Algorithm Theoretical Basis Document for GRACE Level-1B Data Processing V1.2

Sien-Chong Wu

Gerhard Kruizinga

Willy Bertiger

May 9, 2006

Jet Propulsion Laboratory

California Institute of Technology

GRACE 327-741

(JPL D-27672)

Table of Contents

Change Log

Revision	Date	Pages	Description
1.1	Feb. 2, 2004	All	Initial release
1.2	May 9, 2006	24 6, 19, 38	Correct algorithm formulae for GNI_1A Add blank page message

1 Introduction

The Level 1B data processing software system forms the core for the editing, correction and compression of GRACE Level 1A data, and the creation of Level 1B data files. Raw data (Level 0) collected on-board the GRACE spacecraft are sent to the ground and converted into Level 1A by reformatting and calibration of the Level-0. This process is non-destructive, which means that all level-0 is retained and at their original rates. The level 1A software in addition to reformatting collects data into 24-hour and 30-hour data files centered on noon of each day. The Level 1A data includes:

1) Quaternions from two star cameras on the two spacecraft (4 sets of quaternions), for spacecraft attitude determination.

2) GPS data from each spacecraft, on-board navigation solution and dual-frequency GPS data

3) Accelerometer data from each spacecraft (angular and linear accelerations)

4) KBR (K-band ranging) measurements from each spacecraft

5) House keeping data of the spacecraft and science instruments

The Level 1B process converts the Level 1A KBR data into dual-one-way range, range-rate, and range-acceleration data, while editing and time-tagging the Level 1A SCA quaternions, GPS, accelerometer and house keeping data.

This document provides a description of the Level-1B process functions and algorithms of the following main executable modules:

`KBR_debreak` — detecting and flagging KBR K- and Ka-band phase breaks, editing anomalous data and applying timetag correction due to missed interrupts. A missed interrupt refers to the CPU missing a hardware interrupt causing a time tag error in one of the 4 KBR phase measurement streams.

`KBR_order` —filling data gaps and applying timetag correction(also referred to as re-sampling) for KBR K- and Ka-band phases; this is the most critical time tag correction process and more stringent criteria are used here than any other module

`KBR_compress` —digital filtering of K- and Ka-band inter-spacecraft range data to form biased dual-1-way range (DOWR), range rate and range acceleration, computing light-time correction and antenna phase center offset from centers of gravity

`ACC_compress` — editing, applying timetag correction and digital filtering of accelerometer data

SCA_compress — editing, applying timetag correction and compression of star camera quaternion data, combining data from 2 star cameras, computing antenna phase center offset correction along the line-of-sight

GPS_compress — checking phase continuity and editing, applying timetag correction for GPS L-band range and phase data, and compressing phase data

gni1a2gnv1b — creating continuous (across day boundary) GPS navigation file (on-board GPS navigation solution)

TimeCorrHK — applying timetag correction to all housekeeping data

tdp2clk1b — converting tdp (time dependent parameters) clock files from GIPSY to CLK1B file, include clock reset information and validity intervals; CLK1B contains the final time series of the GRACE clock corrections to coordinate time

tim1a2tim1b — determine mapping from OBDH time to receiver time

clk1b2uso1b — calculating daily average GRACE carrier frequencies from clock solutions

Algorithms for major subroutines that are called by one or more of the main modules are separately described in the Appendices. A flow diagram for the Level-1B functions, products, and inter-dependencies is shown in Fig. 1.

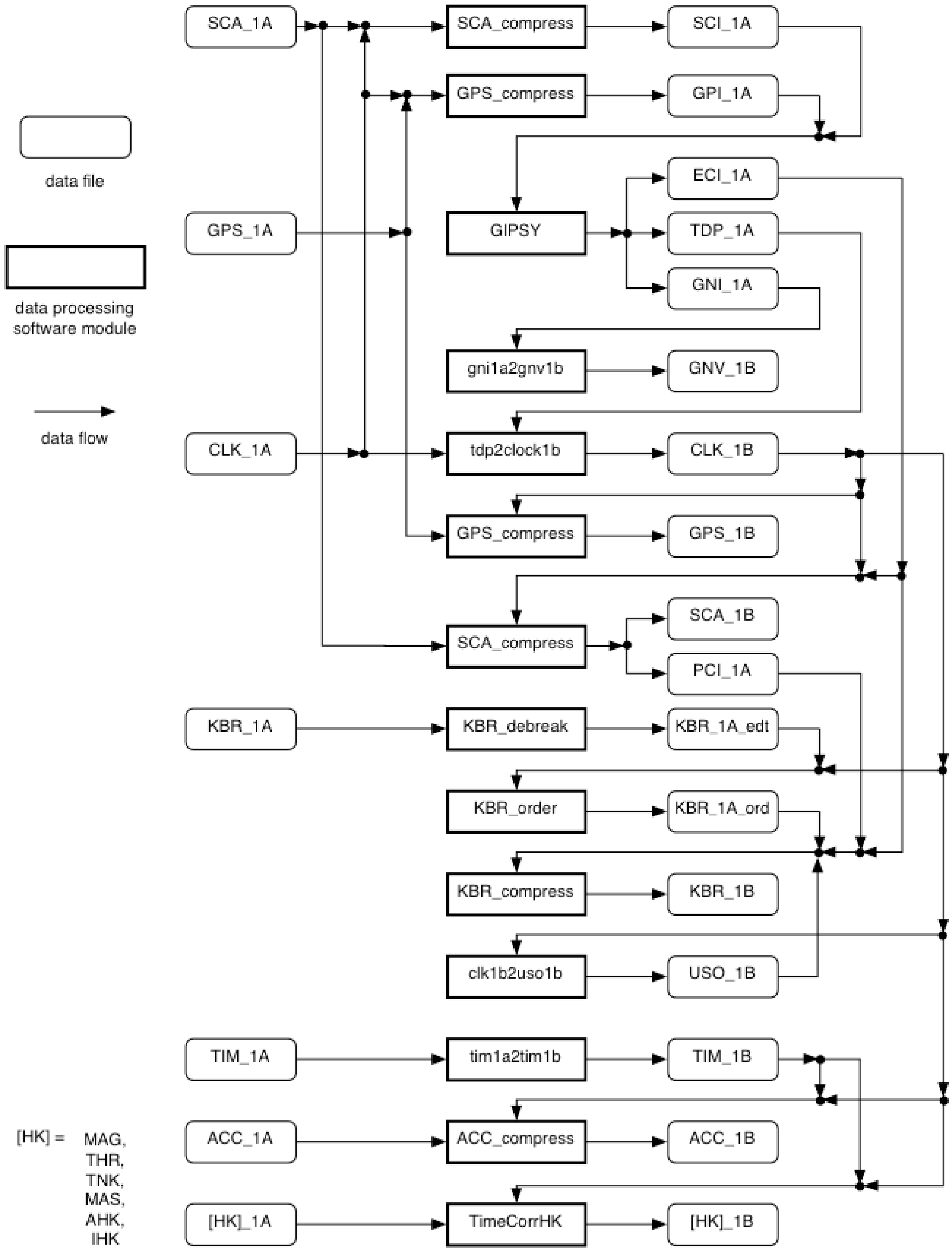


Fig. 1. Flow diagram of GRACE major data products

- This page intentionally blank -

2 KBR_Processing

The generation of KBR_1B data from KBR_1A data involves three processing modules in sequence: **`KBR_debreak`**, **`KBR_order`** and **`KBR_compress`**. Each of these processing modules are individually described in the following sub-chapters. These 3 modules are wrapped by the perl script kbr1a2kbr1b.pl.

2.1 KBR_debreak

`KBR_debreak` detects and flags KBR K- and Ka-band phase breaks, edits anomalous data and applies KBR timetag corrections due to missed interrupts affecting K or Ka phase measurements.

2.1.1 Input Files:

KBR_1A data file (1/10-sec nominal data interval) containing raw K- and Ka-band phases with unknown 10^8-cycle wrapping (applied on board to preserve precision, to accommodate the secular trend due to the ±~500-kHz baseband phase rates); the file name is defined by `–kbr1a_in` flag

(optional) Replacement SOE(Sequence of Events) file

2.1.2 Option Flags:

`-kbr1a_out`	optional KBR_1A output file name
`-tol`	tolerance for K–0.75Ka phase continuity beyond which the phase data are considered bad and discarded
`-maxbreak`	maximum gap width (sec) of phase data beyond which a flag for phase break is to be set
`-lowsnr_edit`	when set, the "low-SNR" editing is to be applied to eliminate bi-level phase observations
`-soe`	SOE (Sequence Of Events) file replacing the default one

2.1.3 Output File:

KBR_1A data file (1/10-sec nominal data interval) containing edited K- and Ka-band phases with unknown 10^8-cycle wrapping

2.1.4 Algorithm:

1. The tolerance for continuity test is set at Tol = 0.05 cycle as default. This default value can be replaced by `-tol` option while running **`KBR_debreak`**.
2. Max_Gap = 21 sec is set as the largest raw data gap beyond which a data break flag is set and a new pass begins at the next data points. This default value can be replaced by invoking `-maxbreak` option while running **`KBR_debreak`**.

3. Get the time series for K- and Ka-band timetag offsets due to missed interrupts from the default SOE file. Currently missed interrupts are detected by plotting linear combinations of the KBR data. The SOE keyword for K-band is K_MI and for Ka-band KAMI. The default SOE file can be replaced by invoking `–soe` option while running **`KBR_debreak`**. The antenna ID in the output data stream is made negative to designate that a non-zero timetag offset due to missed interrupt has been corrected.

   A missed interrupt is defined as a timetag shift in K- or Ka-band phase observation. This shift is a multiple of 0.02 sec depending on the number of missed interrupt events occurred since the last reset of the Instrument Processing Unit (IPU). The timetags of missed interrupt events and the timetag offsets are tabulated in the SOE file.

