CryoSat Level1b Expert Support

CryoSat Quaternion Products: Algorithm description

Document REF	C2-TN-ARS-GS-5220
Internal REF	ARE-016512
Issue	1.3
Date	19 May 2022
Pages	12
Recipients	S. Badessi – ESA V. Torroni – Serco for ESA J. Bouffard – ESA A. Di Bella – Serco for ESA T. Parrinello – ESA

	Name	Signature
Prepared by	M. Scagliola – Aresys	
Checked by	D. Giudici – Aresys	
Approved by	D. Giudici – Aresys	

Change Records

Issue	Date	Description	Author
1.0	23/03/2020	First issue	M. Scagliola
1.1	27/03/2020	Updated following inputs received from CFI Support	M. Scagliola
1.2	02/04/2020	Section 2.3 updated following inputs received from CFI Support	M. Scagliola
1.3	13/05/2022	Document updated for release	L. Recchia

Summary

List of Figures

1. Document Overview

1.1. Scope

This document describes the algorithm to compute the attitude quaternions in the Geocentric Mean of 2000 Inertial Coordinate Frame for CryoSat starting from the mispointing angles in the EOCFI Satellite Nominal Attitude Frame.

1.2. Acronyms

Aresys	Advanced Remote Sensing Systems
EOCFI	Earth Observation Customer Furnished Item
ESOC	European Space Operation Centre
AOCS	Attitude and Orbit Control System
XML	eXtensible Markup Language

1.3. Reference Documents

[RD1] CryoSat-2 Quaternion products file format specifications, C2-TN-ARS-GS-5231, Issue 1.0, 19/05/2022
[RD2] IPF_STRF_PROC, Detailed Processing Model, CS-DD-ARS-GS-5003, Issue 2.0, 22 June 2018
[RD3] Earth Explorer Mission CFI Software, Release Notes Version 3.7.6, EOCFI-DMS-SRN-004, 28 May 2019
[RD4] Earth Explorer Mission CFI Software EXPLORER_DATA_HANDLING SOFTWARE USER MANUAL, EE-MA-DMS-GS-0007, issue 3.7.6
[RD5] Earth Explorer Mission CFI Software, Explorer_lib Software User Manual, EE-MA-DMS-GS-0003, Issue 3.7.6
[RD6] Earth Explorer Mission CFI Software, Explorer_pointing Software User Manual, EE-MA-DMS-GS-0005, Issue 3.7.6
[RD7] Earth Explorer Mission CFI Software, Conventions Document, EE-MA-DMS-GS-0001, Issue 3.7.6
[RD8] CryoSat-2 Product Handbook, BaselineD 1.0, C2-LI-ACS-ESL-5319, 3/4/2018

2. Attitude quaternions

2.1. Objective

A study was requested to provide users a product defined as in [RD1] containing the attitude quaternions in the Geocentric Mean of 2000 Inertial Coordinate Frame for CryoSat. According to [RD7], in case of theoretical local normal pointing the EOCFI Satellite Nominal Attitude Frame is defined as in the following figure, while the EOCFI Satellite Attitude frame is rotated w.r.t. the nominal one, having the –z axis pointed towards the antenna boresight.

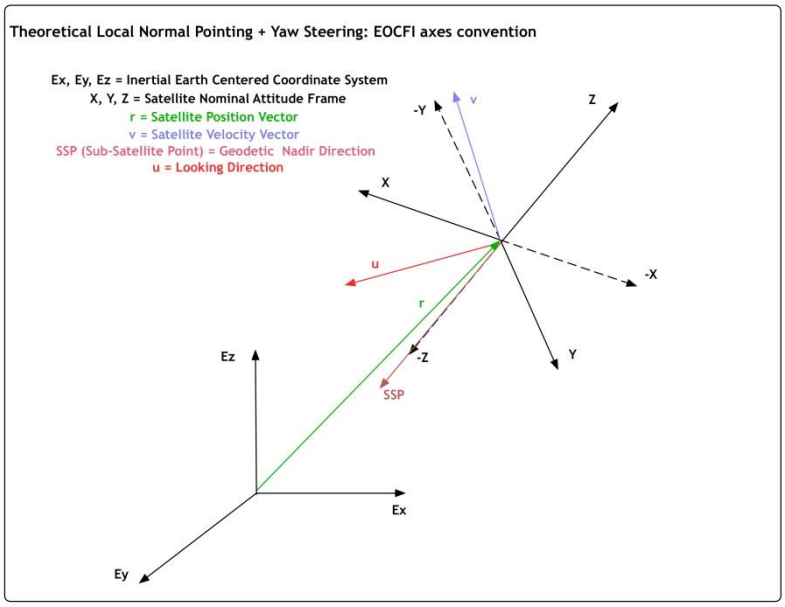

Fig.1 EOCFI Satellite Nominal Attitude Frame.

