu_vectime_widget
visu_polar.sav	RCL_visu_polar
visu_CLUGEOM.sav	RCL_visu_CLUGEOM
visu_CLUPOS.sav	RCL_visu_CLUPOS

*Table 46: Liste des commandes RCL utilisant des SAV d'IDL du répertoire bin*

### 7.4. COMMANDES RCL UTILISANT D'AUTRES COMMANDES RCL

Les commandes RCL peuvent appeler d'autres commandes RCL, notamment toutes les commandes de production par exemple. La liste des commandes RCL qui en nécessitent d'autres (et donc qui ne sont pas indépendantes) est donnée dans la Table 47 ci-dessous.

Si on veut utiliser une commande par simple copie sur une autre machine, et que le logiciel RCL n'est pas installé, il faut également copier les commandes appelées, et éventuellement les .exe et .sav utilisés.

Commande RCL	: Commande(s) utilisée(s)
RCL_download_data_CLUFGM_onemonth_nolog	: RCL_list_days_of_month RCL_download_data_CLUFGM_oneday
RCL_download_data_CLUFGM_oneyear	: RCL_download_data_CLUFGM_onemonth
RCL_get_data_CLUSTA_VTL1_forSPL2	: RCL_next_day RCL_version
RCL_get_data_CLUSTA_VTL1_forVTL2	: RCL_previous_day RCL_next_day RCL_version
RCL_get_data_CLUSTA_WFL1_forVTL1	: RCL_previous_day RCL_version
RCL_list_days_of_month	: RCL_nday_of_month
RCL_move_rff	: RCL_copy_rff
RCL_next_day	: RCL_nday_of_month
RCL_previous_day	: RCL_nday_of_month
RCL_product_CS_oneday	: RCL_spectro_L2_to_cef
RCL_product_CS_onemonth	: RCL_product_CS_onemonth_nolog
RCL_product_CS_onemonth_nolog	: RCL_list_days_of_month RCL_product_CS_oneday
RCL_product_CS_oneyear	: RCL_product_CS_onemonth
RCL_product_CWF_oneday	: RCL_vectime_L2_to_cef
RCL_product_CWF_onemonth	: RCL_product_CWF_onemonth_nolog
RCL_product_CWF_onemonth_nolog	: RCL_list_days_of_month RCL_product_CWF_oneday
RCL_product_CWF_oneyear	: RCL_product_CWF_onemonth
RCL_product_DWF_oneday	: RCL_vectime_L1_to_cef
RCL_product_DWF_onemonth	: RCL_product_DWF_onemonth_nolog
RCL_product_DWF_onemonth_nolog	: RCL_list_days_of_month RCL_product_DWF_oneday
RCL_product_DWF_oneyear	: RCL_product_DWF_onemonth
RCL_product_SPL2_oneday	: RCL_get_data_CLUSTA_VTL1_forSPL2 RCL_vectime_L1_to_spectro_L2
RCL_product_SPL2_onemonth	: RCL_product_SPL2_onemonth_nolog
RCL_product_SPL2_onemonth_nolog	: RCL_list_days_of_month RCL_product_SPL2_oneday
RCL_product_SPL2_oneyear	: RCL_product_SPL2_onemonth
RCL_product_visu_orbit_onemonth	: RCL_product_visu_orbit_onemonth_nolog
RCL_product_visu_orbit_onemonth_nolog	: RCL_list_days_of_month RCL_product_visu_orbit_oneday
RCL_product_visu_orbit_oneyear	: RCL_product_visu_orbit_onemonth

RCL_product_visu_SPL2_oneday	: RCL_visu_spectro_4Bz_3H
RCL_product_visu_SPL2_onemonth	: RCL_product_visu_SPL2_onemonth_nolog
RCL_product_visu_SPL2_onemonth_nolog	: RCL_list_days_of_month RCL_product_visu_SPL2_oneday
RCL_product_VTL1_oneday	: RCL_get_data_CLUSTER_WFL1_forVTL1
RCL_product_VTL1_oneday	: RCL_waveform_to_vectime
RCL_product_VTL1_onemonth	: RCL_product_VTL1_onemonth_nolog
RCL_product_VTL1_onemonth_nolog	: RCL_list_days_of_month RCL_product_VTL1_oneday
RCL_product_VTL1_oneyear	: RCL_product_VTL1_onemonth
RCL_product_VTL2_oneday	: RCL_get_data_CLUSTER_VTL1_forVTL2 RCL_vectime_calibration_CLUSTER
RCL_product_VTL2_onemonth	: RCL_product_VTL2_onemonth_nolog
RCL_product_VTL2_onemonth_nolog	: RCL_list_days_of_month RCL_product_VTL2_oneday
RCL_product_VTL2_onemonth_onesat	: RCL_nday_of_month RCL_product_VTL2_oneday
RCL_visu_ave_spectrum	: RCL_version RCL_decode_datiso RCL_get_data_CLUSTER RCL_get_data_CLUSTERGM
RCL_visu_spectro	: RCL_version RCL_decode_datiso RCL_get_data_CLUSTER RCL_get_data_CLUSTERGM
RCL_visu_spectro_3H	: RCL_version RCL_decode_datiso RCL_get_data_CLUSTER RCL_get_data_CLUSTERGM
RCL_visu_spectro_4Bz	: RCL_version RCL_decode_datiso RCL_get_data_CLUSTER RCL_get_data_CLUSTERGM
RCL_visu_spectro_4Bz_3H	: RCL_version RCL_decode_datiso RCL_get_data_CLUSTER RCL_get_data_CLUSTERGM
RCL_visu_vectime	: RCL_version

**Table 47:**      *Commandes RCL utilisant d'autres commandes RCL*

## 7.5. DESCRIPTION DES PROGRAMMES F90

Cette description est obtenue par l'extraction des « cartouches de commentaires » dans les entêtes des programmes, puis remis en forme. Ces cartouches étant à l'origine en anglais, elles ne sont pas traduites ici.

### 7.5.1 Opération sur les fichiers RFF

- ***check\_rff.f90***

Check CLUSTER/STAFF-SC RFF WaveForm and VecTime files.

Input: name of WF or VT file (must be existing).

Used by **RCL\_check\_rff**

- ***clean\_rff.f90***

Read & clean a RFF WaveForm or VecTime files by removing useless comments.

Input: name of WF or VT file (must be existing).

Output: Cleaned WF or VT file with “\_clean” added to the name (will be created).

Used by **RCL\_clean\_rff**

- ***diff\_rff.f90***

Create RFF file containing difference between 2 given rff files. VT1 and VT2 must be of the same type and have same number of data records. VT3 file will be created.

Input: name of VT1 and VT2 files to compare (must be existing).

Output: VT3 whose data field contains difference between data field values of VT1 and VT2.

Used by **RCL\_diff\_rff**

### 7.5.2 Traitements génériques

- ***waveform\_to\_vectime.f90***

Convert a WF RFF file into a VT RFF file.

Input: name of WFL1 input file (must be existing), name of VTL1 output file.

Output: VTL1 file (will be created).

Used by **RCL\_waveform\_to\_vectime**

- ***vectime\_to\_spectro.f90***

Read input RFF file VecTime type, compute spectra and create spectrogram RFF file.

Input: name of input VT file, name of output SP file, Kernel\_size, Nshift, Weighting function (see section 4.3.3).

Output: SP file (will be created).

Used by **RCL\_vectime\_to\_spectro**



- ***reduce\_time\_rff.f90***

Extract a time period into a WF or VT RFF file and create new file containing reduced time period data.

Input: name of file to read (must be existing), name of file to create, start\_time, end\_time, given as ISO date-time format.

Output: rff file containing data corresponding to given time period.

Used by **RCL\_reduce\_time\_rff**

- ***spectro\_to\_polar.f90***

Read a SP.rff file and compute all polarisation parameters. For correct physical interpretation, SP file should be in MFA coordinates.

Input: name of SP file to read (must be existing).

Output: ascii file named copolar.resu, containing all polarisation parameters.

Used by **RCL\_spectro\_to\_polar**

- ***spectro\_xyz\_to\_lrz.f90***

Transform Cartesian XYZ component of a SP filz into circular LRZ componantes.

Input: name of SP file to read (must be existing, with data in xyz Cartesian coordinates), name of file to create.

Output: rff file containing data in circular LRZ componants.

Used by **RCL\_spectro\_xyz\_to\_lrz**

### 7.5.3 Traitements STAFF

- ***vectime\_calibration\_CLUSTER.f90***

Read a Cluster/STAFF VTL1 file and calibrate data with continuous method [see 13, Robert, 2013] and create new VecTime VTL2 file.

Input: name of VTL1 file to read (must be existing), name of VTL2 file to create, path of calibration tables, detrend frequency, cut-off frequency, frequency filter (f1,f2), calibration step, Kernel size, N\_shift, weighting function (see chapter 4.8.1 and 8.3).

Output: VTL2 rff file containing calibrated waveforms.

Used by **RCL\_vectime\_calibration\_CLUSTER**

- ***vectime\_L1\_to\_spectro\_L2.f90***

Read a Cluster/STAFF VTL1 file, compute calibrated spectra (classical method) and create spectrogram SPL2 file.

Input: name of VTL1 file to read (must be existing), name of SPL2 file to create, path of calibration tables, detrend frequency, cut-off frequency, frequency filter (f1,f2), calibration step, Kernel size, N\_shift, weighting function (see chapter 4.8.2 and 8.2).

Output: SPL2 rff file containing calibrated spectra.

Used by **RCL\_vectime\_L1\_to\_spectro\_L2**

- ***vectime\_to\_mfa.f90***

Read a Cluster/STAFF VT file with coordinates in SR2 or ISR2 and transform it in MFA coordinates. Use FGM SPIN data base in cef format.

Input: name of VTL2 file to read (must be existing), name of VTL2 file to create.

Output: VTL2 rff file containing data in MFA system.

Used by **RCL\_vectime\_to\_mfa**

- ***search\_strong\_dc.f90***

From STAFF-SC/SPL2 data base in ISR2 coordinates, where perpendicular DC field is available, search time period where the XY DC field is greater then a given value.

Input: name of SPL2 file to read (must be existing), value of perp. DC min value.

Output: List of time period wher DC > DCmin

Used by **RCL\_search\_strong\_dc**

### 7.5.4 Traitements FGM

- ***fgm\_cef\_gse\_to\_sr2.f90***

Convert a fgm cef file given in gse system into a fgm cef file in sr2 system. Initial data must be in GSE system.

Input: name of fgm.cef file to read (must be existing), right ascension and declination of spin axis in GEI system.

Output: fgm/cef file containing data in SR2 system.

Used by **RCL\_fgm\_cef\_gse\_to\_sr2**

### 7.5.5 Accès aux données d'orbites

- ***get\_data\_CLUPOS.f90***

Calcul les positions du tétraèdre en GSE, GEI ou GSM.

Input: directory of orbit files (must be existing), date (year, month, day), time (hour, min, sec), duration (min), time resolution (mn), coordinates? (only gei, gse or gsm).

Output: ascii file get\_data\_CLUPOS.resu containing position of the 4 S/C.

Used by **RCL\_get\_data\_CLUPOS**

- ***get\_data\_CLUGEOM.f90***

Calcul les paramètres correspondant à la géométrie du tétraèdre (élongation, planarity, inter-distances, surfaces des faces, etc.) sur une journée.

Input: directory of orbit files (must be existing), date (year, month, day), duration (min), time resolution (mn), coordinates? (only gei, gse or gsm), option short (que position et vitesse) ou long (avec toute la géométrie).

Output: ascii file get\_data\_CLUGEOM.resu containing geometric parameter of the tetrahedron.

Used by **RCL\_get\_data\_CLUGEOM**

### 7.5.6 Conversion RFF vers CEF

- ***vectime\_L1\_to\_cef.f90***

Read a VTL1 RFF file and create header of DWF.cef file. Then, data part will be added by RCL\_vectime\_L1\_to\_cef.

Input: VTL1 file name (mut be existing)

Output: cef DWF file with usual name (CAA/CSA convention).

Used by **RCL\_vectime\_L1\_to\_cef**

- ***vectime\_L2\_to\_cef.f90***

Read a VTL2 RFF file and create header of CWF.cef file. Then, data part will be added by RCL\_vectime\_L2\_to\_cef.

Input: VTL2 file name (mut be existing)

Output: cef CWF file with usual name (CAA/CSA convention).

Used by **RCL\_vectime\_L2\_to\_cef**

- ***spectro\_L2\_to\_cef.f90***

Read a SPL2 RFF file and create header of CS.cef file. Then, data part will be added by RCL\_spectro\_L2\_to\_cef

Input: SPL2 file name (mut be existing)

Output: cef CS file with usual name (CAA/CSA convention).

Used by **RCL\_spectro\_L2\_to\_cef**

- ***w\_cal\_caveat\_header.f90***

Create the caveat header file associated to calibration data.

Input: VTL2 file name (mut be existing)

Output: cef header file with usual name (CAA/CSA convention).

Used by **RCL\_vectime\_L2\_to\_cef**

- ***w\_cwf\_cef\_header.f90***

Program to write CWF ceh file, NBR & HBR.

For archive. Is no longer used.

- ***w\_vectime\_cef\_header.f90***

Program to write DWF ceh file, NBR & HBR.

For archive. Is no longer used.

### 7.5.7 Conversions CEF vers RFF

- ***DWF\_cef\_to\_rff.f90***

Convert a DWF cef file to a VTL1 rff file.

Input: path of DWF cef file, path of include\_dirname

Output: VTL1.rff file

Used by **RCL\_DWF\_cef\_to\_rff**

- ***CWF\_cef\_to\_rff.f90***

Convert CWF cef file to a VTL2 rff file.

Input: path of CWF cef file, path of include\_dirname

Output: VTL2.rff file

Used by **RCL\_CWF\_cef\_to\_rff**

- ***fgm\_cef\_to\_rff.f90***

Convert a fgm cef file to a VTL2 rff file.

Input: FGM cef file name (mut be existing)

Output: VTL2.rff file

Used by **RCL\_fgm\_cef\_to\_rff**

### 7.5.8 Divers

- *dir\_properties\_pretty\_tree.f90*

Compute & print properly directory tree and properties.

Input: None. Read the temporary file “dir\_properties\_tree.tmp” created by the RCL command RCL\_dir\_properties\_pretty\_tree (mut be existing).

Output: pretty tree on standard output.

Used by **RCL\_dir\_properties\_pretty\_tree**

- *dir\_size\_pretty\_tree.f90*

Compute & print properly directory tree and size of each.

Input: None. Read the temporary file “dir\_size\_tree.tmp” created by the RCL command RCL\_dir\_size\_pretty\_tree (mut be existing).

Output: pretty tree on standard output.

Used by **RCL\_dir\_size\_pretty\_tree**

- *check\_efw\_status.f90*

Read CLUSTER/EFW RFF WaveForm or VecTime files & check status.

For archive. Is no longer used.

### 7.5.9 Bibliothèques nécessaires à chaque programme F90

La Table 48 ci-dessous indiquent les bibliothèques nécessaires à la création de chaque exécutable. Ces informations sont issues du fichier Makefile.

Commande RCL	: Commande(s) utilisée(s)
check_rff.exe	: lib_rw_rff.o
check_efw_status.exe	: lib_rw_rff.o
clean_rff.exe	: lib_rw_rff.o lib_time.o
diff_rff.exe	: lib_rw_rff.o lib_time.o
DWF_cef_to_rff.exe	: lib_rw_rff.o lib_rw_cef.o lib_utility.o lib_time.o
fgm_cef_to_rff.exe	: lib_rw_rff.o lib_rw_cef.o lib_utility.o lib_time.o
get_data_CLUPOS.exe	: lib_CLUORB.o rocotlib_V2p0.o
CWF_cef_to_rff.exe	: lib_rw_rff.o lib_rw_cef.o lib_utility.o lib_time.o
waveform_to_vectime.exe	: lib_rw_rff.o lib_time.o

w_vectime_cef_header.exe	: lib_rw_cef.o lib_rw_rff.o lib_time.o
w_cwf_cef_header.exe	: lib_rw_cef.o lib_rw_rff.o lib_time.o
w_cal_caveat_header.exe	: lib_rw_cef.o lib_rw_rff.o lib_time.o
vectime_calibration_CLUSTA.exe	: lib_rw_rff.o lib_time.o lib_gainant.o lib_deconvo.o lib_utility.o rocotlib_V2p0.o
vectime_L1_to_spectro_L2.exe	: lib_rw_rff.o lib_time.o lib_gainant.o lib_deconvo.o lib_utility.o rocotlib_V2p0.o
vectime_to_spectro.exe	: lib_rw_rff.o lib_time.o lib_deconvo.o lib_utility.o rocotlib_V2p0.o
vectime_L1_to_cef.exe	: lib_rw_rff.o lib_rw_cef.o lib_time.o
vectime_L2_to_cef.exe	: lib_rw_rff.o lib_rw_cef.o lib_time.o
reduce_time_rff.exe	: lib_rw_rff.o lib_time.o
spectro_L2_to_cef.exe	: lib_rw_rff.o lib_rw_cef.o lib_time.o

**Table 48:**      *Bibliothèques utilisées par chaque programme exécutable F90*

## 7.6. DESCRIPTION DES BIBLIOTHÈQUES F90

Les programmes F90 des commandes RCL utilisent 9 bibliothèques décrites ci-après. Un résumé des modules disponibles pour chacune d'elle est donné ci-après ( Table 49).

lib_deconvo.f90	Modules de déconvolution/calibration des formes d'onde
lib_gainant.f90	Modules pour calculer les gains des antennes
lib_rw_rff.f90	Modules pour lire et écrire des fichiers RFF
lib_rw_cef.f90	Modules pour lire et écrire des fichiers CEF
lib_time.f90	Modules pour gérer les problèmes de temps et de calendrier
lib_utility.f90	Modules utilitaires généraux
lib_CLUORB.f90	Modules pour calculer l'orbite et la position des 4 sat. CLUSTER
lib_CLU_tools.f90	Modules pour calculer toute la géométrie du tétraèdre de CLUSTER
rocotlib_V2p0.f90	Bibliothèque de changement de coordonnées, voir [8, Robert, 2003]

**Table 49 :** *Récapitulation des bibliothèque F90 utilisées*

### 7.6.1 lib\_deconvo.f90

Cette bibliothèque contient toutes les sousroutines de traitement du signal permettant d'effectuer la déconvolution des formes d'onde et donc leur calibration via la connaissance des fonctions de transfert (Table 50).

<b>subrout. casinus(sig,n,fe,fs,as,phi,com)</b>	calcul d'une sinusoïde contenue dans un signal reel	P. Robert, CRPE, 1977-1984
subrout. casinus_D(sig,n,fe,fs,as,phi,com)	calcul d'une sinusoïde contenue dans un signal reel	P. Robert, CRPE, 1977-1984
subrout. desinus(xio,n,fe,fs,amp,pha)	despinage d'une forme d'onde xio (algorithme demod)	P. Robert, CRPE, 1977-1984
subrout. desinus_D(xio,n,fe,fs,amp,pha)	despinage d'une forme d'onde xio (algorithme demod)	P. Robert, CRPE, 1977-1984
subrout. deconvo_R(xio,tra,TFcor,nbp,ix,DSS,DCS,M_Kern)	deconvolution des formes d'onde x y z	P. Robert, CRPE, 1977-1984
subrout. deconvo_C3(Z,TFcor,N,DSS,DCS,M)	deconvolution optimisee par D-FFT et I-FFT incluses	P. Robert, LPP, 2012
subrout. fftpat_XY(X,Y,N,DSS,DCS,M,IND)	fft directe ou inverse sans table de sinus	P. Robert, CRPE, 1984, C/T
subrout. interpo(tab1,mils1,n1,tab2,mils2,n2,mdtmax,iano)	interpolation de tab1, resultat dans tab	P. Robert, LPP, 2001
subrout. lissage(x,nbp,nlis,iano)	lissage d'un tableau sur nlis points	P. Robert, CETP, 2001
subrout. modpha(s,rm,rp)	lissage d'une courbe et soustraction a l'original	P. Robert, CETP, 2003
subrout. test_blk_cont(nor_index,imsres,ierr)	test sur la continuite des blocks	P. Robert, LPP, 2010
subrout. test_blk_cont_no_cal(nor_index,ext_index,imsres,ierr)	test sur la continuite des blocks/saut des calibrations	P. Robert, LPP, 2010
subrout. w_com(pr_out,com)	write on stdo comment only if pr_out is true	P. Robert, LPP, 2010
subrout. trofix(wave,nbp,fe,fspin,rotdeg)	passage d'un repere en rotation a un repere fixe	P. Robert, CRPE, 1977-2008

**Table 50 :** *Liste des sousroutines de traitement du signal de la bibliothèque lib\_deconvo.f90*

### 7.6.2 lib\_gainant.f90

Cette bibliothèque (Table 51) contient toutes les sousroutines permettant de lire les fichiers de calibrations, c'est à dire les fonctions de transfert des instruments, encore appelé gain des antennes, qui sont des données de type complexe (au sens module et phase).

subrout. initcal(ifcod,dirfic)	initialisation du file code et du directory des f.calib	P. Robert, CETP, Sept 2000
subrout. r_califile(pathfil,ncal,freq,Preal,Pimag)	lecture du fichier des gains complexes	P. Robert, CETP, Sept 2000

**Table 51 :** *Liste des sousroutines de la bibliothèque lib\_gainant.f90*

### 7.6.3 lib\_rw\_rff.f90

Cette bibliothèque (Table 52) contient tous les modules, sousroutines et fonctions permettant de définir, lire et écrire des fichiers RFF. Si des paramètres doivent être ajoutés dans les fichiers, ils doivent être définis dans ces codes. Si des paramètres sont absents, ils seront mis à leur valeur par défaut.

module rff_param_def module rff_data_def	modules for RFF Parameter definition modules for RFF indexed data definition	P. Robert , LPP, 2011 Jan. 23 R. Piberne, LPP, 2011 Feb. 17
subrout. rff_allocate_data_arrays subrout. rff_get_current_date(datiso) subrout. rff_set_default_init subrout. rff_set_default_DATA_DESCRIPTION subrout. rff_set_default_BLOCK_DESCRIPTION subrout. rff_set_default_INDEX_DESCRIPTION subrout. rff_set_default_INDEX_EXTENSION_DES subrout. rff_update_history(credate,com) subrout. rff_R_file(ifc,file) subrout. rff_R_manda_param(ifc) subrout. rff_R_optio_param(ifc) subrout. rff_R_const_data(ifc) subrout. rff_R_metadata(ifc,file) subrout. rff_R_indexed_data(ifc) subrout. rff_R_tail(ifc) subrout. rff_W_file(ifc,file) subrout. rff_W_manda_param(ifc) subrout. rff_W_optio_param(ifc) subrout. rff_W_const_data(ifc) subrout. rff_W_metadata(ifc,file) subrout. rff_W_indexed_data(ifc) subrout. rff_W_tail(ifc) subrout. rff_format_W_to_R(format_W,format_R) subrout. rff_format_R_to_W(format_R,format_W)	Allocate data arrays from mandatory parameters donne la date ISO au moment du call set all parameters to zero or 'undefined'(default) Set DATA_DESCRIPTION parameter value to default Set BLOCK_DESCRIPTION parameter value to default Set INDEX_DESCRIPTION parameter value to default Set INDEX_EXTENSION_DESCRIP parameter val to default Update HISTORY parameter in optional_parameters Read rff file, metadata & data, WaveForm or VecTime Read mandatory parameter in the rff file Read optional parameters in the rff file Read constant data in the rff file Open RFF file, read header, metadata & constant data Read indexed data for waveform and vectime RFF files Read tail of a RFF file Write rff file, metadata & data, WaveForm or VecTime Write mandatory parameters Write optional parameters Write constant data Open RFF file, Write header, metadata & constant data Write indexed data for waveform and vectime RFF files Write tail of a RFF file convert a Write format in a Read format convert a Read format in a Write format	R. Piberne, LPP, 2011 Feb. 17 P. Robert , LPP, 2011 Mar. 08 R. Piberne, LPP, 2012 Mar. 08 P. Robert , LPP, 2012 Mar. 28 P. Robert , LPP, 2012 Mar. 28 P. Robert , LPP, 2012 Mar. 28 P. Robert , LPP, 2012 Mar. 28 P. Robert , LPP, 2012 Mar. 28 P. Robert , LPP, 2012 Mar. 28 R. Piberne, LPP, 2011 Feb. 17 P. Robert , LPP, 2011 Jan. 23 P. Robert , LPP, 2011 Jan. 23 P. Robert , LPP, 2011 Jan. 23 R. Piberne, LPP, 2011 Feb. 17 R. Piberne, LPP, 2011 Feb. 17 P. Robert , LPP, 2011 Mar. 08 P. Robert , LPP, 2011 Mar. 09 P. Robert , LPP, 2011 Jan. 23 P. Robert , LPP, 2011 Jan. 23 P. Robert , LPP, 2011 Jan. 23 P. Robert , LPP, 2011 Mar. 08 P. Robert , LPP, 2011 Mar. 09 P. Robert , LPP, 2011 Mar. 08 P. Robert , LPP, 2011 Oct. 15 P. Robert , LPP, 2012 Aug. 30
function get_pos(param,ifc) function get_paramC(param,ifc) function get_paramI(param,ifc) function get_paramI2(param,ifc) function get_paramR(param,ifc) function get_paramD(param,ifc) function get_paramT(param,ifc,nbli) function Fbasename(path) function Fdirname(path) function give_RCL_version()	Read rff file until searched Character parameter Read rff file until searched Character parameter & return it Read rff file until searched Integer parameter & return it Read rff file until searched 2nd Int. parameter & return it Read rff file until searched Real SNGL parameter & return it Read rff file until searched Real DBLE parameter & return it Read rff file until searched Character parameter & return it Return file name of a path Return directory name of a path Return RCL_version	P. Robert , LPP, 2011 Jan. 23 P. Robert , LPP, 2011 Jan. 23 P. Robert , LPP, 2011 Jan. 23 P. Robert , LPP, 2011 Jan. 23 P. Robert , LPP, 2011 Jan. 23 P. Robert , LPP, 2011 Jan. 23 P. Robert , LPP, 2011 Jan. 23 P. Robert , LPP, 2011 Mar. 08 P. Robert , LPP, 2011 Mar. 08 P. Robert , LPP, 2011 Mar. 08 P. Robert , LPP, 2014 Jan.

Table 52 : Liste des sousroutines de la bibliothèque lib\_rw\_rff.f90

### 7.6.4 lib\_rw\_cef.f90

Cette bibliothèque contient tous les modules, sousroutines et fonctions permettant de lire et écrire des fichiers CEF (Table 53).

module type_def_cef	modules for CEF type definitions	L. Mirioni, LPP, 2008
subrout. go_back_to_the_beginning(ifc, param_name) subrout. go_to_keyword(param, ifc) subrout. cef_find_n_include(ifc,n_include) subrout. cef_find_n_metadata(ifc,n_metadata)	Rewind file until START_VARIABLE=param found Read cef file until searched parameter is found Find number of include files in the cef header Find number of metadata in the cef header	R. Piberne, LPP, 2012 Feb. R. Piberne, LPP, 2011 Nov. R. Piberne, LPP, 2011 Nov. R. Piberne, LPP, 2011 Nov.

subrout. cef_find_n_variables(ifc,n_variables) subrout. cef_R_include(ifc,n_include,include_names,include_dirname) subrout. cef_R_metadata(ifc,n_metadata, metadata_names) subrout. cef_R_entry(ifc,cef_entry,n_metadata, metadata_names) subrout. cef_R_variables(ifc,n_variables,variable_names) subrout. cef_R_parameter(ifc,cef_param,n_variables,variable_names) subrout. cef_R_indexed_data(ifc,n_of_vec_in_file, n_block,indexed_data) subrout. concatenate_include_files(ifc,include_dirname,cef_file_name, n_block,block_first_index,block_last_index)  subrout. w_cef_meta(output_file_unit,dummy_cef_meta) subrout. w_cef_parameter(output_file_unit,dummy_var) subrout. u_cef_clear_meta(dummy_cef_meta) subrout. u_cef_clear_var(dummy_var) subrout. suppress_blank(arg_c) subrout. suppress_tab(arg_c)	Find number of variables in the cef headr Read include filenames in the cef file Read mandatory parameter in the cef file Read metadata entry in the cef file Read variable parameter in the cef file Read parameters inside a variable of a cef file Read indexed data for cef files Concatenate cef files to CEF files if necessary  Write Meta Data dummy_cef_meta in output_file Write var Data dummy_var in file output_file_unit. Fills each field of dummy_cef_meta with blanks. Fills each field of dummy_var with blanks supprime les blancs d'une chaine de caractere supprime les tabulations d'une chaine de caractere	R. Piberne, LPP, 2011 Nov. R. Piberne, LPP, 2011 Nov. R. Piberne, LPP, 2011 Nov. R. Piberne, LPP, 2012 Jan. R. Piberne, LPP, 2011 Nov. R. Piberne, LPP, 2011 Nov. R. Piberne, LPP, 2012 March R. Piberne, LPP, 2012 Jan.  L. Mirioni, LPP, 2005, Rev RP 2011 L. Mirioni, LPP, 2005, Rev RP 2011 L. Mirioni, LPP, 2005, Rev RP 2011 L. Mirioni, LPP, 2005, Rev RP 2011 R. Piberne, 2011 R. Piberne, LPP, 2011
function get_cef_paramC(param,ifc,data_reach,end_of_search) function get_cef_paramC_new(param,ifc,data_reach,end_of_search) function u_present_iso_time()	Read cef file until searched Char. param. & return it Read cef file until searched Char. param. & return it Present time in ISO format	P. Robert , LPP, 2011 Jan. R. Piberne, LPP, 2012 Feb. L. Mirioni, LPP, 2008

**Table 53 :** Liste des sousroutines de la bibliothèque lib\_rw\_cef

### 7.6.5 lib\_time.f90

Cette bibliothèque (Table 54) contient toutes les sousroutines permettant de résoudre les problèmes de temps et de calendrier, et de faire de l'opération sur les dates au format ISO '2000-09-13 12:09:56.868055'

subrout. addsec_datiso(datiso1,sec,datiso2) subrout. addsecdbl_datiso(datiso1,secdbl,datiso2) subrout. cdatdoty(idoy,iyear,imonth,iday) subrout. cdatjd50(jd50,iyear,imonth,iday) subrout. clean_datiso(datiso,datpat) subrout. codecsec(nbdy,ih,im,is,ims,imc, decsec) subrout. codecsecdbl(nbdy,ih,im,is,ims,imc, secdbl) subrout. codecsecinv(decsec,nbdy,ih,im,is,ims,imc) subrout. codecsecdblinv(secdbl,nbdy,ih,im,is,ims,imc) subrout. codoty(imonth,iday,iyear,idoty) subrout. coforstr(str, format) subrout. cojd00(iyear,imonth,iday,jd00) subrout. cojd50(iyear,imonth,iday,jd50) subrout. coleapyear(iyear,ileap) subrout. comilday(ih,im,is,ims, milday) subrout. comildayinv(milday,ih,im,is,ims) subrout. compar_datiso(date1,date2,secdif) subrout. decode_datiso(datiso,iyear,imon,iday,ih,im,is,ims,imc) subrout. decode_datisos(datiso,iyear,imon,iday,ih,im,is,imc,decsec) subrout. encode_datiso(iyear,imon,iday,ih,im,is,ims,imc,datiso) subrout. encode_datpat(iyear,imon,iday,ih,im,is,ims,imus,datpat) subrout. gdatetime(iyear,imon,iday,ih,im,is,ims) subrout. gdatiso(datiso) subrout. print_date_time(com) subrout. read_date(prompt,iday,imonth,iyear) subrout. read_time(prompt,ih,im,is)	additionne a la date ISO un temps en sec. additionne a la date ISO un temps en sec. double compute_date_from_day_of_year and for a given year compute_date_from_julian_day_1950 jd50=0 for jan. 1 converti la date iso ex: '2000-09-13 12:09:56.868055' calcule la seconde decimale d'un temps calcule la seconde decimale dble d'un temps convertie la seconde decimale dble en nbdy,ih,im,is,ims convertie la seconde decimale dble en nbdy,ih,im,is,ims compute_day_of_the_year with idoty=1 for jan1 Renvoie le format associe aux chiffres d une string compute_julian_day_2000 with jd00=0 for jan 1, 2000 compute_julian_day_1950 with jd50=0 for jan 1, 1950 compute_leap_year with ileap=1 for leap year, 0 if not calcule la milliseconde du jour convertie la milliseconde du jour en ih,im,is,ims calcule la difference entre 2 dates ISO decode la date ISO en date et heure decode la date ISO en date et heure encode la date ISO en date et heure cre la 'date Patrick' ex: '2000-09-13 12:09:56.868055' donne la date et le temps au moment du call donne la date ISO au moment du call print le CPU time ecoule depuis le dernier appel lit la date depuis l'input et test sa validite lit l'heure depuis l'input et test sa validite	P. Robert, CETP, 2002 P. Robert, CETP, 2002 P. Robert, CRPE, 1992 P. Robert, CRPE, 1992 P. Robert, CETP, 2002 Sep. P. Robert, CETP, 2001,Jan. P. Robert, CETP, 2001 Jan. P. Robert, CETP, 2001 Jan. P. Robert, CETP, 2001 Jan. P. Robert, CRPE, 1993 Jul. J. Ahmad, CETP, 2005 P. Robert, CETP, 2000 Jul. P. Robert, CRPE, 1993 Jul. P. Robert, CRPE, 1992 P. Robert, CETP, 2001 Jan. P. Robert, CETP, 2001 Jan. P. Robert, CETP, 2002 P. Robert, CETP, 2001 P. Robert, CETP, 2001 P. Robert, CETP, 2001 P. Robert, CETP, 2002 Sep. J. Ahmad, CETP, 2006 P. Robert, LPP, 2011 Mar. P. Robert, LPP, 2011 Mar. P. Robert, CRPE, 1993 P. Robert, CRPE, 1993
---	--	---

**Table 54 :** Liste des sousroutines de la bibliothèque lib\_time.f90



### 7.6.6 lib\_CLUSTER.f90

Cette bibliothèque contient les sousroutines nécessaires à la lecture des fichiers d'orbite (.orb) de CLUSTER, et permet de calculer les positions et les vitesses à n'importe quel moment, et dans différents systèmes de coordonnées (Table 55).

subrout. openorbi(dirpath,iyear,imon,iday,ifc,ierr)	open CLUSTER orbit files	P. Robert, CRPE, 1996
subrout. 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
subrout. 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
subrout. closeorbi(ifc)	close CLUSTER orbit files	P. Robert, CRPE, 1996
subrout. cosatpos(dmjd20,ic1,ic2,ic3,ic4,satpos,satvit,satrev)	compute CLUSTER position & velocity in GEI	P. Robert, CRPE, 1996
subrout. corevmoy(satrev,revmoy)	compute CLUSTER mean revolution number	P. Robert, CRPE, 1996
subrout. cvsatpos(satpos)	compute CLUSTER position in Earth radii	P. Robert, CRPE, 1996
subrout. cbaryclus(s,gx,gy,gz)	compute CLUSTER position of barycentre	P. Robert, CRPE, 1996
subrout. transclus(s,gx,gy,gz)	translate CLUSTER position of a Gi vector	P. Robert, CRPE, 1996
subrout. longmot(mot,nc)	calcul la longueur utile d'une chaîne de caractères	P. Robert, CRPE, 1998
subrout. 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
subrout. gei_to_gse(pos)	convert GEI pos. in GSE system	P. Robert, CRPE, 1996
subrout. gei_to_gse_d6(posvit)	convert GEI pos. in GSE system in DBLE	P. Robert, CRPE, 2001
subrout. gei_to_gsm(pos)	convert GEI pos. in GSM system	P. Robert, CRPE, 1996
subrout. magtosr2(x,y,z,nbp,rx,ry,rz)	convert GSE mag waveform in SR2 system	P. Robert, CRPE, 1996
subrout. pantomfa(xfo,yfo,zfo,xma,yma,zma,nbp,rx,ry,rz, &	convert SR2 waveform in MFA coordinates	P. Robert, CRPE, 2001

Table 55 : Liste des sousroutines de la bibliothèque lib\_CLUSTER.f90

### 7.6.7 lib\_CLU\_tools.f90

Cette bibliothèque (Table 56) contient les sousroutines permettant de calculer les paramètres géométriques du tétraèdre de CLUSTER, ainsi que les coordonnées barycentriques permettant le calcul du rotationnel et de la divergence d'un champ de vecteur d'après la méthode de G. Chanteur [voir 15, Chanteur, 1993].

subrout. cgeompara(s)	compute geometric parameters of a given tetrahedron	P. Robert, CRPE, 1994
subrout. cgeomcrit(s, valcrit, titcrit, n)	compute geometric criterions and give their title	P. Robert, CRPE, 1994
subrout. cdistance(s)	compute all the distances of the tetrahedron	P. Robert, CRPE, 1994
subrout. csurfaces	compute all the surfaces of the tetrahedron	P. Robert, CRPE, 1994
subrout. cnormales	compute all the normales of the tetrahedron	P. Robert, CRPE, 1994
subrout. canglefac	compute all the angles of the tetrahedron	P. Robert, CRPE, 1994
subrout. cvolumeto(s)	compute the volume of the tetrahedron	P. Robert, CRPE, 1994
subrout. cspherins(s)	compute radius&volume of sphere including tetrahedron	P. Robert, CRPE, 1994
subrout. setmatrix(tax,tay,taz, tbx,tby,tbz, tcx,tcy,tcz)	set matrices for further linear resolution system	P. Robert, CRPE, 1994
subrout. detmatrix(det)	compute determinant of the matrix defined by setmat	P. Robert, CRPE, 1994
subrout. invmatrix(gx,gy,gz,px,py,pz)	compute inverse matrix determined by setmat and detmat	P. Robert, CRPE, 1994
subrout. cellicrit(s,a,b,c, dirax,gl,gf)	Compute criterions defined by J. Schoenmaker	P. Robert, CRPE, 1994
subrout. quaellips(POS,SAX,DIR)	compute the properties of the ellipse for a tetrahedron	J. Schoenmaekers, ESOC, 1993
subrout. CEIGENVAL(A,R,N,MV)	computes eigenvalues&eigenvectors of real symm. matrix	J. Schoenmaekers, ESOC, 1993
subrout. csomdebd(s,bmag)	computation of J vector inside the 4 S/C tetrahedron	P. Robert, CRPE, 1994
subrout. inipobary(s)	set the initial position of 4 S/C for barycent. coord.	G. Chanteur, P. Robert, CRPE, 1994
subrout. inimabary(bmag)	set the the magnetic field values for barycent. coord.	G. Chanteur, P. Robert, CRPE, 1994
subrout. calbabary(qfbary)	compute barycentric coordinates	G. Chanteur, P. Robert, CRPE, 1994
subrout. estibary	Estimate vectorial quantities of B by barycent. coord.	G. Chanteur, CETP, 1994
subrout. ccurlinco(iface,curx,cury,curz)	compute curl(B) from a given face of the tetrahedron	P. Robert, CRPE, 1994
subrout. ccurlbary(curx,cury,curz)	compute curl(B) from barycentric method	G. Chanteur, P. Robert, CRPE, 1994
subrout. ccurvbary(rnormax,rnormay,rnormaz,rcurv)	compute radius of curvature of field lines	G. Chanteur, CETP, 2003
subrout. cgrabary(gra1x,gra1y,gra1z, gra2x,gra2y,gra2z, &	compute grad(B) matrix from barycentric metho	G. Chanteur, CETP, 1994
subrout. cdivbary(divdeb)	compute div(B) from barycentric method	G. Chanteur, P. Robert, CRPE, 1994
subrout. ctestbary(iok)	compute test on gradient sign from barycentric method	G. Chanteur, P. Robert, CRPE, 1994
subrout. cellipara(s,scalax,vectax)	computation of the inertial ellipsoide	P. Robert, CRPE, 1994

Table 56 : Liste des sousroutines de la bibliothèque lib\_CLU\_tools.f90

### 7.6.8 lib\_utility.f90

Cette bibliothèque contient des utilitaires pratiques...(Table 57).

subrout. basename(path,dir,nd,name,nf)	calcul le nom d'un fichier/directory a partir du path	P. Robert, CETP, 2002
subrout. cbestfor(x,format)	compute_best_format as '(f3.1)' for x=2.5	P. Robert, CETP, 1995, 2001
subrout. cbestfori(ix,format)	compute_best_format_integer as '(i3)' for ix=124	P. Robert, CETP, 1995, 2001
subrout. crandom01(ranval,size)	genere une valeur aleatoire entre 0 et size	P. Robert, CETP, 1994
subrout. randgen71(ranval,idum)	utilitaire de crandom01	P.W.DALY, sept. 1994
subrout. int_to_char(ival,cval,nc)	converti un entier en character, en le cadrant a gauche)	P. Robert, CETP, 2003
subrout. real_to_string(rx,sx,nc)	met un reel en string, avec le meilleurs format	P. Robert, CETP, 2003
subrout. ringbel	ring bell by printing '007' octal character	P. Robert, CETP, 2000
subrout. upper_case(string,nc)	met la string en majuscule si elle ne l'est pas	P. Robert, CETP, 2000
subrout. lower_case(string,nc)	met la string en minuscule si elle ne l'est pas	P. Robert, CETP, 2000
subrout. uencfor(format,ifg,n,nd)	encode un format a partir de ses caracteristiques	P. Robert, CETP, 2000
subrout. udecfor(format,ifg,n,nd)	decode un format en fournissant ses caracteristiques	P. Robert, CETP, 2000

Table 57 : Liste des subroutines de la bibliothèque lib\_utility.f90

### 7.6.9 Rocotlib\_V2p0.f90

La bibliothèque de changement de coordonnée Rocotlib (Robert's Coordinate Transform Library) a été développée initialement pour l'ESA, puis livrée au CDPP. Cette version a été légèrement upgradée pour CLUSTER. Pour plus d'information, on pourra se référer à [8, Robert, 2003]. Cette bibliothèque est divisée en quatre catégories décrites ci-dessous (Table 58).

#### • « Compute » modules

subrout. cangrat(ux,uy,uz,vx,vy,vz,angle,ratio)	compute_angle_and_ratio between U and V vectors	P. Robert, CRPE, 1992
subrout. cdatdoy(idoy,iyear,imonth,iday)	compute_date_from_day_of_year and for a given year	P. Robert, CRPE, 1992
subrout. cdatjd00(jd00,iyear,imonth,iday)	compute_date_from_julian_day_2000 with jd00=0 for jan. 1	P. Robert, CRPE, 1992
subrout. cdatjd50(jd50,iyear,imonth,iday)	compute_date_from_julian_day_1950 with jd50=0 for jan. 1	P. Robert, CRPE, 1992
subrout. cdatwee(iweek,iyear,imonth,iday)	compute_date_for_first_day_of_week_number	P. Robert, CRPE, 2001
subrout. cdipdir(iyear,idoy,d1,d2,d3)	compute_dipole_direction in GEO system	P. Robert, CRPE, 1992
subrout. cdowweek(iyear,imonth,iday,idow)	compute_day_of_the_week	P. Robert, CRPE, 2001
subrout. cdoyear(iyear,imonth,iday,idoy)	compute_day_of_year with idoy=1 for january 1	P. Robert, CRPE, 1992
subrout. cfrdayn(iday,cday,nbcha)	compute_french_day_name, ex: 'Lundi' for iday=1	P. Robert, CRPE, 2001
subrout. cfrmonn(imonth,cmonth,nchar)	compute_French_month_name	P. Robert, CRPE, 2001
subrout. chouday(ih,im,is,houday)	compute_hour_of_day from hours, minutes, seconds	P. Robert, CRPE, 1992
subrout. cjd1950(iyear,imonth,iday,jd50)	compute_julian_day_1950 with jd50=0 for january 1, 1950	P. Robert, CRPE, 1992
subrout. cjd2000(iyear,imonth,iday,jd00)	compute_julian_day_2000 with jd00=0 for january 1, 2000	P. Robert, CRPE, 1992
subrout. cmilday(ih,im,is,ims,milday)	compute_millsec_of_day from hours, minutes, seconds, ms	P. Robert, CRPE, 2001
subrout. cnbdmon(iyear,imonth,nbdays)	compute_number_of_day_of_the_month	P. Robert, CRPE, 2001
subrout. coleapy(iyear,ileap)	compute_leap_year with ileap=1 for leap year, 0 if not	P. Robert, CRPE, 1992
subrout. csundir(iyear,idoy,ih,im,is,gst,slong,sra,sdec,obliq)	compute_sun_direction in GEI system	C.T. Russel 1971, rev. P.Robert,1992,2001-02
subrout. csunset(iyear,imon,iday,rlat,rlon,tmer,tris,tset,durd)	compute_sunset_time and others	P. Robert, CRPE, 2001
subrout. ctimhou(houday,ih,im,is)	compute_time from decimal hour of the day	P. Robert, CRPE, 1992
subrout. ctimmil(milday,ih,im,is,ims)	compute_time from millsec. of the day	P. Robert, CRPE, 2001
subrout. ctimpa2(jd1950,houday)	compute_time_parameters and time-dependent matrix	P. Robert, CRPE, 2001
subrout. ctimpa3(jd2000,houday)	compute_time_parameters and time-dependent matrix	P. Robert, CRPE, 2001
subrout. ctimpar(iyear,imonth,iday,ih,im,is)	compute_time_parameters and time-dependent matrix	P. Robert, CRPE, 1992
subrout. cusdayn(iday,cday,nbcha)	compute_US_day_name, ex: 'Monday' for iday=1	P. Robert, CRPE, 2001
subrout. cusmonn(imonth,cmonth,nchar)	compute_US_month_name	P. Robert, CRPE, 2001
subrout. cweedoy(iyear,imonth,iday,iweek)	compute_week_of_the_year	P. Robert, CRPE, 2001

#### • « Give » modules

subrout. gdipdir(dxgei,dygei,dzgei,dxgeo,dygeo,dzgeo)	give_dipole_direction in GEI and GEO system	P. Robert, CRPE, 1992
subrout. gdiptan(diptan)	give_dipole_tilt_angle in radians	P. Robert, CRPE, 1992
subrout. gecldir(exgei,eygei,ezgei,exgeo,eygeo,ezgeo)	give_ecliptic_direction in GEI and GEO system	P. Robert, CRPE, 1992
subrout. gsrudir(rxgei,rygei,rzgei,rxgeo,rygeo,rzgeo)	give_sun_rotation_direction in GEI and GEO system	P. Robert, CRPE, 1992
subrout. gsundir(sxgei,sygei,szgei,sxgeo,sygeo,szgeo)	give_sun_direction in GEI and GEO system	P. Robert, CRPE, 1992

subrout. gsunpar(gmst,slon,sras,sdec,obli)	give_sun_parameter dependant of time in GEI system	P. Robert, CRPE, 1992
subrout. gvvernum(vernum,verdat)	give_version_number and modification date of the lib.	P. Robert, CRPE, 1992

## • « Print » and « Read » modules

subrout. plibinf	print_library_informations	P. Robert, CRPE, 1992
subrout. recoor(x,y,z,cs)	read coordinate values from input	P. Robert, CRPE, 2002
subrout. recsys(cs)	read coordinate system from input and check validity	P. Robert, CRPE, 2002
subrout. redat(iyear,imonth,iday)	read_date from input and check validity	P. Robert, CRPE, 1992
subrout. retime(ih,im,is)	read_time from input and check validity	P. Robert, CRPE, 1992

## • « Transform » modules

subrout. tcarsph(x,y,z,r,teta,phi)	transforms_car_to_sph: CAR -> SPH	P. Robert, CRPE, 1992
subrout. tdmegeo(xdme,ydme,zdme,rlat,rlong,xgeo,ygeo,zgeo)	transforms_dme_to_geo: DM -> GEO	P. Robert, CRPE, 1992
subrout. tgeigeo(xgei,ygei,zgei,xgeo,ygeo,zgeo)	transforms_gei_to_geo: GEI -> GEO	P. Robert, CRPE, 1992
subrout. tgeigse(xgei,ygei,zgei,xgse,ygse,zgse)	transforms_gei_to_gse: GEI -> GSE	P. Robert, CRPE, 1992
subrout. tgeigsm(xgei,ygei,zgei,xgsm,ygsm,zgsm)	transforms_gei_to_gsm: GEI -> GSM	P. Robert, CRPE, 1992
subrout. tgeigsq(xgei,ygei,zgei,xgsq,ygsq,zgsq)	transforms_gei_to_gsq: GEI -> GSEQ	P. Robert, CRPE, 1992
subrout. tgeimag(xgei,ygei,zgei,xmag,ymag,zmag)	transforms_gei_to_mag: GEI -> MAG	P. Robert, CRPE, 1992
subrout. tgeisma(xgei,ygei,zgei,xsma,ysma,zsma)	transforms_gei_to_sma: GEI -> SM	P. Robert, CRPE, 1992
subrout. tgeodme(xgeo,ygeo,zgeo,rlat,rlong,xdme,ydme,zdme)	transforms_geo_to_dme: GEO -> DM	P. Robert, CRPE, 1992
subrout. tgeoge(xgeo,ygeo,zgeo,xgei,ygei,zgei)	transforms_geo_to_gei: GEO -> GEI	P. Robert, CRPE, 1992
subrout. tgeogse(xgeo,ygeo,zgeo,xgse,ygse,zgse)	transforms_geo_to_gse: GEO -> GSE	P. Robert, CRPE, 1992
subrout. tgeogsm(xgeo,ygeo,zgeo,xgsm,ygsm,zgsm)	transforms_geo_to_gsm: GEO -> GSM	P. Robert, CRPE, 1992
subrout. tgeogsq(xgeo,ygeo,zgeo,xgsq,ygsq,zgsq)	transforms_geo_to_gsq: GEO -> GSEQ	P. Robert, CRPE, 1992
subrout. tgeomag(xgeo,ygeo,zgeo,xmag,ymag,zmag)	transforms_geo_to_mag: GEO -> MAG	P. Robert, CRPE, 1992
subrout. tgeosma(xgeo,ygeo,zgeo,xsma,ysma,zsma)	transforms_geo_to_sma: GEO -> SM	P. Robert, CRPE, 1992
subrout. tgeovdh(xgeo,ygeo,zgeo,rlat,rlong,xvdh,yvdh,zvdh)	transforms_geo_to_vdh: GEO -> VDH	P. Robert, CRPE, 1992
subrout. tgsegei(xgse,ygse,zgse,xgei,ygei,zgei)	transforms_gse_to_gei: GSE -> GEI	P. Robert, CRPE, 1992
subrout. tgsegeo(xgse,ygse,zgse,xgeo,ygeo,zgeo)	transforms_gse_to_geo: GSE -> GEO	P. Robert, CRPE, 1992
subrout. tgsegsm(xgse,ygse,zgse,xgsm,ygsm,zgsm)	transforms_gse_to_gsm: GSE -> GSM	P. Robert, CRPE, 1992
subrout. tgsegsq(xgse,ygse,zgse,xgsq,ygsq,zgsq)	transforms_gse_to_gsq: GSE -> GSEQ	P. Robert, CRPE, 1992
subrout. tgsestr2(xgse,ygse,zgse,rotx,roty,rotz,xsr2,ysr2,zsr2)	transforms_gse_to_sr2: GSE -> SR2	P. Robert, CETP, 2001
subrout. tgsmgei(xgsm,ygsm,zgsm,xgei,ygei,zgei)	transforms_gsm_to_gei: GSM -> GEI	P. Robert, CRPE, 1992
subrout. tgsmgeo(xgsm,ygsm,zgsm,xgeo,ygeo,zgeo)	transforms_gsm_to_geo: GSM -> GEO	P. Robert, CRPE, 1992
subrout. tgsmgse(xgsm,ygsm,zgsm,xgse,ygse,zgse)	transforms_gsm_to_gse: GSM -> GSE	P. Robert, CRPE, 1992
subrout. tgsmgsq(xgsm,ygsm,zgsm,xgsq,ygsq,zgsq)	transforms_gsm_to_gsq: GSM -> GSEQ	P. Robert, CRPE, 2002
subrout. tgsmmag(xgsm,ygsm,zgsm,xmag,ymag,zmag)	transforms_gsm_to_mag: GSM -> MAG	P. Robert, CRPE, 2002
subrout. tgsmisma(xgsm,ygsm,zgsm,xsma,ysma,zsma)	transforms_gsm_to_sma: GSM -> SM	P. Robert, CRPE, 1992
subrout. tgsqgei(xgsq,ygsq,zgsq,xgei,ygei,zgei)	transforms_gsq_to_gei: GSEQ-> GEI	P. Robert, CRPE, 1992
subrout. tgsqgeo(xgsq,ygsq,zgsq,xgeo,ygeo,zgeo)	transforms_gsq_to_geo: GSEQ-> GEO	P. Robert, CRPE, 1992
subrout. tgsqgse(xgsq,ygsq,zgsq,xgse,ygse,zgse)	transforms_gsq_to_gse: GSEQ-> GSE	P. Robert, CRPE, 1992
subrout. tgsqgsm(xgsq,ygsq,zgsq,xgsm,ygsm,zgsm)	transforms_gsq_to_gsm: GSQ -> GSM	P. Robert, CRPE, 2002
subrout. tmaggei(xmag,ymag,zmag,xgei,ygei,zgei)	transforms_mag_to_gei: MAG -> GEI	P. Robert, CRPE, 1992
subrout. tmaggeo(xmag,ymag,zmag,xgeo,ygeo,zgeo)	transforms_mag_to_geo: MAG -> GEO	P. Robert, CRPE, 1992
subrout. tmaggsma(xmag,ymag,zmag,xgsm,ygsm,zgsm)	transforms_mag_to_gsm: MAG -> GSM	P. Robert, CRPE, 2002
subrout. tmagisma(xmag,ymag,zmag,xsma,ysma,zsma)	transforms_mag_to_sma: MAG -> SM	P. Robert, CRPE, 1992
subrout. tsmagei(xsma,ysma,zsma,xgei,ygei,zgei)	transforms_sma_to_gei: SM -> GEI	P. Robert, CRPE, 1992
subrout. tsmageo(xsma,ysma,zsma,xgeo,ygeo,zgeo)	transforms_sma_to_geo: SM -> GEO	P. Robert, CRPE, 1992
subrout. tsmagsm(xsma,ysma,zsma,xgsm,ygsm,zgsm)	transforms_sma_to_gsm: SM -> GSM	P. Robert, CRPE, 1992
subrout. tsmamag(xsma,ysma,zsma,xmag,ymag,zmag)	transforms_sma_to_mag: SM -> MAG	P. Robert, CRPE, 1992
subrout. tsphcar(r,teta,phi,x,y,z)	transforms_sph_to_car: SPH -> CAR	P. Robert, CRPE, 1992
subrout. tsr2gse(xsr2,ysr2,zsr2,rotx,roty,rotz,xgse,ygse,zgse)	transforms_sr2_to_gse: SR2 -> GSE	P. Robert, CETP, 2001
subrout. tsr2mfa(xsr2,ysr2,zsr2,bx,bz,by,rox,roy,roz,xm,ym,zm)	transforms_sr2_to_mfa: SR2 -> MFA	P. Robert, CETP, 2001
subrout. tsr2sre(xsr2,ysr2,spifre,spipha,deltaT,xsre,ysre)	transforms_sr2_to_sre: SR2 -> SRef	P. Robert, CRPE, 2001
subrout. tsresr2(xsre,ysre,spifre,spipha,deltaT,xsr2,ysr2)	transforms_sre_to_sr2: SRef-> SR2	P. Robert, CRPE, 2001
subrout. tvdhgeo(xvdh,yvdh,zvdh,rlat,rlong,xgeo,ygeo,zgeo)	transforms_vdh_to_geo: VDH -> GEO	P. Robert, CRPE, 1992

Table 58 : Liste des sous-routines de la bibliothèque Rocotlib\_2p0.f90

## 7.7. DESCRIPTION DES PROCÉDURES IDL

Le logiciel IDL est utilisé dans le package RCL pour effectuer toutes les visualisations. Les commandes de visualisation (**RCL\_visu\_\***) utilisent ce logiciel pour produire des fichiers graphiques vectoriels en format PostScript, incluant des images comme les spectrogrammes par exemple. Ces procédures de visualisation, initialement développées pour les Roprocs à partir des années 2000, sont utilisées pour produire des visualisations « standards », essentiellement des tracés de vecteurs (en cartésiens et en sphérique) et de spectrogrammes, mais aussi des tracés plus élaborés telles ceux des trajectoires des 4 satellites CLUSTER en 3D.