4. If `-lowsnr_edit` option is invoked while running **`KBR_debreak`**, the following onboard software glitch (due to unknown cause) is accommodated: The phases appear as alternating bi-level phases and the reported K/Ka-band SNR is erroneously low (~34 dB). Analysis has shown that the actual SNR is closer to 65 dB. Therefore, the bi-level phases are considered valid measurements but have unknown bias between the two levels. Further analysis showed that such bias cannot be determined to micron level and the following editing scheme was adopted to select only one of these levels. The resulting gaps in the edited data streams will be filled in **`KBR_order`**.

   The editing scheme used is the following:

   First, arbitrarily select the very first valid phase observation as the reference. If the next "range-free" (K–0.75Ka) phase data differs by ≤ 0.12 cycle from that at previous time separated by < 2.1 sec, then the data are considered continuous and retained; otherwise the data are considered to be a member of the other bi-level phase and discarded.

   Differences in 10^8-cycle wrapping between data points are unified (Appendix I) while forming the "range-free" phase data for continuity check.

5. If the "range-free" phase data differs by an amount > Tol from that at previous time separated by < 21 sec, the data are considered as anomalous and are discarded. This editing will remove isolated data outliers.

6. Set flag for phase break and restart a new continuous pass if discarded data string is longer than 21 sec (2 data packet periods plus 1 sec). Therefore, a 21-sec data gap is introduced in the data stream when a phase break occurs. Such a flag is also set when a change in missed interrupt level occurs, but without introducing the 21-sec data gap.

7. Correct data timetags by *adding* the offset due to missed interrupts and then resample at integer multiples of 1/10 sec with a cubic Lagrange interpolation (Appendix C) over 4 data time points (2 on either side of the corrected timetag) that remain in phase continuity.

8. Write into the output KBR_1A file the edited data, including the data quality flags for both K and Ka phases whenever a phase break occurs and the negative antenna ID designating a timetag correction made due to missed interrupts.

2.2 KBR_order

KBR_order fills data gaps and applies timetag correction, determined in the precision orbit determination (POD) process using the dual-frequency GPS data and the quaternion attitude information, to the KBR K- and Ka-band phases produced by **KBR_debreak**. Furthermore, a KBR time tag correction is applied to the K and Ka-band phases which is dependent on the IPU software version running in the IPU.

2.2.1 Input Files*:*

KBR_1A data file (1/10-sec nominal data interval) containing edited K- and Ka-band phases with unknown 10^8-cycle wrapping, which is the output from **KBR_debreak**; the file name is defined by `–kbr1a_in` flag

CLK_1B data file containing GRACE clock corrections

(optional) Replacement SOE file

2.2.2 Option Flags:

`-soe`	SOE file replacing the default one

2.2.3 Output File:

KBR_1A data file (1/10-sec nominal data interval) containing time-ordered K- and Ka-band phases with unknown 10^8-cycle wrapping

2.2.4 Algorithm:

1. Get the additional timetag offsets, KTOFF, from the default SOE file. Such timetag offsets (0.04000000 sec for GRACE A and 0.03999918 sec for GRACE B for Version 147 and earlier, 0 for Version 148 and after) on both K and Ka phases are due to an IPU software glitch in time tagging KBR data. These offsets are not accounted for by the clock corrections recorded in CLK_1B file.

   The default SOE file can also be replaced by invoking `-soe` option while running **KBR_order**.

2. Flag and fill data gaps if input data are continuous (phase break flag is not set). The following interpolation schemes are used doe gap filling (see Appendix A):

quadratic	if 2 data points are available on each side of the gap;
linear	if only 1 data point is available on either side of the gap.

   Note: Gaps > 2.1 sec are skipped (without setting any phase break flag) here but will be filled in **KBR_compress** after forming dual-one-way range (DOWR). This is to avoid interpolation of data with high-frequency variations over a large time span. Such high-frequency variations are significantly reduced in DOWR data and hence interpolation over a longer data span has lower interpolation error.

The relative 10^8-cycle wrapping between data at different time points are unified (Appendix I) while filling the data gaps and also while correcting timetags in the following step.

3. Check the availability of valid clock corrections (see the description of **`tdp2clock1b`** in a later section) from the input CLK_1B file in the neighborhood of KBR_1A data timetag:

   if valid corrections are available, compute timetag corrections by linearly interpolating CLK_1B clock corrections (typically given every 5-minutes);

   if valid corrections are not available, the KBR data are discarded.

4. Correct data timetags by *adding* the linearly interpolated timetag corrections and additional constant offset (KTOFF), and resample at the corrected integer multiples of 1/10 sec with a quadratic Lagrange interpolation (Appendix C).

   Care must be taken in the interpolation process when differences of timetags are computed to avoid significant round-off error (see Appendix C).

5. Write into the output KBR_1A file the time-ordered data, the associating phase break flags and filled data flags.

2.3 KBR_compress

`KBR_compress` digitally filters K- and Ka-band inter-spacecraft phase data from the ordered KBR_1A files (which is the output of **`KBR_order`** module) to form biased dual-1-way range (DOWR), range rate and range acceleration; it also computes light-time correction and ingests and combines the antenna phase center offset corrections from centers of gravity.

2.3.1 Input Files:

KBR_1A data files from both spacecraft (1/10-sec nominal data interval), containing time-ordered K- and Ka-band phases with unknown 10^8-cycle wrapping, which are the output from **`KBR_order`**; the files name is defined by `–Akbr1a` and `–Bkbr1a` flags for GRACE A and GRACE B data, respectively

ECI_1A data files containing GRACE orbit solution in inertial coordinates for both spacecraft; the files name is defined by `–Aeci1a` and `–Beci1a` flags for GRACE A and GRACE B data, respectively

PCI_1A data files containing antenna phase center offsets from the center of mass for both spacecraft; the files name is defined by `–Apci1a` and `–Bpci1a` flags for GRACE A and GRACE B data, respectively; these values are determined infrequently through special experiments

USO_1B files containing the daily average of K- and Ka-band carrier frequencies for both spacecraft; the GPS-observed USO frequency determines the length scale of the measurements. Although this input file is optional (see below), this file is a mandatory input for the Level 1B processing

2.3.2 Option Flags:

`-kbr1b`	optional KBR_1B output file name
`-version`	version number to be inserted in the output KBR_1B file name
`-maxbreak`	maximum gap width (sec) of phase data beyond which a flag for phase break is to be set
`-cal`	to write a CAL file (in ASCI format) containing uncompressed (10-Hz) version of KBR_1B data with a file name the same as the output KBR_1B file appended by the keyword "`.cal`"
`-bincal`	the same as flag above except that a binary file is written instead of in ASCI format
`-marginstart`	to extend the starting time (sec) before beginning midnight in the output KBR_1B data
`-marginend`	to extend the ending time (sec) after ending midnight in the output KBR_1B data
`-Aaddtcor`	additional timetag correction to K and Ka phase data for GRACE A
`-Baddtcor`	additional timetag correction to K and Ka phase data for GRACE B
`-K_phase`	nominal value of K-band GRACE A + GRACE B phase from the previous day to be used to maintain phase continuity across day boundary
`-Ka_phase`	nominal value of Ka-band GRACE A + GRACE B phase from the previous day to be used to maintain phase continuity across day boundary
`-Auso1b`	USO_1B file name containing daily average of K and Ka RF frequencies for GRACE A replacing the nominal ones
`-Buso1b`	USO_1B file name containing daily average of K and Ka RF frequencies for GRACE B replacing the nominal ones

2.3.3 Output File:

KBR_1B data file (5-sec data interval) containing dual-one-way range, range rate, range acceleration, Ka-band ionosphere, corrections for light-time and antenna center offsets, and their first and second time derivatives

(optional) CAL file (in ASCI format) containing uncompressed (10-Hz) version of KBR_1B data

(optional) binary CAL file containing uncompressed (10-Hz) version of KBR_1B data

2.3.4 *Algorithm:*

1. Antenna ID that was set negative to designate K or Ka phase timetag corrections due to missed interrupts is restored as positive and a data quality flag designating the missed interrupts is set.

2. Form ionosphere-free dual-one-way range (DOWR) and Ka-band ionospheric delay by proper combinations (see Appendix D) of the phase data between GRACE A & B and between K- & Ka-frequency bands.

   Differences in 10^8-cycle wrappings are unified (Appendix I) on combined phases where the net number of wrapping is generally within ±1.

   To preserve computation precision, the initial (generally large) values are removed from ionosphere-free DOWR and Ka-band ionospheric delay as constant biases and added back only after compression process in Step 4 below.

3. Flag and fill data gaps (Appendix A) with 2.1 < gap width ≤ 21 sec for DOWR and Ka-band ionospheric delay, using cubic interpolation with up to 100 data points on each side of the gap. No filling will be made for data gaps longer than 21 sec or across a time at which a phase break occurs.

4. In order to assure continuity in dual-one-way range, the effects of residual 10^8-cycle wrappings from previous day need to be restored. (Nominal value for the sum of phases between GRACE A and B may need to be input for each of K and Ka phases with `-K_phase` and `-Ka_phase` options).

5. Compress dual-one-way range data and form range rate and range acceleration data at integer multiples of 5 sec with a digital filter (Appendix B) of 7^{th}-order self-convolution with 100-mHz bandwidth over a 70.7-sec data span around the sampled time.

   To avoid the undesirable "peak shaving" effects, a quadratic over a 70.7-sec is removed prior to compression and added back afterward. "Peak shaving" is caused by low-pass filtering of a signal near the peaks when large 2^{nd}- and higher-order terms exist. The results are reduced peak amplitudes (hence "peak shaving").

6. Compress Ka-band ionospheric delay at integer multiples of 5 sec with the same digital filter as in Step 5 above.

7. Compute range, range rate and range acceleration corrections for light-time effects (see Appendix E) using the GRACE orbit positions and velocities from the two ECI_1A orbit data files(POD files). This is the correction to be added onto the DOWR to form the "instantaneous range".

8. Get the range, range rate and range acceleration corrections for antenna phase center offsets from center of mass from the two input PCI_1A and sum between GRACE A and GRACE B.