In case of CryoSat, the satellite fixed frame [RD8] is defined as in the following

1. satellite fixed frame is centred at the centre of mass of the platform;
2. x_S is directed from the from the reference frame centre of origin in the direction of the real antenna boresight;
3. z_S is directed from the origin and parallel with the direction vector adjoining the rx only antenna reference point to the tx only antenna reference point and orthogonal to x_S;
4. y_S is determined by the cross product $y_S = x_S \times z_S$.

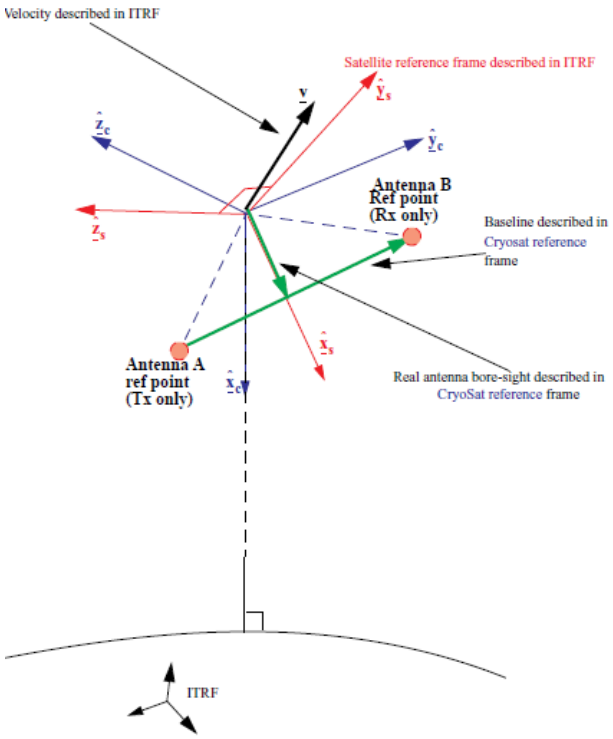

Fig.2 CryoSat Satellite Reference Frame.

Thus, neglecting at this moment the different conventions, the CryoSat Satellite Reference Frame corresponds to the EOCFI Satellite Attitude Frame and the mispointing Euler angles are defined as the rotation angles between the EOCFI Satellite Attitude Frame and the EOCFI Satellite Nominal Attitude Frame.

It is worth recalling here that in the CryoSat science products it is included the information about the platform mispointing in terms of Euler angles (i.e. yaw/pitch/roll) according to the CryoSat conventions, that can be mapped to the EOCFI conventions as in the following table.

	EOCFI conventions	CryoSat conventions	
Pitch	Positive pitch: Nose up	Positive pitch: Nose down	Opposite sign
Yaw	Positive yaw: Nose left	Positive yaw: Nose right	Opposite sign
Roll	Positive roll: Right antenna down	Positive roll: Right antenna down	Same sign

In the CryoSat ground processing the mispointing angles are compensated for known biases in roll and pitch as function of the Star Tracker that is used to compute the mispointing for a given timestamp. As a consequence, aiming at providing users quaternions containing the same spacecraft orientation that is provided as mispointing angles in the science products, it was proposed in [RD1] to compute the quaternions starting from the already compensated mispointing angles. The adopted approach consists in computing the quaternions starting from the mispointing Euler angles defined according to the EOCFI conventions and describing to users the difference between the EOCFI convention and the CryoSat convention.

2.2. CryoSat Star Tracker Processor

The Star Tracker Processing [RD2] currently included in the CryoSat BaselineE processing suite exploits v3.7.6 Earth Explorer Mission CFI [RD3]. The Star Tracker processor is in charge of computing the mispointing angles exploiting the library function of the EECFI software starting from the following inputs

- Orbit file, preliminary orbit file (ORB_DOP) in operations
- Star Tracker Level0 products from any of the three Star Tracker on-board of CryoSat platform
- Housekeeping Telemetry file provided by ESOC, containing the Star Tracker priority list according to the AOCS
- Star Tracker Database file, containing the EECFI configuration, e.g Antenna Bench To Star Tracker rotation angles
- Processor Configuration file, containing the processing configuration, e.g. the posting rate of the mispointing angles in the output product

The output of the Star Tracker processor is in Earth Explorer Format XML file according to the definition given in [RD3], so that the file is to be provided to the EECFI library functions.