Pour chaque programme ou bibliothèque, le détail des arguments est donné en dessous d'une brève description. Des exemples de résultats sont donnés en annexe 9.10.

### 7.7.1 Procédures de lecture

- **read\_data\_CLUPOS.pro**

Read file get\_data\_CLUPOS.resu

- **read\_FGM\_bav.pro**

Reading FGM bav files and select given period.

- **read\_fgm\_cef.pro**

Reading FGM cef files and get other characteristic.

- **rff\_read\_SPfile\_4.pro**

Read 4 Spectrogram RFF file.

- **rff\_read\_SPfile.pro**

Read a Spectrogram RFF file.

- **rff\_read\_VTfile.pro**

Read a Vectime RFF file

- **get\_FGM\_freq.pro**

Reading FGM bav files and get characteristic frequencies.

- **r\_copolarlib.pro**

Procédure pour lire les fichiers au format copolar.resu produits initialement par les Roprocs, puis porté dans les RCL dans la commande RCL\_spectro\_to\_polar.

PRO read_fgm_cef, filename, w = w	reading FGM cef files and get other characteristics	R. Piberne, LPP, 2009 Oct 13
PRO read_FGM_bav, datadir,datafile,msdj1,msdj2, isat,iday,imon,i&	reading FGM bav files and select given period	P. Robert , CETP,2001 Mar
PRO rff_read_SPfile, filename, w=w	Read a Spectrogram RFF file	R. Piberne, LPP, 2011 Oct 13
FUNCTION get_param, header, param	Return parameter value of the name inside the header	R. Piberne, LPP, 2011 Oct 13
FUNCTION get_varia, header, varia	Return variable value of the name inside the header	R. Piberne, LPP, 2011 Oct 13
PRO rff_read_SPfile_4, filename_1,filename_2,filename_3,filename&	Read 4 Spectrogram RFF file	P. Robert , LPP, 2013 Feb 07
PRO rff_read_VTfile, filename, w=w	Read a Vectime RFF file	R. Piberne, LPP, 2011 Oct 13
FUNCTION get_param, header, param	Return parameter value of the name inside the header	R. Piberne, LPP, 2011 Oct 13
PRO get_fgm_freq, msdj,fgm_bmod,spin_per,f1,f2,Fci,FLH,Fci_time,&	reading FGM bav files and get characteristic frequenci	P. Robert , LPP, 2013 Feb 18
PRO read_data_CLUPOS, posfile,jul00,iday,imon,iyear,secduj,posvi&	lecture du fichier get_data_CLUPOS.resu	P. Robert , LPP, 2001 May
PRO codij, pos,d12,d13,d14,d23,d24,d34	compute inter-spacecraft distances	P. Robert , LPP, 2001 May

Table 59: Procédures IDL de lecture de fichiers

### 7.7.2 Procédures de visualisations

- **visu\_spectro.pro**

Visualisation d'un spectrogramme.

- **visu\_spectro\_4Bz.pro**

Visualisation de 4 spectrogrammes Bz.

- **visu\_ave\_spectrum.pro**

Visualisation d'un spectre moyen

- **visu\_polar.pro**

Visualisation des paramètres de polarisation des ondes contenus dans le fichier copolar.resu

- **visu\_vectime.pro**

Visualisation d'un fichier type vectime.rff

- **visu\_vectime\_widget.pro**

Management of the event sent by the main widget.

- **visu\_CLUGEOM.pro**

Visualisation d'un fichier d'orbite get\_data\_CLUGEOM.resu

- **visu\_CLUPOS.pro**

Visualisation d'un fichier d'orbite style cresatpos.resu

PRO visu_ave_spectrum, datiso1, datiso2, f1r, f2r, puimin, puimax, \$	MAIN: visualisation d'un spectre moyen	P. Robert, LPP, 2013 Apr
PRO give_sensi_chambon, f_sensi2, p_sensi2	donne la sensibilité des capteurs mesurée à Chambon	P. Robert, LPP, 2013 Apr
PRO visu_spectro, datiso1, datiso2, f1r, f2r, puimin, puimax, \$	visualisation d'un spectrogramme	P. Robert, CETP/LPP 2000-13
PRO visu_spectro_4Bz, datiso1, datiso2, f1r, f2r, puimin, puimax, \$	visualisation de 4 spectrogrammes Bz	P. Robert, LPP, 2013 Jan
PRO visu_vectime, version, w, print = print	visualisation d'un fichier type vectime.rff	P. Robert, CETP/LPP 2002-12
PRO cre_plot, XPS, nomps, fultit, titp1, titp2, jul00, iday, imon, iyear &		
PRO visu_CLUPOS, rcl_version	visualisation d'un fichier d'orbite style cresatpos.resu	P. Robert, CETP, 2001 May
PRO creps, nomps, fultit, jul00, iday, imon, iyear, \$	creation du fichier PS pour le trace 3D des positions	P. Robert, CETP, 2001 Jun
PRO cobowsho, bowsho, nbow	calcul du bow shock	P. Robert, CETP, 2001 May
PRO codij, pos, d12, d13, d14, d23, d24, d34	calcul des inter-distances du tétraèdre	P. Robert, CETP, 2001 May
PRO visu_CLUGEOM, rcl_version	visualisation d'un fichier d'orbite get_data_CLUGEOM.resu	P. Robert, LPP, 2001 May
PRO creps1, nomps, fultit, jul00, iday, imon, iyear, \$	creation du fichier PS pour les positions et vitesses	P. Robert, LPP, 2001 May
PRO creps2, nomps, fultit, jul00, iday, imon, iyear, \$	creation du fichier PS pour la géométrie du tétraèdre	P. Robert, LPP, 2001 May
PRO ajustphi, bgse, nrec	recalcul phi pour éviter les discontinuités de phase	P. Robert, LPP, 2001 Jan
PRO read_data_CLUGEOM, jul00, iday, imon, iyear, secduj, posvit, re	lecture du fichier get_data_CLUGEOM.resu	P. Robert, LPP, 2001 May

**Table 60 :** Procédures IDL de visualisation de fichiers de données

### 7.7.3 Utilitaires

- *rpws\_bepatient.pro*

Create a widget that indicates that a process is running.

- *rpws\_center\_widget.pro*

Center widget base.

- *t3d.pro*

Implement three-dimensional transforms. This routine accumulates one or more sequences of translation, scaling, rotation, perspective, and oblique transformations and stores the result in !P.T variable, the 3D transformation system variable. All the IDL graphic routines use this [4,4] matrix for output. Copyright (c) 1987-2013, Exelis Visual Information Solutions, Inc. All rights reserved.

- *zoom\_x.pro*

Zoom part of a widget.

### 7.7.4 Bibliothèques ou procédures nécessaires à la fabrication des sav d'IDL

La Table 61 ci-dessous montre comment sont fabriqués les .sav à partir d'une licence IDL. Ces instructions sont extraites du fichier Make\_IDL\_sav.bash.

La variable \$R\_IDL est définie dans le fichier RCL\_config.bash, et pointe vers l'application IDL installée sur la machine.

```
echo
echo "-----"
echo "create visu_vectime.sav"

$R_IDL << !!
.FULL_RESET_SESSION
.COMPILE roploplib_201301.pro
.COMPILE julday.pro
.COMPILE rff_read_VTfile.pro
.COMPILE visu_vectime.pro
save, /routines, filename='visu_vectime.sav'
!!

echo 'move visu_vectime.sav in bin directory'
mv visu_vectime.sav ../bin
echo "done..."
```

**Table 61:** Exemple de fabrication d'un .sav d'IDL

La Table 62 ci-dessous liste les procédures nécessaires à chaque application pour générer les .sav correspondant.

SAV	Procédures requises
visu_vectime.sav :	roplotlib_201301.pro julday.pro rff_read_VTfile.pro visu_vectime.pro
visu_spectro.sav :	roplotlib_201301.pro julday.pro rff_read_SPfile.pro read_data_CLUP05.pro read_FGM_bav.pro get_FGM_freq.pro visu_spectro.pro
visu_polar.sav :	roplotlib_201301.pro julday.pro rff_read_SPfile.pro r_copolarlib.pro visu_polar.pro
visu_spectro_4Bz.sav :	roplotlib_201301.pro julday.pro rff_read_SPfile.pro rff_read_SPfile_4.pro read_data_CLUP05.pro read_FGM_bav.pro get_FGM_freq.pro caldat.pro visu_spectro_4Bz.pro
visu_ave_spectrum.sav :	roplotlib_201301.pro julday.pro rff_read_SPfile.pro read_data_CLUP05.pro read_FGM_bav.pro get_FGM_freq.pro visu_ave_spectrum.pro
visu_CLUP05.sav :	roplotlib_201301.pro julday.pro read_data_CLUP05.pro t3d.pro visu_CLUP05.pro
visu_CLUGEOM.sav :	roplotlib_201301.pro julday.pro visu_CLUGEOM.pro
visu_vectime_widget.sav :	roplotlib_201301.pro julday.pro rpws_bepatient.pro rff_read_VTfile.pro visu_vectime.pro zoom_x.pro rpws_center_widget.pro visu_vectime_widget.pro

**Table 62:** Liste des procédures IDL nécessaire à la fabrication des .SAV.



## 7.8. DESCRIPTION DE LA BIBLIOTHÈQUES IDL ROPLTLIB

La roplotlib.pro dans sa version 201301 est une suite de procédures IDL, écrites dans les années 2000 pour les Roprocs, et qui ont pour but de faciliter la création de plots, avec des graduations horaires, une présentation harmonisée, des mires de couleurs, etc. Elle est principalement utilisée pour produire des visualisations « standards », sous forme de fichiers PostScript, comme expliqué au chapitre précédent. La Table 63 ci-dessous résume les procédures disponibles.

### • Procédures

PRO addsec_datiso, datiso1,sec,datiso2	additionne a la date ISO un temps en sec.	P. Robert, CETP, 2002
PRO gap_detection, time,dth, time2	gap detection in a regular time series	P. Robert, LPP, 2012 May
PRO gap_correction, time2, pui1,pui12	gap correction for computing integrated power	P. Robert, LPP, 2012 May
PRO ajustphideg, phi,nbval	recalcule phi pour éviter les discontinuités de phase	P. Robert, CETP, 2001 Jan
PRO ajustzero, temps,tempsb,nx,t1,t2,routine	ajuste l'origine des temps/arrondi t1 & t2 pour visu 3h	P. Robert, CETP, 2000 Oct
PRO basename, path, dir,nd, name,nf	calcul les noms fichier & directory a partir du path	P. Robert, CETP, 2001 Jan
PRO cadre, tit1,chas1,tit2,chas2,xtvt,xpage,ypage,inoir,ijaune,&	trace du cadre de la page, avec bandau jaune d'entete,	P. Robert, CETP, 2000 Avr
PRO calima_mima, spectro,semin,semax,ifr, rmin,rmax,nbzero,ilog	calcul des min et max de la future image	P. Robert, CETP, 2000 Sep
PRO calimage, spectro,image,vmin,vmax,icr,nbzero,semin,semax,ifr	calcul de l'image correspondant au tableau spectro	P. Robert, CETP, 2000 Sep
PRO calpuiint,spectrog,f1ulf,f2ulf,df,nx,ny,pui1	calcul de la puissance integree du spectrogramme	P. Robert, CETP, 2002 May
PRO calcyl, vec3,vec5,nbp	extension d'un vecteur vec3(3,nbp) a ses comp. perp.	P. Robert, CETP, 2002 Nov
PRO calsphe, vec3,vec6,nbp	extension d'un vecteur vec3(3,nbp) a ses comp. spher.	P. Robert, CETP, 2002 Oct
PRO calt1t2, temps, nx, dth, t1, t2	calcul et arrondi t1 et t2 pour la visu	P. Robert, CETP, 2000 Oct
PRO caltemps, nbp,dt,ih1,im1,is1,ims1, temps,xtit,t1,t2,cstatim,&	calcul du temps et du titre de l'axe.	P. Robert, CETP, 2000 Aug
PRO caltempstrond, nbp,dt,ih1,im1,is1,ims1, temps,xtit,t1,t2,csta&	calcul du temps et du titre de l'axe, debut arrondi	P. Robert, CETP, 2000 Aug
PRO cfleche, tabx,taby,n,flex,fley,ltete,lqueue	calcule une fleche 2D a partir des tab. tabx, taby	P. Robert, CETP, 2002 Mar
PRO cfleche3D, tabx,taby,tabs,n,flex,fley,flez,ltete,lqueue	calcule une fleche 3D a partir des tabs tabx,taby,tabs	P. Robert, CETP, 2002 Mar
PRO codatec, iday,imon,year, datec	converti la date en variable character	P. Robert, CETP, 2001 Fev
PRO codatec3, iday,imon,year, datec	converti la date en variable character 3 carac.	P. Robert, CETP, 2001 Fev
PRO codechour, ih,im,is,ims, dechour	compute decimal hour from hour,min,sec	P. Robert, CETP, 2001 Feb
PRO cogratim, t1, t2, xtickval, xtickgra, nbxti, nbxmi	calcul des graduations de l'axe en HH:MM:SS	P. Robert, CETP, 2000 Oct
PRO cohms, dechour, ih,im,is,ims	compute hour,min,sec from decimal hour	P. Robert, CETP, 2001 Feb
PRO cohms_s, isec, id,ih,im,is	compute day,hour,min,sec from isec (LONG integer)	P. Robert, CETP, 2002 Mar
PRO cojul00, iyear,imon,iday, jul00	calcul du jour julien 2000 (1/1/2000 = Jour 0)	P. Robert, CETP, 2000 Sep
PRO cojul50, iyear,imon,iday, jul50	calcul du jour julien 1950 (1/1/1950 = Jour 0)	P. Robert, CETP, 2000 Sep
PRO defpalrgb8	definition d'une palette RGB a 8 couleurs	P. Robert, CETP, 1998 Dec
PRO defpalhsv256, inoir,iblan,ijaunp, rouge,ivert,ibleu,ijaun&	definition d'une mire HSV 256 couleurs, avec coul. spec.	P. Robert, CETP, 2000 Avr
PRO softinfo, soft, arch,os,release	permet de connaître la version du logiciel	P. Robert, CETP, 2001 Fev
PRO misenpag, nbpan,nbcupan,ypage,maryh,maryb, ry,oy,dyc,dyp	mise en page d'un groupe de courbes dans une page	P. Robert, CETP, 2001 Jan
PRO normascale, xmin,xmax,ymin,ymax,zmin,zmax,dmax	normalisation des echelles pour que dx,dy,dz = dmax	P. Robert, CETP, 2002 Feb
PRO normascale2, delta, xmin,xmax,ymin,ymax,zmin,zmax	normalisation des echelles pour que dx,dy,dz = delta	P. Robert, CETP, 2012 Oct
PRO crepagnu, numps,fultit,jul00,iday,imon,year,signat,\$	ouverture du fichier .ps + cadre, titre et signature,	P. Robert, CETP, 2001 Jun
PRO crepagnu2, fultit,jul00,iday,imon,year,signat,\$	memme chose que crepagnu, mais n'ouvre pas le fichier	P. Robert, CETP, 2001 Jun
PRO crepagps, numps,fultit,jul00,iday,imon,year,t1,t2,signat,\$	ouverture du fichier .ps, mise en page panels standards,	P. Robert, CETP, 2001 Avr
PRO decode_datiso, datiso,iyear,imon,iday,ih,im,is,ims,imc	decode la date ISO en date et heure	P. Robert, CETP, 2001
PRO encode_datiso,iyear,imon,iday,ih,im,is,ims,imc,datiso	encode la date ISO en date et heure	P. Robert, CETP, 2001
PRO ploimagra, image, x1,y1,xs,ys,t1,t2,f1,f2,labt,labf, \$	plot d'une image avec graduation des axes,	P. Robert, CETP, 2000 Oct
PRO plomir253, rmin1,rmax1,rmin2,rmax2,x1m,y1m,xsm,ysm,chasim	plot d'une mire a 253 couleurs, definie sur 256 couleurs	P. Robert, CETP, 2000 Oct
PRO plotcou,xdat,ydat,t1,t2,ymi,yma,tit,xtit,ytit,x1,x2,y1,y2,y&	plot d'une courbe et de ses graduations.	P. Robert, CETP, 2001 Jan
PRO plot_cercle, x0,y0,radius,icol,ifill, thick=thick	trace d'un cercle sur un plot deja defini	P. Robert, CETP, 2002 Mar
PRO plot_arc_cercle, x0,y0,radius,tet1,tet2,icol,ifill,thick=thi&	trace d'un arc de cercle sur un plot deja defini:	P. Robert, CETP, 2002 Mar
PRO plot_sphere, x0,y0,z0,radius,icol	trace d'une sphere sur un plot 3D deja defini	P. Robert, CETP, 2002 Mar
PRO plot_sphere_part, tet1,tet2,phi1,phi2,x0,y0,z0,radius,icol	trace d'une partie de sphere sur un plot 3D	P. Robert, CETP, 2002 Dec
PRO roundt1t2, t1, t2, dth	arrondi t1 et t2 a une minute ronde si differences < dth	P. Robert, CETP, 2000 Oct
PRO ppagarr,orix,oriy,ipos,bodyl,bodyw,headl,headw,angle	plot sur la page une fleche completement definie	P. Robert, PatCie, :1
PRO round_symbol, fil	symbole en cercle a utiliser avec pymbol=8	P. Robert, CETP, 2001



• **Fonctions**

FUNCTION datim_iso	return current date/time as 2001-08-30T14:55:37.000	P. Robert, LPP, 2011 Sep
FUNCTION datim	return current date/time as 2001 Aug 30, 14:55:37	P. Robert, LPP, 2011 Aug
FUNCTION give_signat, projet, appli, noms, version	retourne la signature pour les Roplots	P. Robert, LPP, 2011 Aug
FUNCTION give_roplotlib_version	retourne la version de la roplotlib	P. Robert, LPP, 2011 Aug
FUNCTION interpol_image, spectro, time, dt, Nx_ima, t1, t2	Return interpolated image, from R. Piberne algorithm	P. Robert, LPP, 2012 May
FUNCTION logticks_exp, axis, index, value	Function for log axis tickmarks with exponential output	P.V. Delst, CIMSS, 2000 Nov

**Table 63 :**     *Procédures et fonctions disponibles dans la bibliothèque roplotlib.pro*



## 8. LA CALIBRATION DES SPECTRES ET DES FORMES D'ONDE

### 8.1. INTRODUCTION

La calibration des données issues d'un search-coils, dont la fonction de transfert n'est pas linéaire en fréquence, est un problème ancien datant des premières expériences de ce type (voir [20], Robert, 1971). La calibration des spectres, si le principe semble simple (correction des amplitudes de chaque raie du spectre par la valeur de la fonction de transfert à cette fréquence), la mise en pratique est plus délicate du fait que les capteurs magnétiques sont dans un repère en rotation (satellite spiné) dont la fréquence est incluse dans la bande de fréquence de l'instrument. De plus ces capteurs tournent dans un champ continu de 10 à 1000 fois plus grand que l'amplitude des ondes à détecter.

La production de spectres calibrés est faite de manière discontinue, fenêtre après fenêtre, comme expliqué ci-dessous. Elle conduit à la production de fichiers journaliers de type SPL2.rff ou CS.ccf.

La calibration d'une forme d'onde « en continue » est elle beaucoup plus délicate, comme il est expliqué au chapitre 8.3.

Ces calibrations ont été validées par des comparaisons fines avec les données du magnétomètre FGM, durant de nombreux « cross-calibration meeting » [comme 11, Robert, 2009], résumées dans le « Calibration Report » du CSA [19, Robert 2012] ainsi que dans l'article [13, Robert, 2013]. On rappelle ici de manière détaillée la calibration des spectres et des formes d'onde.

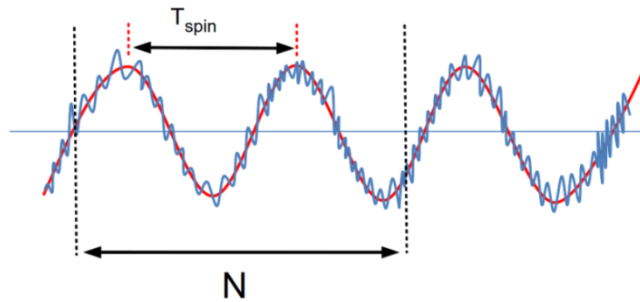
### 8.2. CALIBRATION DES SPECTRES

#### 8.2.1 Théorie

En théorie la calibration des spectres se résume à une FFT complexe d'une portion de forme d'onde (fenêtre de calibration), que l'on divise par la valeur complexe de la fonction de transfert de search-coils, ceci pour chaque fréquence, et composante par composante, dans le repère tournant des capteurs.

En théorie, c'est facile. En pratique il faut prendre certaines précautions.

Dans la pratique, le signal original  $x_{k(Volt)}$  est constitué d'un grand signal sinusoïdal à la fréquence de spin  $f_{spin} \sim 0,25$  Hz (ou avec une période  $T_{spin} \sim 4$  s), dû à la rotation des capteurs dans un champ continu élevé ( $\sim 10$  à  $1000$  nT), alors que les fluctuations qui se superposent à cette sinusoïde « porteuse » sont de l'ordre de 10 à 1000 fois plus faibles (quelques nT). La Figure 12 ci-dessous illustre ce principe.



**Figure 12 :** *Signal utile superposé à la sinusoïde de spin pour un search-coil en rotation*

En outre, une transformée de Fourier sur une période  $N$  suppose un signal périodique. Or, la troncature de la sinusoïde sur les bords de la fenêtre introduit de grandes discontinuités, ce qui génère des composantes hautes fréquences dans le spectre, qui n'ont pas de significations physiques [voir 20, Robert, 1971 et 19, Robert 2012].

### 8.2.2 Les étapes successives de la calibration de la forme d'onde

Il faut au minimum quatre étapes de traitement pour obtenir un spectre calibré dans un repère fixe « dé-Dopplerisé » de la rotation du satellite.

#### • Étape 1 : conversion en Volt

Les données de télémétrie sont des « counts » en entiers, codés sur 16 bits et donc compris dans l'intervalle  $[0 - 65535]$ . La dynamique étant de 10 V et l'offset de 5V (valeurs écrites dans le fichier VTL1.rff à calibrer), le passage des coups de télémétrie en Volt se fait par la règle de trois :

$$X_i(t_n)_V = X_i(t_n)_c \cdot \text{dyna} - \text{offset}$$

Dans le programme de calibration, cela se traduit par les instructions simples suivantes:

```
proc_vec(1,:) = proc_vec(1,:)*dyna -offset
proc_vec(2,:) = proc_vec(2,:)*dyna -offset
proc_vec(3,:) = proc_vec(3,:)*dyna -offset
```

#### • Étape 2 : despin

Ainsi, la première étape consiste à enlever cette grande sinusoïde avec une sous-routine spécialisée qui calcule l'amplitude et la phase de cette sinusoïde (dont on doit connaître la fréquence, mais qui figure dans les VTL1) et qui la supprime (voir § 7.3.1, routine desinus.f90 et desinus\_D.f90). Cette étape se traduit dans le code par les instructions suivantes ( $N\_Kern$  est la taille de la fenêtre de calibration):

```
call desinus(proc_vec(1,:),N_Kern,sam_rate,spin_rate,mod_d1,pha_d1)
call desinus(proc_vec(2,:),N_Kern,sam_rate,spin_rate,mod_d2,pha_d2)
call desinus(proc_vec(3,:),N_Kern,sam_rate,spin_rate,mod_d3,pha_d3)
```

Dans cette étape, on applique aussi éventuellement la procédure de « detrend » expliquée au chapitre 8.2.4 qui se traduit dans le code par les instructions :

```
do i=1,3
!   trend computation
  RA1D_tmp = proc_vec(i,:)
  call lissage(RA1D_tmp,N_Kern,nbp_liss,ierr)
  if (ierr /= 0) then
    RA1D_tmp = 0.
  endif
!   subtract trend to the original signal
  proc_vec(i,:) = proc_vec(i,:) - RA1D_tmp
end do
```

• **Complément à l'étape 2: calcul du champ DC dans le plan de spin**

Lorsque on applique la procédure de despin par la routine `desinus` on récupère en sortie l'amplitude en Volt et la phase de la sinusoïde de spin pour chaque composante, soit  $\tilde{B}_x, \tilde{\varphi}_x$  et  $\tilde{B}_y, \tilde{\varphi}_y$  ainsi que celle de la composante  $B_z, \tilde{B}_z, \tilde{\varphi}_z$  qui, bien que faible, permettra le calcul de l'angle de dépointage (voir paragraphe suivant). Les valeurs  $\tilde{B}_x$  et  $\tilde{B}_y$  sont deux estimations différentes du champ perpendiculaire  $B_\perp$ , en valeur non calibrées (en V), de même que  $\tilde{\varphi}_x$  et  $\tilde{\varphi}_y$  sont deux estimations de la phase  $\varphi$  dans le repère tournant SR.

L'annexe 9.4 donne le calcul complet du champ DC dans le plan de spin, en repère tournant SR2 comme en repère fixe SR2. Les résultats sont rappelés ci-dessous.

- *Amplitude et phase de  $B_\perp$  en repère tournant SR :*

A partir des deux estimations  $\tilde{B}_x$  et  $\tilde{B}_y$  de l'amplitude (en V) on peut calculer deux estimations  $B_{\perp x}$  et  $B_{\perp y}$  de l'amplitude en nT par les formules :

$$B_{\perp x} = \tilde{B}_x \frac{1}{|\alpha_x(f_s)|} \quad B_{\perp y} = \tilde{B}_y \frac{1}{|\alpha_y(f_s)|}$$

Où  $|\alpha_x(f_s)|$  et  $|\alpha_y(f_s)|$  sont les modules de la fonction de transfert à la fréquence de spin pour chacune des deux composantes X et Y. Il est naturel de prendre comme valeur de  $B_\perp$  la valeur moyenne:

$$B_\perp = \frac{1}{2} (B_{\perp x} + B_{\perp y})$$

De même on obtient deux estimations différentes de l'angle  $\varphi$  en corrigeant les phases mesurée par la fonction de transfert :

$$\varphi_x = \tilde{\varphi}_x - \text{phase}[\alpha_x(f_s)]$$

$$\varphi_y = \tilde{\varphi}_y - \text{phase}[\alpha_y(f_s)]$$

Pour la phase on décide de prendre comme phase de référence  $\varphi_x$  (ce choix est arbitraire, on aurait pu prendre  $\varphi_y$ ).

La partie correspondante du code pour l'estimation du  $B_\perp$  en repère SR est la suivante :

```
! 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 :*

En principe Bx et By devraient être égaux, même si les fonctions de transfert sont légèrement différentes. De même la différence entre  $\varphi_x$  et  $\varphi_y$  devrait être de 90°.

On peut donc effectuer un test pour voir si ces hypothèses sont vraies ou non. La mesure des différences permettra d'estimer le « misalignment angle » (voir 9.3.9).

Les tests sont juste fait par un print des valeurs du test :

```
! 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 per3. 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 et phase de  $B_{\perp}$  en repère tournant SR2 :*

Pour obtenir les valeurs Bx et By du champ DC dans le repère SR2, il suffit de faire une rotation d'angle  $\psi$ . Le calcul de cet angle est fait dans le chapitre 9.4 et on a :

$\psi = (\omega_s t + \varphi_s + \alpha)$ . (Pour plus de détail, voir aussi l'étape 4 ci-après).

Les valeurs retenues pour le champ continu dans le plan de spin sont alors :

$$B_x^{DC} = B_{\perp} \sin(\varphi_x + \psi)$$

$$B_y^{DC} = B_{\perp} \cos(\varphi_x + \psi)$$

Par cette méthode, on a préservé le fait que l'amplitude est la même sur les deux composantes, et que celles-ci sont déphasées de 90°.

Et un peu plus loin le calcul des composantes BDCx et BDCy en 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
```

*• Autre complément à l'étape 2: calcul du misalignment angle*

Le “misalignment angle” est l'angle (faible) entre le véritable axe de rotation du satellite et l'antenne Z du search-coils. Le calcul complet est donné au chapitre 9.4. Sa valeur est déduite des amplitudes et phase de la sinusoïde de spin calculée sur les 3 composantes. On a:

$$\theta_z = \sin^{-1} \left( \frac{a_z \sqrt{2}}{\sqrt{a_x^2 + a_y^2 + a_z^2}} \right)$$

Ce calcul est fait toujours à l'étape 2, au moyen des lignes de codes suivantes:

```
! 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
```

### • Étape 3 : calibration en repère tournant

La seconde étape consiste d'abord à appliquer une fonction de pondération sur la fenêtre, avant la FFT, afin d'éviter des discontinuités résiduelles sur les bords. Ceci est fait après centrage du signal sur zéro bien sûr.

La fonction de pondération  $W_k$  doit conserver la forme générale du signal mais aussi faire en sorte que les bords tombent à zéro. Par expérience, le choix d'un très long trapèze fonctionne bien. Les bords du trapèze peuvent aussi être légèrement arrondis comme représenté sur la Figure 13 ci-dessous.

Il est possible d'appliquer d'autres fenêtres de pondération, comme on le verra au paragraphe 8.2.3.

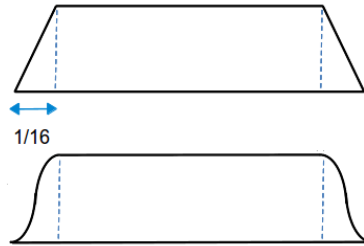


Figure 13 : apodisation en trapèze du signal pour la calibration d'un spectre

Ensuite, on peut voir qu'une estimation du spectre calibré (en nT) est obtenue par :

$$\tilde{X}_n = \frac{1}{\alpha_n} \frac{1}{N} \sum_{-N/2}^{N/2} x_k W_k e^{-2i\pi nk}$$

Où  $\alpha_n$  est la fonction de transfert de l'antenne, définie sur toute la gamme de fréquence. Dans la programmation de cette formule, on utilise généralement une FFT de bibliothèque, ou celle écrite dans les RCL (voir § 7.3.1, routine `fftpat_XY`). Dans la mesure où le signal est réel, son spectre est symétrique, c'est-à-dire que la partie des fréquences négatives du spectre est le conjugué de la partie des fréquences positives. Si on ne compte pas faire de FFT inverse ensuite, on peut se contenter des fréquences positives pour appliquer la correction de la fonction de transfert et donc prendre simplement les premiers  $N/2$  points du spectre complexe. Mais dans ce cas il ne faudra pas oublier un facteur 2 dans le calcul de la densité spectrale de puissance (DSP ou PSD en anglais), qui s'exprime comme suit :

$$DSP_n(nT^2/Hz) = \frac{2(\tilde{X}_n)^2}{\delta f}$$

Par contre, si on envisage ensuite de faire une FFT inverse, il faut bien faire attention de prendre toutes les fréquences quand on applique la correction de la fonction de transfert, et donc de la définir sur les fréquences négatives, à partir de la partie des fréquences positives conjuguées. Pour plus de détail, on pourra se référer au chapitre 9.7.

*En pratique ce sers toujours le cas*, car le spectre obtenu ci-dessus est en repère « SR », donc dans le repère tournant des antennes, ce qui n'a guère d'intérêt sauf pour faire des tests. Or, pour faire le changement de repère pour passer en « SR2 », encore appelé « Despun », il est nécessaire de repasser en forme d'onde.

Dans le programme de calibration des spectres, la déconvolution est faite de la manière suivante

```
!   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=cplx(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
```



• **Étape 4 : passage en SR2**

Cette étape se fait sur la forme d'onde calibrée en repère tournant SCS des antennes (Sensor Coordinate System, voir § 8.4.1). Ces formes d'onde calibrées par séquence de N points (avec l'apodisation sur les bords) sont obtenues à partir du spectre calibré ci-dessus après une FFT inverse (sur toutes les fréquences, négatives comme positives). Au chapitre 9.3.8, on verra qu'on peut négliger la matrice de dépointage, et confondre le repère SCS avec le repère SRS (Spin Reference System).

Dans ce cas le produit des diverses matrices de changement de coordonnées se réduit à une simple rotation dans le plan de spin de l'angle  $\psi$  (voir § 8.4.8). Comme précédemment on a :

$$\psi = (\omega_s t + \varphi_s + \alpha).$$

Dans cette formule,  $\omega_s$  vaut  $2\pi f_s$  où  $f_s$  est la fréquence de spin,  $t$  le temps mesuré depuis le début de la fenêtre, soit  $t = (n - 1)\delta t$  avec  $\delta t = 1/fe$  où  $fe$  est la fréquence d'échantillonnage du satellite considéré (~25. ou ~450. Hz suivant le mode NBR ou HBR).  $\varphi_s$  est le « spin phase angle » lu dans chaque bloc du fichier VTL1. On verra aussi au § 8.4.4 que  $\alpha = 45^\circ$ .

La transformation de SR à SR2 dans le plan de spin est donc :

$$\begin{cases} X_{SR2} = X_{SR} \cos \psi - Y_{SR} \sin \psi \\ Y_{SR2} = X_{SR} \sin \psi + Y_{SR} \cos \psi \end{cases}$$

Dans le code, cette étape correspond aux instructions suivantes :

```
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),mnda_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
```

• **Étape 5 : rajout du champ DC sur X et Y**

On a vu à l'étape 2 que l'on pouvait calculer les deux composantes Bx et By du champ DC dans le plan de spin en repère SR2. Dans cette étape, on rajoute donc ces deux valeurs aux valeurs de forme d'onde simplement par le code suivant :

```
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
```

• **Étape 6: mode de contrôle pour l'étude du DC<sub>⊥</sub>**

Pour ce mode particulier, les étapes 3, 4 et 5 ont été sautées. On ne conserve qu'un seul point par fenêtre, qui contient le champ DC sur x et y, et le misalignement angle sur Z. La portion de code correspondante est la suivante :

```
!   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
```

• **Étape 7: Calcul en ISR2**

C'est la même étape que l'étape 4, mais le résultat est en ISR2, et non plus en SR2. La portion de code correspondante est la suivante :

```
!   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
```

• **Étape 8: Calcul en GSE**

C'est la même étape que l'étape 4, mais les formes d'onde sont calculées en GSE. La portion de code correspondante est la suivante :

```
if (spin_phase >= -360.) then
!   GSE spin axis is supposed to not change during a spectra (a few 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,sygsse,szgsse
    endif

    do i= i1,i2
!      transform data in GSE
        call tsr2gse(proc_vec(1,i),proc_vec(2,i),proc_vec(3,i),sxgse,sygsse,szgsse,&
                    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
endif

```

### 8.2.3 Calcul du spectre complexe

Suivant le niveau de calibration choisie, on a à présent une forme d'onde calibrée (pour les étapes  $\geq 3$ ), dans le repère qu'on a choisi, et sur la période  $T$  correspondant à la durée de la fenêtre de calibration (les  $N\_Kern$  points). Il faut à présent en calculer le spectre complexe.

Pour ce faire, on commence par centrer le signal par soustraction de la valeur moyenne (indispensable pour faire une aposisation correcte), puis on le multiplie par la fonction d'apodisation choisie. On peut ensuite passer dans le domaine fréquence par une FFT améliorée ou les tables de sinus ont été pré-calculées afin de gagner du temps CPU. La valeur du continue, soustraite avant le calcul de la FFT, est ensuite rajoutée dans la fréquence zéro. Enfin, on normalise le spectre pour que l'énergie soit conservée.

Cette opération est faite à l'aide de la portion de code suivante :

```

!      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 pour conservation de la puissance (on a pris que 1/2 spectre)
!      rappel: pui_w est l'amplitude efficace de la fenetre
    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

```

Le spectre est ensuite écrit dans un fichier temporaire, avant transfert dans le champ « INDEXED\_DATA » du fichier RFF au moyen des instructions suivantes :