9. Write into the output KBR_1B file the ionosphere-free compressed range, range rate, range acceleration and Ka-band ionosphere from Step 5 and 6; the light-time and antenna phase corrections from Step 7 and 8; the associating nearest raw data signal-to-noise ratio

(SNR) for both K and Ka phases and both spacecraft; and the quality flags designating phase breaks, filled data, data with K or Ka timetag offsets due to missed interrupts, light-time corrections with extrapolated spacecraft states, and filled antenna phase center corrections.

3 ACC_compress

ACC_compress edits, applies timetag correction and digitally filters on ACC_1A accelerometer data to generate ACC_1B data.

3.1.1 Input Files:

ACC_1A data file containing 3 components of the raw linear acceleration (nominal 10-Hz data rate) and 3 components of the raw angular acceleration (nominal 1-Hz data rate) in ACC coordinate frame(identical to the Science Reference Frame(SRF) except for axis labels); the file name is defined by `-acc1a` flag

CLK_1B data file containing GRACE clock corrections; the file name is defined by `–clk1b` flag

(optional) TIM_1B file containing the conversion of OBDH (On Board Data Handler) time to receiver time for ACC_1A data (see Appendix G)

3.1.2 Option Flags:

`-acc1b`	optional ACC_1B output file name
`-version`	version number to be inserted in the output ACC_1B file
`-cal`	to write a CAL file (in ASCI format) containing uncompressed (10-Hz) version of ACC_1B data with a file name the same as the output ACC_1B file appended by the keyword "`.cal`"
`-tim1b`	TIM_1B file containing the conversion of OBDH time to receiver time for ACC_1A data (no time conversion if file not specified)
`-Butterworth_lag`	replacement of the default Butterworth filter delay of 0.14 sec
`-process_all`	if set, all data before and after midnights will be processed and written in the output ACC_1B data file (normally, only data between consecutive midnights are written)

3.1.3 Output File:

ACC_1B data file (1-sec data interval) containing the edited 3 components of linear and angular accelerations in GRACE Science Reference Frame (SRF)

(optional) CAL file (in ASCI format) containing uncompressed (10-Hz) version of ACC_1B data

3.1.4 Algorithm:

1. Remove data flagged with "no pulse sync" or "invalid timetag"; remove the whole 1-sec blocks of data on both sides of a data gap > 0.2 sec. "no pulse sync" refers to the 1-pps clock on OBDH not being synchronized with the 1-pps clock on the IPU; "invalid

timetag" refers to the timetag spacing on the ACC_1A data being corrupted which implies corrupted ACC data.

2. Convert data timetag from OBDH time to receiver time; and then resample both 10-Hz linear ACC data and 1-Hz angular ACC data at integer multiples of 1/10 sec in corrected time with linear interpolations.

3. Flag and fill data gaps (Appendix A) using cubic interpolation with up to 200 data points on each side of the gap; if a data gap is too wide (>100 sec), no filling will be made.

4. Compute timetag correction to be added to data receiver time by linear interpolation of clock corrections from input CLK_1B file, and subtract out the Butterworth filter delay (nominally 0.14 sec); and the resample at integer multiples of 1/10 sec with a Lagrange quadratic interpolation (Appendix C) over nearest 3 data points.

   In case there is no valid CLK_1B clock correction data for a time span, extrapolation of valid clock corrections outside of this time span is used to retain continuity of ACC data. A quality flag is set when this occurs. For validity of clock correction, see **`tdp2clk1b`** later. Note that time tag corrections for the ACC data do not need to be as accurate as KBR data corrections.

5. Sample angular ACC data at integer multiples of 1 sec; compress linear ACC data with a digital filter (Appendix B) of 7^{th}-order self-convolution with ~35-mHz bandwidth over a 140.7-sec data span around the sampled time.

6. Compute "fit residuals" by differencing compressed linear ACC data from uncompressed data at the integer-second sampling times. A quality flag is set when a residual > 10 microns/sec^2.

7. Rotate both linear and angular acceleration data from ACC frame into GRACE Science Reference Frame (SRF):

   $X_{SRF} = Z_{ACC}$

   $Y_{SRF} = X_{ACC}$

   $Z_{SRF} = Y_{ACC}$

8. Write into the output ACC_1B file the six components of compressed ACC data (3 linear and 3 angular accelerations) , the "fit residuals" and quality flags designating filled data, data with extrapolated clock corrections and data with large residuals.

4 SCA_compress

SCA_compress edits, applies timetag correction and compresses SCA_1A star camera quaternion data, combines data from 2 star cameras and produces SCA_1B data; it also computes antenna phase center offset correction along the line-of-sight and produces PCI_1A data, which is ingested by **KBR_compress** to produce the KBR_1B product.

4.1.1 Input Files:

SCA_1A data file containing raw quaternions rotating inertial frame to star camera frames (nominal data interval is 1 sec; occasionally, 5 sec for data from secondary star camera); the file name is defined by `-sca1a` flag

CLK_1B data file containing GRACE clock corrections; the file name is defined by `–clk1b` flag

ECI_1A data files containing GRACE orbit solutions from both spacecraft(POD); the file name is defined by `-Aeci1a` and `-Beci1a` flags for GRACE A and GRACE B respectively

4.1.2 Option Flags:

`-sca1b`	optional SCA_1B output file name
`-pci1a`	optional PCI_1A output file name
`-version`	version number to be inserted in the output SCA_1B file name
`-qks`	use "QKS" quaternions instead of "QSA" quaternions (Ref. 4) from SOE file for nominal star camera to SRF rotation quaternions
`-rpy`	output RPY file name containing Roll-Pitch-Yaw attitude deviation from the nominal attitudes derived from the line-of-sight between the two GRACE spacecraft
`-sigma`	replacement N value (default is 3) for local N-sigma editing of Roll-Pitch-Yaw attitude deviation from the nominal attitudes derived from the line-of-sight between the two GRACE spacecraft
`-cal`	to write a CAL file (in ASCI format) containing uncompressed (1-Hz) version of SCA_1B data with a file name the same as the output SCA_1B file appended by the keyword "`.cal`"
`-sci`	if set, the SCA_1B output file name is changed to SCI_1A
`-process_all`	if set, all data before and after midnights will be processed and written in the output SCA_1B data file (normally, only data between consecutive midnights are processed)
`-marginstart`	to extend the staring time (sec) before beginning midnight in the output SCA_1B data

`-marginend` to extend the ending time (sec) after ending midnight in the output SCA_1B data

4.1.3 Output File:

SCA_1B data file containing edited quaternion rotating inertial frame to SRF (5-sec data interval): data from primary or secondary star camera at times when only either one exists; combined data at times when both exist; no output data at times when neither exists

PCI_1A data (5-sec data interval): range, range rate and acceleration of DOWR antenna phase center correction for KBR_1B data based on SCA_1B quaternion and ECI_1A

(optional) CAL file (in ASCI format) containing uncompressed (1-Hz) version of SCA_1B data

(optional) RPY file containing Roll-Pitch-Yaw attitude deviation (in degrees) from the nominal attitudes derived from the line-of-sight between the two GRACE spacecraft

4.1.4 Algorithm:

1. Discard SCA_1A data flagged as “invalid”, and data with negative or large (>7) star catalog “fit residuals”. These residuals are reported by the star camera software.
2. Convert the input quaternions from [inertial frame (I) to camera frame (C) rotation] into [inertial frame (I) to SRF (S) rotation] by post multiplying QSA (or QKS) quaternion, which is [camera frame (C) to SRF (S) rotation] read in from SOE file:

   $$R_{I,S} = P_{I,C}\, Q_{C,S}$$

   See Appendix F for quaternion definition and operations.

   The QSA quaternions are defined as the rotation from Star camera frame to SRF and QKS quaterions are defined as the rotation from Star camera frame to KBR bore sight frame used for pointing the spacecraft. QKS is not used in the Level-1B processing but only for attitude control performance tests.
3. Compute the nominal quaternion $E_{I,S}$ (see Appendix H) derived from
   (a) the line-of-sight if ECI orbits of both spacecraft are supplied; or
   (b) the velocity vector if ECI orbit of only the receiving spacecraft is supplied.
4. Compute the “difference” (see Appendix F) of $R_{I,S}$ from $E_{I,S}$:

   $$D_{I,S} = (E_{I,S})^{-1}\, R_{I,S}$$

   and convert the difference into Roll, Pitch and Yaw angles.
5. Edit the quaternion $R_{I,S}$ in Step 2 with a global (over the entire day of data) N-sigma test for the residuals of the differences in Roll, Pitch and Yaw angles in Step 4. Currently, N is set to be big (N = 33) to bypass this editing step in effect.

6. Further edit the residuals from Step 5 with a local 3-sigma test over 300-sec time spans. The local fit is according to the number of raw data points (N_p) involved:

   cubic if $N_p > 20$;

   quadratic if $5 < N_p \leq 20$;

   no editing if $N_p \leq 5$.

7. Flip signs, if needed, for all 4 quaternion components to maintain their continuity in time.

8. Compute timetag corrections by linearly interpolating clock corrections from input CLK_1B file. In case there is no valid CLK_1B clock correction data for a time span, extrapolation of valid clock corrections outside of this time span is used to retain continuity of SCA data. A quality flag is set when this occurs.

   For validity of clock correction, see **`tdp2clock1b`** in a later section.

9. Resample SCA_1A data at integer seconds for both primary and secondary star camera data within ±2.5 sec from the resampled time, and extended up to ±5 sec until 3 data points are available (N = 3). The interpolation scheme is

   quadratic if $N \geq 3$

   linear if $N = 2$

   point sampling if $N = 1$

10. Combine data from 2 star cameras (Appendix J), if available. At times when no data from either camera is available, no data will be output.

11. Rotate KBR antenna phase center from center of mass into inertial space by the combined quaternion from Step 10 and then compute the projection on the line-of-sight vector between GRACE A and GRACE B computed from the input ECI file.