The functional blocks of the CryoSat Star Tracker processor are depicted in the following figure and detailed in the following steps

- The mispointing angles in the EOCFI Satellite Nominal Attitude Frame are computed for each of the available Star Tracker Level0 file exploiting the EECFI library functions
- The mispointing angle biases are compensated for each Star Tracker
- For each time sample, one mispointing angle set is selected according to the priority list provided by AOCS
- An optional moving average is applied to mispointing angles to smooth the data
- The mispointing angles are written in the output Earth Explorer Format XML file

It is worth underlining here that the mispointing angles in the output Earth Explorer Format XML file are compliant to the EOCFI conventions, that have been described in the previous Section.

To generate the attitude quaternions output, a new module has been added to the CryoSat Star Tracker processor that is in charge of:

- computing the attitude quaternions in the Geocentric Mean of 2000 Inertial Coordinate Frame starting from the mispointing angle in the EOCFI Satellite Nominal Attitude Frame
- writing the attitude quaternions in an Earth Explorer Format XML file

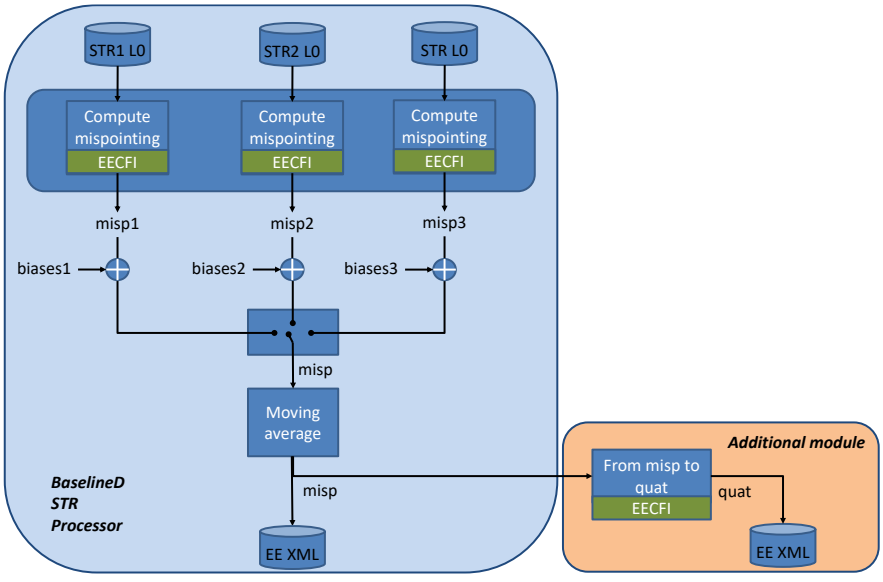

Fig.3 CryoSat Star Tracker Processor: functional blocks.

2.3. From mispointing to attitude quaternions

Throughout this section the processing steps to compute the attitude quaternions in the Geocentric Mean of 2000 Inertial Coordinate Frame starting from the mispointing angle in the EOCFI Satellite Nominal Attitude Frame are detailed. The EECFI library functions refers to [RD5] and [RD6].

1. Get the vectors of the Satellite Attitude Frame (*xs_final_ut, ys_final_out, zs_final_out*) starting from the mispointing angles (*ad_Output_Angles*)

```
xs_initial[0]=1.0;xs_initial[1]=0.0;xs_initial[2]=0.0;
ys_initial[0]=0.0;ys_initial[1]=1.0;ys_initial[2]=0.0;
zs_initial[0]=0.0;zs_initial[1]=0.0;zs_initial[2]=1.0;
double xs_final_out[3], ys_final_out[3], zs_final_out[3];
long int aj_grv_ierr[XL_NUM_ERR_GET_ROTATED_VECTORS];

   j_status = xl_get_rotated_vectors( xs_initial,
                                      ys_initial,
                                      zs_initial,
                                      ad_Output_Angles,
                                      xs_final_out,
                                      ys_final_out,
                                      zs_final_out,
                                      aj_grv_ierr );
```