```
!   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
```

#### 8.2.4 Paramètres de la procédure RCL\_vectime\_L1\_to\_spectro\_L2

##### • Rappel

Cette procédure prend donc en entrée les formes d'onde de niveau L1 (non calibrées) et produit un spectrogramme calibré, c'est-à-dire une suite de spectres calibrés selon la méthode décrite ci-dessus. On a vu au chapitre 4.8.1 que les paramètres de cette procédure étaient les suivants :

Usage : 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)

Certaines précautions sont bien sûr à prendre en ce qui concerne les paramètres à appliquer, en fonction de ce que l'on recherche et du type de signal à l'entrée.

### • Nom des fichiers

Le nom des fichiers VTL1.rff et SPL2.rff sont libres, du moment qu'ils aient le suffixe rff. On peut avoir par exemple pour le VTL1 un nom du genre :

CLU3\_STASC\_VTL1\_NBR\_20021009\_03-12.rff

Par souci de cohérence, il est de bon ton de nommer le SPL2 d'une manière analogue, par exemple :

CLU3\_STASC\_SPL2\_NBR\_20021009\_03-12.rff

Le fichier VTL1 doit être de la classe VecTime et contenir les méta-data nécessaires (voir un exemple au chapitre 9.9.2). Le fichier SPL2 sera généré en préservant ces méta-data, et en modifiant ou en ajoutant celles qui sont nécessaires. Comme toutes les procédures RCL, le paramètre HISTORY est complété.

### • Fréquence de « detrend »

Le « detrend » est une méthode de lissage des formes d'onde, permettant d'enlever les basses fréquences. Le principe est le suivant :

- On effectue un lissage sur Nlis points d'une forme d'onde de Nbp points. Chaque point de la forme d'onde lissée prends la valeur moyenne des Nlis/2 points de chaque coté. Quand la moyenne sur la fenêtre de Nlis point est faite, on décale cette fenêtre d'un point et on recommence. L'ensemble de ces valeurs moyennes, que l'on peut appeler forme d'onde lissée, représente la « tendance » du signal, c'est-à-dire son évolution moyenne sur la séquence considérée. Elle ne contient plus aucune fréquences supérieures à  $F_{det} = Nlis / Fe$ , où  $Fe$  est la fréquence d'échantillonnage. Il est bien évident qu'on doit avoir  $Nlis < Nbp/2$ .
- On soustrait la forme d'onde lissée au signal original. Le signal obtenue est débarrassé des basses fréquences, en dessous de  $F_{det}$ , dite « fréquence de detrend ».

La ci-dessous illustre ce principe. En noir et en haut, on a la signal original, dont on a calculé la tendance, en rouge. En bas, on obtient le signal auquel on a soustrait sa tendance.

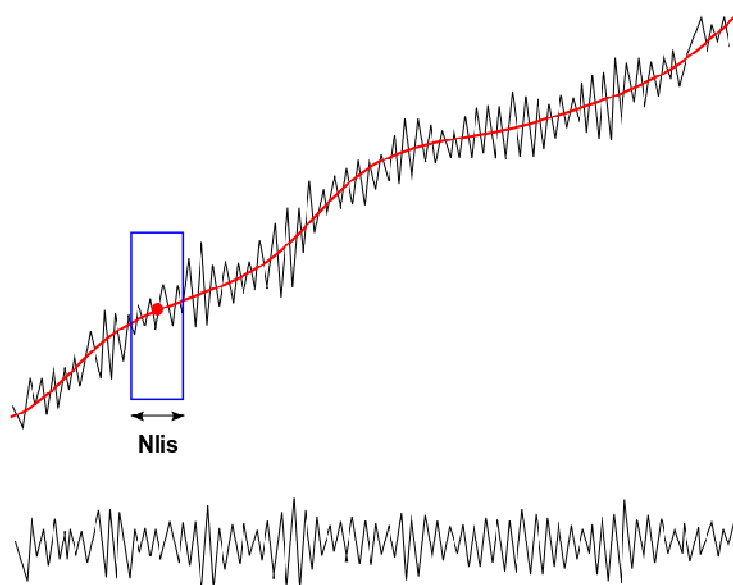


Figure 14: Principe du "detrend"

Cette méthode est particulièrement utile dans deux cas :

- Elle permet de calibrer les données VTL1 sur de courtes périodes, inférieure à celle du spin, pour laquelle la méthode de despin classique ne peut plus s'appliquer. En effet, si on cherche à calibrer sur une portion seulement de la sinusoïde de spin (voir Figure 13), on ne peut plus calculer la sinusoïde de spin. En revanche, on peut effectuer un detrend qui par lissage approximerait la tendance du signal, c'est-à-dire une portion de la sinusoïde, et la retirera.
- En cas de fort champ DC dans le plan de spin, il améliore la fonction de despin, et peut être utilisé conjointement. De plus la sinusoïde de spin peut être saturée, et si la saturation n'est pas trop importante, on peut retirer la sinusoïde saturée par la méthode du detrend, et calibrer les portions de haute fréquence sur les flancs non saturés de la sinusoïde. En mode HBR, cela peut être utile pour récupérer des portions (courtes) de signal intéressantes, et permet de calibrer les données sur des durées correspondantes à une arche de la sinusoïde de spin.

Il faut garder en mémoire que contrairement au despin, le detrend ne retire pas seulement les composantes du signal à la fréquence de spin, mais toutes les basses fréquences. Dans la mesure où la sensibilité du search-coil n'est pas optimum à basse fréquence, cet inconvénient peut être considéré comme mineur. Dans la chaîne de production de masse pour le CSA, cette option n'est pas utilisée.

#### • *Fréquence de coupure*

Lors de la calibration, on divise l'amplitude en Volt d'une raie en fréquence par la valeur de la fonction de transfert à cette fréquence. Dans le cas d'un search-coil, la fonction de transfert à la fréquence zéro est nulle. On définit la fréquence de coupure par la valeur en dessous de laquelle la fonction de transfert n'a plus de signification. Les formes d'onde calibrées sont donc filtrées en dessous de cette fréquence. On rappelle que cette fréquence est donnée dans le repère tournant des antennes. Une valeur usuelle pour STAFF est de 0.1 Hz. Il n'est pas recommandé de descendre en dessous.

#### • *Fréquences min et max*

Ce sont les fréquences entre lesquelles on calibre. Le signal calibré est donc filtré entre ces bornes, toujours dans le repère tournant. Si on mentionne (0., 0.) le signal sera filtré entre  $F_c$  et la fréquence de Nyquist (12.5 ou 225 Hz selon le mode).

#### • *Les étapes de la calibration*

Elles sont décrites au chapitre 8.2.2. Les valeurs 1 à 3 ne sont utilisées que pour la vérification des données et le contrôle des étapes de calibration. L'option 4 est la plus courante, et sert quand on veut calculer et visualiser des spectres ou spectrogrammes. Dans l'option 5, la valeur du DC est additionnée sur les composantes X et Y des formes d'onde. Elles ne sont donc plus centrées sur zéro, et ne peuvent être changées de repère. C'est l'option qui est utilisée lors des comparaisons directes avec les données de FGM, en mode burst (voir 13, Robert 2014).

L'option 6 permet de vérifier la stabilité du champ Bperp estimé lors du despin.

Les options 7 et 8 sont les options de production pour les visualisations « quick look » en ISR2 et les spectres livrées au CSA en GSE.

#### • La taille du noyau de calibration

C'est un paramètre important, qui détermine à la fois la résolution temporelle des spectres et la qualité de la calibration. Pour choisir cette valeur, un compromis est à faire.

- Une valeur élevée donnera une résolution en fréquence très fine, et donc une bonne qualité de la calibration puisque la fonction de transfert varie non linéairement avec la fréquence. Par contre, si on fait un spectre sur une longue durée, on suppose lors du despin que le champ continu perpendiculaire ne varie pas, ou peu. Si ce qui n'est pas le cas, le despin sera mauvais, et il restera des résidus de spin dans le spectre calibré.
- Une valeur courte est bien adaptée aux événements transitoires, variant rapidement avec le temps. Mais il faudra rester au dessus d'une ou deux périodes de spin pour avoir un despin efficace. En dessous, il faudra jouer avec le detrend. Par contre, on aura alors une mauvaise résolution en fréquence, ce qui ne sera pas adapté à des phénomènes du genre onde pseudo-monochromatique. Enfin, la fonction de transfert sera appliquée sur des raies larges, donc avec une mauvaise qualité de calibration.

Dans tous les cas on a  $\Delta t \cdot \Delta f = 1$  et il faut jouer avec ce compromis. Pour la production des spectres de routines, livrés au CSA, on a choisi  $\Delta t \sim 10$  secondes, en NBR comme en HBR (respectivement 256 et 4096 points). Cette valeur permet un despinage correcte (sur deux périodes de spin) et une résolution en fréquence qui permet de suivre correctement les variations de la fonction de transfert.

Dans le cas d'applications scientifiques spécifiques, on peut être amené à changer ces valeurs. Par exemple, pour l'étude d'un sifflement haute fréquence en HBR (vers 150 Hz, de largeur 50 Hz) et relativement transitoire (10 secondes) comme dans le cas de la Figure 40 au chapitre 9.10.8 (montrant la polarisation circulaire droite) on a choisi 128 points ( $\delta t = 0.28$  sec. et  $\delta f = 3.5$  Hz) et utilisé une fréquence de detrend de 50 Hz.

Pour une onde pseudo-monochromatique en NBR, s'étendant sur 20 à 30 mn autour de 3 Hz en champ DC variant peu (voir Figure 43 au chapitre 9.10.8) on a choisi 512 points (5 périodes de spin) soit donc  $\delta t = 10.48$  sec et  $\delta f = 0.049$  Hz ce qui permet de mettre en évidence les structures des paquets d'onde.

#### • La taille du déplacement de la fenêtre

Dans le programme de calcul de spectre, il est prévu un paramètre « d'overlapping », c'est-à-dire la possibilité de faire se chevaucher les spectres, permettant d'augmenter artificiellement la résolution en temps sans altérer celle en fréquence. Cette option est gérée par le paramètre Nshift, permettant de calculer un spectre après un déplacement de Nshift point, en choisissant bien sûr  $Nshift \leq N_{kern}$ . Néanmoins cette option n'a jamais été validée et il est déconseillé de l'utiliser.

#### • Le type d'apodisation

Plusieurs types d'apodisation de la forme d'onde sont possibles avant le calcul du spectre. Dans la version RCL 2.2, les types disponibles sont les suivants, avec leur recommandation.

- « no » aucune apodisation : fenêtre rectangulaire, recommandée pour des signaux transitoires dont la durée est inférieure à la longueur de la fenêtre ou pour la séparation de deux fréquences très proches et avec des amplitudes très proches également.
- « tr » fenêtre trapèze comme sur la Figure 15 : recommandée pour la plupart des spectrogrammes. C'est celle utilisée pour produire les spectres SPL2, puis CS livrée au CSA. Chaque flancs représente  $N_f = 1/16$  du nombre total de points.

- « rr » rounded rectangle : version améliorée du trapèze, mais qui mange un peu plus de signal (Figure 15). Les bords sont calculés sur  $Nf=1/16$  et représentent la première et seconde  $1/4$  de période d'un sinus.
- « rt » rounded trapèze : version dérivée du trapèze. Les bords sont calculés sur  $Nf=1/16$  et représentent la première et seconde  $1/4$  de période d'un  $\sin^2$ .
- « Ri » fenêtre de Riesz. Sa formule est :

$$W_k = 1 - \left[ \frac{2(k-1)}{N} - 1 \right]^2$$

- « Ha » fenêtre de Hann (dites de Hanning) classique, généralement utilisée si on ne connaît pas trop la forme du signal, et recommandée pour des ondes pseudo monochromatiques ou des superpositions de sinusoides, ou encore pour des signaux aléatoires à bande étroite. Sa formule est :

$$W_k = 0.5 \left( 1 - \cos\left(\frac{2\pi k}{N}\right) \right)$$

- « KB » fenêtre de Kaiser-Bessel, adaptée pour la séparation de deux fréquences très proches mais dont les amplitudes sont très différentes. Sa formule est compliquée, d'autant qu'elle fait intervenir une fonction de Bessel d'ordre zéro du premier genre et d'un paramètre  $\alpha$  à choisir. Ce paramètre caractérise le compromis entre la largeur du lobe principal et des lobes secondaires. Une valeur raisonnable est  $\alpha=3$ . Dans ce cas, une approximation suffisante est donnée par la formule suivante (<http://www.diracdelta.co.uk/science/source/k/a/kaiser-bessel-window>):

$$W_k = 0.40243 - 0.49804 \cos\left(\frac{2\pi k}{N}\right) + 0.09831 \cos\left(\frac{4\pi k}{N}\right) - 0.00122 \cos\left(\frac{6\pi k}{N}\right)$$

- « BH » fenêtre de Blackman-Harris. Elle est proche de celle de Kaiser-Bessel. Sa formule est :

$$W_k = 0.35875 - 0.48829 \cos\left(\frac{2\pi k}{N}\right) + 0.14128 \cos\left(\frac{4\pi k}{N}\right) - 0.01168 \cos\left(\frac{6\pi k}{N}\right)$$

- « g » fenêtre gaussienne « sur mesure » utilisée pour la calibration continue (voir chapitre suivant). Elle est donnée par

$$W_k = e^{-x^2} \text{ avec } x = [(k - N/2) - 0.5]/\sigma$$

$$\text{et on prend } \sigma = \frac{N/2}{\sqrt{-\ln(10^{-3})}}$$

La valeur -0.5 est mise pour symétriser la gaussienne avec le sommet entre  $N/2$  et  $N/2 + 1$ .

#### • La forme et la normalisation des fenêtres de pondération

La Figure 15 ci-dessous donne la forme de quelques fonctions de pondération usuelles, telles qu'elle sont disponibles dans la procédure RCL\_vectime\_L1\_to\_spectro\_L2. Néanmoins, afin de conserver l'énergie, il convient de normaliser ces fonctions avant usage par un coefficient qui normalise la surface à 1, afin qu'il n'y ait pas de perte d'énergie.



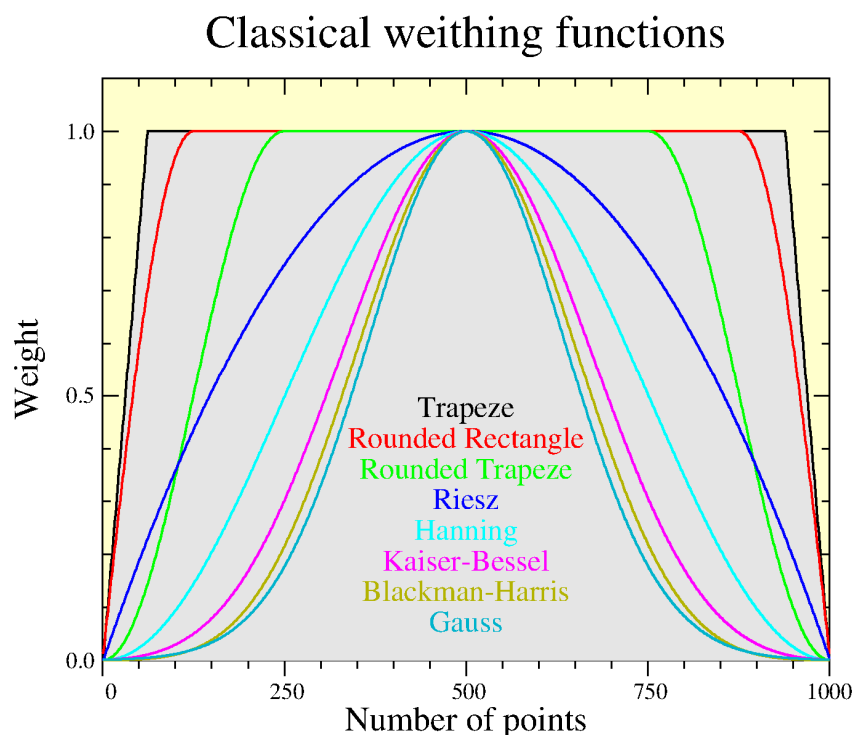


Figure 15: représentation des fenêtres de pondérations disponibles

Les coefficients de normalisation à appliquer sont les suivants :

	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

#### • Exemple choisi pour la production de masse

Pour la production des CS au CSA, et donc des SPL2, les paramètres de production ont été choisis comme indiqué ci-dessous. C'est la commande de base RCL\_product\_SPL2\_oneday qui appelle la procédure RCL\_vectime\_L1\_to\_spectro\_L2 avec les paramètres suivants :

```

VTL1.rff      : de la forme CLU1_STASC_VTL1_NBR_20091214_full.rff
                obtenu par un RCL_get_data_CLUSTA_VTL1_forSPL2
                avec comme arguments $SatNum $year $month $day $BitRate)
SPL2.rff      : produit sous la forme CLU1_STASC_SPL2_GSE_NBR_20091214.rff
Fdet          : 0. (pas de detrend en production de masse)
Fc            : 0.1 pour les spectres en ISR2, 0.6 pour ceux en GSE
F1, F2        : 0. 0. (soit toute la gamme de fréquence)
Step          : 7 pour les spectres en ISR2, 8 pour ceux en GSE
N-Kern        : 256 en NBR, 4096 en HBR (soit environ 10 secondes).
N_shift       : égal à N_Kern
Apod          : t (trapèze)

```

## 8.3. CALIBRATION CONTINUE DES FORMES D'ONDE

### 8.3.1 Théorie

La méthode de calibration des formes d'onde en continue est en fait dérivée de celles de la calibration des spectres. Le détail de cette méthode calibration est décrite dans [13, Robert, 2013] ainsi que sa validation par comparaison avec les données FGM en « Full Résolution ».

Pour obtenir la forme d'onde calibrée à partir du spectre calibré, il suffirait, en théorie de faire une FFT inverse. Néanmoins, la forme d'onde obtenue est entachée d'une discontinuité sur les bords, par une remise à zéro des extrémités. Pour éviter cela, le principe de la calibration continue des formes d'onde (par opposition à « discontinue » quand  $N\_Shift=N\_Kern$ ) est le suivant :

- On calibre un spectre qui débute à l'instant  $t_1$ , sur une durée  $T$ . La fenêtre d'apodisation utilisée est une gaussienne, définie au chapitre 8.3.1.
- On fait la FFT inverse, mais on ne garde que les points centraux de la forme d'onde calibrée, c'est-à-dire  $N\_shift$  points, qui correspondent au sommet de la gaussienne
- On décale le temps  $t_1$  d'un intervalle  $\delta t$  correspondant à  $N\_Shift$  points, et on calibre à nouveau une de fenêtre de durée  $T=N\_Kern/fe$

Dans la pratique, la calibration optimale est obtenue pour  $N\_Shift=2$ . Sur une fenêtre de  $N\_Kern$  points (1024 par exemple) on ne garde donc que les 2 points du sommet de la gaussienne.

### 8.3.2 Les étapes successives de la calibration

Elles sont en fait les mêmes que pour la calibration des spectres, à quelques détails près :

- Dans l'étape 3, la calibration se fait maintenant par une routine optimisée `DECONVO_C3` de la bibliothèque `lib_deconvo` (voir 7.3.1). On retrouvera dans le code les instructions suivantes:

```
! 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,:))
```

- Toujours dans l'étape 3, après calibration, on normalise l'énergie due à la pondération gaussienne :

```
!      On fait une ponderation inverse sur les points gardes, pour ne pas
!      affecter le module et creer des harmoniques a N_Shift
!      que pour le cas Gaussien, avec N_Shift < N_Kern

      if(apod == 'g') then
        do j=1,i2
          proc_vec(i,j)=proc_vec(i,j)/Weight(j)
        enddo
      endif
```

- Dans l'étape 7, on ajoute aux 3 composantes calculées en étape 4, deux composantes donnant la valeur du DC sur les composantes X et Y, le tout en repère ISR2 :

```
!      SR2 to ISR2

      if (spin_phase >= -360.) then
        do i= 1,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
```

### 8.3.1 Paramètres de la procédure RCL\_vectime\_calibration\_CLUSTA

#### • Rappel

Comme pour la procédure de calibration des spectres, cette procédure prend donc en entrée les formes d'onde de niveau L1 (non calibrées) et produit une forme d'onde calibrée continument selon la méthode décrite ci-dessus, détaillée aussi dans l'article [13, Robert, 2013]. On a vu au chapitre 4.8.2 que les paramètres de cette procédure étaient les suivants :

Usage : RCL\_vectime\_calibration\_CLUSTA VTL1.rff VTL2.rff Fdet Fc F1 F2 Step \
N\_Kern N\_shift Apod

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)
	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
N_Kern	: Kernel size
N_shift	: for sliding window
Apod	: trapezium (t) or Gaussian (g)

ex: RCL\_vectime\_calibration\_CLUSTA CLU1\_STASC\_NBR\_VTL1\_20091213.rff \
CLU1\_STASC\_NBR\_VTL2\_20091213.rff \
0. 0.1 0. 0. 1024 2 g



## 9. ANNEXES

9.1. Information sur le statut dans les WFL1 et VTL1 .....	143
9.1.1 Information générales .....	143
9.1.2 Signification des caractères du statut .....	143
9.1.3 Quelques mots sur l'indice de compression .....	145
9.2. Calcul du "spin phase" dans les WFL1 et les VTL1 .....	146
9.2.1 Définition .....	146
9.2.2 Méthode de calcul .....	146
9.2.3 Les fichiers SPINPHASE .....	148
9.2.4 Les fichiers des Sun Pulses interpolés : .....	148
9.3. Définition des systèmes de coordonnées utilisés .....	149
9.3.1 The Sensor Coordinate System (SCS) .....	149
9.3.2 The Orthogonal Sensor System (OSS) .....	149
9.3.3 The Data Sensor System (DSS) .....	150
9.3.4 The Body Build System (BBS) .....	150
9.3.5 The Spin Reference System (SRS) .....	150
9.3.6 The spin reference2 system (SR2) .....	151
9.3.7 The Inverse SR2 system (ISR2) .....	152
9.3.8 Simplification of the cumulative matrix products .....	152
9.3.9 The Geocentric Equatorial Inertial system (GEI) .....	153
9.3.10 The Geocentric Solar Ecliptic system (GSE) .....	153
9.3.11 Geocentric Solar Magnetospheric system (GSM) .....	154
9.3.12 Magnetic Field Aligned system (MFA) .....	154
9.4. Estimation du champ continu dans le plan de spin .....	155
9.4.1 Problématique .....	155
9.4.2 Amplitude et phase de $\mathbf{B}^{\perp}$ en repère tournant SR .....	155
9.4.3 Tests sur $\mathbf{B}^{\perp}\cdot\mathbf{x}$ et $\mathbf{B}^{\perp}\cdot\mathbf{y}$ .....	156
9.4.4 Amplitude et phase de $\mathbf{B}^{\perp}$ en repère fixe SR2 .....	156
9.5. Estimation du "misalignment angle" .....	157
9.5.1 Définitions .....	157
9.5.2 Signal delivered by a rotating antenna into a DC field .....	158
9.5.3 Estimate of the misalignment angle $\theta_z$ for low value .....	158
9.5.4 Estimate of the misalignment angle $\theta_z$ for high value .....	158
9.6. Calcul de la matrice de correction du dépointage .....	161
9.6.1 Passage de coordonnées non orthogonales à un système orthogonal .....	161
9.6.2 Matrice de correction du dépointage .....	163
9.7. Petits rappels sur la transformée de Fourier .....	167
9.7.1 La transformée de Fourier au sens mathématique .....	167
9.7.2 La transformée de Fourier discrète .....	167
9.7.3 La « Fast Fourier Transform » .....	168
9.8. Utilité des composantes circulaires en polarisation .....	171
9.8.1 Définition .....	171
9.8.2 Passages en composantes circulaires Gauche et Droite .....	171
9.9. Exemples de fichiers RFF .....	173
9.9.1 Exemple de WFL1.rff .....	173
9.9.2 Exemple de VTL1.rff .....	176
9.9.3 Exemple de VTL2.rff .....	179
9.9.4 Exemple de SPL2.rff .....	182

9.10. Exemple de visualisations .....	185
9.10.1 Résultat de RCL_visu_spectro à partir des VTL2 .....	185
9.10.2 Résultat de RCL_visu_spectro à partir des VTL1 .....	186
9.10.3 Résultat de RCL_visu_spectro_4Bz_3h .....	187
9.10.4 Résultat de RCL_visu_ave_spectrum à partir des VTL1 .....	188
9.10.5 Résultat de RCL_visu_vectime sur les VTL1 .....	189
9.10.6 Résultat de RCL_visu_vectime sur les VTL2 en ISR2 .....	190
9.10.7 Résultat de RCL_visu_vectime sur les VTL2 en GSE .....	191
9.10.8 Résultat de RCL_visu_polar sur des VTL2 en GSE .....	192
9.10.9 Résultat de RCL_visu_CLUPOS / 3 heures .....	197
9.10.10 Résultat de RCL_visu_CLUPOS / orbite complète .....	199
9.10.11 Résultat de RCL_visu_CLUGEOM .....	201
9.11. Résumé des commandes RCL .....	203
9.11.1 Liste fonctionnelle des commandes RCL .....	203
9.11.2 Liste alphabétique des commandes avec arguments .....	206
9.12. Quelques mots sur les ROPROC .....	209
9.12.1 Historique des Roproc et lien avec les RCL .....	209
9.12.2 Introduction aux Roproc .....	210
9.12.3 Liste thématique des Roproc .....	211
9.12.4 Examples of arguments of Roproc commands .....	217

## 9.1. INFORMATION SUR LE STATUT DANS LES WFL1 ET VTL1

### 9.1.1 Information générales

Dans les WFL1, le statut donnant des informations sur l'expérience se présente sous la forme ci-dessous (voir aussi Table 2 dans §1.4 et § 8.2.1).

2001-09-23T00:00:00.904328Z	000000000001	17.35
34047 33190 32795	0	
34078 33107 32796	0	

Il est donné en même temps que la datation de chaque bloc. Dans les VTL1, il est dupliqué lors de l'interpolation utilisée pour dater chaque vecteur (voir § 2.5.2)

2001-09-23T00:00:00.904328Z,	000000000001	10,	17.35, 34047, 33190, 32795
2001-09-23T00:00:00.944327Z,	000000000001	00,	20.93, 34078, 33107, 32796
2001-09-23T00:00:00.984326Z,	000000000001	00,	24.51, 34098, 33028, 32796

On peut voir que dans le passage WFL1 vers VTL1 le statut a gagné deux caractères.

- Le premier caractère rajouté dans le VTL1 indique si le vecteur a été daté par un block maître dans le WFL1 (**1**) ou si la date a été calculée par interpolation entre le temps de deux blocs des WFL1 (**0**). De même, ce caractère indique si le « spin phase » est une valeur initiale du block correspondant au WFL1, ou si cette valeur a été interpolée entre deux blocs.
- Le deuxième caractère rajouté au statut est simplement le statut de compression, qui était donné après les 3 valeurs ix, iy et iz du WFL1, et qui a été intégré au statut général pour les VTL1.

### 9.1.2 Signification des caractères du statut

L'ensemble de ce paragraphe est extrait du « STAFF User Guide [7, Cornilleau, 2011]. Il est rappelé ici pour plus de commodité.

Le statut des VTL1 est constitué de 12 caractères de celui des WFL1, auxquels on a rajouté le bit d'interpolation et l'indicateur de compression. Les caractères correspondants se reportent aux entités selon la Table 64 ci-dessous.

1	2	3	4	5	6	7	8	9	10	11	12	13	14
Step in cal	EFW Y	EFW Z	Mode SA	Mode SC	Despin SA	WHISPER	Calibration	EFW sweep	Compression	Phase	TCOR	Time Quality	Compression Quality

Table 64 : STAFF status word

La signification de chaque caractère est donnée dans la Table 65 ci-dessous.

N	Min-max	Meanings		
<b>1</b> Step in cal	0-n	0: science mode		
		Step	Mode	Attenuation (dB)
		1	CAL4	0
		2	CAL4	0
		3	CAL4	0
		4	CAL4	0
		5	CAL4	-13
		6	CAL4	-26
		7	CAL4	-39
		8	CAL4	-52
		9	CAL4	-65
		a	CAL4	-78
		b	CAL4	Gnd
		c	CAL3	0
		d	CAL3	-26
		e	CAL1	0
		f	CAL2	0
		g	CAL1	-26
		h	CAL2	-26
		i	CAL1	-52
		j	CAL2	-52
		k	CAL2	Gnd
		l	CAL OFF redundant	
		m	CAL2	-26
		n	CAL Off/On satellite	
		o: after calibration, till reset or new calibration		
<b>2</b> EFW Y boom pair	0-1	0: density mode off 1: density mode on		
<b>3</b> EFW Z boom pair	0-1	0: density mode off 1: density mode on		
<b>4</b> Mode STAFF-SA	0-f	Value	Mode	
		0	NM1	
		1	NM2e	
		2	NM2b	
		3	Illegal	
		4	Emergency	
		5	Special	
		6	NM1'e	
		7	NM1'b	
		8	FM1	
		9	FM3e	
		a	FM3b	
		b	Illegal	
		c	FM2	
		d	Illegal	
		e	Illegal	
		f	Passive	

N	Min-max	Meanings
<b>5</b> Mode STAFF-SC	0-1	0 : SC bandwidth 10 Hz 1 : SC bandwidth 180 Hz
<b>6</b> STAFF-SA on-board despin	0-1	0 : despin on 1 : despin off
<b>7</b> WHISPER transmitter	0-1	0: off 1: active
<b>8</b> Calibration	0-1	0: off 1: active
<b>9</b> EFW sweep progress	0-2	0: no scanning 1: scanning 2: non synchronised block
<b>10</b> Compression Mode	0-2	0: nominal 1: backup 2: no compression
<b>11</b> Phase	0-N	0:Phase calculation OK Right Sun Pulse 1:Phase calculation OK Sun Pulse interpolated 3: Phase calculation OK Sun Pulse suspect 4: Phase calculation OK no Phi_SC SATT 5: Phase calculation OK no Phi_SC SATT + Sun Pulse interpolated 6: Phase calculation OK no Phi_SC SATT + Sun Pulse suspect N:Phase=-500 No Sun Pulse N_Phase=-500 No reference phase in SATT
<b>12</b> Tcor	0-1	0 : No (No Tcor correction available) 1 : Yes (Tcor corrected)
<b>13</b> Time Quality	0-1	0: interpolated time 1: block time
<b>14</b> Compression Quality	0-7	0 = no compression error 1 – 7 = error on 1 to 3 components in instrument frame error in BX error in By error in Bx an in By error in Bz error in Bx an Bz Error in By and Bz Error in the 3 components

Table 65 : Signification de chacun des 14 caractères du statut des VTL1 ou des DWF



### 9.1.3 Quelques mots sur l'indice de compression

*Ce paragraphe a été extrait de l'User Guide [7, Cornilleau 2011] en anglais, et non retraduit ici.*

Wave form data are sample into 16 bits for the first record of each block, but for other records only the difference is kept, coded in 12bits. If the difference between two records is too big, we may encounter compression errors. Fortunately we know on which bit the error occurs, which allows us to maximise it.

Three compression modes are available (see Status word character #10), and may lead to one or another bit to be wrong. The maximum error is then known, see the following Table 66 (where Delta is the difference between the current record and the previous one):

	Delta (16 bits)	Maximum Compression Error (TM counts)	Maximum Error (mV)
No Compression	0-65535	0	0
Normal Compression	0-2015	0	0
	2016-65535	1024	150
Backup compression	0-511	0	0
	512-1535	1	0.15
	1536-3587	2	0.3
	3588-7447	4	0.6
	7448-65535	1024	150

**Table 66 :** Erreur maximum du à la compression

The normal and backup compression are used respectively when we expect to measure “low” and “high” amplitude signals including large spin signals.

Details of the compression error are given in the DWF data sets (one word per component).

In CWF a compression quality is given (see status word, character # 14) in the status word. This compression quality is calculated the following way:

$$Q = 4*Bx\_error + 2*By\_error + 1*Bz\_error$$

With Bx, By and Bz being the components in the STAFF Search coil reference frame and the error set to 0 no error, 1 an error.

## 9.2. CALCUL DU “SPIN PHASE” DANS LES WFL1 ET LES VTL1

Une partie de ce chapitre est issu d'un document trouvé sans date ni auteur. Il a été remis en forme et vérifié ici, notamment à partir des informations disponibles dans le Data Delivery Interface Document (DDID [17, ESA/ESOC 2000]).

### 9.2.1 Définition

La connaissance du « spin phase » est essentielle pour passer du repère tournant des antennes à un repère fixe, ou à n'importe quel autre repère. C'est un angle variant rapidement (à la vitesse de rotation du satellite) et il est présent dans les fichiers de télémétrie WFL1 pour chaque bloc daté, puis interpolés et mis dans chaque vecteur des fichiers VTL1.

C'est le « Sun pulse » qui est utilisé pour définir la direction du Soleil dans le repère tournant du satellite appelé SR (Spin Reference). L'axe Z du SR est aligné avec moment d'inertie principal, et est proche de l'axe X du « Body Build system ». Voir aussi le § 8.6.4 pour la définition du SR2.

Le « spin phase » est ainsi défini dans le DDID (DDID – CL-ESC-ID-2001 – Appendix I, page 134) comme l'angle de rotation du demi-plan défini par les axes +ZSR et +XSR du système de référence SR depuis le moment où la direction du soleil était contenue dans ce plan (voir Figure 16).

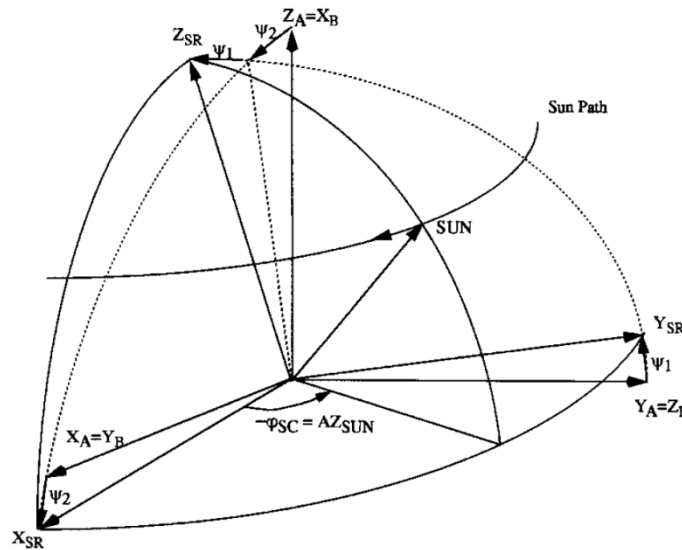


Figure 16 : Définition du « spin phase » dans le repère SR2 (extraite du DDID)

### 9.2.2 Méthode de calcul

Pour calculer ce spin phase, on utilise les Sun Reference Pulse qui sont générées à bord du satellite et qui indiquent le moment où la direction du soleil est comprise dans le demi-plan précédemment défini (Meridian Sun Events). Ces événements sont répertoriés dans le fichier HKsat.

On donc :

$$\Phi_t = 2\pi \cdot f_s (t - t_0) = 2\pi \cdot f_s t + \Phi_0$$

Pour calculer la fréquence de spin il suffit de 2 Sun Events consécutifs, puisque entre ces 2 événements on peut considérer que le satellite a accompli un tour. On a donc :

$$f_s = \frac{1}{T_{i+1} - T_i}$$

En pratique, pour la production des WFL1, il a été écrit une fonction qui :

- Prend en entrée la date à laquelle doit être calculé la phase au format ISO, le numéro de satellite Cluster.
- Donne en sortie la phase calculée et un statut rendant compte du déroulement du calcul de la phase. Les différentes valeurs du statut sont définies dans l'annexe 8.1

Cette fonction se déroule de la manière suivante :

- On initialise la fonction avec la valeur -500.00 pour la phase et U pour le statut.
- On lit la date ISO (date à laquelle on veut la phase).  
S'il y a un problème lors de la lecture de la date ISO, le statut vaut X et le message d'erreur suivant s'affiche sur la fenêtre de commande :  
ERROR / PHASE\_COMPUTE / CAN'T READ ISODATE : dateISO  
et l'exécution s'arrête.
- On vérifie la validité du numéro de satellite.  
Si le numéro de satellite n'est pas valide, le statut vaut X et le message d'erreur suivant s'affiche sur la fenêtre de commande :  
ERROR / PHASE\_COMPUTE / CLUSTER DOESN'T HAVE A SATELLITE numérosat  
et l'exécution s'arrête.
- On recherche le  $\Phi_{SC}$  (Phase du satellite) dans le fichier SPINPHASE. Pour ce faire, on parcourt le fichier SPINPHASE jusqu'à trouver la ligne correspondant dans ce fichier et donc trouver le  $\Phi_{SC}$  ou jusqu'à la fin du fichier (si on ne trouve pas de ligne correspondant à la date).  
Si un problème intervient lors de la lecture du fichier SPINPHASE, le statut vaut X et le message d'erreur suivant s'affiche sur la fenêtre de commande :  
ERROR / PHASE\_COMPUTE / ERROR WHILE READING FILE nomFichierSpinPhase  
et l'exécution s'arrête.  
Si aucun  $\Phi_{SC}$  n'a été trouvé à la fin du parcours du fichier SPINPHASE, nous affectons à  $\Phi_{SC}$  une valeur par défaut. Ces valeurs par défaut sont :  
 $\Phi_{SC} = 333.93$  pour le satellite 1  
 $\Phi_{SC} = 333.87$  pour le satellite 2  
 $\Phi_{SC} = 333.95$  pour le satellite 3  
 $\Phi_{SC} = 333.89$  pour le satellite 4  
Dans ce cas, le statut vaut 4 et le message suivant s'affiche sur la fenêtre de commande :  
USING MEAN PHI\_SC=333.93INSTEAD FOR SAT 1  
si on traite le satellite 1
- On recherche le Sun Pulse le plus proche précédent la date à laquelle on veut la phase. Pour cela, nous parcourons le fichier des Sun Pulses interpolés jusqu'à trouver le bon Sun Pulse. Une fois que ce Sun Pulse est trouvé, on contrôle que le Sun Pulse suivant n'est pas plus proche et est postérieur à la date (à laquelle on veut la phase).  
Si un problème intervient lors de la lecture du fichier de Sun Pulse interpolés, le statut vaut X et le message d'erreur suivant s'affiche sur la fenêtre de commande :  
ERROR / PHASE\_COMPUTE / ERROR WHILE READING FILE nomFichierSunPulseInterpoles  
et l'exécution s'arrête.
- On vérifie que le Sun Pulse trouvé est satisfaisant.

Si ce n'est pas le cas, le statut vaut N et le message d'alerte suivant s'affiche sur la fenêtre de commande :

```
WARNING / PHASE_COMPUTE / UNABLE TO FIND SATISFACTORY SUN PULSE FOR dateISO IN
nomFichierSunPulseInterpoles
```

et l'exécution s'arrête.

- On affecte la valeur du statut pour les cas qui n'ont pas encore été traités. Le satus vaut :
  - 0 si tous est OK,
  - 1 si le Sun Pulse utilisé est interpolé,
  - 2 si le Sun Pulse utilisé est suspect,
  - 5 si le Sun Pulse utilisé est interpolé et si on n'a pas trouvé de  $\Phi_{SC}$  dans le fichier SPINPHASE,
  - 6 si le Sun Pulse utilisé est suspect et si on n'a pas trouvé de  $\Phi_{SC}$  dans le fichier SPINPHASE.
- On calcule la phase par la relation suivante :

$$Phase = \phi_{sc} + 360.0 * \frac{Date - SP_{precedent}}{Periodes_P} \text{ modulo } 360^\circ$$

### 9.2.3 Les fichiers SPINPHASE

Il y a un fichier SPINPHASE par satellite. Les trois « champs » présents dans ces fichiers sont les suivants :

- date de début de validité,
- date de fin de validité,
- valeur de  $\Phi_{SC}$ .

Ces différentes valeurs (date de début de validité, date de fin de validité et valeur de  $\Phi_{SC}$ ) sont obtenues à partir des fichiers SATT présents dans /NFS/nas-cluster1/bouzid/val/SATT/AAMMJJ (où AA est l'année, MM est le mois et JJ est le jour).

### 9.2.4 Les fichiers des Sun Pulses interpolés :

Il y a un fichier de Sun Pulses interpolés par satellite et par jour. Les informations présentes dans ces fichiers sont les suivantes :

- Sun Pulse (date ISO),
- caractère valant T si le Sun Pulse est « vraie » (Sun Pulse lu dans le fichier HK) ou F si le Sun Pulse est interpolé,
- écart moyen entre Sun Pulse. Cet écart est calculé en réalisant la moyenne des écarts entre Sun Pulses successifs pour les cinq Sun Pulses précédents et les cinq Sun Pulses suivants le Sun Pulse qui nous intéresse.

Les fichiers de Sun Pulse brut sont dans :