    The default antenna phase center offset vectors of [1.472584 m, 0, 0] in SRF can be substituted by the time series from the "VKB" records in the SOE file. These vectors are the ionospere-free combinations of offsets between K- and Ka-frequencies and are determined by specialized experiments

12. Linearly interpolate the antenna phase center correction into 10-Hz rate (and flagged as filled data if interpolation over a time span > 5 sec is called for) and compress with a digital filter (Ref. 1; Appendix B) with 100-mHz bandwidth over a 70.7-sec data span around the sampled time.

13. Write into the output SCA_1B file the 4 components of the compressed/combined quaternion, and quality flags designating data with extrapolated clock corrections, data derived from single star camera and data derived from low-rate (5-sec interval) incoming data.

14. Write into the output PCI_1A file the compressed range, range rate and range acceleration of antenna phase center correction (wrt. center of mass), and the quality flag designating filled data.

- This page intentionally blank -

5 GPS_compress

`GPS_compress` checks phase continuity on GPS_1A L-band phase data and edits accordingly, applies timetag correction for both range and phase data, compresses phase data and produce GPS_1B data.

5.1.1 Input Files:

GPS_1A data file containing raw CA, L1 and L2 phases (nominally 1-sec interval) and pseudorange (nominally 10-sec interval) data; the file name is defined by `-gps1a` flag

Clock data file containing GRACE clock corrections; the file name is defined by `-clk1a` flag although CLK_1B file can be used for precise clock corrections

5.1.2 Option Flags:

`-gps1b`	optional GPS_1B output file name
`-version`	version number to be inserted in the output GPS_1B file name
`-gpi`	if set, the output GPS_1B file name is labeled as GPI1A… instead of GPS1B…
`-cal`	to write a CAL file (in ASCI format) containing uncompressed (1-Hz) version of GPS_1B data with a file name the same as the output GPS_1B file appended by the keyword "`.cal`"
`-editsnr_p1`	replacement value for minimum SNR (default is 2) for editing raw L1 phase
`-editsnr_p2`	replacement value for minimum SNR (default is 2) for editing raw L2 phase
`-phtol_l1`	replacement value for tolerance (default is 1 L1-cycle = 0.19 m) in editing raw CA–L2 phase
`-phtol_l2`	replacement value for tolerance (default is 1 L1-cycle = 0.19 m) in editing raw L1–L2 phase
`-process_all`	if set, all data before and after midnights will be processed and written in the output GPS_1B data file (normally, only data between consecutive midnights are processed)
`-marginstart`	to extend the starting time (sec) before beginning midnight in the output GPS_1B data
`-marginend`	to intend the ending time (sec) after ending midnight in the output GPS_1B data

5.1.3 *Output File:*

GPS_1B data file (10-sec data interval) containing edited CA, L1 and L2 phases and pseudorange data

5.1.4 *Algorithm:*

1. Check the validity of incoming data by observing the following signal-to-noise criteria:

   Data are considered bad and discarded if

   $SNR_{L1} < p1\_SNR$ or

   $SNR_{L2} < p2\_SNR$ or

   $SNR_{L1} < 0.4\,(SNR_{CA})^2\,/1000$

   p1_SNR and p2_SNR have a default value of 2, and can be replaced by invoking `-editsnr_p1 and -editsnr_p2` options while running **`GPS_compress`**.

2. Check the availability of valid clock corrections from the input CLK_1B file in the vicinity of GPS data timetag:

   if valid corrections are available, compute timetag corrections by linearly interpolating clock corrections from input CLK_1B file.

   if valid corrections are not available, GPS data are discarded.

   For validity of clock correction, see **`tdp2clock1b`** in a later section.

3. GPS data timetags are corrected by adding the timetag corrections; GPS phases are corrected by adding the product of the timetag corrections and the speed of light.

4. Check for CA–L2 and/or L1–L2 phase continuity with default tolerances, $Tol_{L1} = Tol_{L2} =$ 0.19 m (1 L_1-cycle) which can be replaced by invoking `-phtol_l1` and `-phtol_l2` respectively while running **`GPS_compress`**:

   accumulate L1 (or CA if L1 not available) and L2 phase data over 5 time points which are within 5 sec away from the most recent data point and perform a quadratic fit to compute the RMS fit residual, *Res*;

   if *Res* > Tol/4, replace oldest data point with a new data point and repeat the fit;

   if *Res* ≤ Tol/4, extrapolate L1 and L2 phase data to new time point as predictions;

   if | new phase – prediction | > Tol, set flag at new data point for phase break;

   if new data time – previous (10-sec) output time > 100 sec, set flag for phase break.

5. Form "negative-ion" raw phases, by proper combinations of L1 and L2 phases, ϕ_1 and ϕ_2, so that they have the same (instead of opposite) ionospheric effects as the corresponding range data:

   $$\phi_1^- = (2.546 + 1.546)\ \phi_1 - 2\ (1.546)\ \phi_2$$

$$\phi_2^- = 2\ (1.546)\ \phi_1 - (2.546 + 1.546)\ \phi_2$$

6. Form [range – negative-ion phase] differences, which is essentially a constant bias plus range data noise and multipath errors:

$$B_{CA} = R_{CA} - \phi_1^-$$

$$B_{L1} = R_{L1} - \phi_1^-$$

$$B_{L2} = R_{L2} - \phi_2^-$$

7. Correct raw phase data time tags and phase observables with input clock correction file and then compress/resample at integer multiples of 10 sec with a cubic interpolation over a 10-sec data span. Let this re-sampled phase be Φ_{CA}, Φ_{L1} and Φ_{L2}.

8. Form re-sampled "negative-ion" phases by proper combinations of L1 and L2 resampled phases in Step 7:

$$\Phi_{L1}^- = (2.546 + 1.546)\ \Phi_{L1} - 2\ (1.546)\ \Phi_{L2}$$

$$\Phi_{L2}^- = 2\ (1.546)\ \Phi_{L1} - (2.546 + 1.546)\ \Phi_{L2}$$

9. Form re-sampled range data by summing the [range – negative-ion phases] B_{CA}, B_{L1} And B_{L2} in Step 6 and the resampled "negative-ion" phases Φ_{L1}^- and Φ_{L2}^- in Step 5:

$$\mathcal{R}_{CA} = B_{CA} + \Phi_{L1}^-$$

$$\mathcal{R}_{L1} = B_{L1} - \Phi_{L1}^-$$

$$\mathcal{R}_{L2} = B_{L2} - \Phi_{L2}^-$$

   The "signal" of these resampled range data is precisely resampled and the data noise is slightly increased from the original range data by the far lower noise of ϕ_{Li}^- and Φ_{Li}^-.

10. A constant phase bias is adjusted for each continuous phase stream from Step 7, so that the phase values are close to the range counterparts from Step 9 using the first valid range value for the continuous phase pass.

11. Write into the output GPS_1B file the resampled CA, L1 and L2 ranges from Step 9, the bias-adjusted, resampled, compressed CA, L1 and L2 phases from Step 10, and quality flags designating phase breaks.

6 gni1a2gnv1b

gni1a2gnv1b applies a smoothing process from 23:00 to 01:00(next day) on the GIPSY noon centered 30-hour GNI_1A orbit ephemerides and creates continuous (across day boundary) GPS navigation file GNV_1B .

6.1.1 Input Files:

GNI_1A containing precision orbits estimated by GIPSY; the file name is defined by `–gni1a_curr` flag

6.1.2 Option Flags:

`-gni1a_prev`	optional input GNI_1A filename for previous day
`-gni1a_next`	optional input GNI_1A filename for next day
`-gnv1b`	optional GNV_1B output file name
`-version`	version number to be inserted in the output HK_1B filename
`-smooth_window`	window size to be used to smooth ephemeris solutions for adjacent days
`-tcentroid_sm_start`	start centroid time to center smooth window on
`-tcentroid_sm_end`	end centroid time to center smooth window on
`-tmargin_start`	to extend the starting time (sec) before beginning midnight in the output GNV_1B file
`-tmargin_end`	to extend the ending time (sec) after ending midnight in the output GNV_1B file
`-stats_window`	window centered on midnight to compute clock overlap statisitics, (default: 7200 sec)
`-fac`	factor used in sigma editing of orbit overlap statistics. (default: 3.0)

6.1.3 Output Files:

GNV_1B containing precision orbits estimated by GIPSY which have been smoothed near day boundaries with previous and next day solutions

6.1.4 Algorithm:

1. Use cosine smoothing to smooth GNI_1A state vectors (position vector and velocity vector in ITRF2000 frame) on day boundaries using previous and next days GNI_1A state vector , if either previous and/or next day is specified. Outside these intervals the state vectors remain unchanged.