2. Get the Satellite Attitude Frame matrix (*actual_frame*) from the vectors of the Satellite Attitude Frame (*xs_final_ut, ys_final_out, zs_final_out*)

```
double actual_frame[3][3];
for (j=0;j<3;j++)
{
      if( j == 0 ){
                        actual_frame[0][j] = xs_final_out[0];
                        actual_frame[1][j] = ys_final_out[0];
                        actual_frame[2][j] = zs_final_out[0];
            }
       else if( j == 1 ){
                        actual_frame[0][j] = xs_final_out[1];
                        actual_frame[1][j] = ys_final_out[1];
                        actual_frame[2][j] = zs_final_out[1];
            }
       else if( j == 2){
                        actual_frame[0][j] = xs_final_out[2];
                        actual_frame[1][j] = ys_final_out[2];
                        actual_frame[2][j] = zs_final_out[2];
            }
}
```

3. Compute the rotation matrix (*mat_sat_j2000*) between Geocentric mean of 2000 and the Satellite Nominal Attitude Frame

```
cf_mode = XP_MODE_FLAG_DIRECTION;
frame_flag_input = XP_FRAME_FLAG_EXT;
frame_flag_output = XP_FRAME_FLAG_SAT;
frame_id_input = XL_GM2000;
frame_id_output = XP_SAT_NOMINAL_ATT;
deriv = XP_NO_DER;
long int poi_gm_err[XP_NUM_ERR_CHANGE_FRAME];
j_time_ref_tai = XL_TIME_TAI;

vec_rate_input[0]       =   0.0;
vec_rate_input[1]       =   0.0;
vec_rate_input[2]       =   0.0;
vec_rate_rate_input[0]  =   0.0;
vec_rate_rate_input[1]  =   0.0;
vec_rate_rate_input[2]  =   0.0;
vec_output[0]           =   0.0;
vec_output[1]           =   0.0;
vec_output[2]           =   0.0;
double mat_sat_j2000[3][3];

for (j=0;j<3;j++)
{
      vec_input[j]=1.0;
      j_status = xp_change_frame (&j_sat_id,ptGetTimeID(),
                                  ptGetSatNomTransID(),ptGetSatTransID(),
                                  &instr_trans_id,
                                  &cf_mode,
                                  &frame_flag_input,&frame_id_input,
                                  &frame_flag_output,&frame_id_output,
                                  &j_time_ref_tai,&d_proc_time,
                                  ad_pos,ad_vel,ad_acc,&deriv,
                                  vec_input,vec_rate_input,
                                  vec_rate_rate_input,
                                  vec_output,vec_rate_output,
                                  vec_rate_rate_output,
                                  poi_gm_err);
      mat_sat_j2000[0][j] = vec_output[0];
      mat_sat_j2000[1][j] = vec_output[1];
      mat_sat_j2000[2][j] = vec_output[2];
      vec_input[j]=0.0;
}
```

4. Compute the product matrix (*actual_frame_j2000*) to obtain transformation between the rotation matrix (*mat_sat_j2000*) and the Satellite Attitude Frame (*actual_frame*)

```
double actual_frame_j2000[3][3];
double sum = 0.0;

for (i=0;i<3;i++){
        for (j=0;j<3;j++){
                sum= 0.0;
                for (k=0;k<3;k++){
                        sum +=actual_frame[i][k]*mat_sat_j2000[k][j];
                        actual_frame_j2000[i][j] = sum;
                }
        }
}
```

5. Get the vectors (*ux_vec, uy_vec, uz_vec*) of the Satellite Attitude Frame in Geocentric mean of 2000 from the products matrix (*actual_frame_j2000*)

```
double ux_vec[3], uy_vec[3], uz_vec[3];
for (j=0;j<3;j++){
        if( j == 0 ){
                ux_vec[0]=actual_frame_j2000[j][0] ;
                uy_vec[0]=actual_frame_j2000[j][1] ;
                uz_vec[0]=actual_frame_j2000[j][2] ;
         }
        else if( j == 1 ){
                ux_vec[1]=actual_frame_j2000[j][0] ;
                uy_vec[1]=actual_frame_j2000[j][1] ;
                uz_vec[1]=actual_frame_j2000[j][2] ;
        }
        else if( j == 2 ){
                ux_vec[2]=actual_frame_j2000[j][0] ;
                uy_vec[2]=actual_frame_j2000[j][1] ;
                uz_vec[2]=actual_frame_j2000[j][2] ;
        }
}
```

6. Get the quaternions in Geocentric mean of 2000 (*quaternions*) from the vectors (*ux_vec, uy_vec, uz_vec*) of the satellite fixed frame in Geocentric mean of 2000

```
double quaternions[4];
long int ierr_v2q[XL_NUM_ERR_VEC_TO_QUATERNION];
j_status =xl_vectors_to_quaternions(ux_vec, uy_vec, uz_vec, quaternions,ierr_v2q);
```