```
/NFS/nas-cluster1/CLUSTER/COMMON/SPINPHASE/SP/HK
```

Ils peuvent présenter des « trous », c'est-à-dire que l'écart moyen entre deux Sun Pulses consécutifs est supérieur à la période de spin moyenne. Ces « trous » empêchent de calculer la phase (à l'endroit du trou). Nous essayons, quand c'est possible, de combler ces « trous » en interpolant les Sun Pulses.

Un « trou » est comblé si la durée de celui-ci est inférieure à 600s (c'est-à-dire 1/10ième de degré). La deuxième condition pour combler ce « trou » est que la valeur de  $\Phi_{SC}$  avant et après le « trou » soit la même.

### 9.3. DÉFINITION DES SYSTÈMES DE COORDONNÉES UTILISÉS

Les définitions ci-dessous sont extraites du chapitre 5 de l'article [13, Robert 2013] ainsi que du document de travail [8, Robert, 2003]. Ceux-ci étant en anglais, il n'as pas été fait de traduction.

To transform telemetry data into significant physical units we need to convert the data from the sensor coordinate system into one or another system, and in particular to transform from the spinning system into a fixed one, with respect to Sun and Earth for instance. The following sections are dedicated to define all intermediate coordinate systems required for this operation. Notice than these definitions can be used for other experiment of the same type, one any other mission.

All transformation matrixes are named as:  $A\_to\_B$  where A and B are two different coordinate systems. To convert a vector given in the A system to the same vector expressed in the B system, the following expression is used:

$$\begin{pmatrix} x \\ y \\ y \end{pmatrix}_B = A\_to\_B \begin{pmatrix} x \\ y \\ y \end{pmatrix}_A$$

For general computation of this kind of matrix, see [8, Robert, 2003].

#### 9.3.1 The Sensor Coordinate System (SCS)

This is the system where the original signal is measured (see Figure 17 below). This system could be a non perfect orthogonal system.

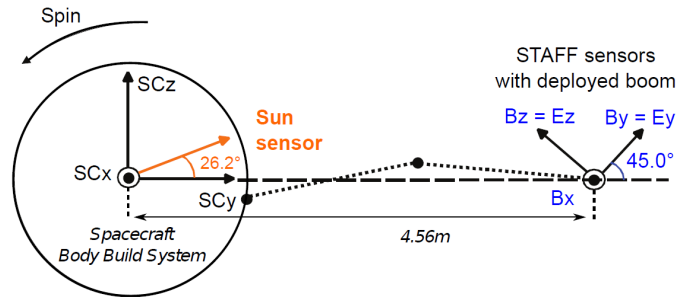


Figure 17 : Position des antennes de STAFF dans le repère « Body Build » lié au satellite.

#### 9.3.2 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.

$$SCS\_to\_OSS \cong \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

If the user wants to do this correction, he can use the formulas given in section 9.6.1 which allows the tranformation from a non orthogonal system to an orthogonal one.

### 9.3.3 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 Fig. 14).

In all our data, the convention taken is  $Z = \text{spin axis}$ . 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 Fig. 1,  $X_{OSS}, Y_{OSS}, Z_{OSS}$ , becomes Y, Z, X in DSS.

This permutation is obtained by the following matrix:

$$OSS\_to\_DSS = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

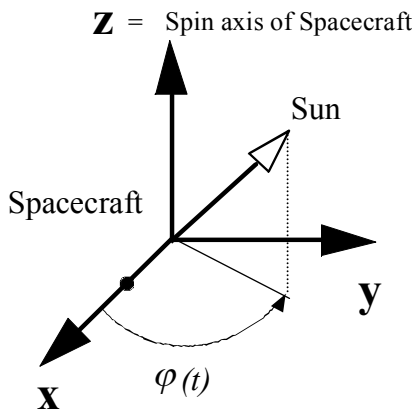
### 9.3.4 The Body Build System (BBS)

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  $Z_{DSS} = X_{BBS}$ . In all cases, 2 other axis may be rotated by an important angle (see Fig. 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  $\alpha = 45^\circ$ .

$$DSS\_to\_BBS = \begin{pmatrix} 0 & 0 & 1 \\ \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \end{pmatrix}$$

### 9.3.5 The Spin Reference System (SRS)

The Spin reference system has its Z axis parallel to the spin axis. This is a spinning system, rotating at the spin frequency. As mentioned above, there is a small misalignment between the  $X_{BBS}$  axis and the  $Z_{SCS}$  axis, as there is another slight misalignment between the  $X_{BBS}$  axis and the  $Z_{DSS}$  axis (see Figure 18).



*This is a spinning local system close to the measurement antenna of a spacecraft.*

*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  $\varphi$  at a given time.*

**Figure 18:** Definition of SR system

This is not easy to separate the two previous angles, but it is possible to estimate the small angle between the  $Z_{SCS}$  axis and the true spin axis which define  $Z_{SRS}$ . This angle  $\theta$  could be estimated by the measurement of the low spin signal on the  $Z_{SCS}$  component (see section 9.5).

If  $B_{xs}, B_{ys}, B_{zs}$ , are the amplitudes in nT of the spin sine on the 3 x, y, z components of the SCS system, this angle is estimated by :

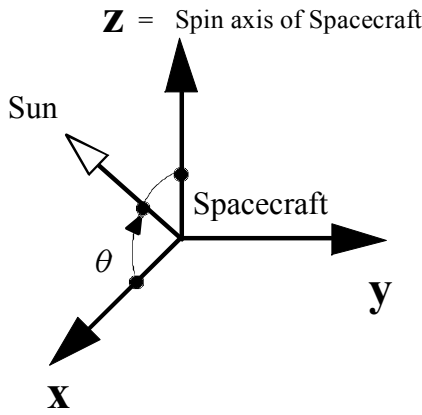
$$\sin \tilde{\theta} = \frac{B_{zs}\sqrt{2}}{\sqrt{B_{xs}^2 + B_{ys}^2 + B_{zs}^2}}$$

This angle could be constant, but can have also small variations during operations on the spacecraft (trajectory modifications, etc.). It has been estimated to an average value of  $\sim 0.5^\circ$ , and, in a first time, has not been taken into account. So, the “BBS\_to\_SRS” matrix is a simple circular permutation set to:

$$BBS\_to\_SRS \cong \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

### 9.3.6 The spin reference2 system (SR2)

The SR2 system, also called “SSS” for Spacecraft-SUN System, or “DS” for despun, is derived from the SRS system by a *despin* operation. The spinning Spacecraft is “stopped” just at the time where the X axis is in the plane containing the Z spin axis and the direction of the Sun. The rotation angle required is derived from the Sun pulse or any other quantity to compute the spin phase angle  $\varphi_s$  (see Figure 19).



*This is a fixed system useful 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  $\theta$  varies continuously.*

**Figure 19: Definition of SR2 system (Despun)**

This spin phase angle  $\varphi_s$ , and the corresponding time measurement, is required to build the “SRS\_to\_SR2” matrix. Terms of this matrix are fast varying with time. The phase angle  $\varphi_s$  is calculated for each time tag of the data thanks to the sun pulse signal. This gives, where  $f_s$  is the spin frequency :

$$SRS\_to\_SR2 = \begin{pmatrix} \sin(2\pi f_s t + \varphi_s) & \cos(2\pi f_s t + \varphi_s) & 0 \\ \cos(2\pi f_s t + \varphi_s) & -\sin(2\pi f_s t + \varphi_s) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

### 9.3.7 The Inverse SR2 system (ISR2)

This is equivalent to the SR2 system (or SSS) 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.

$$SR2\_to\_ISR2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

### 9.3.8 Simplification of the cumulative matrix products

Cumulative matrix product requested to transform original data given in SCS coordinate to a fixed coordinate system such as SR2 can be strongly simplified if we neglect all small misalignment angles mentioned above. By the way, the first mass processing on the STAFF-SC data was to produce a data base for the level 1 data (telemetry data) in the DSS system, which is delivered to the CSA. The only difference between the DSS with the SCS sensor coordinate is a circular permutation of the components to get the Z axis close to the spin axis, since we assume that the SCS is orthogonal and equal to the OSS (see section 8.4.2).

So to transform data expressed in DSS into the “fixed” SR2 we have to apply the cumulative matrix product:

$$\begin{pmatrix} x \\ y \\ y \end{pmatrix}_{SR2} = [SRS\_to\_SR2][BBS\_to\_SRS][DSS\_to\_BBS] \begin{pmatrix} x \\ y \\ y \end{pmatrix}_{DSS}$$

Assuming all small misalignment angles close to zero, we get:

$$[BBS\_to\_SRS][DSS\_to\_BBS] = \begin{pmatrix} \cos \alpha & -\sin \alpha & 1 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Using expression of SRS\_to\_SR2 given in section 5.6, with  $\omega_s = 2\pi f_s$  after some calculus we get:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}_{SR2} = \begin{pmatrix} \sin(\omega_s t + \varphi_s + \alpha) & \cos(\omega_s t + \varphi_s + \alpha) & 1 \\ \cos(\omega_s t + \varphi_s + \alpha) & -\sin(\omega_s t + \varphi_s + \alpha) & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{DSS}$$

By neglecting all the small misalignment angles, the transformation from the Data Sensor System to the fixed SR2 system is simply reduced to a rotation in the spin plane of the fast varying angle:

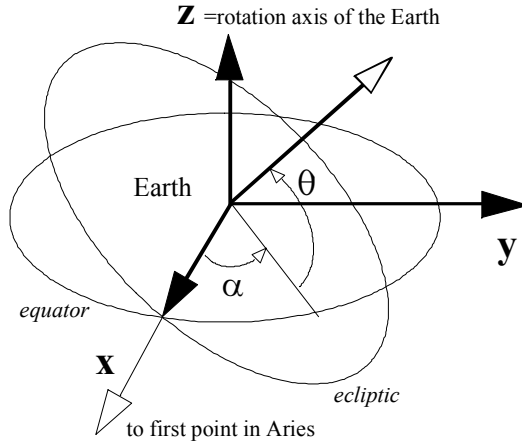
$$\psi = (\omega_s t + \varphi_s + \alpha).$$

This simplification is used for CLUSTER/STAFF calibration, but cannot be used for spacecraft or rocket having precession or nutation, or a non constant direction of the spin axis. In this case, the full computation must be done.



### 9.3.9 The Geocentric Equatorial Inertial system (GEI)

The GSE system is a well known system, with the Z axis perpendicular to the Ecliptic plane, and the X axis toward the Sun. To do the transformation of the SSS to the GSE, the direction of the spin axis in the GSE system is required. Due to the gyroscopic effect of a spinning spacecraft, the spin axis is ~constant in an inertial system, and so has a yearly variation in the GSE system, excepted during spacecraft operations (see Figure 20).



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**  $\alpha$  and the **declination**  $\theta$  as:

**right ascension** :  $\alpha = \tan^{-1}(V_y/V_x)$

with  $\alpha$  in  $[0^\circ, 180^\circ]$  for  $V_y > 0$

$\alpha$  in  $[180^\circ, 360^\circ]$  for  $V_y < 0$

**declination**  $\theta = \sin^{-1}(V_z/V)$

with  $\theta$  in  $[-90^\circ, 90^\circ]$

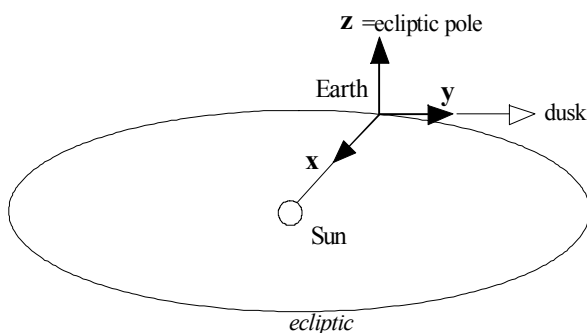
Figure 20 : Definition of GEI system:

SR2 to GSE transformation is done using module “tsr2gse” routine of ROCOTLIB software (see Robert, 1993, 2003, 2004). The Cartesian GSE coordinates of the direction of spin axis is required, as the corresponding time measurement. To transform spin right ascension and spin declination angle, given in STAFF-SC CSA data in Geocentric Equatorial Inertial system (GEI), routine “tgeigse” can be used. Those angles are also available in the auxiliary files available at CSA (latitude and longitude angles of the spin axis direction in GSE).

Note that in GSE system, each component mixes both parallel and perpendicular components to the spin axis. Because sensitivity is strongly different at low frequency on the parallel and perpendicular components in SR2 system, it is recommended to filter the data below ~0.6Hz before coordinate transformation. This is done for CSA Complex Spectra products.

### 9.3.10 The Geocentric Solar Ecliptic system (GSE)

Well known and very used system (see Figure 21).



The X-axis is pointing from the Earth towards the Sun.

The X-axis and the Y-axis are included in the ecliptic plane.

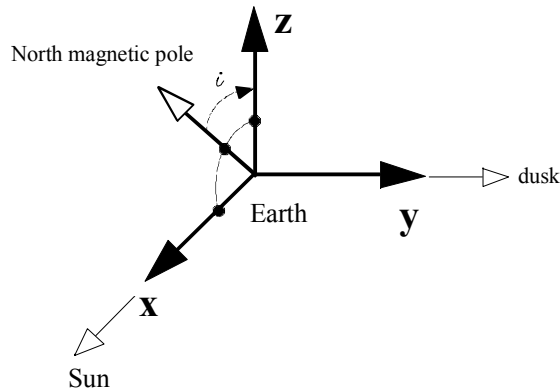
The Y-axis is pointing toward the dusk, opposing to the planetary motion.

The Z-axis is parallel to the ecliptic pole. The GSE system has a yearly rotation with respect to the inertial system.

Figure 21: Definition of GSE system

### 9.3.11 Geocentric Solar Magnetospheric system (GSM)

This system is known in space physics to properly organize the data, insofar as it reconciles the direction of the sun and the plane of the Earth magnetic meridian (see Figure 22).



*The X-axis is pointing from the Earth towards the Sun.*

*The X-Z plane contains the dipole axis.*

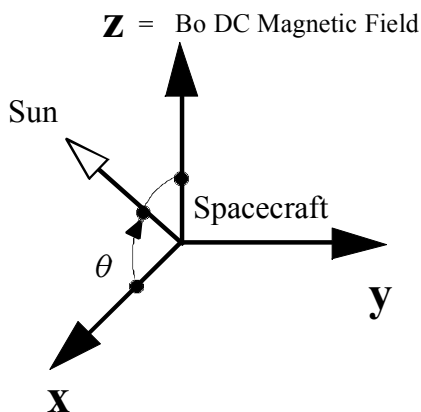
*The Y-axis is perpendicular to the Earth's magnetic dipole, towards the dusk and include in the magnetic equator plane.*

**Figure 22: Definition of GSM system**

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.

### 9.3.12 Magnetic Field Aligned system (MFA)

This system is essential to study the polarization of waves. Indeed, most of the 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 (see Figure 23). It has therefore been introduced for this purpose [16, Robert, 2000].



*This is a system useful for physic, but the meaning of the Bo DC magnetic field must be knew, as its time variation (see ref. [16] ).*

*The Z-axis is the DC magnetic field vector.*

*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 MFA system move continuously with the time variation of the DC magnetic field.*

**Figure 23: Definition of MFA system**

## 9.4. ESTIMATION DU CHAMP CONTINU DANS LE PLAN DE SPIN.

### 9.4.1 Problématique

La rotation des capteurs magnétiques dans le champ continu élevé engendre une composante sinusoïdale forte, d'amplitude  $B_{\perp}$  dans le plan de spin. Il est donc possible, en mesurant l'amplitude et la phase de cette sinusoïde, de recalculer les composantes  $B_x$  et  $B_y$  de ce champ, dans le repère tournant SR comme dans le repère fixe SR2.

Lorsque l'on applique la procédure de despin par la routine `desinus` (voir chapitre 8.2.2) on récupère en sortie l'amplitude en Volts et la phase de la sinusoïde de spin pour chaque composante, soit  $\tilde{B}_x, \tilde{\varphi}_x$  et  $\tilde{B}_y, \tilde{\varphi}_y$  ainsi que celle de la composante  $B_z, \tilde{B}_z, \tilde{\varphi}_z$  qui, bien que faible, permettra le calcul de l'angle de dépointage (voir chapitre 9.5). Les valeurs  $\tilde{B}_x$  et  $\tilde{B}_y$  sont deux estimations différentes du champ perpendiculaire  $B_{\perp}$ , en valeur non calibrées (en V), de même que  $\tilde{\varphi}_x$  et  $\tilde{\varphi}_y$  sont deux estimations de la phase  $\varphi$  dans le repère tournant SR.

La Figure 24 illustre cette mesure dans le repère tournant des antennes.

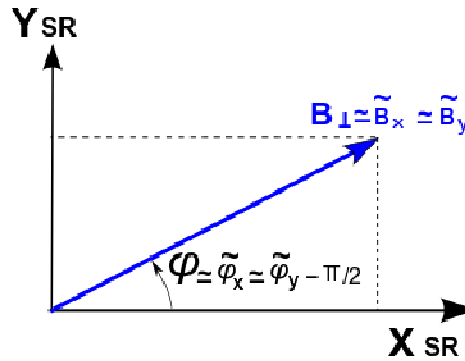


Figure 24: Mesure du champ perpendiculaire et de la phase dans le plan de spin

En associant à ces valeurs le coefficient complexe de la fonction de transfert à la fréquence de spin, en tenant compte d'une part de la position des antennes par rapport aux axes du « Body-Build System » (voir § 8.4.4) et d'autre part du « spin phase » défini au chapitre 9.2, on peut reconstituer les composantes  $x$  et  $y$  du champ  $B_{\perp}$  dans le repère fixe SR2.

### 9.4.2 Amplitude et phase de $B_{\perp}$ en repère tournant SR

A partir des deux estimations  $\tilde{B}_x$  et  $\tilde{B}_y$  de l'amplitude (en V) on peut calculer deux estimations  $B_{\perp x}$  et  $B_{\perp y}$  de l'amplitude en nT par les formules :

$$B_{\perp x} = \tilde{B}_x \frac{1}{|\alpha_x(f_s)|} \quad B_{\perp y} = \tilde{B}_y \frac{1}{|\alpha_y(f_s)|}$$

Où  $|\alpha_x(f_s)|$  et  $|\alpha_y(f_s)|$  sont les modules de la fonction de transfert à la fréquence de spin pour chacune des deux composantes X et Y.

De même on obtient deux estimations différentes de l'angle  $\varphi$  en corrigeant les phases mesurées par la fonction de transfert :

$$\begin{aligned}\varphi_x &= \widetilde{\varphi}_x - \text{phase}[\alpha_x(f_s)] \\ \varphi_y &= \widetilde{\varphi}_y - \text{phase}[\alpha_y(f_s)]\end{aligned}$$

#### 9.4.3 Tests sur $B_{\perp x}$ et $B_{\perp y}$

En principe  $B_{\perp x}$  et  $B_{\perp y}$  devraient être égaux, même si les fonctions de transfert sont légèrement différentes. De même la différence entre  $\varphi_x$  et  $\varphi_y$  devrait être de  $90^\circ$ .

On peut donc effectuer un test pour voir si ces hypothèses sont vraies ou non. La mesure des différences permettra d'estimer la matrice de dépointage des antennes. Le calcul complet de la matrice de dépointage est assez complexe, et sera donné dans un document à venir.

Par contre, dans la mesure où le capteur Z est proche de l'axe de spin, la mesure de l'amplitude  $\tilde{B}_z$  permet d'estimer le « misalignment angle » dont le calcul est donné au chapitre 9.5.

#### 9.4.4 Amplitude et phase de $B_{\perp}$ en repère fixe SR2

Dans la mesure où on a deux estimations du champ  $B_{\perp}$  et de l'angle de phase  $\varphi$  il faut en proposer une seule. Il est naturel de prendre comme valeur de  $B_{\perp}$  la valeur moyenne:

$$B_{\perp} = \frac{1}{2}(B_{\perp x} + B_{\perp y})$$

Pour la phase on décide de prendre comme phase de référence  $\varphi_x$  (ce choix est arbitraire, on aurait pu prendre  $\varphi_y$ ).

Pour obtenir les valeurs  $B_x$  et  $B_y$  du champ DC dans le repère SR2, il suffit de faire une rotation d'angle  $\psi$ . Le calcul de cet angle est fait dans le chapitre 9.3.6 et on a :

$$\psi = (\omega_s t + \varphi_s + \alpha).$$

Les valeurs retenues pour le champ continu dans le plan de spin sont alors :

$$\begin{aligned}B_x^{DC} &= B_{\perp} \sin(\varphi_x + \psi) \\ B_y^{DC} &= B_{\perp} \cos(\varphi_x + \psi)\end{aligned}$$

Par cette méthode, on a préservé le fait que l'amplitude est la même sur les deux composantes, et que celles-ci sont bien déphasées de  $90^\circ$ .

## 9.5. ESTIMATION DU “MISALIGNMENT ANGLE”

Le “misalignment angle” est l’angle (faible) entre le véritable axe de rotation du satellite et l’antenne Z du search-coils. Ce chapitre est issu d’un document « Waveform calibration method for no-linear transfer function » de P. Robert, daté du 11 mai 2007, qui avait été fait pour Themis, mais qui n’as jamais été achevé. Écrit initialement en anglais, ce chapitre n’as pas été retraduis ici.

### 9.5.1 Definitions

Situation is shown by the Figure 25 hereafter : (X,Y,Z) is the SR2 or “Despun” System.

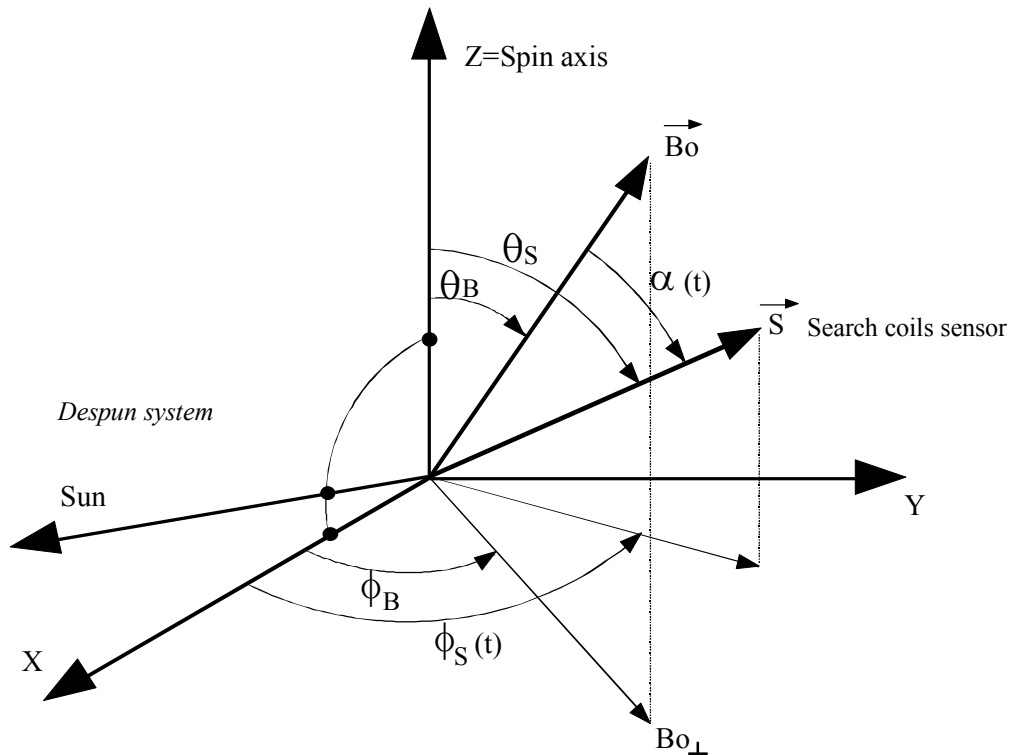


Figure 25: Définition des repères pour un capteur en rotation

In the SR2, or “Despun” system (DS), the Z axis of the spacecraft is along the spin axis. The sensor  $\underline{S}$  (which can be the X, Y, or Z sensors) is rotating around the spin axis with a constant angle  $\theta_S$ . The DC field  $\underline{B}_0$  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  $\theta_z$  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.

### 9.5.2 Signal delivered by a rotating antenna into a DC field

Vectors  $\underline{S}$  and  $\underline{B}_0$  have for components:

$$\underline{S} = \begin{pmatrix} \sin \theta_S \cos \phi_S \\ \sin \theta_S \sin \phi_S \\ \cos \theta_S \end{pmatrix} \quad \underline{B}_0 = \begin{pmatrix} B_0 \sin \theta_B \cos \phi_B \\ B_0 \sin \theta_B \sin \phi_B \\ B_0 \cos \theta_B \end{pmatrix}$$

The signal delivered by the sensor S is  $B_0 \cos \alpha(t) = \underline{B}_0 \cdot \underline{S}$ , 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  $\theta_B$  and  $\theta_S$  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:

$$S(t) = B_{0\perp} \sin \theta_S \cos (\phi_B - \phi_S(t)) \quad \text{with } B_{0\perp} = B_0 \sin \theta_B \quad (\text{no time-dependent})$$

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,  $\theta_S$  is no time-dependant and  $\phi_S(t)$  can be expressed as :

$$\phi_S(t) = 2\pi F_s t + \phi_o$$

Therefore we have:

$$S(t) = B_{0\perp} \sin \theta_S \cos (2\pi F_s t + \phi_o - \phi_B)$$

So, the  $S(t)$  signal is a pure sine wave, with amplitude equal to  $B_{0\perp} \sin \theta_S$

### 9.5.3 Estimate of the misalignment angle $\theta_z$ 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  $a_z = B_{0\perp} \sin \theta_z$  and a phase of  $(\phi_o - \phi_B)$ . 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  $\theta_z$  by :

$$\sin \theta_z \approx a_z / a_x$$

### 9.5.4 Estimate of the misalignment angle $\theta_z$ for high value

So we have (1) :

$$\begin{aligned} S_x(t) &= B_{0\perp} \sin \theta_x \cos (2\pi F_s t + \phi_{ox} - \phi_B) \\ S_y(t) &= B_{0\perp} \sin \theta_y \cos (2\pi F_s t + \phi_{oy} - \phi_B) \\ S_z(t) &= B_{0\perp} \sin \theta_z \cos (2\pi F_s t + \phi_{oz} - \phi_B) \end{aligned}$$

- If the three-axis of the sensors is an orthogonal system, we have:

$$\underline{S}_x \cdot \underline{S}_y = 0 \quad \underline{S}_x \cdot \underline{S}_z = 0 \quad \underline{S}_y \cdot \underline{S}_z = 0$$

After computation, we found **(2)** :

$$\begin{aligned} \cos(\phi_x - \phi_y) &= -\cos \theta_x \cos \theta_y / \sin \theta_x \sin \theta_y = -1 / \operatorname{tg} \theta_x \operatorname{tg} \theta_y \\ \cos(\phi_x - \phi_z) &= -\cos \theta_x \cos \theta_z / \sin \theta_x \sin \theta_z = -1 / \operatorname{tg} \theta_x \operatorname{tg} \theta_z \\ \cos(\phi_y - \phi_z) &= -\cos \theta_y \cos \theta_z / \sin \theta_y \sin \theta_z = -1 / \operatorname{tg} \theta_y \operatorname{tg} \theta_z \end{aligned}$$

with

$$\phi_x(t) = 2\pi F_s t + \phi_{ox} \quad \phi_y(t) = 2\pi F_s t + \phi_{oy} \quad \phi_z(t) = 2\pi F_s t + \phi_{oz}$$

- Always with the properties of an orthogonal system, we have:

$$\underline{S}_x = \underline{S}_y \cdot \underline{S}_z \quad \underline{S}_y = \underline{S}_z \cdot \underline{S}_x \quad \underline{S}_z = \underline{S}_x \cdot \underline{S}_y$$

After computation, we found **(3)** :

$$\begin{aligned} \sin(\phi_z - \phi_y) &= \cos \theta_x / \sin \theta_y \sin \theta_z \\ \sin(\phi_x - \phi_z) &= \cos \theta_y / \sin \theta_z \sin \theta_x \\ \sin(\phi_y - \phi_x) &= \cos \theta_z / \sin \theta_x \sin \theta_y \end{aligned}$$

- By using **(2)** and **(3)**, we can eliminate the  $\phi_i$  angles, and we obtain a system of 3 equations depending of only  $\theta_i$  **(4)** :

$$\begin{aligned} \cos^2 \theta_x &= -\cos(\theta_y - \theta_z)(\theta_y + \theta_z) & \sin^2 \theta_x &= 1 + \cos(\theta_y - \theta_z)(\theta_y + \theta_z) \\ \cos^2 \theta_y &= -\cos(\theta_z - \theta_x)(\theta_z + \theta_x) & \sin^2 \theta_y &= 1 + \cos(\theta_z - \theta_x)(\theta_z + \theta_x) \\ \cos^2 \theta_z &= -\cos(\theta_x - \theta_y)(\theta_x + \theta_y) & \sin^2 \theta_z &= 1 + \cos(\theta_x - \theta_y)(\theta_x + \theta_y) \end{aligned}$$

- 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  $\theta_x$  and  $\theta_y$ .

• At last, by solving the 3 previous systems, we can compute the 3 angles  $\theta_x$ ,  $\theta_y$ ,  $\theta_z$ , 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  $\theta_x \approx \theta_y \approx 90^\circ$ , we found the previous approximation  $\sin \theta_z \approx a_z / a_x$

Likewise, if  $\theta_z$  is equal to zero, from **(5)**  $\theta_x = \theta_y = 90^\circ$ , and  $a_x = a_y = B_{o\perp}$

Nevertheless, if  $\theta_z$  is not equal to zero, and  $\theta_x \neq \theta_y$ .

Finally, that we call the misalignment angle is  $\theta_z$  given by:

$$\theta_z = \sin^{-1} \left( \frac{a_z \sqrt{2}}{(a_x^2 + a_y^2 + a_z^2)^{1/2}} \right)$$



## 9.6. CALCUL DE LA MATRICE DE CORRECTION DU DÉPOINTAGE

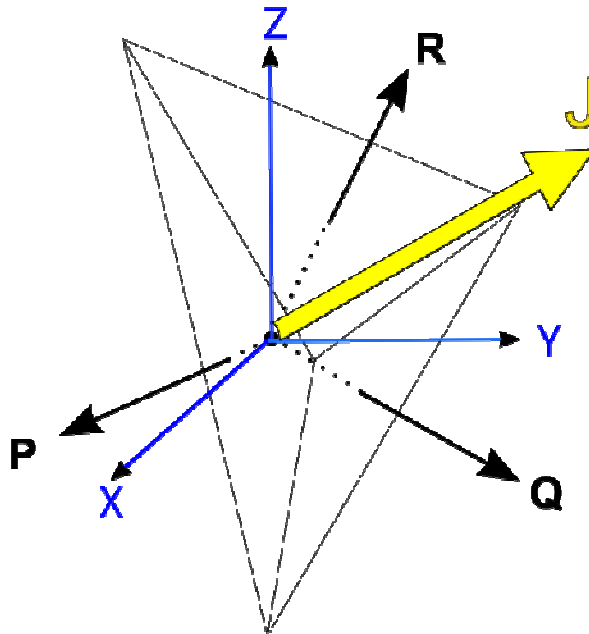
### 9.6.1 *Passage de coordonnées non orthogonales à un système orthogonal*

*Ce chapitre est extrait d'un document de P. Robert du 20 janvier 1994. Merci à G. Chanteur de l'avoir relu, vérifié et suggéré une présentation plus claire que celle d'origine.*

#### • *Position du problème*

Le problème est identique à celui qui se pose lors de la mission CLUSTER pour calculer le courant mesuré par les 4 magnétomètres disposés en tétraèdre, en utilisant la méthode basée sur le théorème d'Ampère.

Dans cette méthode, on mesure la densité de courant  $J$  selon 3 directions non orthogonales qui sont celles des normales aux 3 faces choisies (parmi 4 combinaisons possibles) d'un tétraèdre quelconque mais non dégénéré (ni plat, ni linéaire) par la formule  $\mu_0 I = \oint \vec{B} \cdot d\vec{l}$  sur chacune des faces choisies. La Figure 26 ci-dessous résume le problème.



**Figure 26 :** *Projection du vecteur J sur 3 vecteurs (P, Q, R) non orthogonaux*

Le problème qui se pose ici est analogue : à partir des 3 mesures  $B_{pm}$ ,  $B_{qm}$ ,  $B_{rm}$  faites par les capteurs magnétiques sur des axes de mesure non orthogonaux  $\vec{P}$ ,  $\vec{Q}$ ,  $\vec{R}$  on souhaite recalculer les composantes cartésiennes du repère orthonormé  $\vec{X}$ ,  $\vec{Y}$ ,  $\vec{Z}$ , soit les composantes  $B_x$ ,  $B_y$ ,  $B_z$ .

Au chapitre 9.3.1 le système de mesure non orthogonal (P,Q,R) est appelé « **Sensor Coordinate System** » (SCS), tandis que le repère (X,Y,Z) est appelé « **Orthogonal Sensor System** » (OSS).

• *Passage d'un système non orthogonal à un système orthogonal*

On désigne par  $\vec{P}, \vec{Q}, \vec{R}$  les normales aux trois faces considérées, ces normales n'étant ni coplanaires, ni colinéaires. Leur direction est supposée connue dans le repère orthonormé de référence  $\vec{X}, \vec{Y}, \vec{Z}$  comme :

$$\vec{P} = \begin{pmatrix} P_x \\ P_y \\ P_z \end{pmatrix} \quad \vec{Q} = \begin{pmatrix} Q_x \\ Q_y \\ Q_z \end{pmatrix} \quad \vec{R} = \begin{pmatrix} R_x \\ R_y \\ R_z \end{pmatrix} \quad [1]$$

Les composantes de  $\vec{B}$  dans le repère non orthogonal  $\vec{P}, \vec{Q}, \vec{R}$  sont :

$$\vec{B} = \begin{pmatrix} B_p \\ B_q \\ B_r \end{pmatrix} = B_p \vec{P} + B_q \vec{Q} + B_r \vec{R} \quad [2]$$

En remplaçant dans [2] les vecteurs par leur expression donnée en [1] on obtient :

$$\vec{B} = B_p \begin{pmatrix} P_x \\ P_y \\ P_z \end{pmatrix} + B_q \begin{pmatrix} Q_x \\ Q_y \\ Q_z \end{pmatrix} + B_r \begin{pmatrix} R_x \\ R_y \\ R_z \end{pmatrix}$$

Ce qui s'exprime sous la forme matricielle par :

$$\vec{B} = \begin{pmatrix} B_x \\ B_y \\ B_z \end{pmatrix} = \begin{pmatrix} P_x & Q_x & R_x \\ P_y & Q_y & R_y \\ P_z & Q_z & R_z \end{pmatrix} \begin{pmatrix} B_p \\ B_q \\ B_r \end{pmatrix} \quad [3]$$

On trouve ainsi la matrice de passage du repère non orthogonal  $\vec{P}, \vec{Q}, \vec{R}$  au repère orthonormé  $\vec{X}, \vec{Y}, \vec{Z}$

Mais **ATTENTION** ! Les composantes  $B_p, B_q, B_r$  **NE SONT PAS** les mesures expérimentales de  $\vec{B}$  par les capteurs le long des axes  $\vec{P}, \vec{Q}, \vec{R}$ , comme on peut le voir sur la Figure 27 ci-dessous.

Il faudra donc procéder autrement.

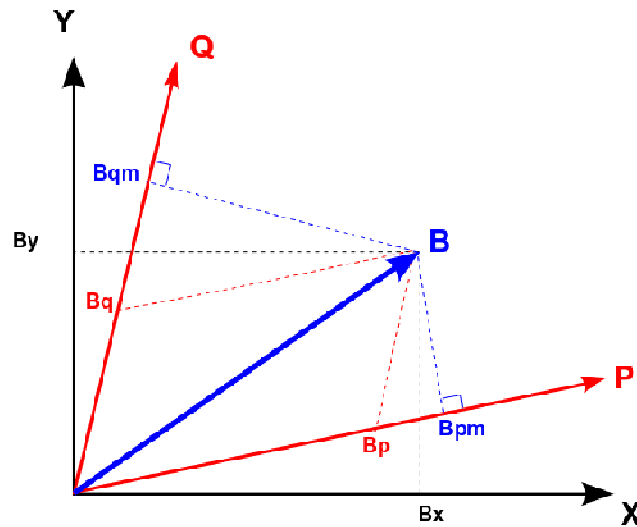


Figure 27: Différence entre les composantes mathématique de  $B$  et les mesures

• **Calcul des composantes cartésiennes XYZ du champ à partir des mesures sur PQR**

Si on appelle  $B_{pm}, B_{qm}, B_{rm}$  les composantes mesurées de  $\vec{B}$  dans le repère  $\vec{P}, \vec{Q}, \vec{R}$  on peut écrire :

$$\begin{aligned} B_{pm} &= \vec{P} \cdot \vec{B} = P_x B_x + P_y B_y + P_z B_z \\ B_{qm} &= \vec{Q} \cdot \vec{B} = Q_x B_x + Q_y B_y + Q_z B_z \\ B_{rm} &= \vec{R} \cdot \vec{B} = R_x B_x + R_y B_y + R_z B_z \end{aligned}$$

Ce système linéaire s'écrivant sous la forme matricielle comme:

$$\vec{B} = \begin{pmatrix} B_{pm} \\ B_{qm} \\ B_{rm} \end{pmatrix} = \begin{pmatrix} P_x & P_y & P_z \\ Q_x & Q_y & Q_z \\ R_x & R_y & R_z \end{pmatrix} \begin{pmatrix} B_x \\ B_y \\ B_z \end{pmatrix}$$

Ce qui correspond à une matrice de passage  $M = (P_i, Q_i, R_i)$  des valeurs de  $\vec{B}$  dans le système orthonormé (X,Y,Z) (ce que l'on cherche) vers les valeurs de  $\vec{B}$  projetées sur les axes  $\vec{P}, \vec{Q}, \vec{R}$  (ce que l'on mesure). Il faudra donc faire l'opération inverse.

• **Définition de la matrice de dépointage**

La matrice  $M = (P_i, Q_i, R_i)$  qui donne la direction des senseurs dans un repère cartésien est dite « matrice de dépointage ». Dans l'application pratique, il faut bien sûr connaître cette matrice **M** si on veut pouvoir faire la correction du dépointage. Elle ne peut qu'avoir été proprement mesurée au sol avant le tir.

Note: Dans le cas où le système de mesure de départ  $(\vec{P}, \vec{Q}, \vec{R})$  est déjà orthonormé, et si on a  $\vec{P} = \vec{X}$ ,  $\vec{Q} = \vec{Y}$ , et  $\vec{R} = \vec{Z}$  la matrice de dépointage se réduit à la matrice unitaire et on a bien sûr :

$$\vec{B} = \begin{pmatrix} B_{pm} \\ B_{qm} \\ B_{rm} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} B_x \\ B_y \\ B_z \end{pmatrix}$$

En l'absence de données suffisamment précise sur la direction des antennes par rapport au système OSS pour les capteurs de CLUSTER/STAFF, c'est la matrice unitaire qui a été appliquée (voir [13], Robert, 2013).

### 9.6.2 Matrice de correction du dépointage

On souhaite donc faire l'opération inverse, c'est à dire déterminer  $\vec{B}$  dans le repère orthonormé XYZ où sont connus les vecteurs  $\vec{P}, \vec{Q}, \vec{R}$ . Il suffit donc de calculer la matrice inverse de M, soit :

$$M' = \begin{pmatrix} U_p & U_q & U_r \\ V_p & V_q & V_r \\ W_p & W_q & W_r \end{pmatrix} \text{ inverse de } M = \begin{pmatrix} P_x & P_y & P_z \\ Q_x & Q_y & Q_z \\ R_x & R_y & R_z \end{pmatrix}$$

Et on obtiendra le résultat cherché :

$$\vec{B} = \begin{pmatrix} B_x \\ B_y \\ B_z \end{pmatrix} = \begin{pmatrix} U_p & U_q & U_r \\ V_p & V_q & V_r \\ W_p & W_q & W_r \end{pmatrix} \begin{pmatrix} B_{pm} \\ B_{qm} \\ B_{rm} \end{pmatrix}$$

La matrice **M'** est dite « matrice de correction du dépointage ».

• **Calcul de la matrice de dépointage inverse**

On procède en quatre étapes :

- 1) calcul de la matrice transposée :

$$M^t = \begin{pmatrix} P_x & Q_x & R_x \\ P_y & Q_y & R_y \\ P_z & Q_z & R_z \end{pmatrix}$$

- 2) remplacement de chaque terme par son déterminant, et changement de signe pour les termes dont la somme des indices est impair :

$$U_p' = (Q_y R_z - Q_z R_y)$$

$$U_q' = -(Q_x R_z - Q_z R_x)$$

$$U_r' = (Q_x R_y - Q_y R_x)$$

$$V_p' = -(P_y R_z - P_z R_y)$$

$$V_q' = (P_x R_z - P_z R_x)$$

$$V_r' = -(P_x R_y - P_y R_x)$$

$$W_p' = (P_y Q_z - P_z Q_y)$$

$$W_q' = -(P_x Q_z - P_z Q_x)$$

$$W_r' = (P_x Q_y - P_y Q_x)$$

- 3) calcul du déterminant

$$D = P_x Q_y R_z + P_y Q_z R_x + P_z Q_x R_y - P_z Q_y R_x - P_x Q_z R_y - P_y Q_x R_z$$

- 4) normalisation de chaque terme par le déterminant de la matrice transposée :

Soit donc :

$$M' = \begin{pmatrix} U_p & U_q & U_r \\ V_p & V_q & V_r \\ W_p & W_q & W_r \end{pmatrix} = \frac{1}{D} \begin{pmatrix} U_p' & U_q' & U_r' \\ V_p' & V_q' & V_r' \\ W_p' & W_q' & W_r' \end{pmatrix}$$

Note: Dans le cas où le système de mesure de départ  $(\vec{P}, \vec{Q}, \vec{R})$  est déjà orthonormé, la matrice inverse  $M'$  se déduit immédiatement de la matrice  $M$  par :

$$M' = M^t = \begin{pmatrix} P_x & Q_x & R_x \\ P_y & Q_y & R_y \\ P_z & Q_z & R_z \end{pmatrix}$$

• **Exemple d'application: cas des antennes électriques d'Interball**

La direction des axes de mesure sont déduites des coordonnées des boules en cm dans le repère cartésien X,Y,Z du satellite (informations S. Perault, 2002), soit:

$$B_{1x} = \begin{pmatrix} 2195 \\ 0 \\ -228 \end{pmatrix} \quad B_{2x} = \begin{pmatrix} 2125 \cdot \cos(15^\circ) \\ -2125 \cdot \sin(15^\circ) \\ 228 \end{pmatrix}$$

On en déduit l'axe  $\mathbf{P}'$  non normalisé :

$$\mathbf{P}' = \begin{pmatrix} 2195 + 2125 \cdot \cos(15^\circ) \\ 2125 \cdot \sin(15^\circ) \\ -456 \end{pmatrix} = \begin{pmatrix} 4247.6 \\ 549.99 \\ -456 \end{pmatrix}$$

$\mathbf{P}$  est déduit de  $\mathbf{P}'$  après normalisation, avec  $N=4307.26$ , soit  $\mathbf{P} = \begin{pmatrix} 0.9860 \\ 0.1277 \\ 0.1059 \end{pmatrix}$

Les axes  $\mathbf{Q}$  et  $\mathbf{R}$  étant identiques aux axes  $Y$  et  $Z$ , on a :

$$\mathbf{Q} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \quad \mathbf{R} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Et la matrice de correction du dépointage est donc :

$$\mathbf{M}' = \begin{pmatrix} U_p & U_q & U_r \\ V_p & V_q & V_r \\ W_p & W_q & W_r \end{pmatrix} = \frac{1}{P_x} \begin{pmatrix} 1 & 0 & 0 \\ -P_y & P_x & 0 \\ -P_z & 0 & P_x \end{pmatrix}$$

Soit :

$$\mathbf{M}' = \begin{pmatrix} 1 & 0 & 0 \\ -0.1295 & 1 & 0 \\ -0.1074 & 0 & 1 \end{pmatrix}$$

On voit donc qu'il apparait des termes diagonaux faibles mais non nuls dus à la non orthogonalité du repère initial des senseurs.

#### • Estimation expérimentale de certains termes

Selon les conventions du chapitre 9.3.1 le système de mesure non orthogonal ( $\mathbf{P}, \mathbf{Q}, \mathbf{R}$ ) est appelé « **Sensor Coordinate System** » (SCS), tandis que le repère ( $X, Y, Z$ ) est appelé « **Orthogonal Sensor System** » (OSS). On a vu que pour CLUSTER/STAFF, on a fait l'approximation :

$$\mathbf{SCS\_to\_OSS} \cong \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Alors que dans le calcul exact il faut remplacer la matrice unité par la matrice de dépointage, soit :

$$\mathbf{SCS\_to\_OSS} = \begin{pmatrix} U_p & U_q & U_r \\ V_p & V_q & V_r \\ W_p & W_q & W_r \end{pmatrix}$$

Ce qui nous donne, pour le passage au DSS (le Data Sensor System , voir chapitre 9.3.3)

$$\mathbf{OSS\_to\_DSS} = \begin{pmatrix} U_r & U_p & U_q \\ V_r & V_p & V_q \\ W_r & W_p & W_q \end{pmatrix}$$

Le passage du DSS au BBS est plus délicat. Au chapitre 9.3.4 on a négligé le petit angle entre l'axe  $Z$  du DSS et l'axe  $X$  du BBS en faisant l'hypothèse  $\vec{Z}_{DSS} \approx \vec{X}_{BBS}$ .

En vérité, il existe aussi un petit dépointage entre l'axe du bras portant l'instrument et le Body Build, et la formule du chapitre 9.3.4 devrait s'écrire avec une matrice de correction supplémentaire comme :

$$DSS\_to\_BBS = \begin{pmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \end{pmatrix}$$

En admettant que le passage du repère du bras au repère du Body Build soit entre repère orthogonaux.

Dans ce cas la matrice  $D = \begin{pmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \end{pmatrix}$  est une matrice de rotation proche de l'unité.

On as donc  $a_x \simeq b_y \simeq c_z \simeq 1$

tandis que  $a_y \simeq a_z \simeq \varepsilon$

$$b_x \simeq b_z \simeq \varepsilon$$

$$c_x \simeq c_y \simeq \varepsilon$$

De même comme on a :

$$OSS\_to\_DSS = \begin{pmatrix} U_r & U_p & U_q \\ V_r & V_p & V_q \\ W_r & W_p & W_q \end{pmatrix} \simeq \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

On peut affirmer que  $W_r \simeq U_p \simeq V_q \simeq 1$

tandis que  $U_r \simeq U_q \simeq \varepsilon$

$$V_r \simeq V_p \simeq \varepsilon$$

$$W_p \simeq W_q \simeq \varepsilon$$

Si l'enjeu en vaut la peine, on peu faire le calcul complet à l'ordre 1, en négligeant les termes en  $\varepsilon^2$  afin de trouver la matrice de correction totale. On peut ensuite la comparer avec le « misalignment angle » mesuré expérimentalement par la formule du chapitre 9.3.5.

Expérimentalement, les valeurs différentes de l'amplitude des deux composantes  $B_{xs}, B_{ys}$  de la sinusoïde de spin mesurée dans le repère SCS, ainsi que l'écart de leur différence de phase avec la valeur théorique de  $90^\circ$ , devrait pouvoir permettre une estimation expérimentale de la matrice de dépointage totale : 3 angles à déterminer, à partir de 3 paramètres, le problème est donc a priori soluble.

Néanmoins, pour CLUSTER/STAFF, le misalignment angle étant de l'ordre de  $0.5^\circ$ , et en tout cas inférieur au degré, ce calcul complet n'as pas été fait.

## 9.7. PETITS RAPPELS SUR LA TRANSFORMÉE DE FOURIER

*La littérature sur la transformée de Fourier est abondante et diversifiée. Ce qui est donné ici est un simple rappel de base de ce qu'on doit savoir pour le calcul de spectre ou spectrogramme.*

### 9.7.1 La transformée de Fourier au sens mathématique

La transformée de Fourier (TF) permet de passer du domaine « temps », c'est à dire d'une fonction dépendant du temps  $x(t)$ , que l'on appelle aussi « forme d'onde », au domaine « fréquence », c'est-à-dire une fonction dépendant de la fréquence  $X(f)$ , que l'on appelle « spectre ».

Pour que ce soit possible, il faut que la fonction  $x(t)$  soit intégrable sur les réel  $\mathbb{R}$ .

Cette opération se fait par la formule :

$$X(f) = \int_{-\infty}^{+\infty} x(t) e^{-2i\pi f t} dt$$

À partir du spectre  $X(f)$  on peut revenir à la forme d'onde par :

$$x(t) = \int_{-\infty}^{+\infty} X(f) e^{+2i\pi f t} df$$

Bien sur, en pratique on ne connaît jamais un signal  $x(t)$  sur une durée infinie. Si on suppose la fonction  $x(t)$  périodique, et de période  $T$ , on peut calculer son spectre par :

$$X(f) = \frac{1}{T} \int_{-T/2}^{+T/2} x(t) e^{-2i\pi f t} dt$$

Ce qui nous rapproche déjà un peu du cas réel. Mais avec les hypothèses de périodicité, attention aux effets de bords ! Il faut savoir qu'une troncature abrupte entraînera du bruit sur toute la gamme de fréquence qui n'aura pas de signification (voir Figure 30 du chapitre 9.7.3 page 168).

En pratique, les signaux sont échantillonnés et on utilise la TF discrète.

### 9.7.2 La transformée de Fourier discrète

En traitement du signal numérique, la fonction  $x(t)$  est échantillonnée à la fréquence  $f_e$ , dite « fréquence d'échantillonnage », et la fonction  $x(t)$  devient un ensemble de valeurs  $x_n$ . Dans ce cas, si le point  $n=0$  est l'origine des temps, le point  $n$  est situé au temps  $t_n = n \delta t$  avec  $\delta t = 1/f_e$ .

Il faut noter que la fréquence  $f_e$  doit être choisie de telle manière (théorème de Shannon) qu'il n'existe aucune fréquence supérieure à  $f_e/2$  dans le signal d'origine  $x(t)$ . En pratique, lors de la numérisation par des CAD (Convertisseur Analogique/Digital), il est mis en place dans le hardware un filtre anti-aliasing pour éviter ce problème.

La transformée de Fourier discrète s'exprime alors par :

$$X_k = \frac{1}{N} \sum_{n=0}^{N-1} x_n e^{-2i\pi n k / N}$$

$X_k$  est alors le spectre de la forme d'onde  $x_n$ . Comme le spectre est discret, on parle aussi de spectre de raies.

La transformée inverse est bien sûr :

$$x_n = \sum_{k=-N/2}^{N/2-1} X_k e^{-2i\pi n k / N}$$

### 9.7.3 La « Fast Fourier Transform »

C'est un algorithme rapide de calcul de la transformée de Fourier discrète. Celle qui est utilisée dans les RCL est du type « Cooley-Tukey » du nom de ses inventeurs. Il utilise l'algorithme dit « du papillon » et demeure une FFT extrêmement rapide. Son seul défaut est de fonctionner sur un nombre de point qui est de la forme  $N = 2^m$ , avec m entier.

C'est cet algorithme qui a été utilisé pour écrire la sous-routine de déconvolution `deconvo_C3` utilisée dans le programme de calibration continue (voir chapitres 7.6.1 et 8.3).

#### • Les fréquences négatives et positives

Dans les programmes de calcul de FFT, on introduit généralement un argument d'entrée-sortie **Sn** qui est un tableau complexe de dimension N. En entrée il correspond à une forme d'onde (qui peut être complexe)  $x_n$ , et en sortie il correspond au spectre complexe  $X_k$ .

Les raies du spectre en sortie sont généralement classées selon la figure Figure 28 ci-dessous :

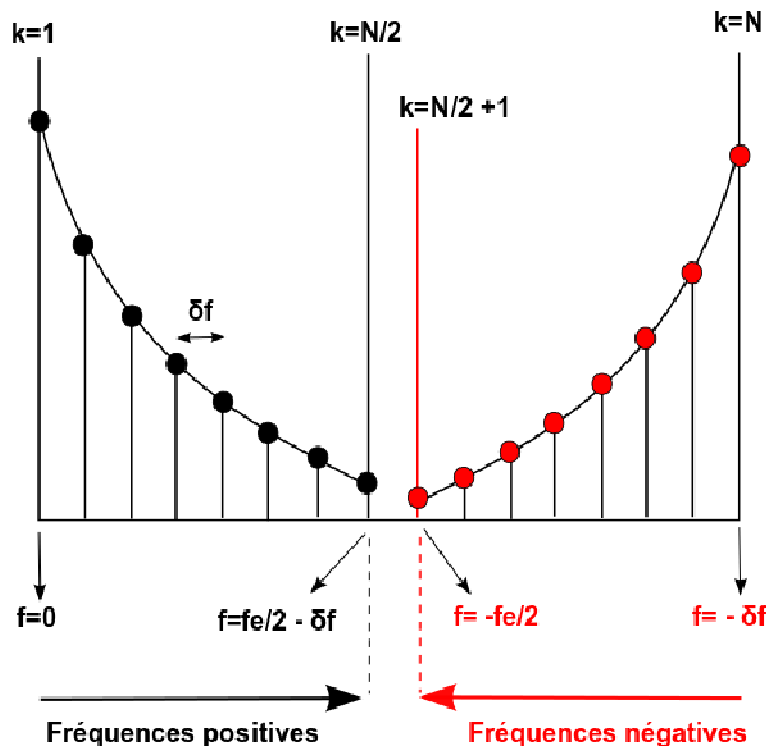


Figure 28: Ordre des raies du spectre en sortie de FFT



• **Quelques relations utiles :**

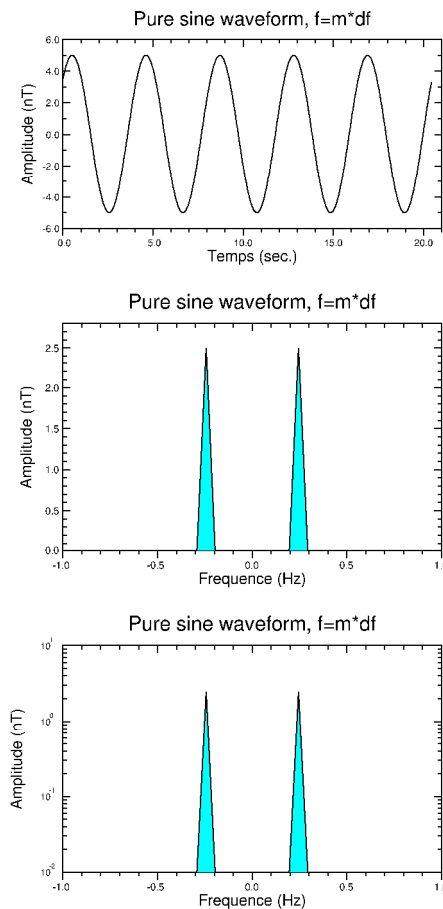
$\delta t$  : période d'échantillonnage,  $\delta f$  : résolution spectrale,  $T$  : Longueur de la fenêtre de  $N$  points.

$$\delta t = \frac{1}{f_e} = \frac{T}{N} \quad \delta f = \frac{1}{T} = \frac{f_e}{N} \quad T = N\delta t = \frac{N}{f_e} \quad f_{max} = \frac{f_e}{2}$$

$f_{max}$ , fréquence la plus élevée du spectre, est dite fréquence de Nyquist.

• **Test de normalisation d'une FFT**

Si on prend comme signal d'entrée une fonction réelle, le spectre doit être symétrique et conjugué, ce qui veut dire que l'amplitude sur les fréquences négatives et positives doit être la même, et les phases inversées. Pour vérifier qu'une FFT est bien normalisée, il suffit de prendre comme signal d'entrée une sinusoïde pure, d'amplitude  $a = 0.5$  comme dans l'exemple de la Figure 29 ci-dessous (l'amplitude du spectre est donnée en échelle linéaire et logarithmique).



**Figure 29:** Spectre d'une sinusoïde pure dont la fréquence est un multiple de  $\delta f$

Si on prend la précaution de prendre une fréquence  $f_{test}$  multiple de  $\delta f$ , on respecte bien l'hypothèse selon laquelle le signal est périodique (si on le répète d'une fenêtre à l'autre, il y a continuité).

Dans ce cas on obtient deux raies pures, à  $\pm \delta f$ , de même amplitude égale à  $a/2$ . On peut vérifier que l'énergie en temps comme en fréquence est bien conservée : en temps, l'amplitude efficace d'une sinusoïde est  $a/\sqrt{2}$  soit une énergie de  $a^2/2$ .

En fréquence, on obtient  $2(a/2)^2$  soit bien  $a^2/2$ . C'est le théorème de Parseval :

$$E = \int_{-\infty}^{+\infty} |x(t)|^2 dt = \int_{-\infty}^{+\infty} |X(f)|^2 df.$$

Ce théorème est parfois bien utile pour vérifier la conservation de l'énergie lors de traitement complexe. Dans cet exemple, on a choisi  $f_{\text{test}} = 5 \cdot \delta f = 5 \cdot 0.048828125 = 0.2441406$  Hz et une amplitude de 0.5 nT, et on peut vérifier sur la Figure 29 qu'on obtient bien 2 raies à 0.25 nT.

Dans ce cas le calcul de phase (que l'on peut vérifier) donne aussi la valeur exacte d'origine. C'est de cette manière qu'est conçue le soft du despin : on utilise une FFT sur une seule raie (la fréquence de spin) et on prend comme longueur de l'échantillon un nombre de points correspondant exactement à une période de spin. On peut ainsi calculer l'amplitude et la phase de la sinusoïde de spin.

#### • Effets de bords

Si on refait le test ci-dessus, avec une fréquence proche mais non multiple de  $\delta f$ , l'hypothèse de périodicité du signal n'est plus vérifiée, et on a des « cassures » du signal d'une fenêtre à l'autre. La conséquence principale est que d'une part de l'énergie se trouve distribuée sur toutes les fréquences, et que d'autre part le spectre qu'on obtient n'est plus exactement celui qu'on attendait, comme le montre la Figure 30 ci-dessous. Néanmoins dans tous les cas l'énergie totale du signal est toujours conservée, ce qui fait que l'amplitude des raies est diminuée.

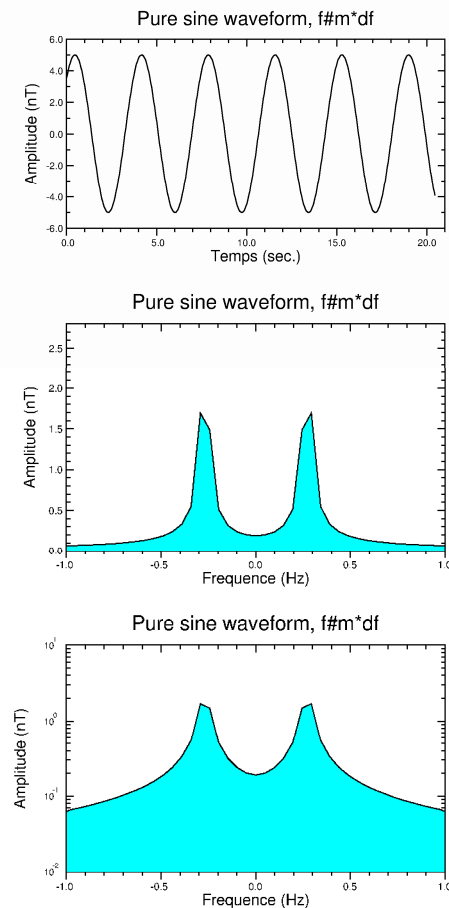


Figure 30: Spectre d'une sinusoïde pure dont la fréquence n'est pas un multiple de  $\delta f$

## 9.8. UTILITÉ DES COMPOSANTES CIRCULAIRES EN POLARISATION

### 9.8.1 Définition

On rappelle qu'une ellipse décrivant un vecteur d'onde tournant à droite par exemple, peut être décomposée en la somme de deux cercles dont l'un tourne à gauche, et l'autre à droite, ce dernier ayant un rayon plus élevée. Ce qui signifie qu'une onde elliptique plane peut être décomposée en deux ondes circulaires planes droite et gauche. L'intérêt principal de cette décomposition est de pouvoir déterminer la polarisation d'une onde plane dans le plan perpendiculaire au champ magnétique, afin de déterminer simplement le mode (gauche ou droit). L'introduction de ces composantes circulaire a été faite par K. Kodera dans sa thèse d'université sur l'analyse de signaux géophysiques non stationnaires (Thèse Université Paris 1976).

### 9.8.2 Passages en composantes circulaires Gauche et Droite

#### • Calcul depuis les formes d'onde

Les spectres gauches et droits peuvent être directement obtenu par la FFT complexe du signal  $S_{xy}(t) = x(t) + iy(t)$ . Le mode gauche est donné par les fréquences négatives, et le mode droit par les fréquences positives. Pour plus de détails, on pourra se reporter à [20], Robert, 1979.

La Figure 31 ci-dessous résume l'intérêt de cette décomposition.

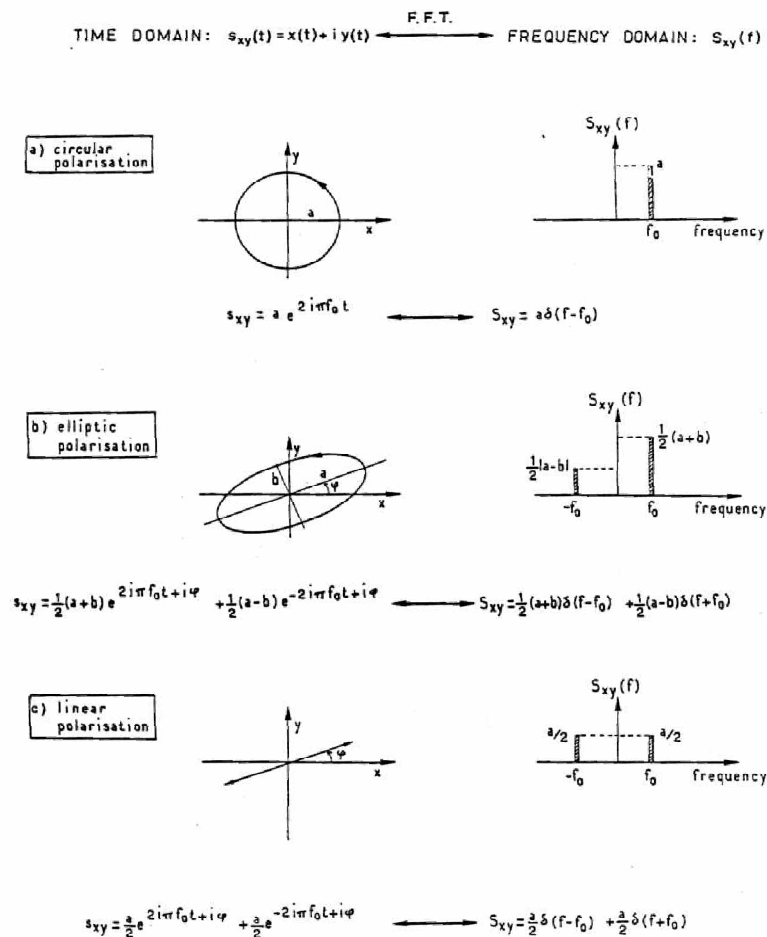


Figure 31: Transformée de Fourier du signal complexe  $S_{xy} = x + iy$  associé à une onde plane monochromatique pour 3 exemples particuliers (Robert, 1971, [20]).

• *Calcul depuis les spectres*

Si on calcule la TF de  $S_{xy}(t) = x(t) + iy(t)$  on obtient :

$$\mathcal{F}(S_{xy}(t)) = \mathcal{F}(x(t)) + i\mathcal{F}(y(t)) = X(f) + iY(f)$$

Comme la TF d'une fonction réelle est symétrique, on peut obtenir les spectres du mode gauche et du mode droit uniquement à partir des fréquences positives des spectres  $X(f)$  et  $Y(f)$  par :

$$R_+(f) = \frac{(X_+ + iY_+)}{2}$$

$$G_+(f) = \frac{(X_+^* + iY_+^*)}{2}$$

$X_+^*$  et  $Y_+^*$  étant les conjugués de la partie des fréquences positives des spectres  $X(f)$  et  $Y(f)$ . Ces formules sont utilisées dans la procédure `visu_ave_spectrum.pro` avec l'option « LR ».

• *Conséquences de la rotation sur le spectre  $S_{xy}(f)$*

Dans l'espace des coordonnées circulaires gauche et droite, l'effet Doppler de la rotation du satellite se traduit par une simple translation en fréquence, comme le montre la Figure 32 ci-dessous :

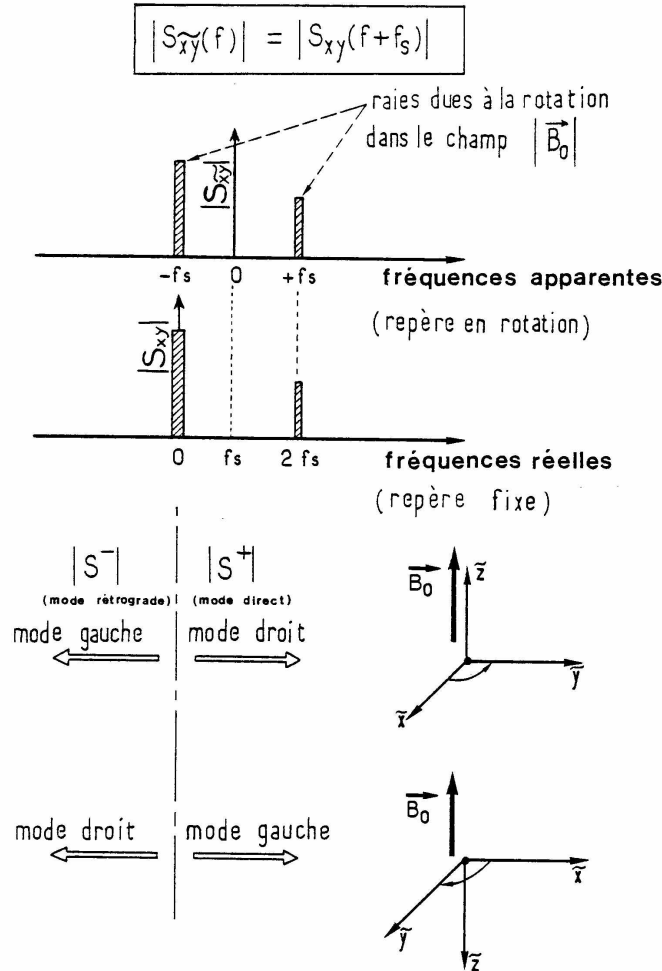


Figure 32: *Passage du repère en rotation à un repère fixe par les composantes circulaires gauche et droite (Robert, 1971, [20]).*

## 9.9. EXEMPLES DE FICHIERS RFF

### 9.9.1 Exemple de WFL1.rff