On the start day boundary, the default smooth window parameters are:

$tcentroid\_sm\_start = midnight\_start;\quad \mathrm{smooth\_window} = 1800\ \mathrm{sec}$

These parameters may be specified by using options `-tcentroid_sm_start` and `-smooth_window`. The smooth window is then defined as:

$$tcentroid\_sm\_start - smooth\_window \le t \le tcentroid\_sm\_start + smooth\_window$$

The smoothed state vector *GNV_1B_SMOOTH(t)* in this window is then calculated as a weighted average of the previous day *GNI_1A_prev(t)* and current day *GNI_1A_curr(t)* state vector according to:

$$GNV\_1B\_SMOOTH(t) = W(t) * GNI\_1A\_prev(t) + (1 - W(t)) * GNI\_1A\_curr(t)$$

where the cosine weight *W(t)* is calculated according to:

$$W(t) = (1 + \cos(\omega(t - tcentroid\_sm\_start + smooth\_window)))/2.0$$

and

$$\omega = 0.5\pi / \mathrm{smooth\_window}$$

On the end day boundary, the default smooth window parameters are:

$$tcentroid\_sm\_end = midnight\_end;\quad \mathrm{smooth\_window} = 1800\ \mathrm{sec}$$

These parameters may be specified by using options `-tcentroid_sm_end` and `-smooth_window`. The smooth window is then defined as:

$$tcentroid\_sm\_end - smooth\_window \le t \le tcentroid\_sm\_end + smooth\_window$$

The smoothed state vector *GNV_1B_SMOOTH(t)* in this window is then calculated as a weighted average of the current day *GNI_1A_curr(t)* and the next day *GNI_1A_next(t)* state vector according to:

$$GNV\_1B\_SMOOTH(t) = W(t) * GNI\_1A\_curr(t) + (1 - W(t)) * GNI\_1A\_next(t)$$

where the cosine weight *W(t)* is calculated according to:

$$W(t) = (1 + \cos(\omega(t - tcentroid\_sm\_end + smooth\_window)))/2.0$$

and

$$\omega = 0.5\pi / \mathrm{smooth\_window}$$

2. Similar to step 1., use cosine smoothing to smooth GNV_1B the formal state vector error on the day boundaries using previous and next days GNI_1A data, if either previous and/or next day is specified. Outside these intervals the formal state vecor

errors remain unchanged. The smoothed formal errors are calculated for the start day boundary according to:

$$\left[GNV\_1B\_SMOOTH\_ERR(t)\right]^2 = \left[W(t) * GNI\_1A\_prev\_ERR(t)\right]^2 + \left[(1-W(t)) * GNI\_1A\_curr\_ERR(t)\right]^2$$

Likewise for the end day boundary the smoothed formal clock error is calculated according to:

$$\left[GNV\_1B\_SMOOTH\_ERR(t)\right]^2 = \left[W(t) * GNI\_1A\_curr\_ERR(t)\right]^2 + \left[(1-W(t)) * GNI\_1A\_next\_ERR(t)\right]^2$$

3. Set flags when smoothing can't be performed because a state vector is missing in the smoothing window. In this case the state vector and formal error that is available in the adjacent days will be used unchanged.
4. Write GNV_1B records to GNV_1B output file including flags.

7 Timing Processing

This chapter covers three modules for the mapping and conversion and between different time frames (Appendix G): OBDH time, receiver time and GPS time(a realization of coordinate time/UTC).

Onboard the GRACE spacecraft two timing frames are used for time tagging all science and housekeeping data. The OBDH provides time tags (OBDH time) for all housekeeping data and ACC data. All other science data are time tagged by the IPU (receiver time). Under nominal conditions, OBDH time and receiver time are synchronized to within 5 milliseconds. During an IPU reboot, however, these time frames are not synchronized and require a correction, which is determined by **`tim1a2tim1b`**. The IPU time tag (receiver time) is only synchronized with GPS time at the time of reboot and then is allowed the freely drift. To correct for this drift a clock solution is computed during the precision orbit determination process. The clock solution necessary to correct to GPS time, is determined by **`tdp2clk1b`**. The resulting clock solution (tdp2clk1b) and OBDH time mapping (tim1a2tim1b) are then used to correct the time tags for all housekeeping file using **`TimeCorrHk`**. The science data processing modules ingest these corrections directly as part of their processing strategy. Finally a fourth module (**`clk1b2uso1b`**) is described for estimating the nominal RF frequencies onboard the two GRACE spacecraft.

7.1 TimeCorrHK

`TimeCorrHK` converts the OBDH time tags for all housekeeping into GPS time tags

7.1.1 Input Files:

AHK_1A data file containing ACC housekeeping data; the file name is defined by `-hk1a` flag

IHK_1A data file containing IPU housekeeping data; the file name is defined by `-hk1a` flag

MAG_1A data file containing magnetometer and magnetorquer data; the file name is defined by `-hk1a` flag

MAS_1A data file containing spacecraft mass data; the file name is defined by `-hk1a` flag

THR_1A data file containing thruster activation data; the file name is defined by `-hk1a` flag

TNK_1A data file containing gas tank measurements; the file name is defined by `-hk1a` flag

Clock data file containing GRACE clock corrections; the file name is defined by `-clk1b` flag although CLK_1A file can be used for precise clock corrections

7.1.2 Option Flags:

`-hk1b`	optional GPS_1B output file name
`-version`	version number to be inserted in the output HK_1B file

`-tim1b` TIM_1B file containing the conversion of OBDH time to receiver time for HK_1A data (no time conversion if file not specified)

7.1.3 Output Files:

AHK_1B data file containing ACC housekeeping data with time tag corrected to GPS time

IHK_1B data file containing IPU housekeeping data with time tag corrected to GPS time

MAG_1B data file containing magnetometer and magnetorquer data with time tag corrected to GPS time

MAS_1B data file containing spacecraft mass data with time tag corrected to GPS time

THR_1B data file containing thruster activation data with time tag corrected to GPS time

TNK_1B data file containing gas tank measurements with time tag corrected to GPS time

7.1.4 Algorithm:

1. Convert data timetag from OBDH time to receiver time if TIM_1B is specified
2. Compute timetag correction to be *added* to data receiver time by linear interpolation of clock corrections from input CLK_1B file
3. Set flag if no OBDH mapping is available for a given time tag
4. Set flag if no CLK1B correction is available for a given time tag
5. If MAG_1A input file is specified then rotate magnetometer measurements into the SRF according to rotation matrix A:

$$A = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ -\sin(\alpha) & -\cos(\alpha) & 0 \\ 0 & 0 & -1 \end{bmatrix}; \qquad \alpha = 25^o; \qquad \vec{M}_{SRF} = A\,\vec{M}_{meas}$$

   where $\vec{M}_{meas}$ is the magnetometer measurement in the magnetometer frame and

   where $\vec{M}_{SRF}$ is the rotated magnetometer measurement in the SRF

6. Write into output file HK_1B file the updated records, including the data quality flags

7.2 tdp2clk1b

`tdp2clk1b` combines the apriori clock solution in CLK_1A and the GISPY tdp (time dependent parameters) clock solution in to the CLK_1B file, which also includes clock reset information and validity intervals. Furthermore **`tdp2clk1b`** applies a smoothing process from 23:00 to 01:00(next day) on the GIPSY noon centered 30-hour TDP_1A combined with CLK_1A clock solution and creates the continuous (across day boundary) clock solution file CLK1B_1B

7.2.1 Input Files:

TDP_1A containing clock correction estimates by GIPSY; the file name is defined by `—tdp_curr` flag

CLK_1A data file containing GRACE clock corrections based on quadratic fits of the onboard clock solution; the file name is defined by `—clk1a_curr` flag. This input is optional

7.2.2 Option Flags:

`-tdp_prev`	optional input TDP_1A filename for previous day
`-tdp_next`	optional input TDP_1A filename for next day
`-clk1a_prev`	optional input CLK_1A filename for previous day
`-clk1a_next`	optional input CLK_1A filename for next day
`-tdp1b`	option to output a TDP_1B file in TDP_1A file format
`-clk1b`	optional CLK_1B output file name
`-version`	version number to be inserted in the output HK_1B filename
`-no_sat_name_chk`	turn of satellite name checking on input files
`-smooth_window`	window size to be used to smooth clock solutions for adjacent days
`-tcentroid_sm_start`	start centroid time to center smooth window on
`-tcentroid_sm_end`	end centroid time to center smooth window on
`-tmargin_start`	to extend the starting time (sec) before beginning midnight in the output CLK_1B file
`-tmargin_end`	to extend the ending time (sec) after ending midnight in the output CLK_1B file
`-stats_window`	window centered on midnight to compute clock overlap statisitics
`-sigedt`	factor used in sigma editing of the linear fit residuals of the total clock solution
`-sigrate`	factor used in sigma editing of the linear fit residuals of the total clock rate solution
`-maxformerror`	all clock solutions with a formal error > maxformerror will be edited
`-mingap`	minimum gap for which a new validity interval must be set

7.2.3 Output Files:

CLK_1B data file containing the total GRACE clock corrections

TDP_1B data file containing the total GRACE clock corrections for the same time span as the TDP_1A file and with no smoothing applied.

7.2.4 Algorithm:

1. For each time tag (t) in TDP_1A add clock solution *TDP_1A(t)* and *CLK_1A(t)* together according to

   $TDP\_1B(t) = TDP\_1A(t) + CLK\_1A(t) \times CSPEED$ (km)

   where

CSPEED	International defined standard speed of light (299792.458 km/sec)
$TDP\_1A(t)$	Clock solution at time (t) based on precision orbit determination in TDP_1A file
$CLK\_1A(t)$	Clock solution at time (t) computed by linearly interpolating the parabolic fit of the onboard clock solution in CLK_1A file

2. Repeat step 1. for previous and next day TDP_1A files if they are specified
3. Get the time series for the IPU resets from the default SOE file . The SOE keyword for IPU resets is "IPUR". From this time series determine the continuous intervals during the day to be processed
4. Compute for each continuous interval the clock rate by using a linear least squares fit for each interval. The linear least squares procedure is iterated according to the following procedure
   a) Select time series for a continuous interval (t, *,TDP_SEL(t)*) and compute data weight for each data point based on the formal sigma of the TDP_1B clock solution
   b) Compute linear least square fit to time series (*t,TDP_SEL(t)*)
   c) Compute fit residual RMS
   d) Edit all data points from TDP_SEL for which the fit residual satisfies: $|TDP\_1B(t) - fit(t)| \geq sigrate\_factor \cdot RMS\_fit$. The *sigrate_factor* can be set with the `–sigrate` option. Default *sigrate_factor* = 3.0
   e) If the number of data points edit is not equal to zero goto step b)
   f) If remaining number of data points in TDP_SEL is less than 2 then set clock rate to zero, otherwise record clock rate and formal error for this continuous interval.
5. Use cosine smoothing to smooth TDP_1B clock solution on day boundaries using previous and next days TDP_1B data, if either previous and/or next day is specified. Outside these intervals the clock solutions remain unchanged.

On the start day boundary, the default smooth window parameters are:

$$tcentroid\_sm\_start = midnight\_start; \quad smooth\_window = 3600 \text{ sec}$$

These parameters may be specified by using options `-tcentroid_sm_start` and `-smooth_window` . The smooth window is then defined as:

$$tcentroid\_sm\_start - smooth\_window \le t \le tcentroid\_sm\_start + smooth\_window$$

The smoothed clock solution *TDP_1B_SMOOTH(t)* in this window is then calculated as a weighted average of the previous day *TPD_1B_prev(t)* and current day *TDP_1B_curr(t)* clock solution according to:

$$TDP\_1B\_SMOOTH(t) = W(t) * TDP\_1B\_prev(t) + (1 - W(t)) * TDP\_1B\_curr(t)$$

where the cosine weight *W(t)* is calculated according to:

$$W(t) = (1 + \cos(\omega(t - tcentroid\_sm\_start + smooth\_window)))/2.0$$

and

$$\omega = 2\pi / smooth\_window$$

On the end day boundary, the default smooth window parameters are:

$$tcentroid\_sm\_end = midnight\_end; \quad smooth\_window = 3600 \text{ sec}$$

These parameters may be specified by using options `-tcentroid_sm_end` and `-smooth_window` . The smooth window is then defined as:

$$tcentroid\_sm\_end - smooth\_window \le t \le tcentroid\_sm\_end + smooth\_window$$

The smoothed clock solution *TDP_1B_SMOOTH(t)* in this window is then calculated as a weighted average of the current day *TPD_1B_curr(t)* and the next day *TDP_1B_next(t)* clock solution according to:

$$TDP\_1B\_SMOOTH(t) = W(t) * TDP\_1B\_curr(t) + (1 - W(t)) * TDP\_1B\_next(t)$$

where the cosine weight *W(t)* is calculated according to:

$$W(t) = (1 + \cos(\omega(t - tcentroid\_sm\_end + smooth\_window)))/2.0$$

and

$$\omega = 2\pi / smooth\_window$$

6. Similar to step 5, use cosine smoothing to smooth TDP_1B the formal error of the clock solution on day boundaries using previous and next days TDP_1B data, if either previous and/or next day is specified. Outside these intervals the formal errors remain unchanged. The smoothed formal errors are calculated for the start day boundary according to:

$$[TDP\_1B\_SMOOTH\_ERR(t)]^2 = [W(t) * TDP\_1B\_prev\_ERR(t)]^2 + [(1-W(t)) * TDP\_1B\_curr\_ERR(t)]^2$$

Likewise for the end day boundary the smoothed formal clock error is calculated according to:

$$[TDP\_1B\_SMOOTH\_ERR(t)]^2 = [W(t) * TDP\_1B\_curr\_ERR(t)]^2 + [(1-W(t)) * TDP\_1B\_next\_ERR(t)]^2$$

7. Remove TDP_1B_SMOOTH(t) clock soltutions for which the formal error is larger than the maximum formal error of 10 cm. This parameter can be changed using the `–maxformerror option`
8. Convert TDP_1B_SMOOTH(t) clock solution into CLK_1B clock representation (seconds) according to:

   $$CLK\_1B(t) = TDP\_1B\_SMOOTH(t) / CSPEED$$

9. Set clock validity intervals flags using the following criteria:
   a) If an IPU reboot occurs then insert clock records with validity flags set. The reason is that the clock after reboot is reset to coincide with an on-board solution GPS time(typically much better than a microsecond) and then freely drifts thereafter, hence a discontinuity exist in the clock at the reboot time.
   b) If a time gap in the valid clock solutions is larger than 900 seconds then insert clock records with validity flags set . The reason is that for KBR and GPS product a very accurate clock is required and the linear interpolation error is too large for gaps greater than 900 seconds. The gap size can be changed with the `–mingap` option.

The insertion of clock records with validity flags set, are shown in Fig. 2.

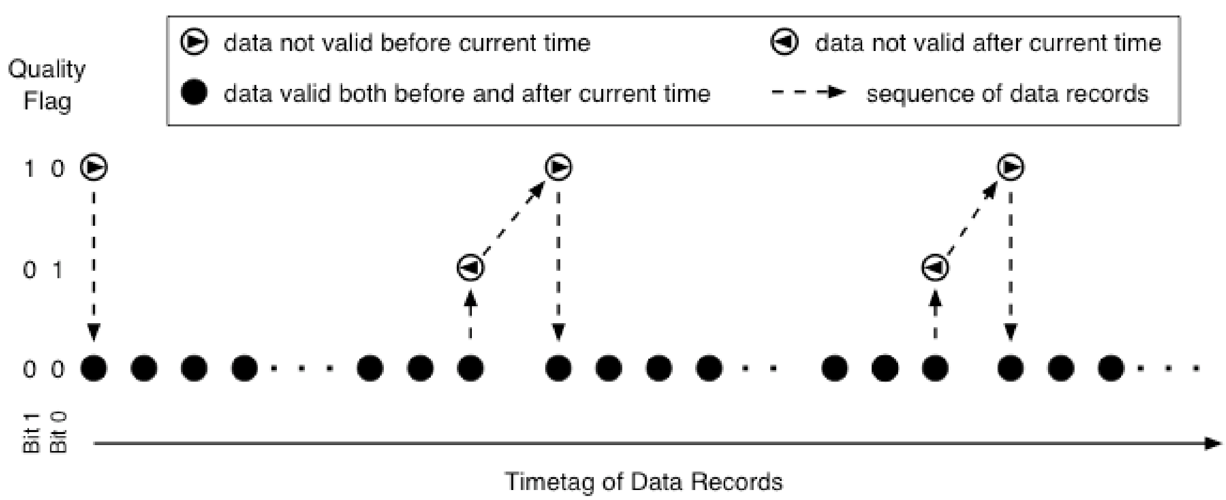


Fig. 2. Clock validity intervals designation

The validity intervals are determined by two bits in the CLK_1B quality flags. Bit 0 designates that a solution is not valid before the current time tag and bit 1 designates that a clock solution is not valid after the current time tag. When an interval is to be assigned, the boundary records are duplicated and the appropriate validity flags are set. If the clock solution is not valid before the current time then the duplicate record is inserted before to the boundary record. Likewise, if the clock solution is not valid after the current time then the duplicate record is inserted after the boundary record. The proper sequence is also illustrated in the figure above as well as the validity flag settings.

10. Set flags when smoothing can't be performed because a clock solution is missing in the smoothing window. In this case the clock solution and formal error that is available in the adjacent days will be used unchanged.
11. Write into output file CLK_1B the CLK_1B records with the clock solution, formal clock error and clock drift and the data quality flags

7.3 tim1a2tim1b

`tim1a2tim1b` determines mapping from OBDH time to receiver time.

7.3.1 Input Files:

TIM_1A data file containing OBDH timing information and ACC counter information; the file name is defined by `–tim1a` flag

ILG_1A data file containing IPU log message including reboot information; the file name is defined by `–ilg1a` flag

7.3.2 Option Flags:

`-tim1a_prev`	optional input TIM_1A filename for previous day
`-tim1a_next`	optional input TIM_1A filename for next day
`-ilg1a_prev`	optional input ILG_1A filename for previous day
`-ilg1a_next`	optional input ILG_1A filename for next day
`-tim1b`	optional TIM_1B output file name
`-version`	version number to be inserted in the output HK_1B filename
`-preserve_edges`	do not remove time tags with no IPU nav update available at beginning and end of specified time span

7.3.3 Output Files:

TIM_1B data file containing OBDH time to Reciever time mapping information

7.3.4 *Algorithm:*

1. Determine all IPU clock nudges within the specified time span from the ILG_1A file IPU log messages. A clock nudge is defined as the amount of time the IPU needs to shift the 1PPS to get the 1PPS back in sync with the GPS time realization based on the onboard navigation solution.

2. Determine all time spans for which OBDH time (appendix G) equals Receiver Time based on if the absolute time transfer confirmation flag is set in the TIM_1A file. The time spans that do not satisfy the criteria above require an OBDH to Receiver time mapping that is not the identity. These intervals are matched up with the IPU nudge times from step 1.

3. One of the time spans as determined in step 2 is illustrated in Fig. 3. In this case the absolute time transfer is not set from 0 till 191 seconds.

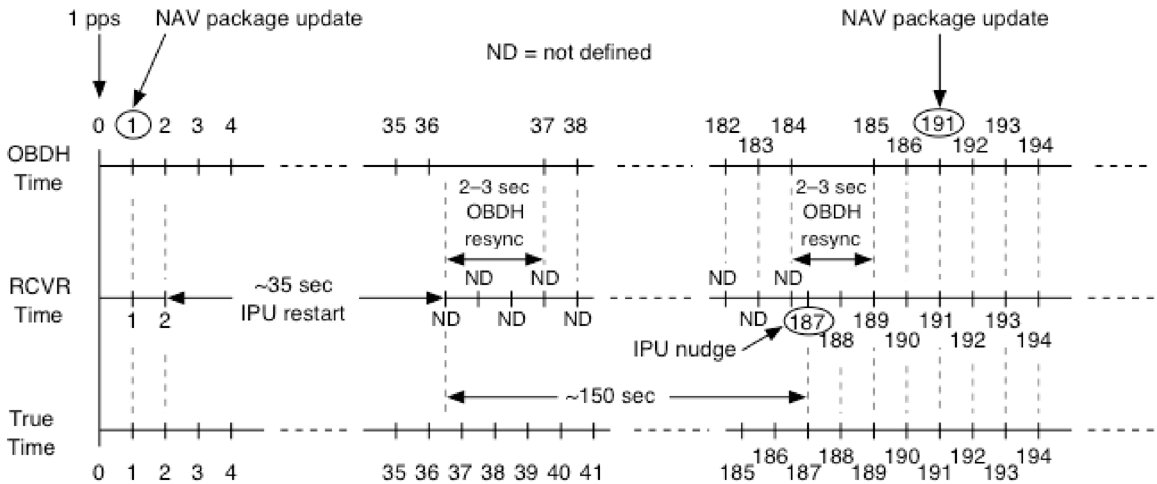


Fig. 3. Relationship between OBDH and receiver times during an IPU reboot

Fig. 3 illustrates the relationship between OBDH, Receiver and True time during an IPU reboot. The following time sequence of events occurs during a nominal reboot:

a) IPU seizes all operations, including 1PPS, at the start of the reboot process. All IPU data products disappear including the realization of receiver time. The OBDH, however, continues to count but the out of sync flag is set in the time stamp packet.