```

START ROPROC_FORMAT_FILE
#-----
# The Roproc Format File is a general format mainly used for data from
# spatial missions (magnetometer, waveform units, S/C trajectory ...).
# For readability, all lines starting with character '#' are comments,
# while empty or blank lines are ignored. Others lines are keywords,
# parameters variables or data.
# This file contents Metadata (description of the data), Constant Data
# (one value per file, or a few, but in limited number) and indexed data
# (data versus time or other INDEX).
# Any group of data begins with a "START" keyword, and stops by a "END".
# Metadata are given as parameters series (one value per parameter).
# Data are described by Metadata parameters.
# Examples of different files are given in the document:
# "The Roproc Format File, a dedicated file format for vectorial data
# processing"
# Author: P. Robert, CNRS/LPP (formerly CETP)
# V 1.0, 1996-2000 (for archive of old mission data)
# V 1.1, October 2001 (for CLUSTER/STAFF-SC NBR & HBR wave data)
# V 1.2, January 2002 (for GEOS wave data)
# V 1.3, August 2003 (for CUSP and any kind of wave data)
# V 1.4, January 2003 (for general titles management)
# V 2.0, March 2004 (for compatibility with Roproc Vector format)
# V 2.1, May 2004 (to be coherent with Cluster Exchange Format)
# V 2.2, March 2007 (some useful upgrades)
# Any comment or suggestion: Patrick.Robert@lpp.polytechnique.fr
#-----

START METADATA
#-----
# Two metadata categories: Mandatory Parameters and Optional Parameters
#-----

START MANDATORY_PARAMETERS
PAR FILE_NAME (STR): CLU1_STASC_NBR_WFL1_20010923.rff
PAR FILE_CLASS (STR): WaveForm
PAR FILE_FORMAT_VERSION (STR): Roproc_Format_File V 2.2
PAR FILE_CREATION_DATE (STR): 2012-08-08T21:55:09.000Z
PAR MISSION_NAME (STR): CLUSTER
PAR OBSERVATORY_NAME (STR): Rumba
PAR OBSERVATORY_NUMBER (INT): 1
PAR EXPERIMENT_NAME (STR): STAFF-SC
PAR EXPERIMENT_MODE (STR): NBR
PAR INSTRUMENT_TYPE (STR): Search Coils
PAR MEASUREMENT_TYPE (STR): B-AC Magnetic field waveform
PAR INDEX_LABEL (STR): Time
PAR INDEX_TYPE (STR): STR
PAR INDEX_UNITS (STR): ISO_TIME
PAR INDEX_FORMAT (STR): (a27)
PAR INDEX_FORM (STR): Scalar
PAR INDEX_DIMENSION (INT): 1
PAR INDEX_PROPERTIES (STR): Regularly Spaced

# One uses here the index extension option: 2 more words are added to
# the time index: status, and spin phase angle in S/C system,
# according INDEX_EXTENSION_FORMAT format.
PAR INDEX_EXTENSION_LABEL (STR): Status : Phase_angle
PAR INDEX_EXTENSION_TYPE (STR): STR : DBL
PAR INDEX_EXTENSION_UNITS (STR): None : degree
PAR INDEX_EXTENSION_FORMAT (STR): (a12,1x,f7.2)
PAR INDEX_EXTENSION_LENGTH (INT): 20

PAR DATA_LABEL (STR): Bx : By : Bz : Compression Factor
PAR DATA_TYPE (STR): INT
PAR DATA_UNITS (STR): TM_counts : TM_counts : TM_counts : None
PAR DATA_FORMAT (STR): (24((3(I5,1x),i1),/),(3(I5,1x),i1))
PAR DATA_FORM (STR): Matrix
PAR DATA_DIMENSION (INT): 4 25
PAR DATA_REPRESENTATION (STR): xyz Cartesian
PAR DATA_COORDINATE_SYSTEM (STR): SSW6RF
PAR DATA_FILL_VALUE (INT): -999

PAR BLOCK_NUMBER (INT): 85701
PAR BLOCK_FIRST_INDEX (STR): 2001-09-23T00:00:00.904328Z
PAR BLOCK_LAST_INDEX (STR): 2001-09-23T23:59:54.886749Z

END MANDATORY_PARAMETERS
#-----

```

```

START OPTIONAL_PARAMETERS

PAR TIME_RESOLUTION          (DBL): 0.0399990667
PAR TIME_SPAN_FROM          (STR): 2001-09-23T00:00:00.000000Z
PAR TIME_SPAN_TO            (STR): 2001-09-23T23:59:59.999999Z

PAR TITLE                    (STR): CLUSTER / STAFF-SC / Rumba (#1)
PAR SUB_TITLE                (STR): TM data in spinning system
PAR DISCIPLINE_NAME          (STR): Space and Magnetospheric Physics
PAR EXPERIMENT_PI_NAME       (STR): Nicole Cornilleau
PAR EXPERIMENT_PI_MAIL       (STR): Nicole.Cornilleau@lpp.polytechnique.fr

PAR MISSION_DESCRIPTION      (TXT): {
Cluster is a set of 4 spacecrafts, launched in summer 2000. Salsa & Samba
(C2-C3, FM6-FM7 on July 16, Rumba & Tango (C1-C4, FM5-FM8) on August 9.
The aim of the CLuster mission is to study small-scale structures of the
magnetosphere and its environment in three dimensions.
To achieve this, Cluster is constituted of four identical spacecrafts that
will fly in a tetrahedral configuration. The separation distances
between the spacecrafts will vary from ~40 km to 10,000 km,
according to the key scientific regions.}

PAR EXPERIMENT_DESCRIPTION   (TXT): {
The Spatio Temporal Analysis Field Fluctuation experiment (STAFF) is a
Tri-axes search coils magnetometer measuring the 3 components of the
magnetic field up to 4kHz. The STAFF-SC waveform unit produces waveform
up to either 10 or 180 Hz, according to telemetry rate.
In Normal Bit Rate (NBR), the sample frequency is about 25.0 Hz, while it is
450 Hz in High Bit Rate (HBR). Owing to telemetry limitations, a reduction
of the dynamic data range from 16 to 12 bits is performed inside DWP.
The principle is to transmit the full 16-bit word at the beginning of
each telemetry packet, and later the difference between the successive
samples, coded on 12 bits. The 4th word of telemetry allows determination
of the maximum error on each component.
The data are given in the "SSW6RF" coordinates which means " STAFF Sensors
WEC 6 Reference Frame " where z is close to the spin axis.}

PAR INDEX_DESCRIPTION        (TXT): {
Time is given in ISO format, accepting any digits for second field as
"2001-02-18T19:16:17.550934Z": "T" and "Z" separators are required.}

PAR INDEX_EXTENSION_DESCRIP  (TXT): {
The extension of the index field is used to add some auxiliary data
varying with the same rate as the index itself.
The index extension field must contain only a limited series of scalar
data: label, type and units can be different, but in accordance with the
given format.
Here, extended index contains a status word of 14 characters, and the
spin phase. Note that S/C HK used to calculate the phase are not
corrected by TCOR, so the maximum error induced is 0.2 degrees. }

PAR DATA_DESCRIPTION        (TXT): {
This file contains "Matrix" DATA FORM which are indexed by the time.
So, the file class is WaveForm also called MatTime. This is a temporal
series of data blocks. Each block begins with the INDEX value,
(ISO epoch) and INDEX_EXTENSION values, the rest of the block is a
series of DATA values, over several lines, according to the DATA FORMAT
value. The number of values per line, or group of lines, is the first
dimension of the matrix, the number of lines, or group of lines, is the
second dimension of the matrix. Sample rate, given in CONSTANT_DATA,
allows time interpolation inside a block.}

PAR BLOCK_DESCRIPTION        (TXT): {
A data block is composed of the index, the index extension and the data.
A block can be entirely read or written by a single operation, by using
a format composed with the concatenation of the INDEX_FORMAT, the
INDEX_EXTENSION_FORMAT, and the DATA_FORMAT.}

PAR FILE_ANOMALIES           (TXT): {
None.}

PAR HISTORY                  (TXT): {
2012-08-08T21:55:09.000Z : N1_TO_RFF V.20120801}

END OPTIONAL_PARAMETERS
END METADATA
#-----

```

```

START DATA
#-----
# Two categories: Constant data (one value per file), and Indexed data
# (data versus time or other INDEX)
#-----

START CONSTANT_DATA

VAR TED_VERSION          (STR), u=None      : 2.5.0.109
VAR TCOR_OPTION          (STR), u=None      : yes
VAR SAMPLE_RATE          (DBL), u=Hz        : 25.0005833
VAR VOLT_RANGE           (FLT), u=Volts     : 10.00
VAR TM_RANGE_MIN         (INT), u=TM_counts : 0
VAR TM_RANGE_MAX         (INT), u=TM_counts : 65535

VAR MISALIGNMENT_MATRIX_L1_C1 (FLT), u=None : 1.000000
VAR MISALIGNMENT_MATRIX_L1_C2 (FLT), u=None : 0.000000
VAR MISALIGNMENT_MATRIX_L1_C3 (FLT), u=None : 0.000000
VAR MISALIGNMENT_MATRIX_L2_C1 (FLT), u=None : 0.000000
VAR MISALIGNMENT_MATRIX_L2_C2 (FLT), u=None : 1.000000
VAR MISALIGNMENT_MATRIX_L2_C3 (FLT), u=None : 0.000000
VAR MISALIGNMENT_MATRIX_L3_C1 (FLT), u=None : 0.000000
VAR MISALIGNMENT_MATRIX_L3_C2 (FLT), u=None : 0.000000
VAR MISALIGNMENT_MATRIX_L3_C3 (FLT), u=None : 1.000000

VAR CONSTANT_TIME_MEASUREMENT (STR), u=ISO_TIME : 2001-09-20T00:00:01Z
VAR SPIN_PERIOD              (DBL), u=second   : 4.023121
VAR SPIN_GEI_RIGHT_ASCENSION (FLT), u=degree  : 74.59
VAR SPIN_GEI_DECLINATION     (FLT), u=degree  : -67.09
VAR MASS_CENTER_X            (FLT), u=mm       : 760.9
VAR MASS_CENTER_Y            (FLT), u=mm       : -0.1
VAR MASS_CENTER_Z            (FLT), u=mm       : 0.1
VAR EULER_ANGLE_FIRST        (FLT), u=degree  : 0.00
VAR EULER_ANGLE_SECOND       (FLT), u=degree  : 0.06

END CONSTANT_DATA
#-----

START INDEXED_DATA
2001-09-23T00:00:00.904328Z 000000000001 17.35
34047 33190 32795 0
34078 33107 32796 0
34098 33028 32796 0
34107 32941 32801 0
34110 32857 32795 0
34109 32776 32799 0
34105 32693 32797 0
34101 32611 32799 0
34090 32528 32796 0
34064 32445 32795 0
34041 32365 32794 0
34009 32290 32790 0
33979 32217 32792 0
33937 32139 32796 0
33898 32071 32794 0
33848 32000 32792 0
33797 31932 32790 0
33746 31873 32793 0
33681 31812 32791 0
33614 31757 32790 0
33552 31704 32790 0
33478 31663 32791 0
33406 31617 32787 0
33330 31584 32789 0
33255 31551 32786 0
2001-09-23T00:00:01.904304Z 000000000001 106.83
33175 31522 32787 0
33092 31498 32783 0
33012 31477 32786 0
32929 31466 32785 0

. . . . .

33588 34500 32803 0
33693 34445 32802 0
33792 34388 32804 0
END INDEXED_DATA
END DATA
END ROPROC_FORMAT_FILE

```

Table 67: Exemple de WFL1.rff (3 pages)

### 9.9.2 Exemple de VTL1.rff

```

START ROPROC_FORMAT_FILE
#=====
# The Roproc.Format.File is a general format mainly used for data from
# spatial missions (magnetometer, waveform units, S/C trajectory ...).
# For readability, all lines starting with character "#" are comments,
# while empty or blank lines are ignored. Others lines are keywords,
# parameters variables or data.
# This file contents Metadata (description of the data), Constant Data
# (one value per file, or a few, but in limited number) and indexed data
# (data versus time or other INDEX).
# Any group of data begins with a "START" keyword, and stops by a "END".
# Metadata are given as parameters series (one value per parameter).
# Data are described by Metadata parameters.
# Examples of different files are given in the document:
# "The Roproc Format File, a dedicated file format for vectorial data
# processing"
# Author: P. Robert, CNRS/LPP (formerly CETP)
# V 1.0, 1996-2000 (for archive of old mission data)
# V 1.1, October 2001 (for CLUSTER/STAFF-SC NBR & HBR wave data)
# V 1.2, January 2002 (for GEOS wave data)
# V 1.3, August 2003 (for CUSP and any kind of wave data)
# V 1.4, January 2003 (for general titles management)
# V 2.0, March 2004 (for compatibility with Roproc Vector format)
# V 2.1, May 2004 (to be coherent with Cluster Exchange Format)
# V 2.2, March 2007 (some useful upgrades)
# Any comment or suggestion: Patrick.Robert@lpp.polytechnique.fr
#=====

START METADATA
#-----
START MANDATORY_PARAMETERS

PAR FILE_NAME (STR): CLU1_STASC_VTL1_NBR_20010923.rff
PAR FILE_CLASS (STR): VecTime
PAR FILE_FORMAT_VERSION (STR): Roproc.Format.File V 2.2
PAR FILE_CREATION_DATE (STR): 2012-09-07T15:44:02.739Z

PAR MISSION_NAME (STR): CLUSTER
PAR OBSERVATORY_NAME (STR): Rumba
PAR OBSERVATORY_NUMBER (INT): 1
PAR EXPERIMENT_NAME (STR): STAFF-SC
PAR EXPERIMENT_MODE (STR): NBR
PAR INSTRUMENT_TYPE (STR): Search Coils
PAR MEASUREMENT_TYPE (STR): B-AC Magnetic field waveform

PAR INDEX_LABEL (STR): Time
PAR INDEX_TYPE (STR): STR
PAR INDEX_UNITS (STR): ISO_TIME
PAR INDEX_FORMAT (STR): (a27)
PAR INDEX_FORM (STR): Scalar
PAR INDEX_DIMENSION (INT): 1
PAR INDEX_PROPERTIES (STR): Regularly Spaced

PAR INDEX_EXTENSION_LABEL (STR): Status : Phase_angle
PAR INDEX_EXTENSION_TYPE (STR): STR : FLT
PAR INDEX_EXTENSION_UNITS (STR): None : degree
PAR INDEX_EXTENSION_FORMAT (STR): (a14,"",f7.2)
PAR INDEX_EXTENSION_LENGTH (INT): 22

PAR DATA_LABEL (STR): Bx : By : Bz
PAR DATA_TYPE (STR): INT
PAR DATA_UNITS (STR): TM_counts : TM_counts : TM_counts
PAR DATA_FORMAT (STR): (I5,"",I5,"",I5)
PAR DATA_FORM (STR): Vector
PAR DATA_DIMENSION (INT): 3
PAR DATA_REPRESENTATION (STR): xyz Cartesian
PAR DATA_COORDINATE_SYSTEM (STR): INSTRUMENT
PAR DATA_FILL_VALUE (STR): -999

PAR BLOCK_NUMBER (INT): 2142522
PAR BLOCK_FIRST_INDEX (STR): 2001-09-23T00:00:00.024349Z
PAR BLOCK_LAST_INDEX (STR): 2001-09-23T23:59:55.846726Z

END MANDATORY_PARAMETERS
#-----

```



```

START OPTIONAL_PARAMETERS

PAR TIME_RESOLUTION          (DBL):  0.0399990667
PAR FREQUENCY_RESOLUTION    (DBL):  0.0000000000
PAR TIME_SPAN_FROM          (STR):  2001-09-23T00:00:00.000000Z
PAR TIME_SPAN_TO            (STR):  2001-09-23T23:59:59.999999Z
PAR TITLE                   (STR):  CLUSTER / STAFF-SC / Rumba (#1)
PAR SUB_TITLE               (STR):  TM data in spinning system
PAR DISCIPLINE_NAME         (STR):  Space and Magnetospheric Physics
PAR EXPERIMENT_PI_NAME      (STR):  Nicole Cornilleau
PAR EXPERIMENT_PI_MAIL      (STR):  Nicole.Cornilleau@lpp.polytechnique.fr

PAR MISSION_DESCRIPTION      (TXT): {
Cluster is a set of 4 spacecrafts, launched in summer 2000. Salsa & Samba
(C2-C3, FM6-FM7 on July 16, Rumba & Tango (C1-C4, FM5-FM8) on August 9.
The aim of the CLuster mission is to study small-scale structures of the
magnetosphere and its environment in three dimensions.
To achieve this, Cluster is constituted of four identical spacecrafts that
will fly in a tetrahedral configuration. The separation distances
between the spacecrafts will vary from ~40 km to 10,000 km,
according to the key scientific regions.}

PAR EXPERIMENT_DESCRIPTION   (TXT): {
The Spatio Temporal Analysis Field Fluctuation experiment (STAFF) is a
Tri-axes search coils magnetometer measuring the 3 components of the
magnetic field up to 4kHz. The STAFF-SC waveform unit produces waveform
up to either 10 or 180 Hz, according to telemetry rate.
In Normal Bit Rate (NBR), the sample frequency is about 25.0 Hz, while it is
450 Hz in High Bit Rate (HBR). Owing to telemetry limitations, a reduction
of the dynamic data range from 16 to 12 bits is performed inside DWP.
The principle is to transmit the full 16-bit word at the beginning of
each telemetry packet, and later the difference between the successive
samples, coded on 12 bits. The 4th word of telemetry allows determination
of the maximum error on each component.
The data are given in the "SSW6RF" coordinates which means " STAFF Sensors
WEC 6 Reference Frame " where z is close to the spin axis.}

PAR INDEX_DESCRIPTION        (TXT): {
Time is given in ISO format, accepting any digits for second field as
"2001-02-18T19:16:17.550934Z": "T" and "Z" separators are required.}

PAR INDEX_EXTENSION_DESCRIP  (TXT): {
The extension of the index field is used to add some auxiliary data
varying with the same rate as the index itself.
The index extension field must contain only a limited series of scalar
data: label, type and units can be different, but in accordance with the
given format.
Here, extended index contains a status word of 14 characters, and the
spin phase. Note that S/C HK used to calculate the phase are not
corrected by TCOR, so the maximum error induced is 0.2 degrees.}

PAR DATA_DESCRIPTION        (TXT): {
Class of this file is "VecTime", meaning a data vector dependant of the time.
DATA_FORM and INDEX_FORM must be "Vector" and "Scalar". This is a temporal
series of data blocks. Each block begins with the INDEX value giving the
time (for example in ISO epoch) and INDEX_EXTENSION values. The rest of the
block is a series of values, corresponding to the vector components,
and according to the DATA_FORMAT value. Sample rate, which can be found in
CONSTANT_DATA, must be consistant with the value of time_resolution.}

PAR BLOCK_DESCRIPTION        (TXT): {
A data block is composed of the index, the index extension and the data.
A block can be entirely read or written by a single operation, by using
a format composed with the concatenation of the INDEX_FORMAT, the
INDEX_EXTENSION_FORMAT, and the DATA_FORMAT.}

PAR FILE_ANOMALIES           (TXT): {
None.}

PAR HISTORY                  (TXT): {
2012-08-08T21:55:09.000Z : N1_TO_RFF V.20120801
2012-09-07T15:43:38.000Z : RCL_get_data_CLUSTER WFL1 RCL_V1p4_Linux_x86_64
2012-09-07T15:44:02.702Z : RCL_waveform_to_vectime}

END OPTIONAL_PARAMETERS
END METADATA
#-----

```

```

START DATA
#-----
START CONSTANT_DATA
VAR TED_VERSION          (STR), u=None      : 2.5.0.109
VAR TCOR_OPTION          (STR), u=None      : yes
VAR SAMPLE_RATE          (DBL), u=Hz        : 25.0005833
VAR VOLT_RANGE_MIN       (FLT), u=Volts     : -5.00
VAR VOLT_RANGE_MAX       (FLT), u=Volts     : 5.00
VAR TM_RANGE_MIN         (INT), u=TM_counts : 0
VAR TM_RANGE_MAX         (INT), u=TM_counts : 65535

VAR MISALIGNMENT_MATRIX_L1_C1 (FLT), u=None : 1.0000
VAR MISALIGNMENT_MATRIX_L1_C2 (FLT), u=None : 0.0000
VAR MISALIGNMENT_MATRIX_L1_C3 (FLT), u=None : 0.0000
VAR MISALIGNMENT_MATRIX_L2_C1 (FLT), u=None : 0.0000
VAR MISALIGNMENT_MATRIX_L2_C2 (FLT), u=None : 1.0000
VAR MISALIGNMENT_MATRIX_L2_C3 (FLT), u=None : 0.0000
VAR MISALIGNMENT_MATRIX_L3_C1 (FLT), u=None : 0.0000
VAR MISALIGNMENT_MATRIX_L3_C2 (FLT), u=None : 0.0000
VAR MISALIGNMENT_MATRIX_L3_C3 (FLT), u=None : 1.0000

VAR CONSTANT_TIME_MEASUREMENT (STR), u=ISO_TIME : 2001-09-20T00:00:01Z
VAR SPIN_PERIOD               (DBL), u=Hz        : 4.0231210
VAR SPIN_GEI_RIGHT_ASCENSION (FLT), u=degree    : 74.59
VAR SPIN_GEI_DECLINATION     (FLT), u=degree    : -67.09
VAR MASS_CENTER_X             (FLT), u=mm        : 760.90
VAR MASS_CENTER_Y             (FLT), u=mm        : -0.10
VAR MASS_CENTER_Z             (FLT), u=mm        : 0.10
VAR EULER_ANGLE_FIRST        (FLT), u=degree    : 0.00
VAR EULER_ANGLE_SECOND       (FLT), u=degree    : 0.06

END CONSTANT_DATA
#-----
START INDEXED_DATA
2001-09-23T00:00:00.024349Z,00000000000100, 298.61,32632,34114,32778
2001-09-23T00:00:00.064348Z,00000000000100, 302.19,32714,34126,32789
2001-09-23T00:00:00.104347Z,00000000000100, 305.77,32806,34122,32789
2001-09-23T00:00:00.144346Z,00000000000100, 309.35,32886,34124,32790
2001-09-23T00:00:00.184345Z,00000000000100, 312.92,32967,34113,32791
2001-09-23T00:00:00.224344Z,00000000000100, 316.50,33050,34094,32790
2001-09-23T00:00:00.264343Z,00000000000100, 320.08,33134,34079,32790
2001-09-23T00:00:00.304342Z,00000000000100, 323.66,33213,34053,32795
2001-09-23T00:00:00.344341Z,00000000000100, 327.24,33289,34019,32796
2001-09-23T00:00:00.384340Z,00000000000100, 330.82,33367,33985,32794
2001-09-23T00:00:00.424339Z,00000000000100, 334.40,33446,33952,32795
2001-09-23T00:00:00.464338Z,00000000000100, 337.98,33514,33904,32794
2001-09-23T00:00:00.504338Z,00000000000100, 341.56,33578,33857,32800
2001-09-23T00:00:00.544337Z,00000000000100, 345.14,33644,33808,32793
2001-09-23T00:00:00.584336Z,00000000000100, 348.72,33707,33753,32798
2001-09-23T00:00:00.624335Z,00000000000100, 352.30,33766,33691,32800
2001-09-23T00:00:00.664334Z,00000000000100, 355.88,33814,33627,32798
2001-09-23T00:00:00.704333Z,00000000000100, 359.45,33867,33565,32798
2001-09-23T00:00:00.744332Z,00000000000100, 3.03,33919,33494,32802
2001-09-23T00:00:00.784331Z,00000000000100, 6.61,33958,33418,32798
2001-09-23T00:00:00.824330Z,00000000000100, 10.19,33998,33342,32799
2001-09-23T00:00:00.864329Z,00000000000100, 13.77,34027,33263,32798
2001-09-23T00:00:00.904328Z,00000000000110, 17.35,34047,33190,32795
2001-09-23T00:00:00.944327Z,00000000000100, 20.93,34078,33107,32796
2001-09-23T00:00:00.984326Z,00000000000100, 24.51,34098,33028,32796
2001-09-23T00:00:01.024325Z,00000000000100, 28.09,34107,32941,32801

. . . . .

2001-09-23T23:59:55.686730Z,00000000000100, 130.45,30987,33392,32773
2001-09-23T23:59:55.726729Z,00000000000100, 134.03,31025,33502,32774
2001-09-23T23:59:55.766728Z,00000000000100, 137.61,31070,33608,32780
2001-09-23T23:59:55.806727Z,00000000000100, 141.19,31122,33724,32771
2001-09-23T23:59:55.846726Z,00000000000100, 144.77,31183,33821,32775
END INDEXED_DATA
END DATA
END ROPROC_FORMAT_FILE

```

Table 68: Example de VTL1.rff (3 pages)

### 9.9.3 Exemple de VTL2.rff

```

START ROPROC_FORMAT_FILE
#=====
# The Roproc_Format_File is a general format mainly used for data from
# spatial missions (magnetometer, waveform units, S/C trajectory ...).
# For readability, all lines starting with character "#" are comments,
# while empty or blank lines are ignored. Others lines are keywords,
# parameters variables or data.
# This file contents Metadata (description of the data), Constant Data
# (one value per file, or a few, but in limited number) and indexed data
# (data versus time or other INDEX).
# Any group of data begins with a "START" keyword, and stops by a "END".
# Metadata are given as parameters series (one value per parameter).
# Data are described by Metadata parameters.
# Examples of different files are given in the document:
# "The Roproc Format File, a dedicated file format for vectorial data
# processing"
# Author: P. Robert, CNRS/LPP (formerly CETP)
# V 1.0, 1996-2000 (for archive of old mission data)
# V 1.1, October 2001 (for CLUSTER/STAFF-SC NBR & HBR wave data)
# V 1.2, January 2002 (for GEOS wave data)
# V 1.3, August 2003 (for CUSP and any kind of wave data)
# V 1.4, January 2003 (for general titles management)
# V 2.0, March 2004 (for compatibility with Roproc Vector format)
# V 2.1, May 2004 (to be coherent with Cluster Exchange Format)
# V 2.2, March 2007 (some useful upgrades)
# Any comment or suggestion: Patrick.Robert@lpp.polytechnique.fr
#=====

START METADATA
#-----
START MANDATORY_PARAMETERS

PAR FILE_NAME (STR): CLU4_STASC_VTL2_ISR2_NBR_20140124.rff
PAR FILE_CLASS (STR): VecTime
PAR FILE_FORMAT_VERSION (STR): Roproc_Format_File V 2.2
PAR FILE_CREATION_DATE (STR): 2014-10-05T12:15:57.850Z

PAR MISSION_NAME (STR): CLUSTER
PAR OBSERVATORY_NAME (STR): Tango
PAR OBSERVATORY_NUMBER (INT): 4
PAR EXPERIMENT_NAME (STR): STAFF-SC
PAR EXPERIMENT_MODE (STR): NBR
PAR INSTRUMENT_TYPE (STR): Search Coils
PAR MEASUREMENT_TYPE (STR): B-AC Magnetic field waveform

PAR INDEX_LABEL (STR): Time
PAR INDEX_TYPE (STR): STR
PAR INDEX_UNITS (STR): ISO_TIME
PAR INDEX_FORMAT (STR): {a27}
PAR INDEX_FORM (STR): Scalar
PAR INDEX_DIMENSION (INT): 1
PAR INDEX_PROPERTIES (STR): Regularly Spaced

PAR INDEX_EXTENSION_LABEL (STR): Status : Phase_angle
PAR INDEX_EXTENSION_TYPE (STR): STR : FLT
PAR INDEX_EXTENSION_UNITS (STR): None : degree
PAR INDEX_EXTENSION_FORMAT (STR): {a14,"",f7.2)
PAR INDEX_EXTENSION_LENGTH (INT): 22

PAR DATA_LABEL (STR): Bx : By : Bz : Dx : Dy
PAR DATA_TYPE (STR): FLT
PAR DATA_UNITS (STR): nT : nT : nT : nT : nT
PAR DATA_FORMAT (STR): {E13.6,1x,E13.6,1x,E13.6,1x,E13.6,1x,E13.6)
PAR DATA_FORM (STR): Vector
PAR DATA_DIMENSION (INT): 5
PAR DATA_REPRESENTATION (STR): xyz Cartesian
PAR DATA_COORDINATE_SYSTEM (STR): ISR2
PAR DATA_FILL_VALUE (STR): -0.1000E+31

PAR BLOCK_NUMBER (INT): 1454900
PAR BLOCK_FIRST_INDEX (STR): 2014-01-24T07:47:48.641359Z
PAR BLOCK_LAST_INDEX (STR): 2014-01-24T23:59:59.964621Z

END MANDATORY_PARAMETERS
#-----

```

## START OPTIONAL\_PARAMETERS

```

PAR TIME_RESOLUTION      (DBL):  0.0399997333
PAR FREQUENCY_RESOLUTION (DBL):  0.0000000000
PAR TIME_SPAN_FROM       (STR):  2014-01-24T00:00:00.000000Z
PAR TIME_SPAN_TO         (STR):  2014-01-24T23:59:59.999999Z
PAR TITLE                 (STR):  CLUSTER / STAFF-SC / Tango (#4)
PAR SUB_TITLE             (STR):  Step 7: Data in ISR2 system [nT] + separated DC (Fc=0.1 Fdet=0.)
PAR DISCIPLINE_NAME       (STR):  Space and Magnetospheric Physics
PAR EXPERIMENT_PI_NAME    (STR):  Nicole Cornilleau
PAR EXPERIMENT_PI_MAIL    (STR):  Nicole.Cornilleau@lpp.polytechnique.fr

```

```

PAR MISSION_DESCRIPTION   (TXT): {
Cluster is a set of 4 spacecrafts, launched in summer 2000. Salsa & Samba
(C2-C3, FM6-FM7 on July 16, Rumba & Tango (C1-C4, FM5-FM8) on August 9.
The aim of the CLuster mission is to study small-scale structures of the
magnetosphere and its environment in three dimensions.
To achieve this, Cluster is constituted of four identical spacecrafts that
will fly in a tetrahedral configuration. The separation distances
between the spacecrafts will vary from ~40 km to 10,000 km,
according to the key scientific regions.}

```

```

PAR EXPERIMENT_DESCRIPTION (TXT): {
The Spatio Temporal Analysis Field Fluctuation experiment (STAFF) is a
Tri-axes search coils magnetometer measuring the 3 components of the
magnetic field up to 4kHz. The STAFF-SC waveform unit produces waveform
up to either 10 or 180 Hz, according to telemetry rate.
In Normal Bit Rate (NBR), the sample frequency is about 25.0 Hz, while it is
450 Hz in High Bit Rate (HBR). Owing to telemetry limitations, a reduction
of the dynamic data range from 16 to 12 bits is performed inside DWP.
The principle is to transmit the full 16-bit word at the beginning of
each telemetry packet, and later the difference between the successive
samples, coded on 12 bits. The 4th word of telemetry allows determination
of the maximum error on each component.
The data are given in the "SSW6RF" coordinates which means " STAFF Sensors
WEC 6 Reference Frame " where z is close to the spin axis.}

```

```

PAR INDEX_DESCRIPTION      (TXT): {
Time is given in ISO format, accepting any digits for second field as
"2001-02-18T19:16:17.550934Z": "T" and "Z" separators are required.}

```

```

PAR INDEX_EXTENSION_DESCRIP (TXT): {
The extension of the index field is used to add some auxiliary data
varying with the same rate as the index itself.
The index extension field must contain only a limited series of scalar
data: label, type and units can be different, but in accordance with the
given format.
Here, extended index contains a status word of 14 characters, and the
spin phase. Note that S/C HK used to calculate the phase are not
corrected by TCOR, so the maximum error induced is 0.2 degrees.}

```

```

PAR DATA_DESCRIPTION      (TXT): {
Class of this file is "VecTime", meaning a data vector dependant of the time.
DATA_FORM and INDEX_FORM must be "Vector" and "Scalar". This is a temporal
series of data blocks. Each block begins with the INDEX value giving the
time (for example in ISO epoch) and INDEX_EXTENSION values. The rest of the
block is a series of values, corresponding to the vector components,
and according to the DATA_FORMAT value. Sample rate, which can be found in
CONSTANT_DATA, must be consistant with the value of time_resolution.}

```

```

PAR BLOCK_DESCRIPTION      (TXT): {
A data block is composed of the index, the index extension and the data.
A block can be entirely read or written by a single operation, by using
a format composed with the concatenation of the INDEX_FORMAT, the
INDEX_EXTENSION_FORMAT, and the DATA_FORMAT.}

```

```

PAR FILE_ANOMALIES         (TXT): {
None.}

```

```

PAR HISTORY                 (TXT): {
2014-09-23T23:11:06.000Z : N1 TO RFF V.20120801
2014-09-30T19:15:59.000Z : RCL_get_data_CLUSTA_WFL1_forVTL1 RCL_V2p1
2014-09-30T19:16:15.772Z : RCL_waveform_to_vectime - RCL_V2.1, January 2014
2014-10-05T12:09:24.000Z : RCL_get_data_CLUSTA_VTL1_forVTL2 RCL_V2p1
2014-10-05T12:09:25.479Z : RCL_vectime_calibration_CLUSTA - RCL_V2.1, January 20}

```

```

END OPTIONAL_PARAMETERS
END METADATA
#-----

```

```

START DATA
#-----
START CONSTANT_DATA

VAR TED_VERSION          (STR), u=None      : 2.5.0.109
VAR TCOR_OPTION          (STR), u=None      : yes
VAR SAMPLE_RATE          (DBL), u=Hz        : 25.0001667
VAR VOLT_RANGE_MIN       (FLT), u=Volts     : -5.00
VAR VOLT_RANGE_MAX       (FLT), u=Volts     : 5.00
VAR TM_RANGE_MIN         (INT), u=TM_counts : 0
VAR TM_RANGE_MAX         (INT), u=TM_counts : 65535

VAR MISALIGNMENT_MATRIX_L1_C1 (FLT), u=None : 1.0000
VAR MISALIGNMENT_MATRIX_L1_C2 (FLT), u=None : 0.0000
VAR MISALIGNMENT_MATRIX_L1_C3 (FLT), u=None : 0.0000
VAR MISALIGNMENT_MATRIX_L2_C1 (FLT), u=None : 0.0000
VAR MISALIGNMENT_MATRIX_L2_C2 (FLT), u=None : 1.0000
VAR MISALIGNMENT_MATRIX_L2_C3 (FLT), u=None : 0.0000
VAR MISALIGNMENT_MATRIX_L3_C1 (FLT), u=None : 0.0000
VAR MISALIGNMENT_MATRIX_L3_C2 (FLT), u=None : 0.0000
VAR MISALIGNMENT_MATRIX_L3_C3 (FLT), u=None : 1.0000

VAR CONSTANT_TIME_MEASUREMENT (STR), u=ISO_TIME : 2014-01-24T00:34:56Z
VAR SPIN_PERIOD              (DBL), u=Hz        : 4.1247060
VAR SPIN_GEI_RIGHT_ASCENSION (FLT), u=degree   : 102.80
VAR SPIN_GEI_DECLINATION     (FLT), u=degree   : -64.72
VAR MASS_CENTER_X            (FLT), u=mm        : 762.40
VAR MASS_CENTER_Y            (FLT), u=mm        : -0.10
VAR MASS_CENTER_Z            (FLT), u=mm        : -0.20
VAR EULER_ANGLE_FIRST        (FLT), u=degree   : -0.05
VAR EULER_ANGLE_SECOND       (FLT), u=degree   : 0.07

VAR FREQUENCY_FILTER_MIN     (FLT), u=Hz        : 0.00
VAR FREQUENCY_FILTER_MAX     (FLT), u=Hz        : 12.50
VAR FREQUENCY_CUT_OFF        (FLT), u=Hz        : 0.10
VAR FREQUENCY_DETREND        (FLT), u=Hz        : 0.00
VAR CALIB_KERNEL_SIZE        (INT), u=None      : 1024
VAR CALIB_SHIFT_SIZE         (INT), u=None      : 2

END CONSTANT_DATA

START INDEXED_DATA
2014-01-24T07:47:48.641359Z,00000000000100, 80.83, 0.265783E+00, 0.170511E+00, 0.174759E-01, -0.276829E+02, 0.333295E+02
2014-01-24T07:47:48.681358Z,00000000000100, 84.32, 0.290429E+00, 0.163142E+00, 0.937258E-02, -0.276829E+02, 0.333295E+02
2014-01-24T07:47:48.721358Z,00000000000100, 87.81, 0.313820E+00, 0.157474E+00, 0.165855E-02, -0.276847E+02, 0.333276E+02
2014-01-24T07:47:48.761358Z,00000000000100, 91.31, 0.331832E+00, 0.143035E+00, 0.425409E-02, -0.276847E+02, 0.333276E+02
2014-01-24T07:47:48.801357Z,00000000000100, 94.80, 0.358689E+00, 0.110676E+00, 0.657462E-03, -0.276867E+02, 0.333247E+02
2014-01-24T07:47:48.841357Z,00000000000100, 98.29, 0.369281E+00, 0.736732E-01, -0.134483E-01, -0.276867E+02, 0.333247E+02
2014-01-24T07:47:48.881357Z,00000000000100, 101.78, 0.370587E+00, 0.524259E-01, -0.119783E-01, -0.276830E+02, 0.333270E+02
2014-01-24T07:47:48.921357Z,00000000000100, 105.27, 0.373347E+00, 0.303290E-01, -0.251347E-01, -0.276830E+02, 0.333270E+02
2014-01-24T07:47:48.961356Z,00000000000100, 108.76, 0.375900E+00, 0.112435E-01, -0.284219E-01, -0.276840E+02, 0.333247E+02
2014-01-24T07:47:49.001356Z,00000000000100, 112.25, 0.377905E+00, -0.509962E-02, -0.276502E-01, -0.276840E+02, 0.333247E+02
2014-01-24T07:47:49.041356Z,00000000000100, 115.74, 0.376592E+00, -0.223075E-01, -0.228977E-01, -0.276847E+02, 0.333226E+02
2014-01-24T07:47:49.081356Z,00000000000100, 119.23, 0.375102E+00, -0.288379E-01, -0.224956E-01, -0.276847E+02, 0.333226E+02
2014-01-24T07:47:49.121355Z,00000000000100, 122.73, 0.371775E+00, -0.435999E-01, -0.156042E-01, -0.276846E+02, 0.333227E+02
2014-01-24T07:47:49.161355Z,00000000000100, 126.22, 0.360669E+00, -0.646611E-01, -0.160881E-01, -0.276846E+02, 0.333227E+02
2014-01-24T07:47:49.201352Z,00000000000110, 129.71, 0.353939E+00, -0.868491E-01, -0.626252E-02, -0.276808E+02, 0.333256E+02
2014-01-24T07:47:49.241351Z,00000000000100, 133.20, 0.349823E+00, -0.912005E-01, -0.207030E-01, -0.276808E+02, 0.333256E+02
2014-01-24T07:47:49.281351Z,00000000000100, 136.69, 0.337359E+00, -0.898363E-01, -0.330107E-01, -0.276816E+02, 0.333249E+02
2014-01-24T07:47:49.321351Z,00000000000100, 140.18, 0.326682E+00, -0.920013E-01, -0.284343E-01, -0.276816E+02, 0.333249E+02
2014-01-24T07:47:49.361350Z,00000000000100, 143.67, 0.325380E+00, -0.960225E-01, -0.210023E-01, -0.276841E+02, 0.333231E+02
2014-01-24T07:47:49.401350Z,00000000000100, 147.17, 0.320030E+00, -0.111014E+00, -0.120692E-01, -0.276841E+02, 0.333231E+02

...

2014-01-24T23:59:59.724622Z,00000000000100, 45.75, -0.184306E+00, -0.339997E+00, -0.789752E-01, -0.586408E+00, -0.395538E+01
2014-01-24T23:59:59.764622Z,00000000000100, 49.24, -0.192907E+00, -0.346306E+00, -0.708003E-01, -0.586226E+00, -0.395388E+01
2014-01-24T23:59:59.804622Z,00000000000100, 52.73, -0.215125E+00, -0.331000E+00, -0.740648E-01, -0.586226E+00, -0.395388E+01
2014-01-24T23:59:59.844621Z,00000000000100, 56.22, -0.238742E+00, -0.307663E+00, -0.643426E-01, -0.585331E+00, -0.395373E+01
2014-01-24T23:59:59.884621Z,00000000000100, 59.72, -0.274305E+00, -0.294713E+00, -0.788636E-01, -0.585331E+00, -0.395373E+01
2014-01-24T23:59:59.924621Z,00000000000100, 63.21, -0.303181E+00, -0.269631E+00, -0.947920E-01, -0.585389E+00, -0.395445E+01
2014-01-24T23:59:59.964621Z,00000000000100, 66.70, -0.316007E+00, -0.255062E+00, -0.858673E-01, -0.585389E+00, -0.395445E+01
END INDEXED_DATA
END DATA
END ROPROC_FORMAT_FILE

```

Table 69 : Exemple de VTL2.rff (3 pages)

### 9.9.4 Exemple de SPL2.rff

```

START ROPROC_FORMAT_FILE

#=====
# The Roproc Format File is a general format mainly used for data from
# spatial missions (magnetometer, waveform units, S/C trajectory ...).
# For readability, all lines starting with character "#" are comments,
# while empty or blank lines are ignored. Others lines are keywords,
# parameters variables or data.
# This file contents Metadata (description of the data), Constant Data
# (one value per file, or a few, but in limited number) and indexed data
# (data versus time or other INDEX).
# Any group of data begins with a "START" keyword, and stops by a "END".
# Metadata are given as parameters series (one value per parameter).
# Data are described by Metadata parameters.
# Examples of different files are given in the document:
# "The Roproc Format File, a dedicated file format for vectorial data
# processing"
# Author: P. Robert, CNRS/LPP (formerly CETP)
# V 1.0, 1996-2000 (for archive of old mission data)
# V 1.1, October 2001 (for CLUSTER/STAFF-SC NBR & HBR wave data)
# V 1.2, January 2002 (for GEOS wave data)
# V 1.3, August 2003 (for CUSP and any kind of wave data)
# V 1.4, January 2003 (for general titles management)
# V 2.0, March 2004 (for compatibility with Roproc Vector format)
# V 2.1, May 2004 (to be coherent with Cluster Exchange Format)
# V 2.2, March 2007 (some useful upgrades)
# Any comment or suggestion: Patrick.Robert@lpp.polytechnique.fr
#=====

START METADATA

#-----
START MANDATORY_PARAMETERS

PAR FILE_NAME (STR): CLU4_STASC_SPL2_ISR2_NBR_20131224.rff
PAR FILE_CLASS (STR): Spectrogram
PAR FILE_FORMAT_VERSION (STR): Roproc Format File V 2.2
PAR FILE_CREATION_DATE (STR): 2014-06-14T00:37:53.582Z

PAR MISSION_NAME (STR): CLUSTER
PAR OBSERVATORY_NAME (STR): Tango
PAR OBSERVATORY_NUMBER (INT): 4
PAR EXPERIMENT_NAME (STR): STAFF-SC
PAR EXPERIMENT_MODE (STR): NBR
PAR INSTRUMENT_TYPE (STR): Search Coils
PAR MEASUREMENT_TYPE (STR): B-AC Magnetic field spectra

PAR INDEX_LABEL (STR): Time
PAR INDEX_TYPE (STR): STR
PAR INDEX_UNITS (STR): ISO_TIME
PAR INDEX_FORMAT (STR): (a27)
PAR INDEX_FORM (STR): Scalar
PAR INDEX_DIMENSION (INT): 1
PAR INDEX_PROPERTIES (STR): Regularly Spaced

PAR INDEX_EXTENSION_LABEL (STR): none
PAR INDEX_EXTENSION_TYPE (STR): none
PAR INDEX_EXTENSION_UNITS (STR): none
PAR INDEX_EXTENSION_FORMAT (STR): none
PAR INDEX_EXTENSION_LENGTH (INT): 0

PAR DATA_LABEL (STR): BxR : BxI : ByR : ByI : BzR : BzI
PAR DATA_TYPE (STR): FLT
PAR DATA_UNITS (STR): nT : nT : nT : nT : nT
PAR DATA_FORMAT (STR): (5(E11.4,2x),E11.4)
PAR DATA_FORM (STR): Matrix
PAR DATA_DIMENSION (INT): 6 128
PAR DATA_REPRESENTATION (STR): xyz Cartesian
PAR DATA_COORDINATE_SYSTEM (STR): ISR2
PAR DATA_FILL_VALUE (STR): -0.1000E+31

PAR BLOCK_NUMBER (INT): 8295
PAR BLOCK_FIRST_INDEX (STR): 2013-12-24T00:00:00.024074Z
PAR BLOCK_LAST_INDEX (STR): 2013-12-24T23:59:52.756120Z

END MANDATORY_PARAMETERS
#-----

```

```

START OPTIONAL_PARAMETERS

PAR TIME_RESOLUTION          (DBL): 10.2399317248
PAR FREQUENCY_RESOLUTION    (DBL): 0.0976569011
PAR TIME_SPAN_FROM          (STR): 2013-12-24T00:00:00.000000Z
PAR TIME_SPAN_TO            (STR): 2013-12-24T23:59:59.999999Z
PAR TITLE                   (STR): CLUSTER / STAFF-SC / Tango (#4)
PAR SUB_TITLE               (STR): Step 7: Data in ISR2 system [nT] + separated DC (Fc=0.1 Fdet=0.)
PAR DISCIPLINE_NAME         (STR): Space and Magnetospheric Physics
PAR EXPERIMENT_PI_NAME      (STR): Nicole Cornilleau
PAR EXPERIMENT_PI_MAIL      (STR): Nicole.Cornilleau@lpp.polytechnique.fr

PAR MISSION_DESCRIPTION      (TXT): {
Cluster is a set of 4 spacecrafts, launched in summer 2000. Salsa & Samba
(C2-C3, FM6-FM7 on July 16, Rumba & Tango (C1-C4, FM5-FM8) on August 9.
The aim of the Cluster mission is to study small-scale structures of the
magnetosphere and its environment in three dimensions.
To achieve this, Cluster is constituted of four identical spacecrafts that
will fly in a tetrahedral configuration. The separation distances
between the spacecrafts will vary from ~40 km to 10,000 km,
according to the key scientific regions.}

PAR EXPERIMENT_DESCRIPTION   (TXT): {
The Spatio Temporal Analysis Field Fluctuation experiment (STAFF) is a
Tri-axes search coils magnetometer measuring the 3 components of the
magnetic field up to 4kHz. The STAFF-SC waveform unit produces waveform
up to either 10 or 180 Hz, according to telemetry rate.
In Normal Bit Rate (NBR), the sample frequency is about 25.0 Hz, while it is
450 Hz in High Bit Rate (HBR). Owing to telemetry limitations, a reduction
of the dynamic data range from 16 to 12 bits is performed inside DWP.
The principle is to transmit the full 16-bit word at the beginning of
each telemetry packet, and later the difference between the successive
samples, coded on 12 bits. The 4th word of telemetry allows determination
of the maximum error on each component.
The data are given in the "SSW6RF" coordinates which means " STAFF Sensors
WEC 6 Reference Frame " where z is close to the spin axis.}