b) After ~35 seconds the IPU 1PPS is restarted at an arbitrary time. No realization of receiver time information is received because no IPU data products are available at this time. On the other hand the ACC starts taking data after the IPU 1PPS returns and starts counting at the ND tick marks of receiver time

c) As the IPU 1PPS returns after about ~35 sec, the OBDH initiates the re-sync process of the OBDH 1PPS with the IPU 1PPS. This normally takes about 2-3 seconds. After the OBDH 1PPS returns, the OBDH counter resumes with the count before the initialization process. Hence OBDH time at this point is 2-3 seconds behind. Since the IPU is not producing any products, including the navigation packets containing the absolution timing information, no update will be performed by the OBDH and OBDH time remains late by 2-3 seconds.

d) Once the IPU has obtained a valid onboard navigation solution (~150 seconds), the size of the clock nudge is calculated to sync the IPU 1PPS with the realization of GPS time. After the nudge is applied the IPU starts producing data products and thus the realization of receiver time, which is synched with GPS time.

e) As the IPU applies the nudge, the OBDH initiates again the re-sync process of the OBDH 1PPS with the IPU 1 PPS. Again after about 2-3 seconds the OBDH 1PPS returns with a further delay of 2-3 seconds. The OBDH time at this time is synced with the IPU 1PPS but the absolute time count is off by about 4-6 seconds.

f) The first IPU navigation packet that arrives after the IPU clock nudge will correct the absolute time count. In the figure at 191 seconds. From this moment OBDH and Receiver time are synched

4. The OBDH time to Receiver time mapping is determined according to the following procedure by working backwards in time from the time tag where the first IPU navigation packet updates the absolute OBDH time count after the IPU clock nudge:

   a) At the navigation packet update past the IPU clock nudge, determine relationship ACC_OFFSET between ACC record count ACC_count and OBDH time according to

   $$ACC\_OFFSET(t_{nav\_update}) = OBDH(t_{nav\_update}) - ACC\_count(t_{nav\_update})$$

   b) From IPU nudge time till navigation packet update the OBDH time to receiver time mapping OBDH2RCVR(t) is then described by:

   $$OBDH2RCVR(t) = OBDH(t) - ACC\_count(t) - ACC\_OFFSET(t_{nav\_update})$$

   where $t_{ipu\_nudge} < t \leq t_{nav\_update}$

   c) Prior to the IPU clock nudge time t_{ipu_nudge} the IPU 1PPS offset is shifted by the nudge size Δt_{nudge} which is reported in the ILG_1A IPU log file. There fore the OBDH time to receiver mapping prior to t_{ipu_nudge} is described by:

   $$OBDH2RCVR(t) = OBDH(t) - ACC\_count(t) - ACC\_OFFSET(t_{nav\_update}) + \Delta t_{nudge}$$

   where $t_{ipu\_restart} < t \leq t_{ipu\_nudge}$ and $t_{ipu\_restart}$ denotes the time tag for which a time gap occurs looking backwards in time starting from t_{ipu_nudge}. Nominally this time represents the start of the 1PPS after an IPU reboot or after the automatic ADC reset sent to the ACC after about 40 seconds of the 1PPS

Write OBDH time to Receiver time, ACC count and TIM_1A flags to TIM_1B output file.

7.4 clk1b2uso1b

`clk1b2uso1b` calculates daily average GRACE carrier frequencies from CLK_1B clock solutions, and produce USO_1B data.

7.4.1 Input Files:

CLK_1B data file containing GRACE clock corrections; the file name is defined by `–clk1b` flag

7.4.2 Option Flags:

`-uso1b`	optional USO_1B output file name
`-version`	version number to be used in the output USO_1B filename

7.4.3 Output Files:

USO_1B data file containing GRACE USO, K and Ka band frequency estimates(cycles per coordinate second) based on clock drift estimates from the GIPSY Precision Orbit Determination process

7.4.4 Algorithm:

1. For each continuous clock validity interval as determined by **tdp1a2clk1b** compute the current USO frequency $f(t)$ according to:

$$f(t) = \frac{f_0}{1 + clock\_drift}$$

where f_0 is the nominal specified frequency for each GRACE and *clock_drift* is the clock drift estimated by **tdp1a2clk1b**

f_0 for GRACE A = 4832000 Hz

f_0 for GRACE B = 4832099 Hz

2. For each continuous clock validity interval compute the K (f_K) and Ka (f_{Ka}) frequencies according to:

$$f_K(t) = 5076 \cdot f(t)$$

$$f_{Ka}(t) = 6768 \cdot f(t)$$

3. Write out USO_1B record to USO_1B output file which includes the USO,K, Ka frequencies and validity interval flags.

Appendix A: Filling Data Gaps

Short data gaps can be filled with interpolation over surrounding data points. This will preserve the low-frequency information of the data and also allow digital filtering of the data without having to skip over data gaps. Note that an unfilled single 0.1-sec data dropout would result in a filtered data gap wider than T_f using a filter with a fit interval of T_f.

Let the number of missing data points in the gap be N_G; the number of available continuous data points to the left of the gap be N_1, with data values $\phi_{-N_1}, \ldots, \phi_{-2}, \phi_{-1}$; and the number of available continuous data points to the right of the gap be N_2, with data values $\phi_{+1}, \phi_{+2}, \ldots, \phi_{+N_2}$. Then the interpolation scheme is

a) linear if $N_1 = 1$ or $N_2 = 1$; and $N_G \leq 21$

Only the two data points ϕ_{-1} and ϕ_{+1} to the immediate left and right of the data gap are included for the interpolation. The interpolated data values are given by

$$\phi_n = \frac{1}{N+1}(N+1-n)\ \phi_{-1} + n\ \phi_{+1} \quad ; \quad n = 1, 2, \ldots, N_G$$

b) quadratic if $N_1 = 2$ and $N_2 \geq 2$, or $N_2 = 2$ and $N_1 \geq 2$; and $N_G \leq 21$

Only four data points (two on each side of the gap) are included for the interpolation. The interpolated values are given by

$$\phi_n = C_0 + C_1\Delta_n + \frac{1}{2}C_2\Delta_n^2 \quad ; \quad \Delta_n = n - \frac{N_G+1}{2} \quad ; \quad n = 1, 2, \ldots, N_G$$

where

$$C_0 = \frac{1}{8(N_G+2)}\left[(N_G+3)^2(\phi_{-1}+\phi_{+1}) - (N_G+1)^2(\phi_{-2}+\phi_{+2})\right]$$

$$C_1 = \frac{1}{2(N_G+1)(N_G+3)+4}\left[(N_G+1)(\phi_{+1}-\phi_{-1}) + (N_G+3)(\phi_{+2}-\phi_{-2})\right]$$

$$C_2 = \frac{1}{N_G+2}\left[(\phi_{-2}+\phi_{+2}) - (\phi_{-1}+\phi_{+1})\right]$$

c) cubic if $N_1 > 2$ and $N_2 > 2$

Up to 100 (KBR) or 200 (ACC) continuous data points on each side of the gap are included for the interpolation. The regular least-squares cubic interpolation algorithm is used.

Note that a linear or a quadratic interpolation applies only to filling small gaps ($N_G \leq 21$) in KBR data. Wider (KBR and ACC) data gaps are filled with a cubic interpolation.

- This page intentionally blank -

Appendix B: CRN-Class Digital Filter

A comprehensive description of digital filters has been given in Ref. 1. In the following, the algorithm and the key parameters needed to perform a CRN-class digital filter for the processing of KBR and ACC data are described.

The parameters dictating the CRN filter are

f_s = raw data rate = 10 samples/sec

N_c = self convolution number (odd integer) = 7

T_f = fit interval = 70.7 sec for KBR data
= 140.7 sec for ACC data

B = target low-pass bandwidth ~ 0.1 Hz for KBR data
~ 0.035 Hz for ACC data

f_0 = dominant (J2) signal frequency = 0.37×10^{-3} Hz

$N_B = B\,T_f$ = the number of frequency bins in the passband

$N_f = f_s\,T_f$ = number of raw data points in the fit interval (odd integer)

The filtered data R_i^{out} at the ith time point can be expressed as the weighted sum of N_f raw data points R^{raw} within $\pm N_h$ time intervals:

$$R_i^{out} = \sum_{n=-N_h}^{N_h} F_n\, R_{i-n}^{raw}$$

where $N_h = (N_f - 1) / 2$ and the weighting function is

$$F_n = \frac{1}{F^{Norm}} \sum_{k=-N_h}^{N_h} H_k \cos\left(\frac{2\pi kn}{N_f}\right) \quad , \quad \text{for } |\,n\,| \le N_h$$

with

$$H_k = \sum_{m=-N_B}^{N_B} \left(\frac{\sin\left[\pi(k-m)/N_c\right]}{\sin\left[\pi(k-m)/N_f\right]} \right)^{N_c}$$

and the normalizing factor

$$F^{Norm} = \sum_{i=-N_h}^{N_h} \left[\cos\left(\frac{2\pi f_0 i}{f_s}\right) \sum_{k=-N_h}^{N_h} H_k \cos\left(\frac{2\pi kn}{N_f}\right) \right]$$

The first and second time derivatives of R_i^{out} are computed by the same algorithm except that the weighting function F_n is replaced by its time derivatives:

$$\dot{R}_i^{out} = \sum_{i=-N_h}^{N_h} \dot{F}_n \, R_{i-n}^{raw} \quad ; \quad \dot{F}_n = \frac{1}{F^{Norm}} \sum_{k=-N_h}^{N_h} -(2\pi k / T_f) \, H_k \, \sin\left(\frac{2\pi kn}{N_f}\right)$$

and

$$\ddot{R}_i^{out} = \sum_{i=-N_h}^{N_h} \ddot{F}_n \, R_{i-n}^{raw} \quad ; \quad \ddot{F}_n = \frac{1}{F^{Norm}} \sum_{k=-N_h}^{N_h} -(2\pi k / T_f)^2 \, H_k \, \cos\left(\frac{2\pi kn}{N_f}\right)$$

with H_k and F^{Norm} being the same as for R_i^{out}.