PAR INDEX_DESCRIPTION        (TXT): {
Time is given in ISO format, accepting any digits for second field as
"2001-02-18T19:16:17.550934Z": "T" and "Z" separators are required.}

PAR INDEX_EXTENSION_DESCRIP  (TXT): {
The extension of the index field is used to add some auxiliary data
varying with the same rate as the index itself.
The index extension field must contain only a limited series of scalar
data: label, type and units can be different, but in accordance with the
given format.
Here, extended index contains a status word of 14 characters, and the
spin phase. Note that S/C HK used to calculate the phase are not
corrected by TCOR, so the maximum error induced is 0.2 degrees.}

PAR DATA_DESCRIPTION        (TXT): {
Class of this file is "Spectrogram", indexed by time. Spectrogram class is also
called MatTime. This is a temporal series of data blocks. Each block begins
with the INDEX value (for example ISO epoch) and INDEX_EXTENSION values.
The rest of the block is a series of data values, over several lines,
according to the DATA_FORMAT value. The number of values per line is the first
dimension of the matrix, the number of lines is the second dimension of the
matrix. In case of Spectrogram class rather than MatTime, each line of
the matrix is a frequency-dependant vector. Sample rate given in CONSTANT_DATA,
correspond to the initial WaveForm or Vectime data, whereas the time resolution
given in the OPTIONAL_PARAMETERS correspond to the time period between two
spectra. Frequency resolution between two rays is given OPTIONAL_PARAMETERS,
and start from zero. Spectra can be overlapped.}

PAR BLOCK_DESCRIPTION        (TXT): {
A data block is composed of the index, the index extension and the data.
A block can be entirely read or written by a single operation, by using
a format composed with the concatenation of the INDEX_FORMAT, the
INDEX_EXTENSION_FORMAT, and the DATA_FORMAT.}

PAR FILE_ANOMALIES           (TXT): {
None.}

PAR HISTORY                  (TXT): {
2014-03-07T07:30:12.000Z : N1_TO_RFF V.20120801
2014-04-17T04:35:51.000Z : RCL_get_data_CLUSTA_WFL1_forVTL1 RCL_V2p1
2014-04-17T04:36:15.574Z : RCL_waveform_to_vectime - RCL_V2.1, January 2014
2014-06-14T00:37:26.000Z : RCL_get_data_CLUSTA_VTL1_forSPL2 RCL_V2p1
2014-06-14T00:37:27.335Z : RCL_vectime_L1_to_spectro_L2 - RCL_V2.1, January 2014}

END OPTIONAL_PARAMETERS
END METADATA
#-----

```



```

START DATA
#-----
START CONSTANT_DATA
VAR TED_VERSION          (STR), u=None      : 2.5.0.109
VAR TCOR_OPTION          (STR), u=None      : yes
VAR SAMPLE_RATE          (DBL), u=Hz        : 25.0001667
VAR VOLT_RANGE_MIN       (FLT), u=Volts     : -5.00
VAR VOLT_RANGE_MAX       (FLT), u=Volts     : 5.00
VAR TM_RANGE_MIN         (INT), u=TM_counts : 0
VAR TM_RANGE_MAX         (INT), u=TM_counts : 65535
VAR MISALIGNMENT_MATRIX_L1_C1 (FLT), u=None : 1.0000
VAR MISALIGNMENT_MATRIX_L1_C2 (FLT), u=None : 0.0000
VAR MISALIGNMENT_MATRIX_L1_C3 (FLT), u=None : 0.0000
VAR MISALIGNMENT_MATRIX_L2_C1 (FLT), u=None : 0.0000
VAR MISALIGNMENT_MATRIX_L2_C2 (FLT), u=None : 1.0000
VAR MISALIGNMENT_MATRIX_L2_C3 (FLT), u=None : 0.0000
VAR MISALIGNMENT_MATRIX_L3_C1 (FLT), u=None : 0.0000
VAR MISALIGNMENT_MATRIX_L3_C2 (FLT), u=None : 0.0000
VAR MISALIGNMENT_MATRIX_L3_C3 (FLT), u=None : 1.0000
VAR CONSTANT_TIME_MEASUREMENT (STR), u=ISO_TIME : 2013-12-19T04:45:00Z
VAR SPIN_PERIOD          (DBL), u=Hz        : 4.1243830
VAR SPIN_GEI_RIGHT_ASCENSION (FLT), u=degree : 87.37
VAR SPIN_GEI_DECLINATION  (FLT), u=degree : -60.72
VAR MASS_CENTER_X         (FLT), u=mm       : 762.40
VAR MASS_CENTER_Y         (FLT), u=mm       : -0.10
VAR MASS_CENTER_Z         (FLT), u=mm       : -0.20
VAR EULER_ANGLE_FIRST     (FLT), u=degree : 0.02
VAR EULER_ANGLE_SECOND   (FLT), u=degree : 0.09
VAR FREQUENCY_FILTER_MIN  (FLT), u=Hz      : 0.00
VAR FREQUENCY_FILTER_MAX  (FLT), u=Hz      : 12.50
VAR FREQUENCY_CUT_OFF     (FLT), u=Hz      : 0.10
VAR FREQUENCY_DETREND     (FLT), u=Hz      : 0.00
VAR CALIB_KERNEL_SIZE     (INT), u=None     : 256
VAR CALIB_SHIFT_SIZE      (INT), u=None     : 256
VAR SPECTRA_KERNEL_SIZE   (INT), u=None     : 256
VAR SPECTRA_SHIFT_SIZE    (INT), u=None     : 256
VAR SPECTRA_WEIGHTING     (STR), u=None     : t
END CONSTANT_DATA
START INDEXED_DATA
2013-12-24T00:00:00.024074Z,
-0.2207E+02, 0.0000E+00, 0.1292E+00, 0.0000E+00, 0.2448E+00, 0.0000E+00,
-0.1693E+00, -0.2965E+00, 0.1043E+00, -0.2622E+00, 0.2242E-01, -0.2698E-03,
-0.1900E+00, -0.1349E+00, 0.2789E+00, -0.4043E+00, -0.1718E+00, 0.1020E+00,
0.3127E+00, 0.1412E+00, -0.4353E-01, 0.2110E+00, -0.8497E-01, -0.1774E-01,
0.1048E+00, -0.4658E-01, -0.4681E-01, 0.1664E+00, 0.8656E-01, -0.1963E-01,
-0.2873E-01, 0.7817E-01, -0.8721E-01, -0.1811E-01, 0.8397E-01, 0.3584E-02,
0.3369E-02, 0.3090E-02, -0.1105E+00, 0.1413E-02, 0.1631E-01, 0.1710E-01,
-0.3288E-02, -0.1242E-01, 0.3994E-01, -0.2851E-01, -0.1280E-01, 0.5689E-02,
0.1167E-01, -0.1046E-01, -0.4420E-01, 0.3149E-01, -0.7822E-02, -0.1710E-01,
-0.2663E-01, 0.1046E-01, -0.5222E-01, -0.1696E-01, 0.6717E-02, 0.5713E-02,
-0.6640E-03, 0.1154E-02, -0.1782E-02, -0.4440E-02, 0.3765E-01, -0.1412E-02,
-0.6916E-03, -0.1623E-01, -0.2058E-02, -0.1541E-01, 0.7065E-02, 0.2518E-02,
-0.1846E-01, 0.1334E-03, 0.4576E-02, 0.1063E-01, -0.8076E-02, -0.6042E-02,
0.6641E-02, -0.1662E-01, -0.5905E-02, -0.5162E-02, 0.1910E-01, -0.2074E-02,
-0.3284E-03, 0.1508E-01, -0.4365E-02, -0.9563E-03, -0.5731E-02, -0.5096E-03,
0.5434E-02, 0.1398E-01, 0.8761E-02, -0.1575E-02, -0.2858E-02, 0.3182E-03,
.....
0.4921E-03, 0.2274E-03, -0.4626E-03, 0.3484E-03, -0.9728E-03, -0.2420E-03,
-0.1395E-03, 0.4235E-03, 0.1005E-02, -0.4714E-03, -0.4557E-03, -0.1525E-03,
0.9723E-03, 0.6099E-03, -0.2750E-03, 0.3045E-03, 0.2924E-03, -0.2726E-03
2013-12-24T00:00:10.263980Z,
-0.2205E+02, 0.0000E+00, -0.1009E+01, 0.0000E+00, 0.1737E-01, 0.0000E+00,
-0.1401E+00, 0.6873E-01, -0.5232E-01, -0.2781E+00, 0.3883E-02, 0.9452E-04,
-0.1532E+00, 0.1542E+00, -0.6176E-01, -0.2536E+00, -0.5802E-01, -0.5190E-01,
0.1504E+00, -0.4472E-01, 0.1331E+00, 0.5195E-01, 0.2569E-02, -0.6270E-01,
0.5278E-03, 0.5423E-02, 0.8388E-01, -0.4152E-01, 0.6023E-01, 0.2719E-01,
0.5183E-01, -0.1640E-01, -0.1055E-02, 0.5150E-01, 0.2449E-01, 0.4206E-01,
0.5314E-01, -0.1015E-01, -0.3300E-01, 0.7556E-01, -0.3008E-01, 0.2176E-01,
0.1880E-01, -0.5660E-02, -0.3689E-01, 0.2242E-01, -0.6885E-01, 0.7839E-02,
.....
-0.1436E-02, -0.1231E-03, 0.8421E-03, 0.8098E-03, -0.1521E-02, 0.8132E-04,
0.4990E-03, -0.6353E-03, -0.5708E-03, 0.1283E-02, 0.1566E-02, -0.7331E-03,
0.5658E-03, -0.2206E-03, 0.4665E-03, 0.1471E-03, 0.5705E-03, -0.2625E-03,
0.1792E-02, -0.5134E-03, -0.1172E-02, 0.9439E-03, 0.1342E-02, 0.9402E-03,
-0.2401E-02, 0.3270E-03, 0.1546E-03, -0.7356E-03, -0.2079E-02, 0.4944E-03
END INDEXED_DATA
END DATA
END ROPROC_FORMAT_FILE

```

Table 70: Exemple de SPL2.rff (3 pages)



## 9.10. EXEMPLE DE VISUALISATIONS

### 9.10.1 *Résultat de RCL\_visu\_spectro à partir des VTL2*

Cette image a été produite avec le script et les paramètres suivants :

**PR\_make\_staff\_spectro\_from\_VTL2** 4 NBR ISR2 20010923 092000 101000 512 t

La base de données d'orbite et la base des fichiers FGM « SPIN\_RES » étant disponible lors du lancement de la commande, les données de positions et la gyrofréquence des protons ont été automatiquement ajoutées.

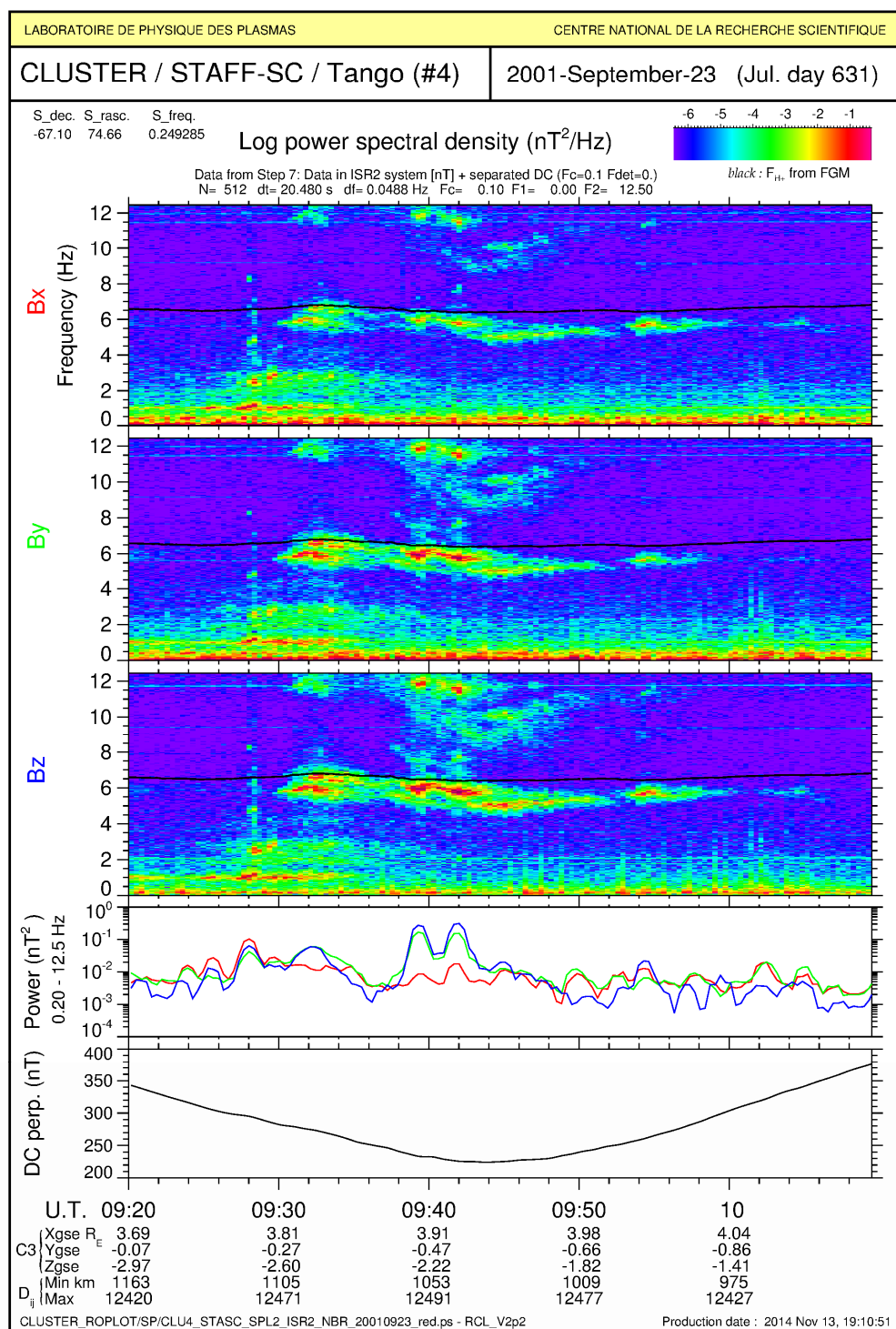


Figure 33: *Exemple de visu\_spectro à partir des VTL2*

### 9.10.2 Résultat de RCL\_visu\_spectro à partir des VTL1

Cette image a été produite avec le script et les paramètres suivants :

**PR\_make\_staff\_spectro\_from\_VTL1** 4 NBR ISR2 20010923 092000 101000 0. 0.6 512 t

Comme précédemment, la base de données d'orbite et la base des fichiers FGM « SPIN\_RES » étant disponible lors du lancement de la commande, les données de positions et la gyrofréquence des protons ont été automatiquement ajoutées.

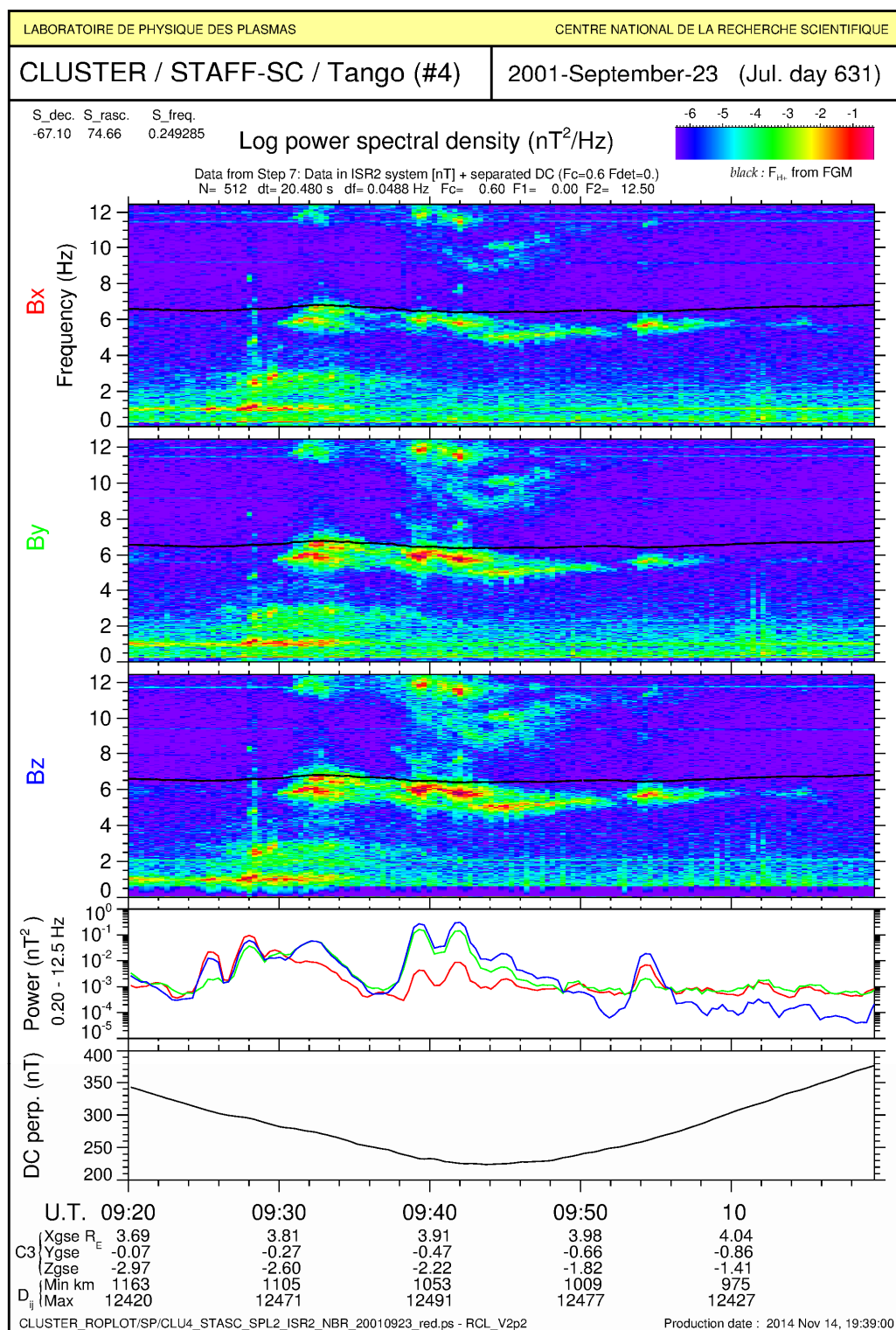


Figure 34: Exemple de visu\_spectro à partir des VTL1

### 9.10.3 Résultat de RCL\_visu\_spectro\_4Bz\_3h

Ce sont les graphiques « quick looks » produit par la chaîne de production décrite au chapitre 2.5.9 et qui travaille sur le base des SPL2 calculés en repère ISR2 avec une résolution temporelle d'environ 10 sec. (256 points en NBR et 4096 en HBR). Un exemple est donné Figure 35.

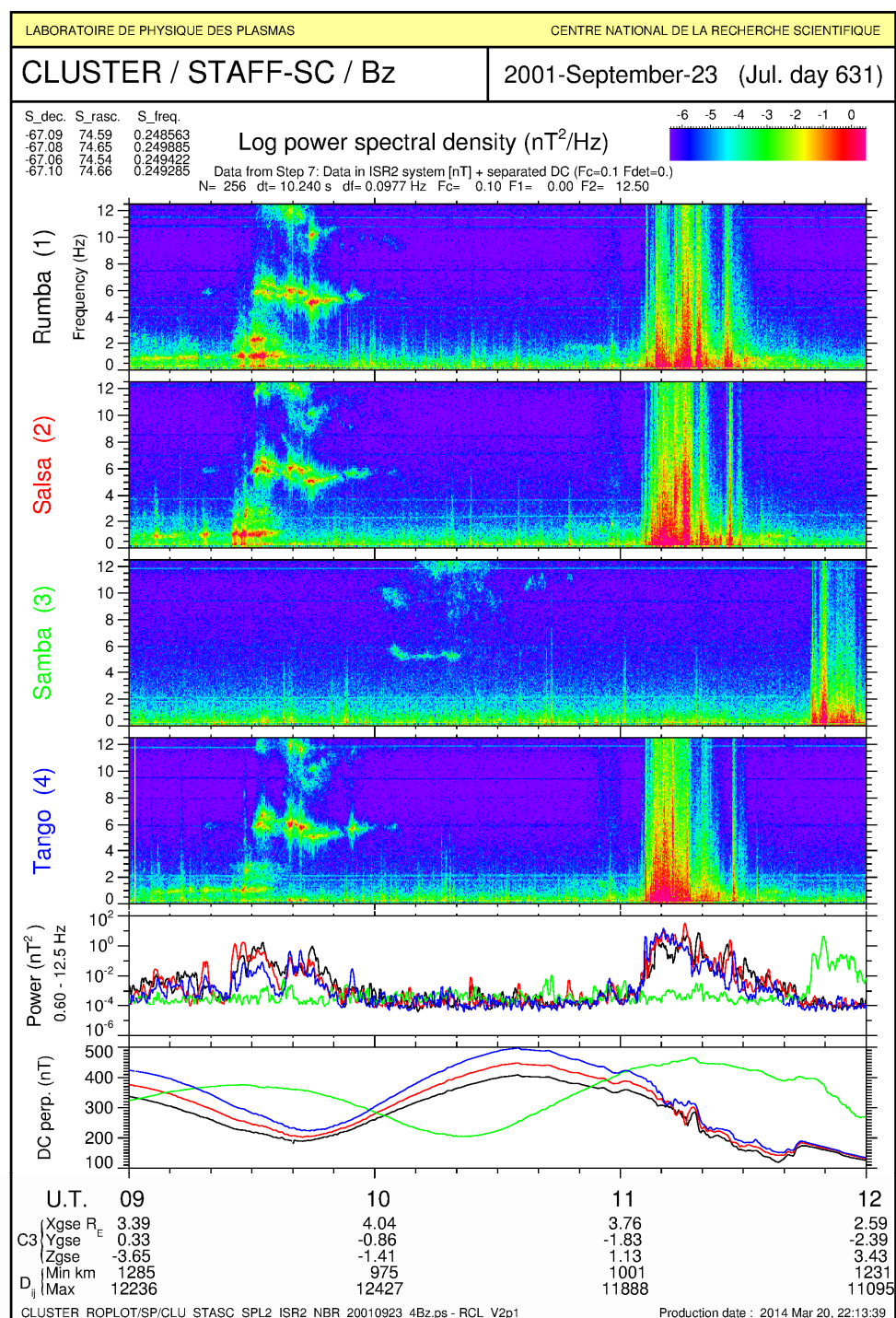


Figure 35 : Exemple de visu\_spectro\_4Bz utilisée comme « Quick looks »

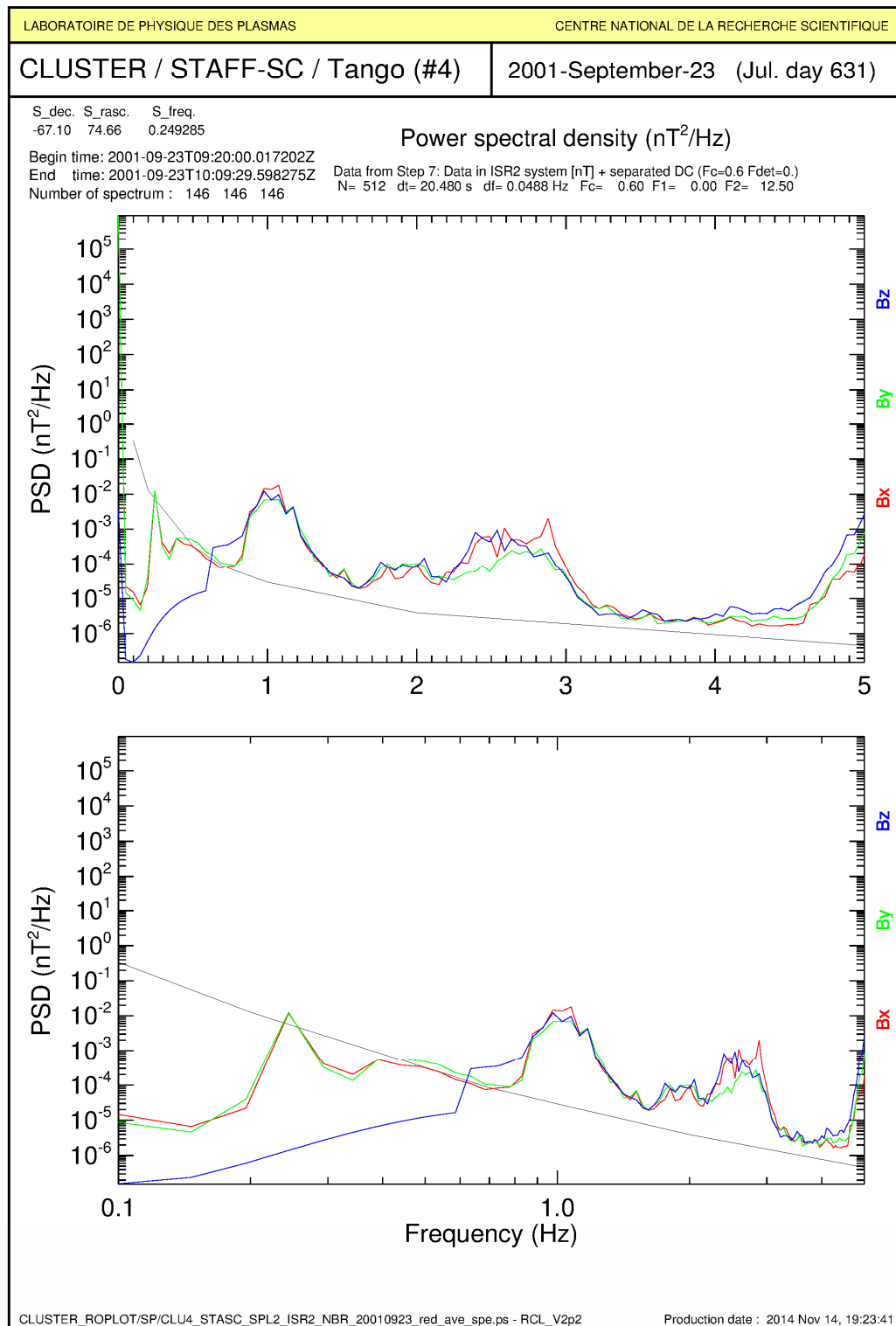
### 9.10.4 Résultat de RCL\_visu\_ave\_spectrum à partir des VTL1

Cette image a été produite avec le script et les paramètres suivants :

**PR\_make\_staff\_spectro\_from\_VTL1** 4 NBR ISR2 20010923 092000 101000 0. 0.6 512 t

Suivi de :

**RCL\_visu\_ave\_spectrum** CLU4\_STASC\_SPL2\_ISR2\_NBR\_20010923\_red.rff 0 0 0. 5. 0. 0. XY y n



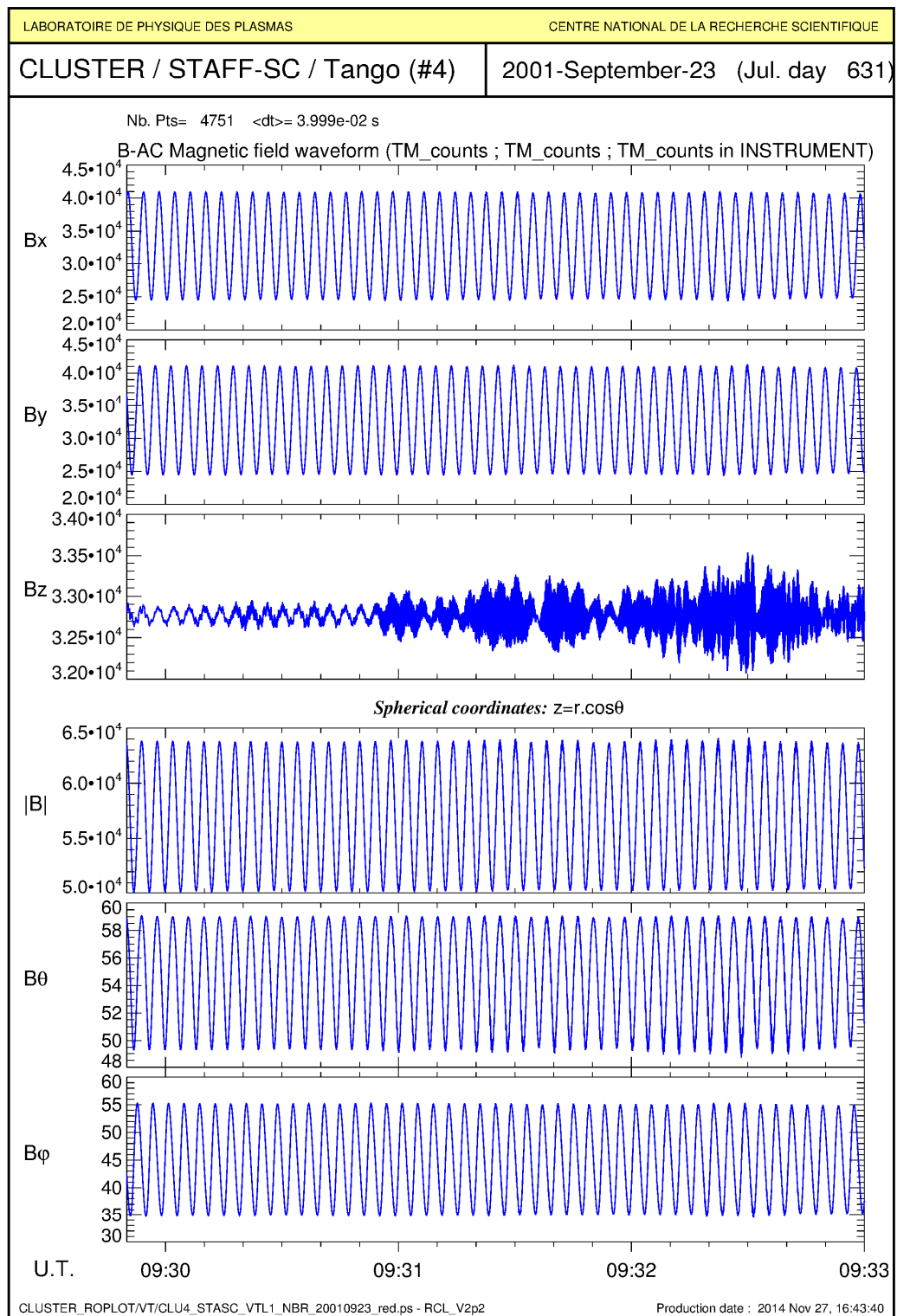
**Figure 36:** Exemple de visu\_spectra à partir des VTL1

### 9.10.5 Résultat de RCL\_visu\_vectime sur les VTL1

Le plot ci-dessous a été obtenu avec le script et les arguments suivants :

**PR\_visu\_staff\_VTL1 4 NBR 20010923\_092800 20010923\_093200**

Les codes couleurs des satellites de CLUSTER sont appliqués (Noir-rouge-vert-bleu).



**Figure 37:**    *Exemple de visu\_vectime sur un VTL1*



### 9.10.6 Résultat de RCL\_visu\_vectime sur les VTL2 en ISR2

Le plot ci-dessous a été obtenu avec le script et les arguments suivants :

```
PR_visu_staff_VTL2 4 NBR ISR2 20010923_092800 20010923_093200
```

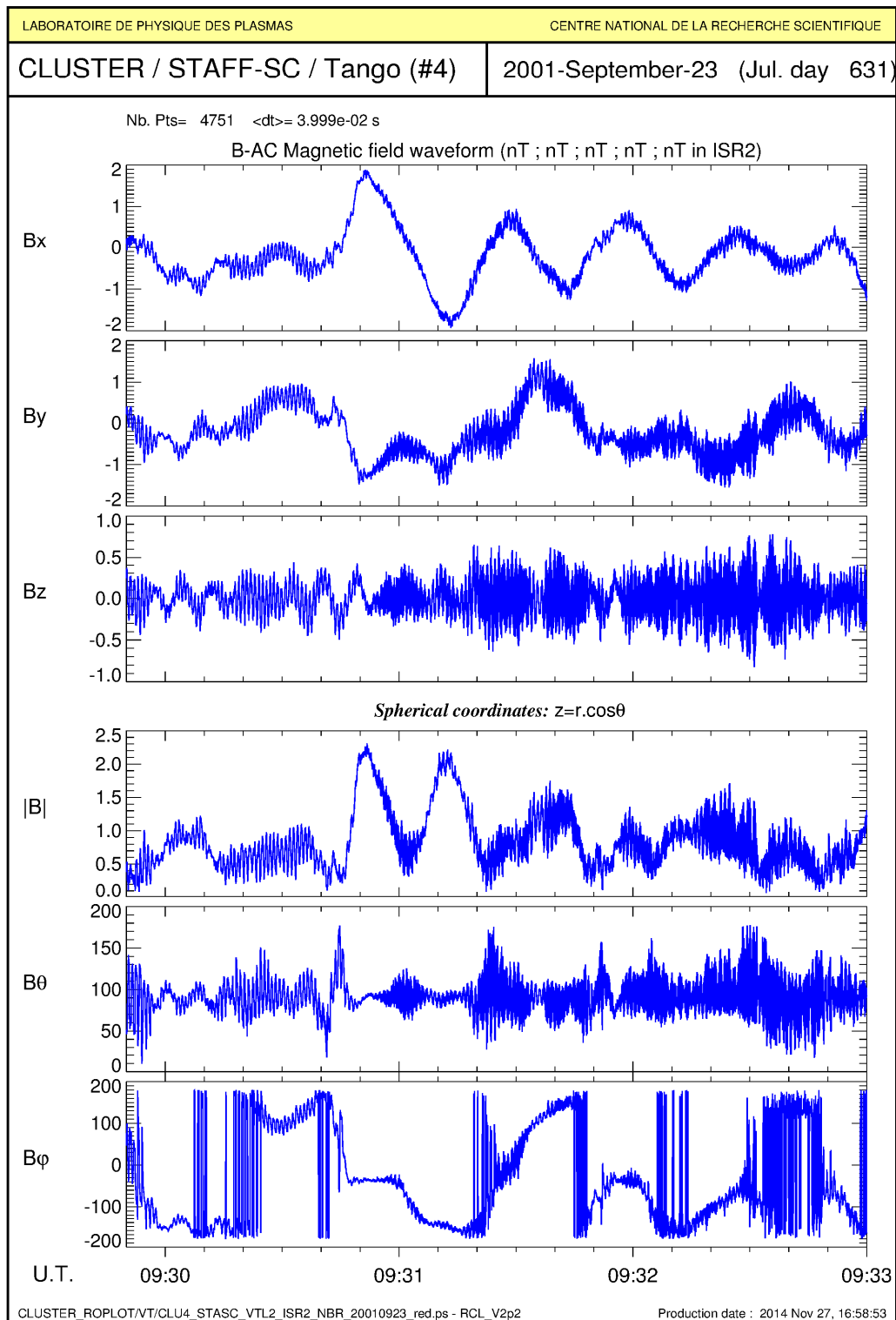
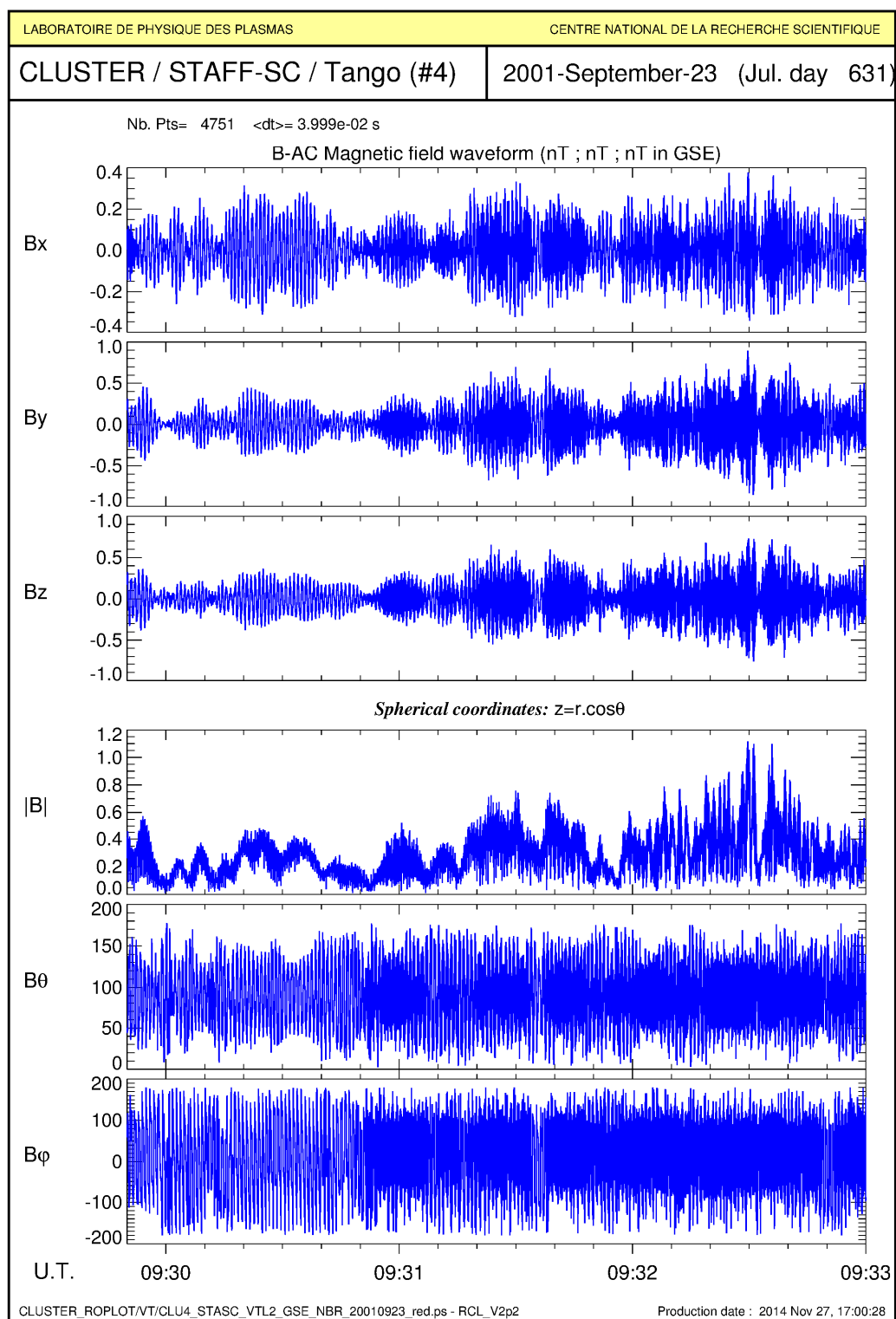


Figure 38: Exemple de visu\_vectime sur un VTL2 en ISR2

### 9.10.7 Résultat de RCL\_visu\_vectime sur les VTL2 en GSE

Le plot ci-dessous a été obtenu avec le script et les arguments suivants :

**PR\_visu\_staff\_VTL2 4 NBR GSE 20010923\_092800 20010923\_093200**

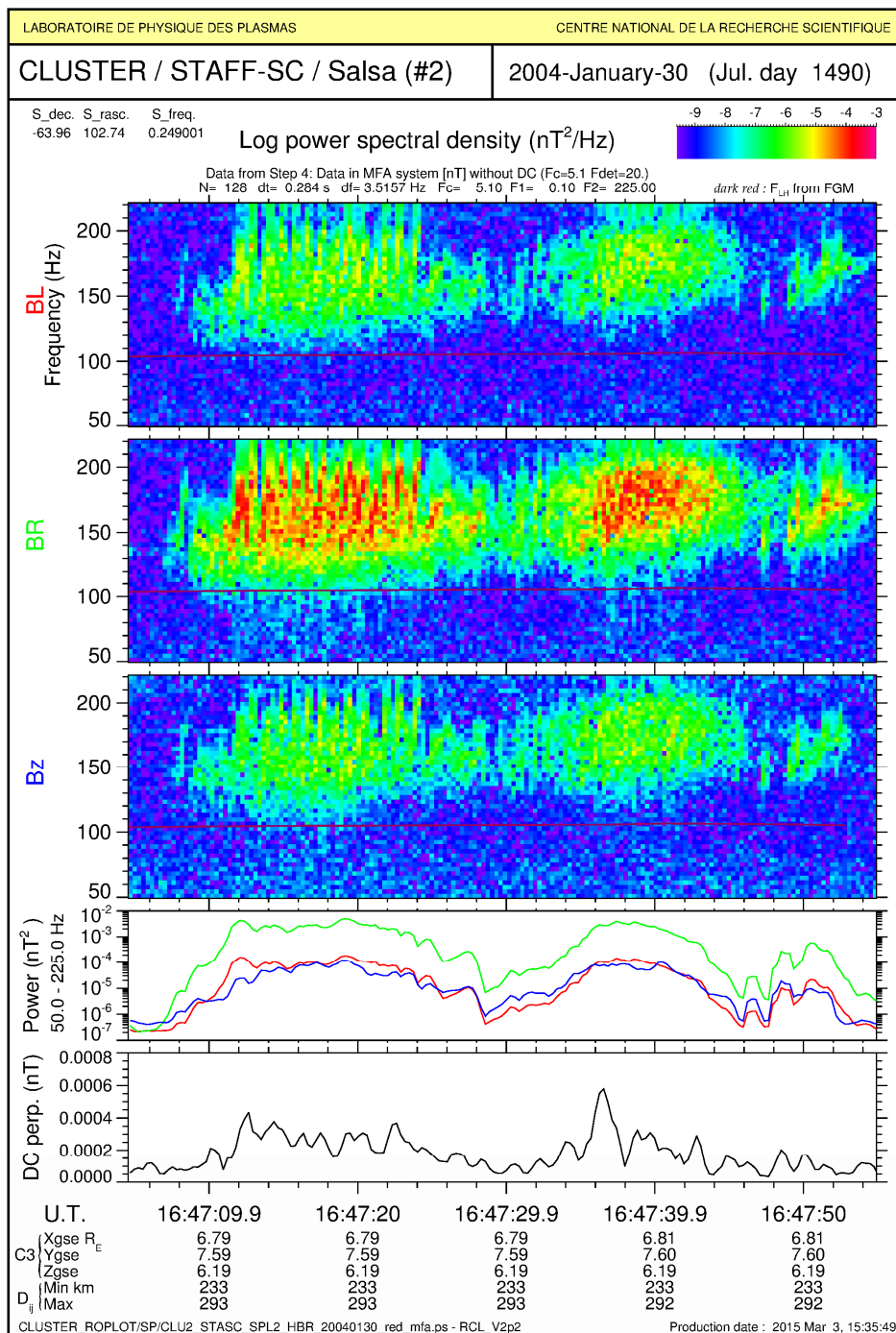


**Figure 39:** Exemple de visu\_vectime sur un VTL2 en GSE

### 9.10.8 Résultat de RCL\_visu\_polar sur des VTL2 en GSE

La Figure 40 ci-dessous a été obtenu avec le script et les arguments suivants :

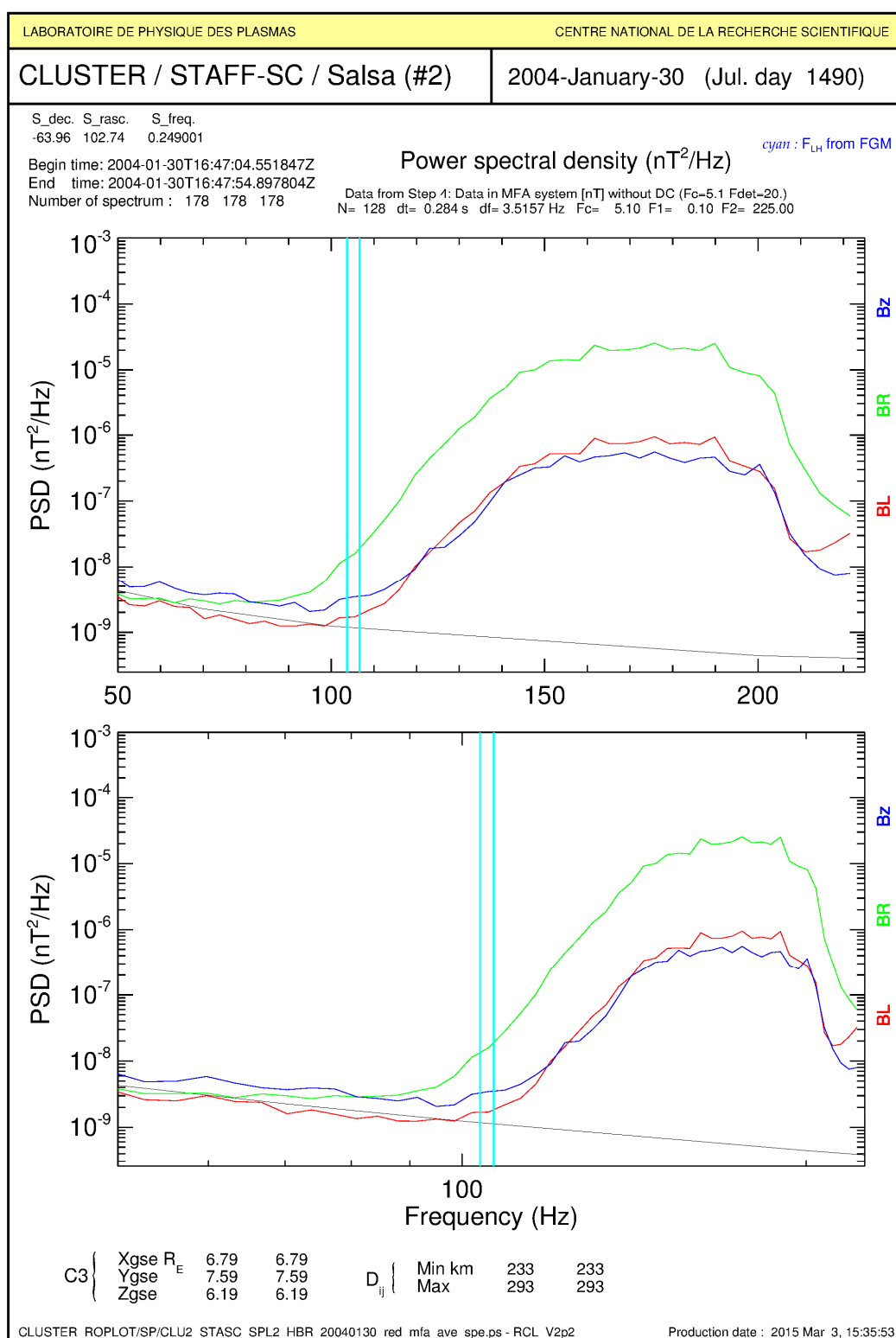
```
PR_compute_staff_polarisation_from_VTL1 2 20040130_164700 20040130_164800 \
128 20. 5.10 50.0 225.0 4 HBR -9.6 -3.0
```



**Figure 40:** Exemple de spectrogramme en repère MFA et en composantes circulaires, montrant la polarisation circulaire droite d'un whisler dans le plan perpendiculaire au champ magnétique DC local.



La figure Figure 41 ci-dessous montre un exemple de spectre moyen en MFA.



**Figure 41:** Spectre moyen du spectrogramme précédent, montrant la nette prépondérance du mode droit (vert).

La Figure 42 montre un exemple de résultat de la commande RCL\_spectro\_to\_polar lancée sur le spectrogramme en MFA de l'exemple précédent, suivi de la commande RCL\_visupolar copolar.resu -5.5 50. 225. De l'examen de cette figure on peut constater que :

- l'excentricité (en bleu cyan) est environ de 0.3, donc une polarisation quasi circulaire ;
- l'angle Theta (bleu foncé) entre le vecteur d'onde  $\mathbf{K}$  et le champ continu  $\mathbf{B}_0$  est de l'ordre de quelques degrés, donc on a une polarisation circulaire droite perpendiculaire à  $\mathbf{B}_0$  (Theta=180° pour le mode gauche);
- l'angle de phase phi est donc dans ce cas indéfini (toutes les couleurs) ;
- l'angle Theta (vert) entre le grand axe de l'ellipse et  $\mathbf{B}_0$  est bien défini et égal à 90°.

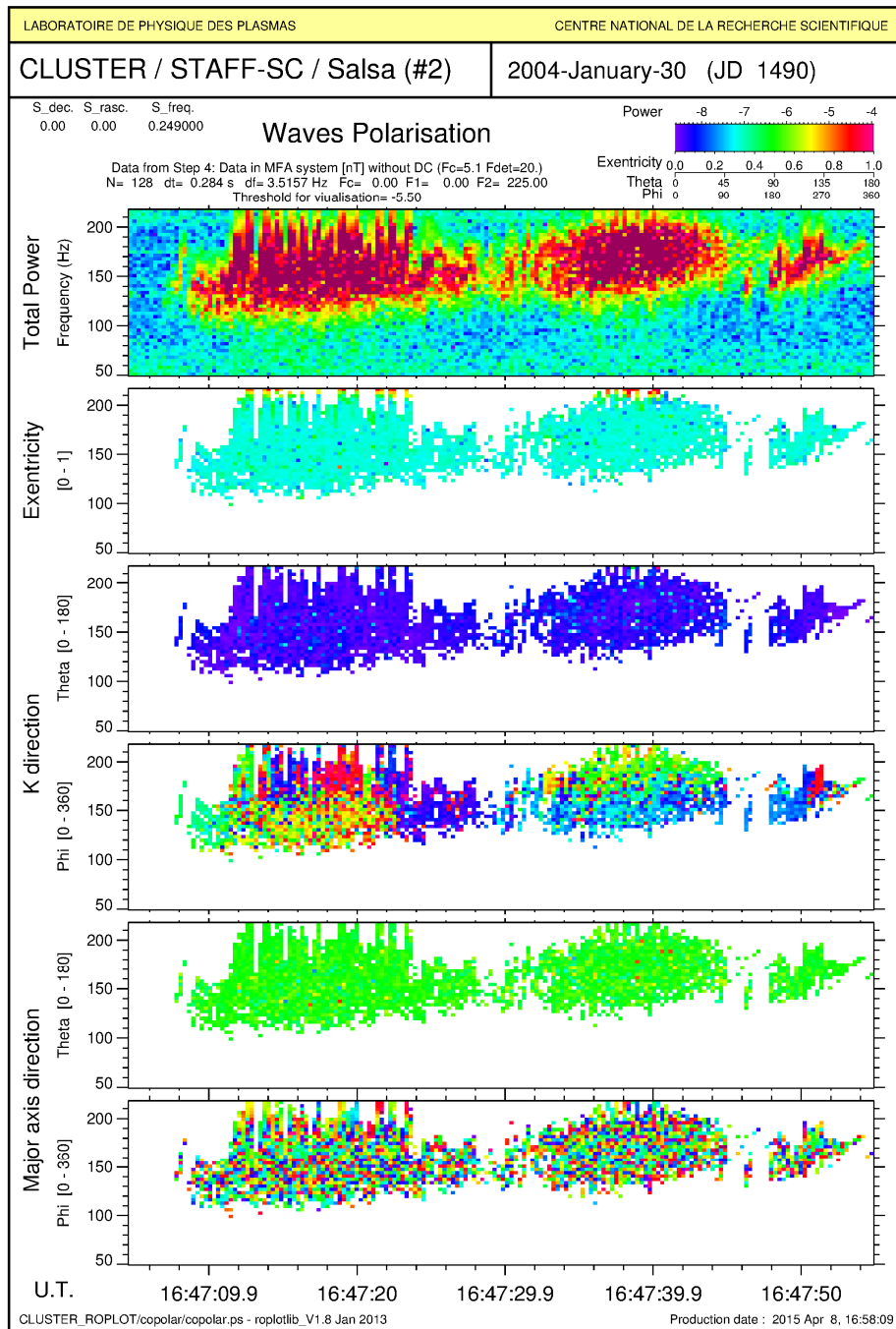


Figure 42: Exemple de visualisation du résultat du programme de calcul de polarisation

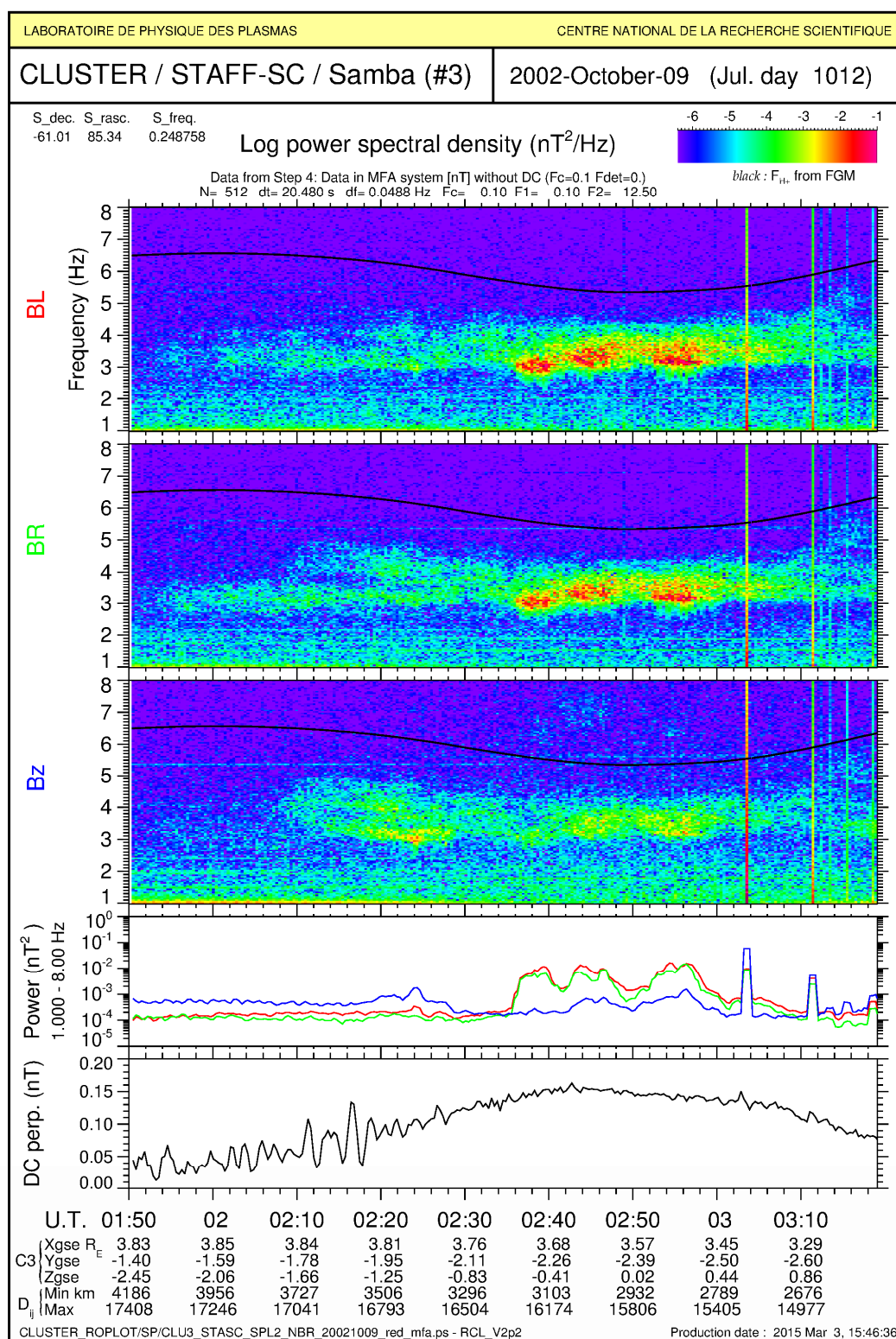


Figure 43: Exemple d'onde pseudo monochromatique linéaire dans le plan perpendiculaire au champ magnétique DC local

La Figure 44 montre un autre exemple de résultat du calcul de polarisation sur le spectrogramme en MFA de l'exemple précédent, suivi de la commande RCL\_visupolar copolar.resu -5.0  
 1. 8. . De l'examen de cette figure, pour la partie où les ondes sont fortes, on peut constater que :

- l'excentricité (en orange) est environ de 0.8, donc une polarisation presque linéaire ;
- l'angle Theta (mélange bleu et rouge) entre le vecteur d'onde  $\mathbf{K}$  et le champ continu  $B_0$  est de l'ordre de 0 ou 180° degrés, donc on a une polarisation quasi linéaire perpendiculaire à  $B_0$  ;
- l'angle de phase phi de  $\mathbf{K}$  (en rouge sur fond vert) est de l'ordre de 350°, et donc la vibration est perpendiculaire à l'axe Soleil-Terre ;
- l'angle Theta (vert) entre le grand axe de l'ellipse et  $B_0$  est très bien défini et égal à 90°, ce qui est normal puisque l'ellipse est quasiment dégénéré en une vibration linéaire, le grand axe étant dans ce cas le sens de la vibration.

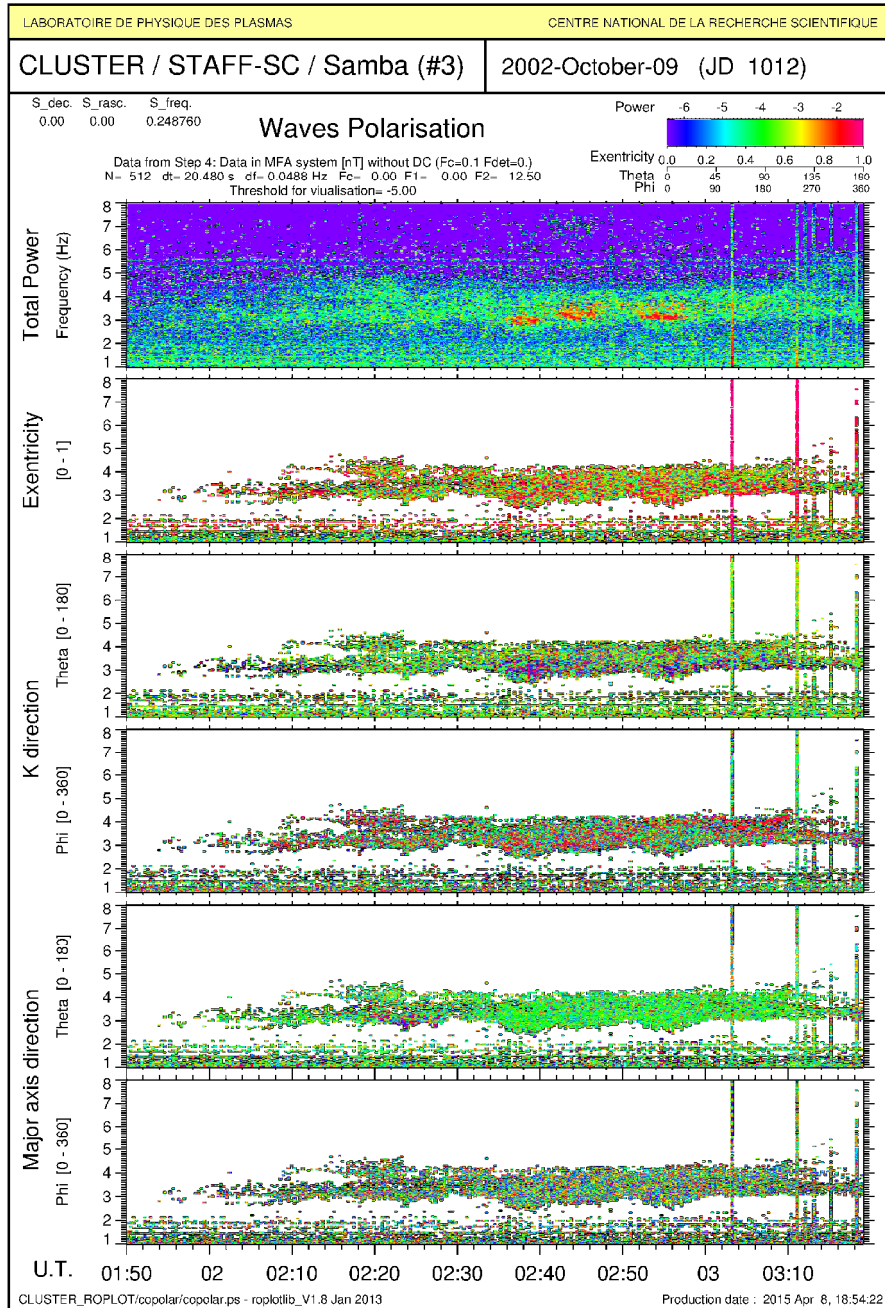


Figure 44: Autre exemple de résultat du calcul de polarisation

### 9.10.9 Résultat de RCL\_visu\_CLUPOS / 3 heures

Cette image a été produite avec le script et les paramètres suivants :

**RCL\_get\_data\_CLUPOS** 2008 12 09 00 00 00 190.1 10. gse

Suivi de : **RCL\_visu\_CLUPOS**

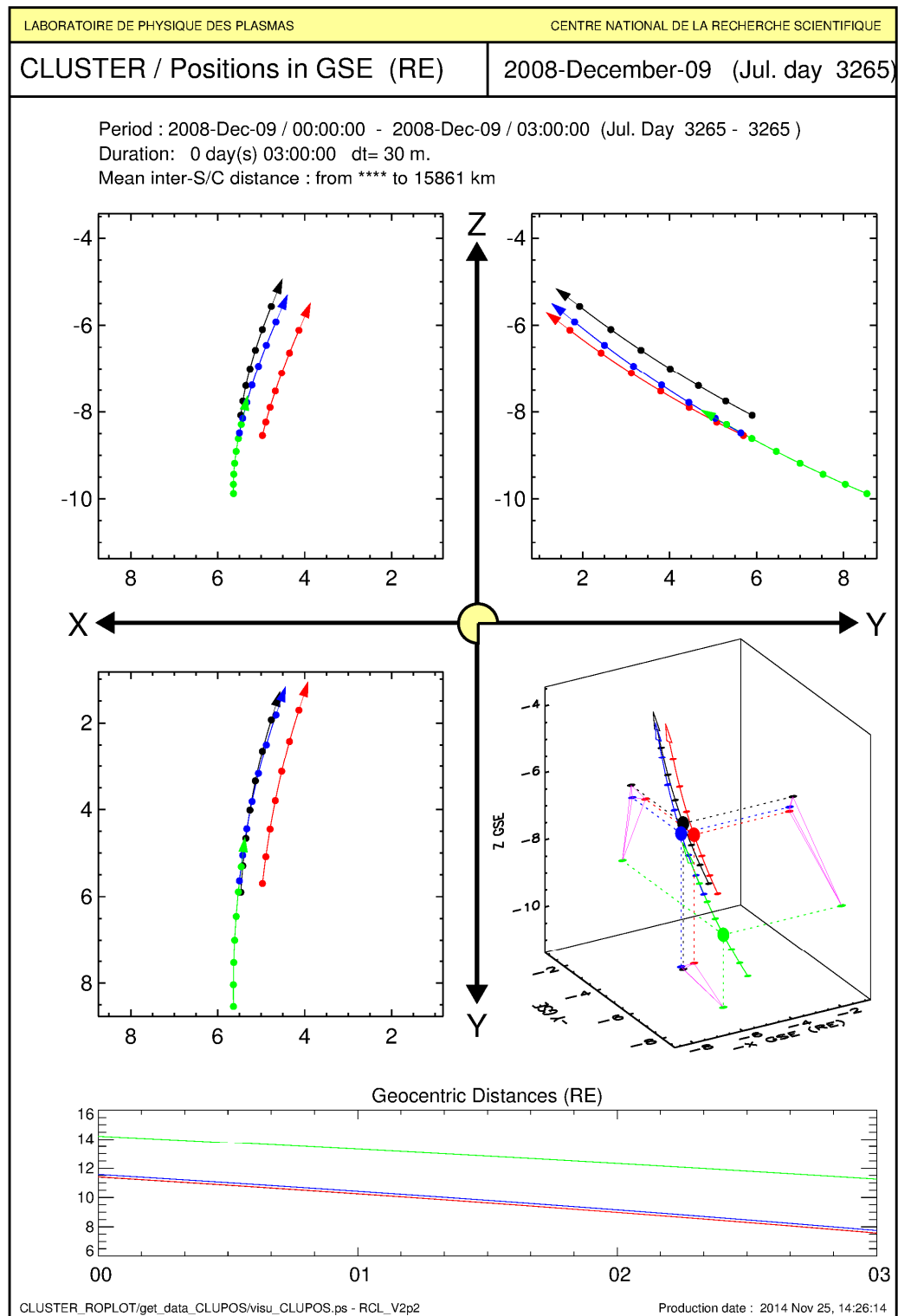
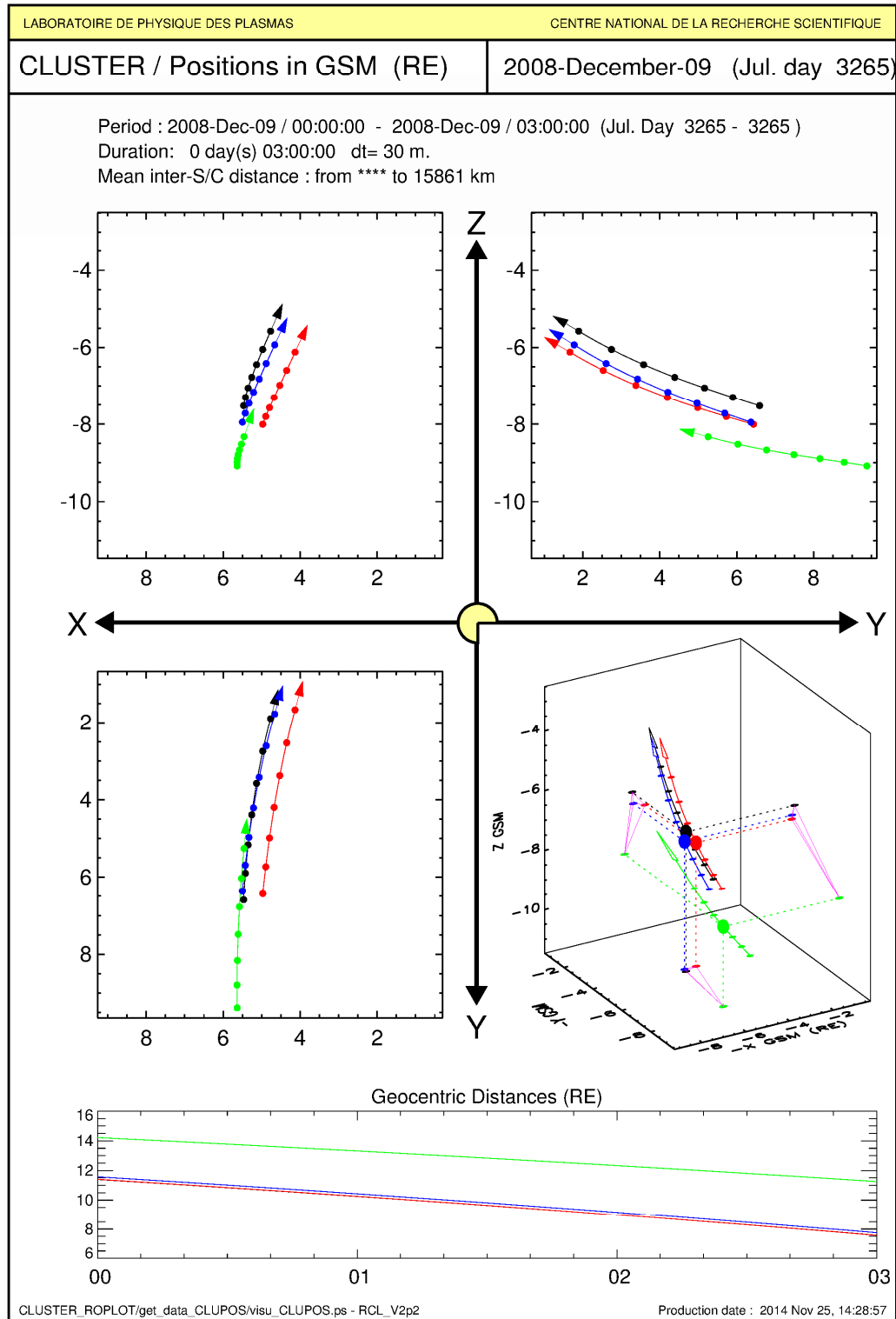


Figure 45: Exemple de visu\_CLUPOS sur 3 heures en GSE

Même chose en GSM avec :

**RCL\_get\_data\_CLUPOS** 2008 12 09 00 00 00 190.1 10. gsm

Suivi de : **RCL\_visu\_CLUPOS**



**Figure 46:** Exemple de visu\_CLUPOS sur 3 heures en GSM

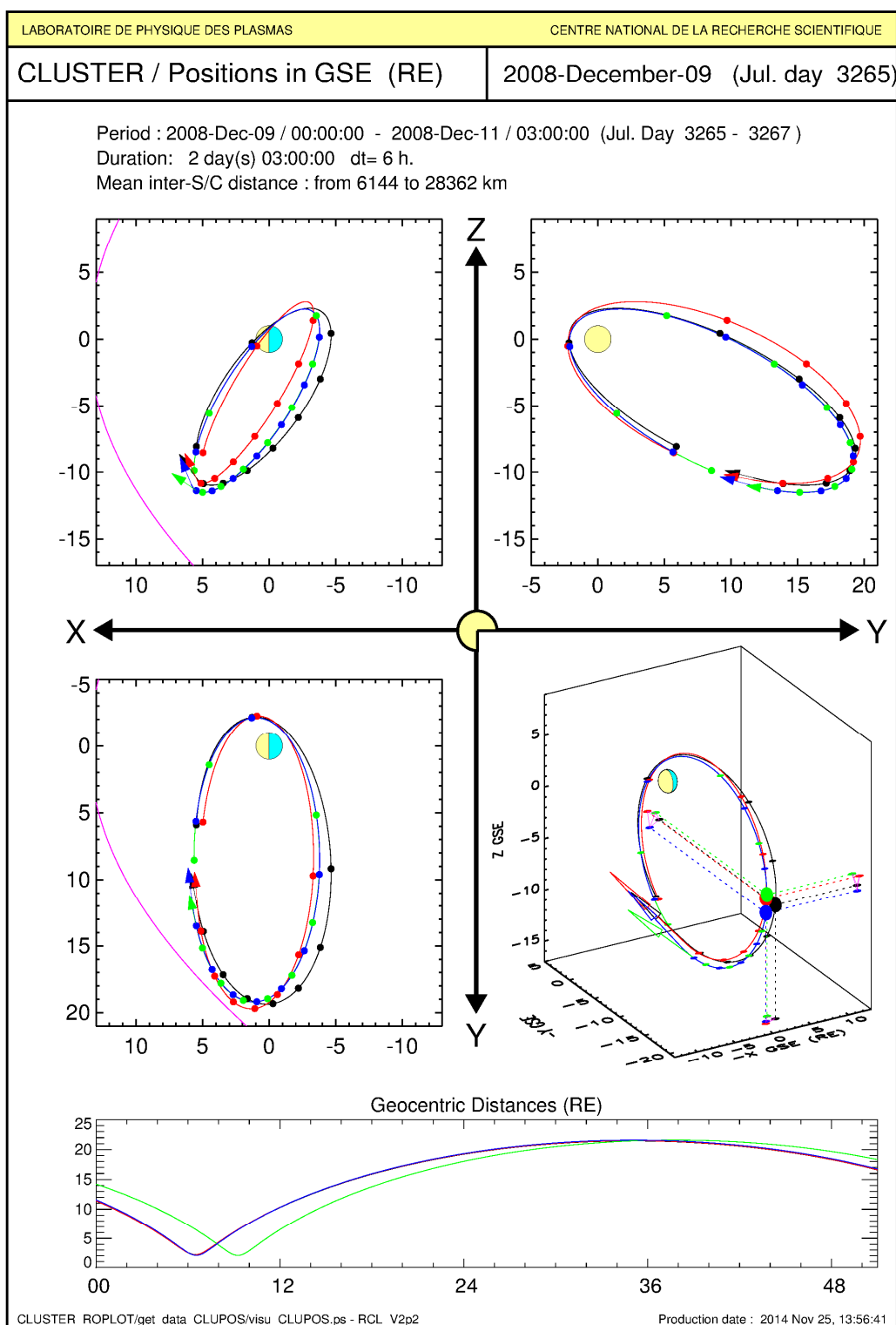


### 9.10.10 Résultat de RCL\_visu\_CLUPOS / orbite complète

Cette image a été produite avec le script et les paramètres suivants :

**RCL\_get\_data\_CLUPOS** 2008 12 09 00 00 00 3070.1 10. gse

Suivi de : **RCL\_visu\_CLUPOS**

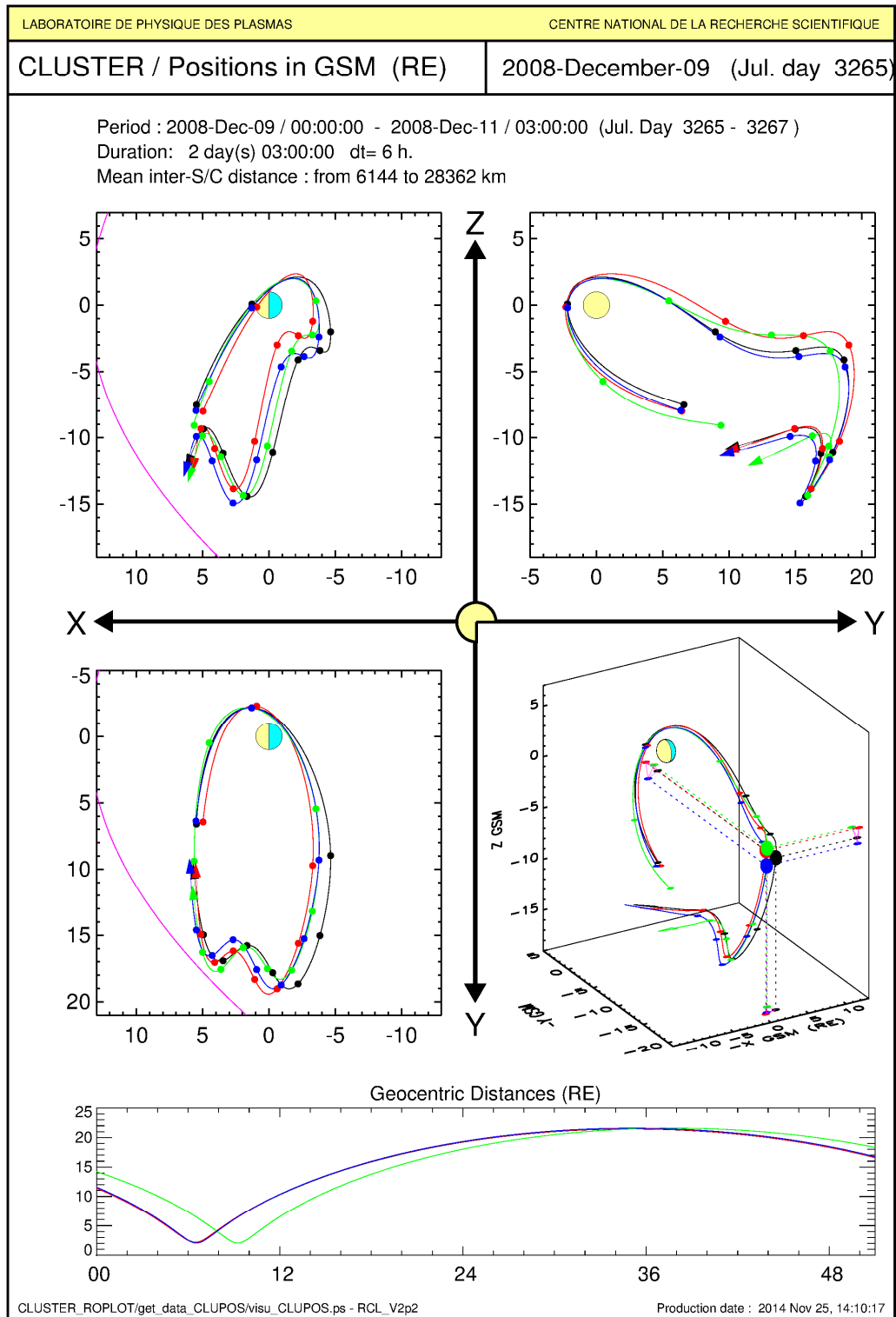


**Figure 47:** Exemple de visu\_CLUPOS sur une orbite complète en GSE

Même chose mais en GSM:

**RCL\_get\_data\_CLUPOS** 2008 12 09 00 00 00 3070.1 10. gse

Suivi de : **RCL\_visu\_CLUPOS**



**Figure 48:** *Exemple de visu\_CLUPOS sur une orbite complète en GSM*



### 9.10.11 Résultat de RCL\_visu\_CLUGEOM

Cette image a été produite avec le script et les paramètres suivants :

**RCL\_get\_data\_CLUGEOM** 2008 12 09 15. gse l

Suivi de : **RCL\_visu\_CLUGEOM**

Qui produit les fichiers visu\_CLUGEOM\_posit.png et visu\_CLUGEOM\_tetra.png ci-dessous.

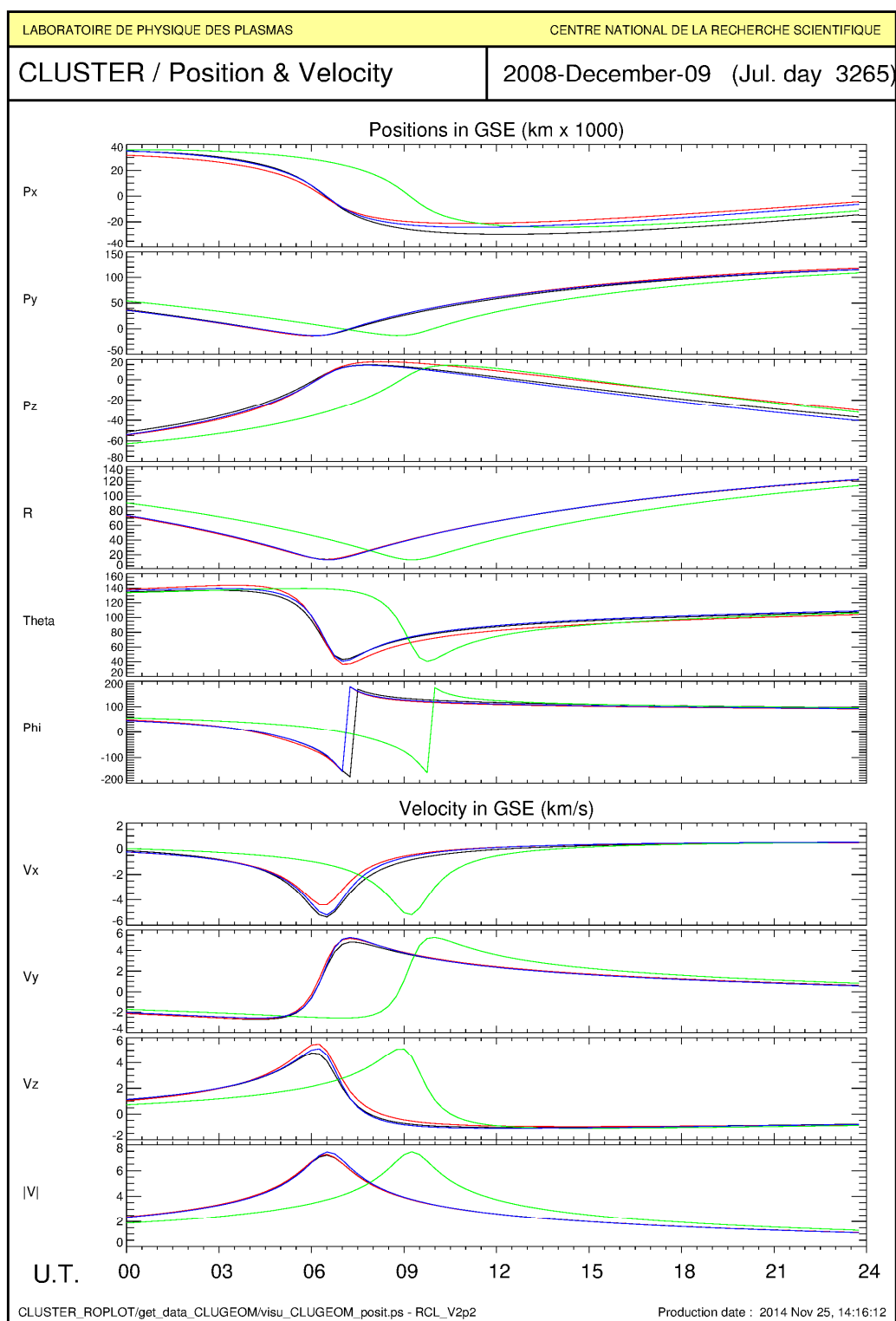


Figure 49: Exemple de visu\_CLUGEOM, partie « positions et vitesses »

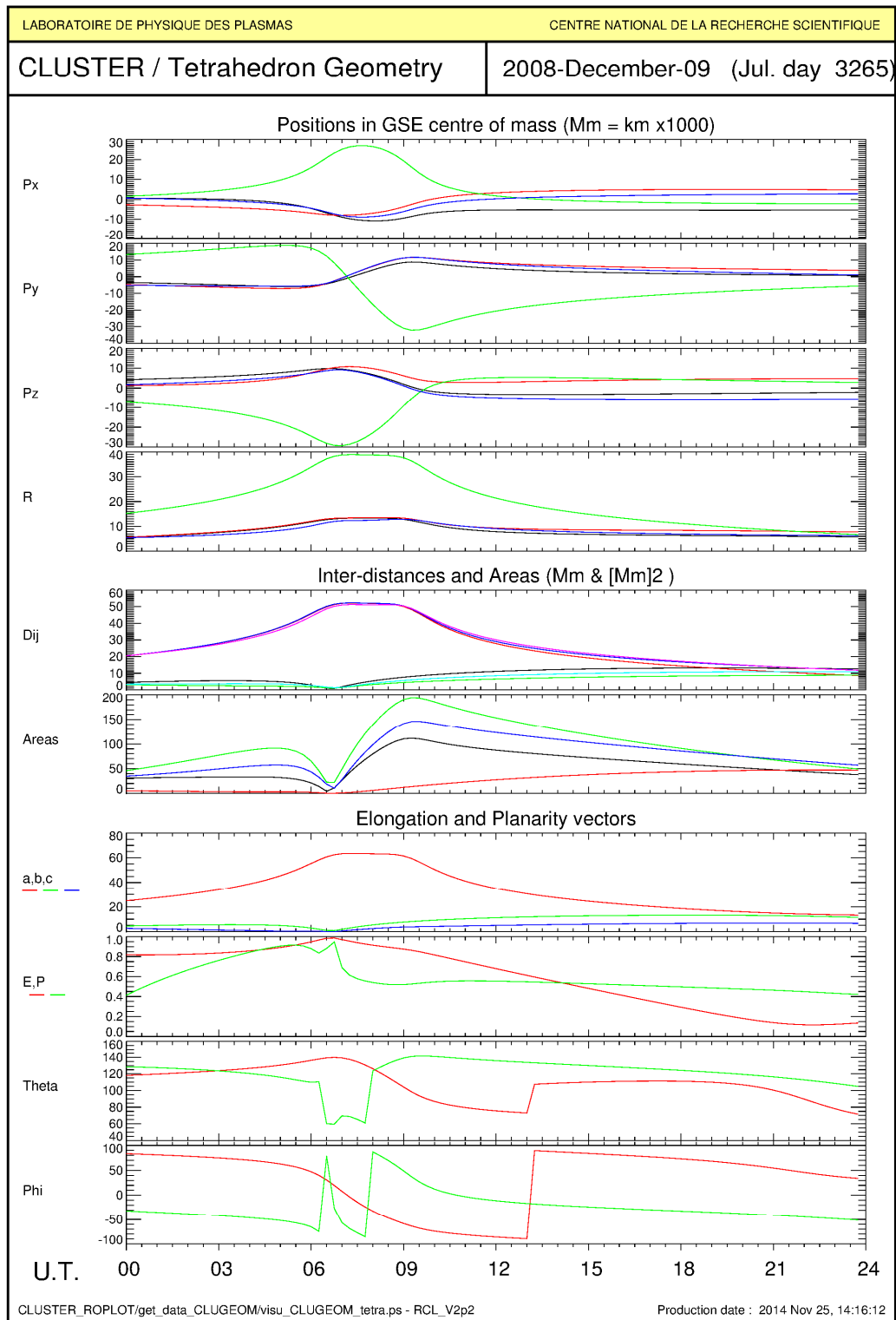


Figure 50: Exemple de visu\_CLUGEOM, partie « géométrie du tétraèdre »

## 9.11. RÉSUMÉ DES COMMANDES RCL

### 9.11.1 Liste fonctionnelle des commandes RCL

Cette liste est celle obtenue par la commande **RCL\_menu**, simplement remise en forme.

#### • *Software Informations :*

RCL_menu	: return this list in ASCII format.
RCL_list	: return list of all RCL commands: create RCL_list.txt
RCL_version	: return current RCL version used, ex: RCL_V1p8
RCL_make_minidoc	: create file mini_doc.txt
RCL_compare_versions	: compare two different versions of RCL package
RCL_compare_versions_long	: same as previous but more deeper

#### • *Tools to handle RFF files (general)*

RCL_check_rff	: check content of a RFF file
RCL_check_rff_dir	: check of all RFF files of a given directory
RCL_info_rff	: return main info of given RFF file
RCL_info_rff_dir	: return main info of all RFF files of a given directory
RCL_clean_rff	: clean content of a RFF file
RCL_copy_rff	: copy a RFF file into another, name updated
RCL_diff_rff	: compute difference between 2 RFF files
RCL_move_rff	: move a RFF file with update of file name inside
RCL_reduce_time_rff	: reduce the time period of a RFF file

#### • *Generic commands*

RCL_list_block_WF	: Return the list of all blocks in a WF file
RCL_waveform_to_vectime	: Convertie un RFF de classe WaveForm vers un classe Vectime
RCL_vectime_to_spectro	: Produce a spectrogram from a VecTime file
RCL_spectro_to_polar	: Produce a spectrogram from a VecTime file

#### • *Access to local CLUSTER/STAFF-SC L1 data base*

RCL_get_data_CLUSTA_VTL1	: get entire day of CLUSTER/STA VTL1
RCL_get_data_CLUSTA_VTL2	: get entire day of CLUSTER/STA VTL2
RCL_get_data_CLUSTA_WFL1_forVTL1	: get entire day of CLUSTER/STA WFL1 + epsi before
RCL_get_data_CLUSTA_VTL1_forVTL2	: get entire day of CLUSTER/STA VTL1 + epsi before/after
RCL_get_data_CLUSTA_VTL1_forSPL2	: get entire day of CLUSTER/STA VTL1 + epsi before

#### • *Access to local CLUSTER/Orbit data base*

RCL_get_data_CLUP0S	: read .orb from local database and get 4 S/C positions/velocity
RCL_get_data_CLUGEOM	: read .orb from local database and get tetrahedron geometry
RCL_give_CLUORB_period	: give list of orbit duration for all the mission

#### • *Access to local CLUSTER/FGM L2 data base*

RCL_get_data_CLUFGM	: read the day file.cef in the local database and create a .bav
RCL_fgm_cef_to_rff	: convert a fgm.cef file into a fgm_VTL2.rff file
RCL_fgm_cef_to_bav	: convert a fgm.cef file into a fgm.bav file
RCL_fgm_cef_gse_to_sr2	: convert a fgm.cef file with data in GSE into SR2 system

• *Access to CSA/FGM L2 data base*

RCL\_download\_data\_CLUFGM\_oneday : download cef data from CSA (SPIN, 5VPS or FULL)  
 RCL\_download\_data\_CLUFGM\_onemonth : download cef data from CSA with tree creation  
 RCL\_download\_data\_CLUFGM\_onemonth\_nolog : same without log file  
 RCL\_download\_data\_CLUFGM\_oneyear : download cef data from CSA with tree creation

• *CLUSTER/STAFF-SC processing*

RCL\_vectime\_L1\_to\_spectro\_L2 : Calibrate a L1 STAFF-SC RFF and produce spectrogram  
 RCL\_vectime\_calibration\_CLUSTA : Calibrate a L1 STAFF-SC RFF VecTime file  
 RCL\_vectime\_L1\_to\_cef : convert a RFF VTL1 into a cef DWF file  
 RCL\_vectime\_L2\_to\_cef : convert a RFF VTL2 into a cef CWF file  
 RCL\_spectro\_L2\_to\_cef : convert a SPL2.rff into a CS.cef  
 RCL\_give\_spin\_dir : give spin direction from VTL1 data base

• *Mass production for STAFF-SC :*

RCL\_product\_VTL1\_oneday : product VTL1.rff file 1 day 1 sat  
 RCL\_product\_VTL1\_onemonth\_nolog : product VTL1.rff files 1 month 4 sat  
 RCL\_product\_VTL1\_onemonth : product VTL1.rff files 1 month 4 sat + log\_file  
 RCL\_product\_VTL1\_oneyear : product VTL1.rff files 1 year 4 sat  
 RCL\_product\_VTL1\_daily : product VTL1.rff file 1 day 1 sat without Tcor

RCL\_product\_VTL2\_oneday : product VTL2.rff files 1 day 1 sat  
 RCL\_product\_VTL2\_onemonth\_nolog : product VTL2.rff files 1 month 4 sat  
 RCL\_product\_VTL2\_onemonth : product VTL2.rff files 1 month 4 sat + log\_file  
 RCL\_product\_VTL2\_onemonth\_onesat : product VTL2.rff files 1 month 1 sat + log\_file

RCL\_product\_SPL2\_oneday : product SPL2.rff files 1 day 1 sat  
 RCL\_product\_SPL2\_onemonth\_nolog : product SPL2.rff files 1 month 4 sat  
 RCL\_product\_SPL2\_onemonth : product SPL2.rff files 1 month 4 sat + log\_file  
 RCL\_product\_SPL2\_oneyear : product SPL2.rff files 1 year 4 sat  
 RCL\_product\_SPL2\_oneday\_fordaily : product SPL2.rff files 1 day 1 sat without Tcor  
 RCL\_product\_SPL2\_daily : product SPL2.rff files 1 day 4 sat 2 modes

RCL\_product\_DWF\_oneday : product DWF.cef files 1 day 1 sat  
 RCL\_product\_DWF\_onemonth\_nolog : product DWF.cef files 1 month 4 sat  
 RCL\_product\_DWF\_onemonth : product DWF.cef files 1 month 4 sat + log\_file  
 RCL\_product\_DWF\_oneyear : product DWF.cef files 1 year 4 sat

RCL\_product\_CWF\_oneday : product CWF.cef file 1 day 1 sat  
 RCL\_product\_CWF\_onemonth\_nolog : product CWF.cef files 1 month 4 sat  
 RCL\_product\_CWF\_onemonth : product CWF.cef files 1 month 4 sat + log\_file  
 RCL\_product\_CWF\_oneyear : product CWF.cef files 1 year 4 sat

RCL\_product\_CS\_oneday : product CS.cef files 1 day 1 sat  
 RCL\_product\_CS\_onemonth\_nolog : product CS.cef files 1 month 4 sat  
 RCL\_product\_CS\_onemonth : product CS.cef files 1 month 4 sat + log\_file  
 RCL\_product\_CS\_oneyear : product CS.cef files 1 year 4 sat

RCL\_product\_visu\_SPL2\_oneday : product SPL2.ps files 1 day 4 sat 4Bz  
 RCL\_product\_visu\_SPL2\_onemonth\_nolog : product SPL2.ps files 1 month 4 sat 4Bz  
 RCL\_product\_visu\_SPL2\_onemonth : product SPL2.ps files 1 month 4 sat 4Bz + log\_file  
 RCL\_product\_visu\_SPL2\_daily : product SPL2.ps files 1 day 4 sat 4Bz  
 RCL\_product\_PNG\_LowRes : product PNG\_LowRes files from PS data base

• **Mass production for orbit visualization :**

RCL\_product\_visu\_orbit\_oneday : product orbit graphical files 1 day 4 sat  
 RCL\_product\_visu\_orbit\_onemonth\_nolog : product orbit graphical files 1 month 4 sat  
 RCL\_product\_visu\_orbit\_onemonth : same with log file  
 RCL\_product\_visu\_orbit\_oneyear : product orbit graphical files 1 year 4 sat

• **Time Utilities :**

RCL\_current\_date : return date as 2012-08-28 14:08:16 day 241 Julsec 1346155696  
 RCL\_nday\_of\_month : return number of day in a month  
 RCL\_next\_day : return date of next day, as '20070925'  
 RCL\_previous\_day : return date one day before, as '20070925'  
 RCL\_decode\_datiso : for date as 2001-09-23T09:17:03.000Z return 2001 09 23 09 17 03  
 RCL\_decode\_datim : for date as 20010923\_091703 return 2001 09 23 09 17 03  
 RCL\_encode\_datiso : return ISO\_date from year month day hour min sec  
 RCL\_list\_days\_of\_month : return a list of day for a given year/month as 01 02 03 30

• **Visualization tools :**

RCL\_visu\_spectro : visualization of a Spectrogram RFF file  
 RCL\_visu\_spectro\_3H : visualization of a Spectrogram RFF file (8 x 3H files)  
 RCL\_visu\_spectro\_4Bz : visualization of 4 Spectrogram RFF file Bz only  
 RCL\_visu\_spectro\_4Bz\_3H : visualization of 4 Spectrogram RFF file Bz only (8 x 3H files)  
 RCL\_visu\_ave\_spectrum : visualization of an average spectrum from Spectrogram file  
 RCL\_visu\_vectime : visualization of a VecTime RFF file  
 RCL\_visu\_vectime\_widget : launch a widget for using visu\_vectime  
 RCL\_visu\_polar : visualization of a copolar.resu file  
 RCL\_visu\_CLUP0S : visualization of a clupos.resu file  
 RCL\_visu\_CLUGE0M : visualization of a clugeom.resu file

• **Graphics Utilities :**

RCL\_ps\_to\_pdf : create PDF file with or without image compression  
 RCL\_ps\_to\_png : create PNG file for a given resolution (dpi, 300=good) 16M colors  
 RCL\_ps\_to\_png\_256 : create PNG file for a given resolution (dpi, 300=good) 256 colors

• **System & Directories properties :**

RCL\_system\_info : return system information (host, system, machine, OS etc.)  
 RCL\_dir\_size : give directory size (octets or Mo)  
 RCL\_dir\_size\_tree : give directory size (octets or Mo) for all the tree  
 RCL\_dir\_size\_pretty\_tree : same with pretty presentation  
 RCL\_dir\_properties : give directory size (octets or Mo), number of files /directories  
 RCL\_dir\_properties\_tree : give directory properties for all the tree  
 RCL\_dir\_properties\_pretty\_tree : same with pretty presentation  
 RCL\_search\_duplicates : search and list duplicate files in a tree

**Table 71:** Liste fonctionnelle des commandes RCL (3 pages)

### 9.11.2 Liste alphabétique des commandes avec arguments

Usage and list of arguments can be given by using help option of each command. Table 72 hereafter give alphabetic list of all RCL commands.

RCL_check_rff_dir RCL_check_rff RCL_clean_rff RCL_compare_versions RCL_compare_versions_long RCL_copy_rff	toto toto.rff toto.rff RCL_previous_dir RCL_previous_dir toto1.rff toto2.rff
RCL_current_date RCL_decode_datiso RCL_decode_datim RCL_diff_rff RCL_dir_properties RCL_dir_properties_tree RCL_dir_properties_pretty_tree RCL_dir_size RCL_dir_size_tree RCL_dir_size_pretty_tree RCL_download_data_CLUFGM_oneday RCL_download_data_CLUFGM_onemonth RCL_download_data_CLUFGM_onemonth_nolog RCL_download_data_CLUFGM_oneyear RCL_encode_datiso	/ No arguments Datiso as 2001-09-23T09:17:03.000Z Date_time as 20010923_091703 toto1.rff toto2.rff /Optional : Mo, DIR, DIR Mo /Optional : Mo, DIR, DIR Mo /Optional : Mo, DIR, DIR Mo /Optional : Mo, DIR, DIR Mo /Optional : Mo, DIR, DIR Mo /Optional : Mo, DIR, DIR Mo SatNum year month day Mode year month Mode year month Mode year Mode year month day hour min sec
RCL_fgm_cef_to_bav RCL_fgm_cef_to_rff	toto.cef toto.cef
RCL_get_data_CLUFGM RCL_get_data_CLUGEOM RCL_get_data_CLUPOS RCL_get_data_CLUSTA_VTL1 RCL_get_data_CLUSTA_VTL2 RCL_get_data_CLUSTA_VTL1_forSPL2 RCL_get_data_CLUSTA_VTL1_forVTL2 RCL_get_data_CLUSTA_WFL1_forVTL1 RCL_give_CLUORB_period RCL_give_spin_dir	SatNum year month day Mode year month day hh mm ss duration dt coord year month day hh mm ss duration dt coord SatNum year month day BitRate SatNum year month day BitRate Coord SatNum year month day BitRate SatNum year month day BitRate SatNum year month day BitRate / No arguments SatNum year month day
RCL_info_rff RCL_info_rff_dir	toto.rff option toto
RCL_list RCL_list_block_WF RCL_list_days_of_month	/ No arguments toto.rff year month
RCL_make_minidoc RCL_menu RCL_move_rff RCL_nday_of_month RCL_next_day RCL_previous_day	/ No arguments / No arguments toto1.rff toto2.rff year month date date

RCL_product_CS_oneday RCL_product_CS_onemonth RCL_product_CS_onemonth_nolog RCL_product_CS_oneyear	SatNum year month day BitRate Coord year month BitRate Coord year month Coord year BitRate Coord
RCL_product_CWF_oneday RCL_product_CWF_onemonth RCL_product_CWF_onemonth_nolog RCL_product_CWF_oneyear	SatNum year month day Coord year month Coord year month Coord year Coord
RCL_product_DWF_oneday RCL_product_DWF_onemonth_nolog RCL_product_DWF_onemonth RCL_product_DWF_oneyear	SatNum year month day BitRate year month BitRate year month BitRate year BitRate
RCL_product_SPL2_daily RCL_product_SPL2_oneday RCL_product_SPL2_oneday_fordaily RCL_product_SPL2_onemonth RCL_product_SPL2_onemonth_nolog RCL_product_SPL2_oneyear	year month day BitRate Coord SatNum year month day BitRate Coord SatNum year month day BitRate Coord year month BitRate Coord year month BitRate Coord year BitRate Coord
RCL_product_visu_orbit_oneday RCL_product_visu_orbit_onemonth RCL_product_visu_orbit_onemonth_nolog RCL_product_visu_orbit_oneyear	year month day year month year month year
RCL_product_visu_SPL2_daily RCL_product_visu_SPL2_oneday RCL_product_visu_SPL2_onemonth RCL_product_visu_SPL2_onemonth_nolog	year month day BitRate Coord year month day BitRate year month BitRate year month BitRate Coord
RCL_product_VTL1_daily RCL_product_VTL1_oneday RCL_product_VTL1_onemonth RCL_product_VTL1_onemonth_nolog RCL_product_VTL1_oneyear	year month day BitRate SatNum year month day BitRate year month BitRate year month BitRate year BitRate
RCL_product_VTL2_oneday RCL_product_VTL2_onemonth RCL_product_VTL2_onemonth_nolog RCL_product_VTL2_onemonth_onesat	SatNum year month day BitRate Coord year month BitRate Coord year month BitRate Coord year month BitRate Coord
RCL_ps_to_pdf.sh RCL_ps_to_png.sh	toto.ps toto.ps 80
RCL_reduce_time_rff	toto1_VT.rff toto2_VT.rff datiso1 datiso2
RCL_search_duplicates RCL_spectro_L2_to_cef RCL_spectro_to_polar RCL_system_info	mode (I or B) toto_VT.rff toto_VT.rff / No arguments
RCL_vectime_calibration_CLUSTA RCL_vectime_L1_to_cef RCL_vectime_L1_to_spectro_L2 RCL_vectime_L2_to_cef RCL_vectime_to_spectro	VTL1.rff VTL2.rff Fdet Fc F1 F2 Step N-Kern N_shift Apod toto_VT.rff VTL1.rff SPL2.rff Fdet Fc F1 F2 Step N-Kern N_shift Apod toto_VTL2.rff VT.rff SP.rff N-Kern N_shift Apod

RCL_version	/ No arguments
RCL_visu_ave_spectrum	SP.rff datiso1 datiso2 f1 f2 Pmin Pmax XY/LR info_orb info_fgm
RCL_visu_CLUGEOM	/ No arguments
RCL_visu_CLUPOS	/ No arguments
RCL_visu_polar	copolar_file threshold
RCL_visu_spectro	SP.rff datiso1 datiso2 f1 f2 pmin pmax
RCL_visu_spectro_3H	SP.rff f1 f2 puimin puimax f_int1 f_int2
RCL_visu_spectro_4Bz	SP1.rff SP2.rff SP3.rff SP4.rff datiso1 datiso2 \
RCL_visu_spectro_4Bz_3H	SP1.rff SP2.rff SP3.rff SP4.rff f1 f2 \
RCL_visu_vectime	toto_VT.rff
RCL_visu_vectime_widget	/ No arguments
RCL_waveform_to_vectime	toto_WF.rff toto_VT.rff

**Table 72:**      *Liste alphabétique des commandes RCL avec arguments (3 pages)*



## 9.12. QUELQUES MOTS SUR LES ROPROC

### 9.12.1 *Historique des Roproc et lien avec les RCL*

Les procédures Roproc (Les « Robert's processing ») ont été développées au CETP dans les années 2000 pour l'exploitation scientifique des données CLUSTER, et ont été en constante évolution durant les premières années d'exploitation. Elles s'appuient pour STAFF sur la base en ascii des données N1 décommutées, pour FGM sur la base des PPDB (fichiers ascii au format standard FGM, dits fichiers « bav » comme « B average »), et sont aussi valables pour la bases des fichiers FGM à haute résolution mais au même format, dits « bhr » comme « B Hight resolution ». Pour les fichiers d'orbite, les procédures s'appuient sur la base des fichiers d'orbite (extension .orb) produits lors du traitement de routine des données STAFF.

Ces procédures produisent des fichiers de résultats en ascii, auto-documentés, et de format aisément compréhensible même en l'absence de programme de lecture : ils sont repris par des procédures de visualisation créant des fichiers PostScript. Les programmes de lecture de ces fichiers de résultats sont disponibles en Fortran 90 ou en IDL.

Les procédures pour STAFF-SC sont largement dérivées des anciennes procédures de traitement des données GEOS, remises à niveau à l'époque pour l'archivage de ces données au CDDP, et étendues à 4 satellites. Les procédures pour FGM et pour les orbites ont été développées dans le même esprit, en utilisant au maximum les bibliothèques existantes, éventuellement étendues. Elles utilisent notamment la bibliothèque de changement de coordonnées Rocotlib [voir 8, Robert, 2003].

Beaucoup de codes de traitement originaux étaient en FORTRAN 77, et ont été reconvertis en F90. Les nouveaux codes ont été écrits en F90. Toutes les visualisations sont en IDL. Les visualisations s'appuient sur un standard développé pour CLUSTER, et sont regroupées en bibliothèques, avec des modules spécialisés, ce qui facilite grandement la création d'éventuelles nouvelles visualisations.

Les procédures Roproc sont livrés sous la forme d'une arborescence de fichiers exécutables, et peuvent être installées sur une machine Linux ou Windows par simple recopie de l'arborescence correspondante du logiciel : Un pack IDL de démonstration est inclus pour les plateformes Sun, Linux, Windows et Mac PPC, ce qui fait qu'une licence IDL n'est pas nécessaire pour travailler et pour produire des graphiques PostScript, PDF ou PNG.

Les Roproc ont servi d'inspiration au projet RCL. En effet plusieurs programmes, ou éléments de programmes, ont été récupérée pour fabriquer les RCL. Néanmoins la programmation en F77 convertie en F90 des Roproc n'étaient pas très propres ni très performante pour un projet devant traiter en production 10 à 15 ans de données. Les codes F90 des RCL ont donc été soigneusement révisés, voire réécrits à la norme F90, en soignant les codes d'erreurs et les fins anormales, afin d'obtenir des codes fiables et performants.

La courte documentation des Roproc données ci-après peut être utile si on souhaite développer d'autres commandes RCL. Néanmoins une adaptation aux bibliothèques RCL décrite au § 7.3 peut être nécessaire.

Les informations ci-dessous sont directement issue du répertoire « doc » du pack Roproc 4.3. Une documentation des Roproc est prévue, mais elle n'existe pas encore à l'heure ou est écrit ce document. A l'origine, le projet RCL devait englober la totalité des Roproc, rendant ce dernier obsolète. Mais le temps a manqué pour mener ce projet à terme, et les deux produits existent aujourd'hui simultanément.

### 9.12.2 Introduction aux Roproc

<hr/> <p style="text-align: center;">ROPROC SHORT DOCUMENTATION</p> <p style="text-align: center;">Creation date : 2010-12-14 18:01:16</p> <p style="text-align: center;">Roproc version= Roproc_4p3</p> <p style="text-align: center;">HOST NAME= cactus HOST TYPE= x86_64-linux OS TYPE= linux MACH TYPE= x86_64</p> <p style="text-align: center;">Author : Patrick ROBERT, 2000-2010, CNRS/CETP-LPP</p> <hr/> <p style="text-align: center;">             %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%              %%              %% INTRODUCTION %%              %%              %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%           </p> <p>RoProc software is a set of procedures, or commands, allowing data processing and visualisations of time series data. Some general info commands are available (see Part II).</p> <p>There is two mains categories of data:</p> <ul style="list-style-type: none"> <li>- waveform data, where the sample rate is constant. Fast Fourier Transform, polarisation computation, etc. are available on this kind of data.</li> </ul> <p>data format are vector block data, time tagged. Example is STAFF experiment of the CLUSTER mission.</p> <p>The list of available Roproc Wave commands are given hereafter.</p> <ul style="list-style-type: none"> <li>- vector data, where the sample rate can be varying, or could have data gaps</li> </ul> <p>data format are vector data, where each vector is time tagged. Example is FGM experiment of the CLUSTER mission</p> <p>In case of constant sampling rate, conversion with wave format is possible. The list of available Roproc Vector commands are given hereafter.</p> <p>There is also a set of Roproc Orbit commands, only available for CLUSTER mission, and a set of Roproc Magnetis Tools, for magnetic field computaion (dipole, IGRF, Tsyganenko 87 until 04), conjugated points, fieldline computation, etc.</p>
--

**Table 73 :**     *Introduction aux Roproc*

### 9.12.3 Liste thématique des Roproc

#### LIST OF ROPROC COMMANDS, SCRIPTS AND MACROS

V 1.1 : P. Robert, CNRS/CETP, 2001 November 9  
Last update : V 4.5 : P. Robert, CNRS/LPP, 2015 January 13

#### A) ROPROC COMMANDS

##### I) Roproc Info commands

rpc\_menu : give menu of all available commands (this file)  
rpc\_list : give full alphabetic list of all rpc\_commands  
rpc\_stat : List of last used Roproc commands  
rpc\_version : give version number of Roproc software

##### II) Roproc Wave commands

###### 1) creating cowave.resu data files

rpc\_cowave\_rcs : Calibration of CLUSTER/STAFF-SC L1 data (by windows)  
rpc\_cowave\_rce : Calibration of CLUSTER/EPW flat data \*\*\* **disused**  
rpc\_cowave\_rds : Calibration of DSP/STAFF-SC N1p data \*\*\* **disused**  
rpc\_cowave\_rcs4 : Run 4 times rpc\_cowave\_rcs for 4 spacecraft  
rpc\_cowave\_rce4 : Run 4 times rpc\_cowave\_rce for 4 spacecraft \*\*\* **disused**

###### 2) reading cowave.resu files

rpc\_readwave : Read and check a cowave.resu file

###### 3) filters : transform cowave.resu into another cowave.resu

rpc\_wave\_filter : Filtering of a cowave.resu file  
rpc\_wave\_to\_mfa\_rce : Transform a cowave.resu file into mfa system \*\*\* **disused**  
rpc\_wave\_to\_mfa\_rcs : Transform a cowave.resu file into mfa system  
rpc\_wave\_to\_mfa\_rds : Transform a cowave.resu file into mfa system \*\*\* **disused**  
rpc\_wave\_to\_gse : Transform a cowave.resu file into gse system  
rpc\_wave\_to\_gsm : Transform a cowave.resu file into gsm system  
rpc\_wave\_to\_mva : Transform a cowave.resu file into MVA system  
rpc\_wave\_to\_mat : Transform a cowave.resu file via a given matrix  
rpc\_wave\_filter4 : Run 4 times rpc\_wave\_filter for 4 S/C  
rpc\_wave\_to\_mfa\_rce4 : Run 4 times rpc\_wave\_to\_mfa\_rce for 4 S/C  
rpc\_wave\_to\_mfa\_rcs4 : Run 4 times rpc\_wave\_to\_mfa\_rcs for 4 S/C  
rpc\_wave\_to\_gse4 : Run 4 times rpc\_wave\_to\_gse for 4 S/C  
rpc\_wave\_to\_gsm4 : Run 4 times rpc\_wave\_to\_gsm for 4 S/C  
rpc\_wave\_to\_mva4 : Run 4 times rpc\_wave\_to\_mva for 4 S/C  
rpc\_wave\_to\_mat4 : run 4 times rpc\_vave\_to\_mat for 4 S/C

###### 4) level 3: transform cowave.resu into L3 product

rpc\_copolar : Computation of wave polarisation from cowave.resu data  
rpc\_cospectro : Computation of spectrogram from cowave.resu data  
rpc\_copolar4 : Run 4 times rpc\_copolar for 4 spacecraft  
rpc\_cospectro4 : Run 4 times rpc\_cospectro for 4 spacecraft

*Roproc\_menu V4.5 Page 1/6*

### III) Roproc Vector commands

#### 1) creating covector.resu data files

rpc\_get\_data\_clufgm : Get FGM data and create the covector.resu file  
 rpc\_get\_data\_clufgm4 : Run 4 times rpc\_get\_data\_clufgm for 4 spacecraft

#### 2) reading covector.resu files

rpc\_readvector : Read and check a readvector.resu file

#### 3) filters : transform covector.resu into another covector.resu

rpc\_vector\_geo\_to\_gse : Change data coordinate system from GEO to GSE  
 rpc\_vector\_gse\_to\_gsm : Change data coordinate system from GSE to GSM  
 rpc\_vector\_gse\_to\_sma : Change data coordinate system from GSE to SMA  
 rpc\_vector\_gse\_to\_sr2 : Change data coordinate system from GSE to SR2  
 rpc\_vector\_to\_mfa : Change data coordinate system from any to MFA  
 rpc\_vector\_to\_mva : Change data coordinate system from any to MVA  
 rpc\_vector\_to\_mat : Change data coordinate system with given matrix

rpc\_vector\_minus\_line : Soustrait a line at covector.resu data  
 rpc\_vector\_minus\_trend : Remove the tendency at covector.resu  
 rpc\_vector\_minus\_sine : compute and remove a sine signal from a file

rpc\_vector\_reduce\_time : extract a time period from a file  
 rpc\_vector\_subsample : subsample a series on n points by averaging  
 rpc\_vector\_timeshift : Time shift of covector.resu data  
 rpc\_vector\_to\_model : Change a covector.resu FGM data by a mag. model

rpc\_vector\_minus\_vector : Difference between two covector.resu files  
 rpc\_vector\_cross\_product : compute A X B for each vector of files A & B  
 rpc\_vector\_average : Average several covector.resu files  
 rpc\_vector\_alitime : Align the time of covector.int.resu on this of covector\_ref.resu, result in covector.resu

rpc\_vector\_geo\_to\_gse4 : Run 4 times rpc\_vector\_geo\_to\_gse for 4 S/C  
 rpc\_vector\_gse\_to\_gsm4 : Run 4 times rpc\_vector\_gse\_to\_gsm for 4 S/C  
 rpc\_vector\_gse\_to\_sma4 : Run 4 times rpc\_vector\_gse\_to\_sma for 4 S/C  
 rpc\_vector\_gse\_to\_sr24 : Run 4 times rpc\_vector\_gse\_to\_sr2 for 4 S/C  
 rpc\_vector\_to\_mfa4 : Run 4 times rpc\_vector\_to\_mfa for 4 S/C  
 rpc\_vector\_to\_mva4 : Run 4 times rpc\_vector\_to\_mva for 4 S/C  
 rpc\_vector\_to\_mat4 : Run 4 times rpc\_vector\_to\_mat for 4 S/C

rpc\_vector\_minus\_line4 : Run 4 times rpc\_vector\_minus\_line for 4 S/C  
 rpc\_vector\_minus\_trend4 : Run 4 times rpc\_vector\_minus\_trend for 4 S/C  
 rpc\_vector\_reduce\_time4 : Run 4 times rpc\_vector\_reduce\_time for 4 S/C  
 rpc\_vector\_subsample4 : Run 4 times rpc\_vector\_subsample for 4 S/C  
 rpc\_vector\_timeshift4 : Run 4 times rpc\_vector\_timeshift for 4 S/C  
 rpc\_vector\_minus\_vector4 : Run 4 times rpc\_vector\_minus\_vector for 4 S/C  
 rpc\_vector\_to\_model4 : Run 4 times rpc\_vector\_to\_model for 4 S/C

#### 5) level 3: transform covector.resu into L3 product

rpc\_addvecpos : Add position of the satellite in covector.resu  
 rpc\_addvecpos4 : Run 4 times rpc\_addvecpos for 4 spacecraft  
 rpc\_alivetime : Time synchronisation of 4 files covector\_i.resu  
 rpc\_cocurldiv : Compute curl & div from 4 time-aligned addvecpos\_N.resu

rpc\_merge\_alivector : Merge 4 time-aligned covector\_N.resu files into a single one

#### IV) Roproc files conversion

##### 1) conversion to vector format

rpc\_wave\_to\_vector : Convert a cowave.resu file into covector.resu  
 rpc\_wave\_to\_vector4 : Run 4 times rpc\_wave\_to\_vector for 4 S/C  
 rpc\_fgm\_bav\_to\_vector : Convert a fgm bav file into a covector.resu

##### 2) conversion to wave format

rpc\_vector\_to\_wave : Convert a covector.resu file into a cowave.resu  
 rpc\_vector\_to\_wave4 : Run 4 times rpc\_vector\_to\_wave for 4 S/C

##### 3) other data file conversions

rpc\_vector\_to\_bav : Convert a covector.resu file into a fgm bav file  
 rpc\_fgm\_cef\_to\_bav : Convert a fgm cef file into a fgm bav file  
 rpc\_vectime\_to\_covector : Convert RFF vectime files to covector format  
 rpc\_rff\_reduce\_time : reduce the time period of a RFF file (WF or VT class)

##### 4) image conversion

rpc\_ps\_to\_pdf : Convert PostScript into a PDF file  
 rpc\_ps\_to\_png : Convert PostScript into a PNG file

#### V) Roproc Orbit commands

rpc\_cresatpos\_rco : Creating list of Cluster positions  
 rpc\_listorbit\_rco : Creating list of Cluster orbit over 24 hours  
 rpc\_get\_param\_orbi\_clu : Computation of Cluster orbit parameters  
 rpc\_compute\_trajectory : Compute elliptical trajectory of a Earth satellite  
 rpc\_test\_keplerlib : Compute Earth orbit, to check routines  
 rpc\_co\_cov\_matrix : Compute covariance matrix for a given date  
 rpc\_co\_discont : Computation of discontinuity velocity from time delays

#### VI) Roproc Magnetic tools

rpc\_conjug : Computing the conjugate points by Tsyganenko model  
 rpc\_cotsyfield : Computation of Tsyganenko magnetic Field  
 rpc\_cofieldline : Computation of a field line model  
 rpc\_cofieldvector : Computing a field vector by Tsyganenko model

#### VII) Roproc Utilities

##### 1) STAFF files

rpc\_datafiles : Print the list of the available L1 files  
 rpc\_libloc : List of all blocs of a L1 file  
 rpc\_exit\_on\_error : used only with arg \$? \$0 into a shell  
 rpc\_testgain : Test the gain of the antennas  
 rpc\_test\_time : Test the time continuity in a L1 file  
 rpc\_give\_spin\_direction : Give rght asc. and decl. of spin axis for given date  
 rpc\_cal\_i\_CLUSTA : Continuous calibration of a L1 STAFF-SC RFF Vectime file

4

**2) Time utilities**

rpc\_caljul : compute Julian day for a given date  
 rpc\_current\_date : return current date and Julian sec.  
 rpc\_today : same, but pretty formatted  
 rpc\_nday\_of month : give number of day of a given month

**3) System utilities**

rps\_sys\_info : give system info (host, processor, operating system etc.)  
 rpc\_dir\_size : give size of a given directory  
 rpc\_dir\_size\_tree : same for all the subtree  
 rpc\_dir\_properties : give size & nb dir. & nb files in a given directory  
 rpc\_dir\_properties\_tree : same for all the subtree

**VIII) Roproc Visualization commands****1) wave visualization**

rpc\_visuwave : Visualization of cowave.resu waveforms  
 rpc\_visuspectro : Visualization of cospectro.resu spectrogram  
 rpc\_visuspectro\_4Bz : Visualization of 4 Bz Spectrogrammes  
 rpc\_visuspectra : Visualization of average spectrum from cospectro.resu  
 rpc\_visupower : Visualization of integrated power from cospectro.resu  
 rpc\_visupolar : Visualization of copolar.resu file  
  
 rpc\_visuwave4 : Run 4 times rpc\_visuwave for 4 spacecraft  
 rpc\_visuspectro4 : Run 4 times rpc\_visuspectro for 4 spacecraft  
 rpc\_visuspectra4 : Run 4 times rpc\_visuspectra for 4 spacecraft  
 rpc\_visupower4 : Run 4 times rpc\_visupower for 4 spacecraft  
 rpc\_visupolar4 : Run 4 times rpc\_visupolar for 4 spacecraft

**2) vector visualization**

rpc\_visuvector : Visualization of covector.resu file  
 rpc\_visuvector\_4sat : Visualization of 4 covector.resu files  
 rpc\_visuaddvecpos : Visualization of Cluster orbit over 3 planes  
 rpc\_visucocurldiv : Visualization of cocurldiv.resu file  
  
 rpc\_visuvector4 : Run 4 times rpc\_visuvector for 4 spacecraft  
 rpc\_visuaddvecpos4 : Run 4 times rpc\_visuaddvecpos for 4 spacecraft  
  
 rpc\_visuvector\_3D : Visualization 3D of a covector.resu file

**3) orbit visualization**

rpc\_visuorbit : Visualization of listorbit files  
 rpc\_visusatpos : Visualization of Cluster orbit over 3 planes +3D persp.

## IX) Some more about rpc commands

### 1) On line help

Usage and list of arguments can be given by using help option.

Example:

```
rpc_vector_to_mva help
```

### 2) rpc\_cowave\_xxx

Same sample rate is used for CLUSTER/STAFF-SC, CLUSTER/EFW, DSP/STAFF-SC. So, the nbp parameter of cowave procedure correspond to the same time duration for theses 3 experiments. Correspondence between nbp and time duration is given below. "\*" is a recommended value to produce spectrograms.

Nb. of pts	Duration in NBR (25 Hz)	Duration in HBR (450 Hz)
128	5.12 s *	0.2844 s
256	10.24 s *	0.5689 s
512	20.48 s *	1.138 s
1024	40.96 s *	2.275 s
2048	81.92 s or 1 m 22 s	4.551 s
4096	163.8 s or 2 m 44 s	9.102 s *
8192	327.7 s or 5 m 28 s	18.20 s *
16384	655.4 s or 10 m 55 s	36.40 s *
32768	1310.7 s or 21 m 51 s	72.82 s or 1 m 13 s
65536	2621.4 s or 43 m 41 s	145.6 s or 2 m 26 s

### 3) rpc\_mave\_to\_mfa

rpc\_wave\_to\_mfa\_rcs : use CLUSTER/FGM PPD data to define the MFA system.  
Filtering on PPD data is done, and STAFF data are processed.

rpc\_wave\_to\_mfa\_rcs4 : Run 4 times rpc\_wave\_to\_mfa\_rcs for 4 S/C

rpc\_mave\_to\_mfa\_rce : Same as above, but EFW data are processed \*\*\* **disused**  
on the X and Y components of the SR2 system;  
3 components in MFA are computed, with assumption  
of  $E//=0$  (no EZ MFA component).

### B) ROPROC SCRIPTS

Roproc scripts are shell scripts, running without arguments. Examples available in sh directories are following:

```

rps_polar_CLUSTER_ex1.sh  rps_polar_CLUSTER_ex2.sh
rps_CLUSTER_ex1.sh        rps_CLUSTER_ex2.sh
rps_polar_CLUSTER_1sat.sh rps_polar_CLUSTER_4sat.sh *recommended

rps_CLUSTERAFGM_ex1.sh    rps_CLUSTERAFGM_ex2.sh    rps_CLUSTERAFGM_ex3.sh

rps_CLUORB_ex.sh

rps_crefigdoc.sh

```

The user can copy these script files (from sh directory) into his directory, modify it, and create its own script files.

### C) ROPROC MACROS

Roproc macros are Roproc commands built from a sequence of several Roproc commands. This enables automation of certain processing by the use of a single command.

Examples available in sh directories are following.

Note: To avoid splash visu, strike: export R\_VISU\_SPLASH=no

```

rpm_spectro_4sat_CLUSTER : produce spectrograms 4 S/C for CLUSTER/STAFF
ex: rpm_spectro_4sat_CLUSTER 2001 09 23 09 20 00 40. 512 0 0.1 0. 0. 5 NBR

rpm_polar_4sat_CLUSTER : produce polar plots for 4 S/C for CLUSTER/STAFF
ex: rpm_polar_4sat_CLUSTER 2001 09 23 09 20 00 40. 512 0 0.1 0. 0. 5 NBR

rpm_visu_4sat_CLUSTER : Produce 4 S/C plots of CLUSTER/FGM data
ex: rpm_visu_4sat_CLUSTER 2001 01 26 11 30 00 60. SPIN

rpm_curl_div_CLUSTER : Produce plots of curlB & divB from CLUSTER/FGM data
ex: rpm_curl_div_CLUSTER 2001 01 26 11 30 00 60. SPIN

```

The user can copy these macros (from sh directory) into his directory, modify it, and create its own macro commands.

Any problem ? [patrick.robert@lpp.polytechnique.fr](mailto:patrick.robert@lpp.polytechnique.fr)

**Table 74:** *Liste thématique des Roproc (6 pages)*



### 9.12.4 Examples of arguments of Roproc commands

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%%      EXAMPLES OF ARGUMENTS OF ROPROC COMMANDS      %%
%%      (Alphabetic order)                            %%
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

rpc_addvecpos CLUSTER
rpc_addvecpos4 CLUSTER
rpc_alivetime 6000
rpc_caljul 2002 09 23
rpc_cocurldiv
rpc_cofieldline 1996 12 15 21 00 00 t87s 6 1 0 0 0 0 0 0 0 0 \
    gsm c RE -23.04 6.04 -1.4
rpc_cofieldvector 1996 12 15 21 00 00 t87s 6 1 0 0 0 0 0 0 0 0 \
    gsm c RE -23.04 6.04 -1.4
rpc_compute_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
rpc_conjug 1996 12 15 21 00 00 t87s 6 1 0 0 0 0 0 0 0 0 \
    gsm c RE -23.04 6.04 -1.4
rpc_copolar
rpc_copolar4
rpc_cospectro 1. 0. 0 0
rpc_cospectro4 0.5 0. 1 0
rpc_cotsyfield 1996 12 15 21 00 00 t89c 4 1 0 0 0 0 0 0 0 0 \
    gsm c -23.04 6.04 -1.4
rpc_cowave_rce $RCE_DATA/efw4_20020909_SC.flat \
    2002 09 09 4 41 0 0.5 4096 0.25 1 NBR
rpc_cowave_rce4 $RCE_DATA/efw4_20020909_SC.flat \
    2002 09 09 4 41 0 0.5 4096 0.25 1 NBR
rpc_cowave_rcs 3 2001 09 23 09 20 00 40.1 512 0. 0.1 0. 0. 5 NBR
rpc_cowave_rcs4 2001 09 23 09 20 00 40.1 512 0 0.1 0. 0. 5 NBR
rpc_cowave_rds 1 2004 02 19 19 10 00 30. 512 0. 0.2 1.5 0. 5 NBR
rpc_cresatpos_rco 2001 09 23 09 00 00 360.0 5.0 gse
rpc_datafiles
rpc_get_data_clufgm 3 2001 09 23 09 20 00 40.1 bav
rpc_get_data_clufgm4 2001 09 23 09 20 00 40.1 bav
rpc_get_param_orbi_clu 2001 09 23 12 00 00
rpc_libloc $RCS_DATA/2001_09/010923_sat4_sc.nlp CLUSTER 4 NBR
rpc_list
rpc_listorbit_rco 2001 09 23 15 gse l
rpc_menu
results are in standard output
rpc_merge_alivector
rpc_readvector
rpc_readwave
rpc_stat 20
rpc_test_gain CLUSTER 1
rpc_test_keplerlib
rpc_test_time 3 2001 09 23 0 100 CLUSTER NBR
rpc_vector_alitime 6000
rpc_vector_average covector_1.resu covector_2.resu \
    covector_3.resu covector_4.resu
rpc_vector_geo_to_gse
rpc_vector_geo_to_gse4
rpc_vector_gse_to_gsm
rpc_vector_gse_to_gsm4
rpc_vector_gse_to_sma
rpc_vector_gse_to_sma4
rpc_vector_gse_to_sr2 102.50 -63.90
rpc_vector_gse_to_sr24 102.50 -63.90
rpc_vector_minus_line -0.00833 10. 0. -22.5 0.01666 -8.
rpc_vector_minus_line4 -0.00833 10. 0. -22.5 0.01666 -8.

```

```

rpc_vector_minus_trend 100
rpc_vector_minus_trend4 100
rpc_vector_minus_vector vector_average.resu -Bmoy
rpc_vector_minus_vector4 bmoy.resu Bcm
rpc_vector_subsample 10
rpc_vector_subsample4 10
rpc_vector_timeshift 10.
rpc_vector_timeshift4 10.
rpc_vector_to_mat "MVA Bcm" \
    0.673415E+00 0.508084E+00 -0.536994E+00 \
    -0.470780E+00 -0.265316E+00 -0.841412E+00 \
    -0.569982E+00 0.819425E+00 0.605279E-01
rpc_vector_to_mat4 "MVA Bcm" \
    0.673415E+00 0.508084E+00 -0.536994E+00 \
    -0.470780E+00 -0.265316E+00 -0.841412E+00 \
    -0.569982E+00 0.819425E+00 0.605279E-01
rpc_vector_to_mfa
rpc_vector_to_mfa4
rpc_vector_to_model CLUSTER t87s 3 1 0 0 0 0 0 0 0 0
rpc_vector_to_model4 CLUSTER t87s 3 1 0 0 0 0 0 0 0 0
rpc_vector_to_mva
rpc_vector_to_mva4
rpc_vector_to_wave 512 0.25 25. 102.64 -63.88
rpc_vector_to_wave4 512 0.25 25. 102.64 -63.88
rpc_version
rpc_visuaddvecpos 2400. nT/RE 3
rpc_visuaddvecpos4 2400. nT/RE 3
rpc_visucocurldiv
rpc_visuorbit
rpc_visupolar -5.5
rpc_visupolar4 -5.5
rpc_visupower 0.5 0. 0. 0.
rpc_visupower4 0.5 0. 0. 0.
rpc_visusatpos
rpc_visuspectra 0. 0. 0. 0.
rpc_visuspectra4 0. 0. 0. 0.
rpc_visuspectro 0. 0. 0. 0.
rpc_visuspectro4 0. 0. 0. 0.
rpc_visuspectro_4Bz
rpc_visuvector 0. 0. 0. 0. 0. 0. 0. 0.
rpc_visuvector4 0. 0. 0. 0. 0. 0. 0. 0.
rpc_visuvector_4sat 0. 0. 0. 0. 0. 0. 0. 0.
rpc_visuwave 0 0. 0. 0. 0. 0. 0.
rpc_visuwave4 0 0. 0. 0. 0. 0. 0.
rpc_wave_filter 0.2 5.
rpc_wave_filter4 0.2 5.
rpc_wave_to_gse
rpc_wave_to_gse4
rpc_wave_to_gsm
rpc_wave_to_gsm4
rpc_wave_to_mat TPN_12.0 0.71279 -0.46706 -0.52324 \
    0.00000 0.74602 -0.66592 \
    0.70138 0.47466 0.53176
rpc_wave_to_mat4 TPN_12.0 0.71279 -0.46706 -0.52324 \
    0.00000 0.74602 -0.66592 \
    0.70138 0.47466 0.53176
rpc_wave_to_mfa_rce
rpc_wave_to_mfa_rce4
rpc_wave_to_mfa_rcs
rpc_wave_to_mfa_rcs4
rpc_wave_to_mfa_rds
rpc_wave_to_mva
rpc_wave_to_mva4
rpc_wave_to_vector
rpc_wave_to_vector4

```

---

END OF ROPROC SHORT DOCUMENTATION

---

**Table 75:**      *Examples of arguments of Roproc commands (2 pages)*





## 10. LISTE DES ACRONYMES

STAFF : Spatio Temporal Analyse of Field Fluctuations

FGM : Flux Gate Magnetometer

HBR : High Bit Rate

NBR : Normal Bit Rate

RFF : Roproc File Format

RCL : Roproc Command Language

WFL1 : RFF Waveform file Level 1 (Uncalibrated)

VTL1 : RFF Vectime file Level 1 (Uncalibrated)

VTL2 : RFF Vectime file Level 2 (Calibrated)

SPL2 : RFF Spectrogram file Level 2 (Calibrated)

CAA : Cluster Active Archive

CSA : Cluster Science Archive

CEF : Cluster Exchange Format

DWF : Decommutated WaveForms

CWF : Calibrated WaveForms

CS : Complex (Calibrated) Spectra

FFT : Fast Fourier Transform



## 11. LISTE DES TABLES ET FIGURES

### 11.1. LISTE DES FIGURES

Figure 1 :	Chaîne de production des WFL1.rff. Les CD sont préalablement concaténés, ainsi que les SATT files et les HK satellites associés. ....	25
Figure 2 :	Chaîne de production des VTL1.rff. Les fichiers WFL1.rff du jour courant ET du jour précédent sont requis. ....	26
Figure 3 :	Chaîne de production des SPL2.rff. Les fichiers VTL1.rff du jour courant ET du jour suivant sont requis. ....	27
Figure 4 :	Chaîne de production des VTL2.rff. Les fichiers WFL1.rff du jour courant, du jour précédent ET du jour d'après sont requis. ....	28
Figure 5 :	Chaîne de production des DWF.cdf. Celle-ci se résume à un simple filtre de changement de format, ne concernant que la partie méta data. ....	29
Figure 6 :	Chaîne de production des CS.cdf. Celle-ci se résume à un simple filtre de changement de format, ne concernant que la partie méta data. ....	29
Figure 7 :	Chaîne de production des CWF.cdf. Celle-ci se résume à un simple filtre de changement de format, ne concernant que la partie méta data. ....	30
Figure 8 :	Chaîne de production des images à mettre sur le web. Celles-ci servent aussi à la validation des données et des chaînes de traitement. ....	31
Figure 9 :	Chaîne de production des spectrogrammes pour la validation des CWF. ....	32
Figure 10 :	Schéma général des chaînes de production STAFF-SC ....	33
Figure 11 :	Exemple de spectrogramme des données FGM produit à partir des commandes RCL ....	87
Figure 12 :	Signal utile superposé à la sinusoïde de spin pour un search-coil en rotation ....	124
Figure 13 :	apodisation en trapèze du signal pour la calibration d'un spectre. ....	127
Figure 14 :	Principe du "detrend" ....	133
Figure 15 :	représentation des fenêtres de pondérations disponibles ....	137
Figure 16 :	Définition du « spin phase » dans le repère SR2 (extraite du DDID) ....	146
Figure 17 :	Position des antennes de STAFF dans le repère « Body Build » lié au satellite. ....	149
Figure 18 :	Definition of SR system ....	150
Figure 19 :	Definition of SR2 system (Despun) ....	151
Figure 20 :	Definition of GEI system: ....	153
Figure 21 :	Definition of GSE system ....	153
Figure 22 :	Definition of GSM system ....	154
Figure 23 :	Definition of MFA system ....	154
Figure 24 :	Mesure du champ perpendiculaire et de la phase dans le plan de spin ....	155
Figure 25 :	Définition des repères pour un capteur en rotation ....	157
Figure 26 :	Projection du vecteur J sur 3 vecteurs (P, Q, R) non orthogonaux ....	161
Figure 27 :	Différence entre les composantes mathématique de B et les mesures ....	162
Figure 28 :	Ordre des raies du spectre en sortie de FFT ....	168
Figure 29 :	Spectre d'une sinusoïde pure dont la fréquence est un multiple de $\Delta f$ ....	169
Figure 30 :	Spectre d'une sinusoïde pure dont la fréquence n'est pas un multiple de $\Delta f$ ....	170
Figure 31 :	Transformée de Fourier du signal complexe $S_{xy} = x + jy$ associé à une onde plane monochromatique pour 3 exemples particuliers (Robert, 1971, [20] ). ....	171
Figure 32 :	Passage du repère en rotation à un repère fixe par les composantes circulaires gauche et droite (Robert, 1971, [20] ). ....	172
Figure 33 :	Exemple de visu_spectro à partir des VTL2 ....	185

Figure 34:	Exemple de visu_spectro à partir des VTL1 .....	186
Figure 35 :	Exemple de visu_spectro_4Bz utilisée comme « Quick looks » .....	187
Figure 36:	Exemple de visu_spectra à partir des VTL1 .....	188
Figure 37:	Exemple de visu_vectime sur un VTL1 .....	189
Figure 38:	Exemple de visu_vectime sur un VTL2 en ISR2 .....	190
Figure 39:	Exemple de visu_vectime sur un VTL2 en GSE .....	191
Figure 40:	Exemple de spectrogramme en repère MFA et en composantes circulaires, montrant la polarisation circulaire droite d'un whisler dans le plan perpendiculaire au champ magnétique DC local.....	192
Figure 41:	Spectre moyen du spectrogramme précédent, montrant la nette prépondérance du mode droit (vert).....	193
Figure 42:	Exemple de visualisation du résultat du programme de calcul de polarisation.....	194
Figure 43:	Exemple d'onde pseudo monochromatique linéaire dans le plan perpendiculaire au champ magnétique DC local .....	195
Figure 44:	Autre exemple de résultat du calcul de polarisation.....	196
Figure 45:	Exemple de visu_CLUPOS sur 3 heures en GSE .....	197
Figure 46:	Exemple de visu_CLUPOS sur 3 heures en GSM.....	198
Figure 47:	Exemple de visu_CLUPOS sur une orbite complète en GSE .....	199
Figure 48:	Exemple de visu_CLUPOS sur une orbite complète en GSM.....	200
Figure 49:	Exemple de visu_CLUGEOM, partie « positions et vitesses » .....	201
Figure 50:	Exemple de visu_CLUGEOM, partie « géometrie du tétraèdre ».....	202

## 11.2. LISTE DES TABLES

Table 1 :	Exemple de script RCL pour calculer et visualiser un spectrogramme de données calibrées .....	16
Table 2:	Exemple de passage d'un format WF (WaveForm) à un format VT (VecTime) par l'application RCF_waveform_to_vectime .....	18
Table 3 :	Nomenclature des noms de fichiers RFF .....	22
Table 4:	Nomenclature des noms de fichiers RFF .....	22
Table 5 :	Arborescence des fichiers RFFet PNG.....	23
Table 6:	Arborescence des fichiers CEF.....	23
Table 7:	Arborescence des fichiers CEF.....	24
Table 8:	Cluster WEC (DWP) exact sampling frequency derived from the DWP clock .....	36
Table 9 :	Valeurs des paramètres de la commande RCL_product_VTL2_oneday .....	36
Table 10:	Valeurs des paramètres de la commande RCL_product_SPL2_oneday.....	37
Table 11:	Valeurs des paramètres de la commande RCL_product_visu_SPL2_oneday .....	37
Table 12:	Taille et nombre de fichiers de l'arborescence des fichiers WFL1 « zippés » .....	40
Table 13:	Taille et nombre de fichiers de l'arborescence des fichiers VTL1 .....	41
Table 14:	Taille et nombre de fichiers de l'arborescence des fichiers VTL2 « zippés » .....	42
Table 15:	Taille et nombre de fichiers de l'arborescence des fichiers SPL2 « zippés » .....	42
Table 16:	Taille et nombre de fichiers de l'arborescence des fichiers PNG.....	43
Table 17:	Résumé des volumes occupés par les bases de données locales (Go).....	44
Table 18:	Exemple de résultat de la commande RCL_list .....	45
Table 19:	Exemple de résultat de la commande RCL_list_block_WF.....	48
Table 20:	Exemple de résultat de la commande RCL_give_CLUORB_period .....	50
Table 21:	Exemple de résultat de la commande RCL_give_spin_dir .....	54
Table 22:	Exemple de résultat de la commande RCL_system_info .....	64
Table 23:	Exemple de résultat de la commande RCL_dir_size_tree .....	65
Table 24:	Exemple de résultat de la commande RCL_dir_size_tree Mo.....	65
Table 25:	Exemple de résultat de la commande RCL_dir_size_pretty_tree .....	66
Table 26:	Exemples de résultats de la commande RCL_dir_properties_tree .....	67
Table 27:	Exemple de résultat de la commande RCL_dir_properties_pretty_tree .....	68
Table 28:	Exemple de résultat de la commande RCL_list .....	70
Table 29:	Contenu du script «PR_make_staff_spectro_from_VTL2 » .....	72
Table 30:	Contenu du script «PR_make_staff_spectro_from_VTL1 » .....	75



Table 31:	Contenu du script «PR_visu_staff_VTL1 » .....	77
Table 32:	Contenu du script «PR_visu_staff_VTL2 » .....	78
Table 33:	Contenu du script « PR_compute_staff_polarisation_from_VTL1 » .....	82
Table 34:	Exemple de script de traitement de masse pour l'étude de la polarisation .....	83
Table 35:	Contenu du script « PR_compute_staff_polarisation_from_VTL2 » .....	85
Table 36:	Script permettant de calculer et visualiser un spectrogramme de FGM .....	86
Table 37:	Exemple de fichier RCL_config.bash.....	90
Table 38:	Options de compilation du fichier Makefile.....	92
Table 39 :	Options « make clean » du fichier Makefile.....	92
Table 40:	Options « make mrproper » du fichier Makefile .....	93
Table 41:	Compilation desIDL.pro dans le fichier Make_IDL_sav.bash .....	93
Table 42:	Arborescence des répertoires créés par le test de production sur un mois.....	95
Table 43:	Arborescence des répertoires du projet RCL .....	97
Table 44:	Fichiers à la racine du projet RCL .....	97
Table 45:	Liste des commandes RCL utilisant les exécutable du répertoire bin.....	100
Table 46:	Liste des commandes RCL utilisant des SAV d'IDL du répertoire bin .....	101
Table 47:	Commandes RCL utilisant d'autres commandes RCL.....	103
Table 48:	Bibliothèques utilisées par chaque programme exécutable F90.....	109
Table 49 :	Récapitulation des bibliothèque F90 utilisées .....	110
Table 50 :	Liste des sousroutines de traitement du signal de la bibliothèque lib_déconvo.f90.....	110
Table 51 :	Liste des sousroutines de la bibliothèque lib_gainant.f90 .....	110
Table 52 :	Liste des sousroutines de la bibliothèque lib_rw_rff.f90 .....	111
Table 53 :	Liste des sousroutines de la bibliothèque lib_rw_cef.....	112
Table 54 :	Liste des sousroutines de la bibliothèque lib_time.f90 .....	112
Table 55 :	Liste des sousroutines de la bibliothèque lib_CLUORB.f90 .....	113
Table 56 :	Liste des sousroutines de la bibliothèquelib_CLU_tools.f90 .....	113
Table 57 :	Liste des sousroutines de la bibliothèquelib_utility.f90 .....	114
Table 58 :	Liste des sousroutines de la bibliothèque Rocotlib_2p0.f90.....	115
Table 59:	Procédures IDL de lecture de fichiers.....	116
Table 60 :	Procédures IDL de visualisation de fichiers de données .....	117
Table 61:	Exemple de fabrication d'un .sav d'IDL .....	118
Table 62:	Liste des procédures IDL nécessaire à la fabrication des .SAV.....	119
Table 63 :	Procédures et fonctions disponibles dans la bibliothèque roploplib.pro.....	121
Table 64 :	STAFF status word.....	143
Table 65 :	Signification de chacun des 14 caractères du statut des VTL1 ou des DWF .....	144
Table 66 :	Erreur maximum du à la compression.....	145
Table 67:	Exemple de WFL1.rff (3 pages).....	175
Table 68:	Exemple de VTL1.rff (3 pages).....	178
Table 69 :	Exemple de VTL2.rff (3 pages).....	181
Table 70:	Exemple de SPL2.rff (3 pages).....	184
Table 71:	Liste fonctionnelle des commandes RCL (3 pages) .....	205
Table 72:	Liste alphabétique des commandes RCL avec arguments (3 pages).....	208
Table 73 :	Introduction aux Roproc .....	210
Table 74:	Liste thématique des Roproc (6 pages).....	216
Table 75:	Exemples of arguments of Roproc commands (2 pages).....	218



## 12. BIBLIOGRAPHIE

- [1] Patrick Robert, “Les procédures Roproc pour le traitement des données de CLUSTER /STAFF-SC, FGM et ORBITE”, CETP, V 2.0, Septembre 2003.  
[ftp://ftp.lpp.polytechnique.fr/robert/keep/ROPROC/Roproc\\_V2p0.pdf](ftp://ftp.lpp.polytechnique.fr/robert/keep/ROPROC/Roproc_V2p0.pdf)
- [2] C. Perry, T. Eriksson, P. Escoubet, S. Esson, H. Laakso, S. McCaffrey, T. Sanderson, H. Bowen, “**The ESA Cluster Active Archive**”, in *Proceedings of the Cluster and Double Star Symposium 5th Anniversary of Cluster in Space, ESTEC*, Noordwijk, 2005.  
<http://caa.estec.esa.int/caa/home.xml>
- [3] 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://caa.estec.esa.int/documents/DS-QMW-TN-0010.pdf>
- [4] 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](ftp://ftp.lpp.polytechnique.fr/robert/keep/ROPROC/The_Roproc_File_Format_20110216.pdf)
- [5] Patrick ROBERT, “**Roproc Short Documentation**”, Version 4.3 CNRS/CETP-LPP, 2000-2010, Last update January 11, 2011  
[ftp://ftp.lpp.polytechnique.fr/robert/keep/ROPROC/Short\\_doc\\_Roproc\\_V4p3\\_c.pdf](ftp://ftp.lpp.polytechnique.fr/robert/keep/ROPROC/Short_doc_Roproc_V4p3_c.pdf)
- [6] Nicole Cornilleau Wehrlin : Chauveau, P.: Louis, S.: Meyer, A.: Nappa, J. M.: Perraut S.: Rezeau, L.: Robert, P.: Roux, A. and C. De Villerdary: “**The cluster spatiotemporal analysis of field fluctuations (STAFF) experiment**”. *Space Science Review* 79: 107- 136, 1997.
- [7] N. Cornilleau-Wehrlin, C. Burlaud and the STAFF team, “**User Guide to the STAFF measurements in the Cluster Active Archive (CAA)**”, ESA CAA-EST-UG-002, April 26, 2011.  
[http://caa.estec.esa.int/documents/UG/CAA\\_EST\\_UG\\_STA\\_v30.pdf](http://caa.estec.esa.int/documents/UG/CAA_EST_UG_STA_v30.pdf)
- [8] Patrick Robert, “**ROCOTLIB: a Coordinate Transformation Library for SolarTerrestrial studies**”, version 1.8, update of CETP Internal report n° RI-CETP/01/2003, Novembre 2003.  
[ftp://ftp.lpp.polytechnique.fr/robert/keep/ROCOTLIB/ROCOTLIB\\_v1.8u\\_TOC.pdf](ftp://ftp.lpp.polytechnique.fr/robert/keep/ROCOTLIB/ROCOTLIB_v1.8u_TOC.pdf)
- [9] xxx, “Description de la chaîne de décommutation des données STAFF-SC”, version ..., année...
- [10] C.C. Harvey, A.J. Allen, F. D’eriot, C. Huc, M. Nonon-Latapie, C.H. Perry, S.J. Schwartz, T. Eriksson and S. McCaffrey, “**Cluster Metadata Dictionary**”, CAA Metadata Working Group, March 5, 2008.  
<http://caa.estec.esa.int/documents/DataDic.pdf>

- [11] Patrick ROBERT, C. Burlaud & STAFF Team, “**STAFF/SC cross-calibration activities**”, 10th CAA Cross-Calibration meeting, Paris, 2-4 November 2009  
<ftp://ftp.rssd.esa.int/pub/Cluster/MoM/CAA/CrossCalibration/CrossCal10/CrossCal10-Annex21-STAFF-SC.ppt>
- [12] Patrick ROBERT, “**STAFF CAA products & Cross-Calibration activities**”, 13th CAA Cross-Calibration meeting, Uppsala, 13-15 Apr 2011
- [13] 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 Cluster and Double Star, Research Article, gi-2013-15  
<http://www.geosci-instrum-method-data-syst.net/3/153/2014/>
- [14] 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).
- [15] Chanteur, G. and Mottez, F., “**Geometrical tools for Cluster data analysis**”, in Proc. International Conf. “Spatio-Temporal Analysis for Resolving plasma Turbulence (START)”, Aussois, 31 Jan.–5 Feb. 1993 , ESA WPP-047, pp. 341–344, European Space Agency, Paris, France, 1993.
- [16] Patrick Robert and C. de Villedary , “**Transformation of a STAFF waveform into a Magnetic Field Aligned coordinate system**”, Rapport interne CNRS-UVSQ/CETP n° RI-CETP/6/2000, Octobre 2000.
- [17] ESA/ESOC, “**Data Delivery Interface Document (DDID)**”, Directorate of Technical and Operational Support, European Space Operations Centre, CLUSTER Data Disposition System, Issue 3, Document Reference: CL-ESC-ID-2001, 19 May 2000.
- [18] Patrick Robert, “**A, easy method to compute plane waves polarisation from a three components waveform**”, Document de travail CRPE, 1980. Révision et application à CLUSTER, Aout 2008.
- [19] Patrick Robert, C. Burlaud and M. Maksimovic, “**Calibration Report of the STAFF Measurements in the Cluster Active Archive (CAA)**”, Version 3.0, Updated by N. Cornilleau-Wehrin, P. Robert and R. Piberne, doc. CAA/ESA Num. CAA-STA-CR-002, Issue: 3.0, 2012-05-16.  
[http://caa.estec.esa.int/documents/CR/CAA\\_EST\\_CR\\_STA\\_v30.pdf](http://caa.estec.esa.int/documents/CR/CAA_EST_CR_STA_v30.pdf)
- [20] Robert, P.: « **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 rencontrés et solutions pratiques** », Note Technique CRPE/ETE/71, May, available at:  
[ftp://ftp.lpp.polytechnique.fr/robert/keep/Biblio\\_et\\_CV/Working\\_documents/1979\\_Robert\\_NTCRPE71\\_Intensite\\_et\\_Polarisation\\_des\\_ondes\\_UBF.pdf](ftp://ftp.lpp.polytechnique.fr/robert/keep/Biblio_et_CV/Working_documents/1979_Robert_NTCRPE71_Intensite_et_Polarisation_des_ondes_UBF.pdf)

## - NOTES -

**— NOTES —**