Note that once F_n and its derivatives are computed, they are applicable to all data time points without the need to be re-computed.

Appendix C: Resampling of Data with Lagrange Interpolation

Let T_{out} be the corrected timetag at which the resampled observable is to be computed. The N raw observables to be used in the interpolation process have uncorrected timetags at $T_0, T_1, \ldots, T_N$ such that

$$T_0 + C_0 < T_{out} < T_N + C_N$$

where C_i is the timetag corrections at time T_i for i = 0, 1, … , N with N = 2 for a quadratic and N = 3 for a cubic interpolation.

The interpolated observable $K(T_{out})$ at the corrected timetag T_{out} is efficiently computed in Lagrange form, which gives the weighted sum of the raw observables $K(T_i)$:

$$K(T_{out}) = \sum_{i=0}^{N} l_i \, K(T_i)$$

where the weights l_i are the Lagrange interpolation coefficients

$$l_i = \frac{\prod_{\substack{j=0 \\ j \neq i}}^{N} \left(T_{out} - (T_j + C_j)\right)}{\prod_{\substack{j=0 \\ j \neq i}}^{N} \left(T_i - T_j + C_i - C_j\right)}$$

For KBR_1A data, $T_0, T_1, \ldots, T_N$ and T_{out} are all integer multiples of 0.1 sec. To reduce round-off error, which would be significant for the high-frequency phases, integer arithmetic is used instead of taking explicit time differences in the interpolation process.

Let $\Delta_{ij} = T_i - T_j$ and $\Delta_i = T_{out} - T_i$. Then the above Lagrange interpolation coefficients can be written as

$$l_i = \frac{\prod_{\substack{j=0 \\ j \neq i}}^{N} \left(\Delta_i + C_i\right)}{\prod_{\substack{j=0 \\ j \neq i}}^{N} \left(\Delta_{ij} + C_i - C_j\right)}$$

Since C_i are small and Δ_i , Δ_{ij} are small (typically within ±2) integer multiples of 0.1, the coefficients l_i and, hence, $K(T_{out})$ can be computed with high precision.

Appendix D: KBR Data Combinations

Ionosphere-free dual one-way range (DOWR) data are formed by proper combinations of KBR data between the two GRACE spacecraft and between K and Ka bands, as has been described in detail in Ref. 2. In the following, only the equations (Eqs. 4.31, 2.19 and 2.20 of Ref. 2) to be applied in the algorithm are given.

Let

f_A^K = K-band carrier frequency transmitted by GRACE A

f_B^K = K-band carrier frequency transmitted by GRACE B

$\phi_{A,B}^K$ = K-band phase transmitted by GRACE A and received by GRACE B

$\phi_{B,A}^K$ = K-band phase transmitted by GRACE B and received by GRACE A

The DOWR for K-band phases is formed by

$$R_K = c\left(\frac{\phi_{A,B}^K + \phi_{B,A}^K}{f_A^K + f_B^K}\right)$$

where c is the speed of light = 299,792,458 m/sec.

The DOWR for Ka-band phases, R_{Ka}, is formed in the same way with K replaced by Ka in the above equation and parameter definitions.

The ionosphere-free DOWR is formed by the linear combination

$$DOWR = C_{Ka}R_{Ka} - C_K R_K$$

with the coefficients

$$C_{Ka} = \frac{f_A^{Ka}\ f_B^{Ka}}{f_A^{Ka}\ f_B^{Ka} - f_A^K\ f_B^K} = 16/7$$

$$C_K = \frac{f_A^K\ f_B^K}{f_A^{Ka}\ f_B^{Ka} - f_A^K\ f_B^K} = 9/7$$

The Ka-band ionospheric delay is

$$ION_{Ka} = DOWR - R_{Ka}$$

Appendix E: Light-Time Correction for DOWR

The light-time correction that has to be applied on DOWR has been described in Ref. 2 (Eq. 4.39) is

$$LTC_K = \frac{1}{f_A^K + f_B^K}\left[f_A^K \dot{\rho}\tau_B^A - f_A^K \eta_B(\tau_A^B - \tau_B^A) + (f_A^K - f_B^K)\eta_B \tau_A^B\right]$$

for K-band (and similarly for Ka band) where the frequencies f's are as defined in Appendix D and

$\dot{\rho}$ = inter spacecraft range rate

η_B = GRACE B velocity component along the line-of-sight vector

τ_B^A = light-time of signal from GRACE A to GRACE B

τ_A^B = light-time of signal from GRACE B to GRACE A

While $\dot{\rho}$ and η_B can be calculated from orbit positions and velocities of the two GRACE spacecraft, the light-times τ_B^A and τ_A^B can be precisely calculated only with an iteration process, as in the following:

Initial values of τ_B^A and τ_A^B are estimated using the spacecraft positions at the *receiving* time, which are recorded in the ECI files. In each iteration step, the position of each spacecraft at the *transmitting* time is calculated from its position and velocity at the receiving time with the light-time from the previous iteration step. The light-time is then updated with the newly estimated position at the transmitting time. In practice, only 2 iterations are needed for convergence.

The light-time corrections LTC_K and LTC_{Ka} for K and Ka band corrections are then combined (Eq. 4.43 of Ref. 2) to form the ionosphere-free LTC:

$$LTC = C_{Ka}(LTC_{Ka}) - C_K(LTC_K)$$

where the coefficients C_K and C_{Ka} are as defined in Appendix D.

Appendix F: Quaternion Operations

A quaternion Q rotating a coordinate frame A into a coordinate frame B is defined by four elements:

$$Q = (q_0 \quad q_1 \quad q_2 \quad q_3)$$

where

$$q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$$

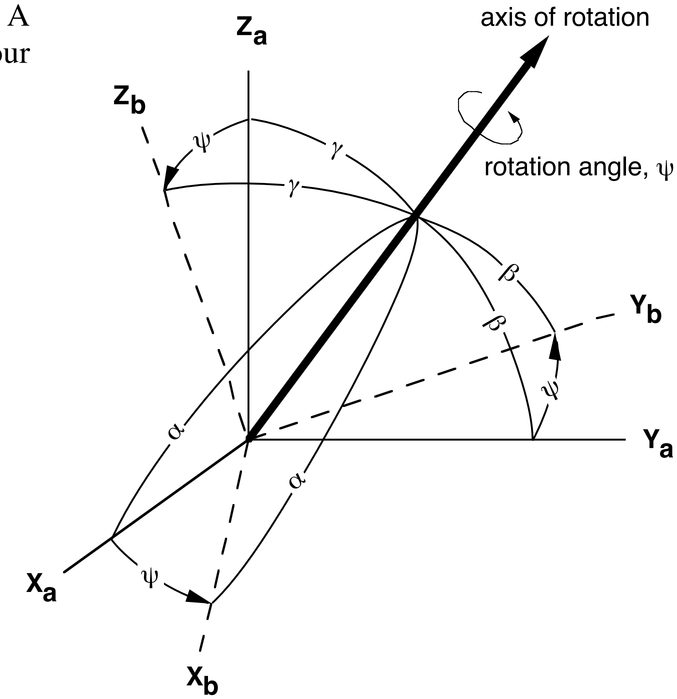
and

$$q_0 = \cos(\psi/2)$$

$$q_1 = \sin(\psi/2)\cos\alpha$$

$$q_2 = \sin(\psi/2)\cos\beta$$

$$q_3 = \sin(\psi/2)\cos\gamma$$



with ψ being the angle of rotation; and α, β and γ the angles between the rotation axis and the X_a , Y_a , Z_a axes (also X_b , Y_b , Z_b axes), respectively, as shown in the above figure.

Rotation of a vector by a quaternion

When a quaternion $Q_{A,B}$ is *operated* on a vector $\mathbf{V}_A$ in coordinate frame A it will rotate $\mathbf{V}_A$ into a vector $\mathbf{V}_B$ in coordinate frame B:

$$\mathbf{V}_B = Q_{A,B}\,\mathbf{V}_A$$

The operation is equivalent to pre-*multiplying* the vector $\mathbf{V}_A$ by a rotation matrix $M_{A,B}$:

$$\mathbf{V}_B = M_{A,B}\,\mathbf{V}_A$$

The rotation matrix $M_{A,B}$ is said to be *corresponding* to the quaternion $Q_{A,B}$.

The rotation of a vector **V** by a quaternion Q is calculated in the following way. Define the "vector" part of a quaternion Q as

$$\mathbf{q} = (q_1 \quad q_2 \quad q_3)$$

such that the quaternion

$$Q = (q_0 \quad q_1 \quad q_2 \quad q_3) = (q_0 \quad \mathbf{q})$$

Then the rotation of **V** by the quaternion Q is

$$Q\ \mathbf{V} = (2q_0^2 - 1)\ \mathbf{V} + (2\mathbf{q} \bullet \mathbf{V})\ \mathbf{q} - 2q_0\ (\mathbf{q} \times \mathbf{V})$$

Product of two quaternions

Let $Q_{A,B}$ be a quaternion rotating from frame A to frame B, and $P_{B,C}$ a quaternion rotating from frame B to frame C. When a vector **V** in frame A is first operated by $Q_{A,B}$, and then operated by $P_{B,C}$ is said to be operated by the product of quaternion $Q_{A,B}\ P_{B,C}$, which is a quaternion $S_{A,C}$ rotating from frame A to frame C:

$$P_{B,C}\left(Q_{A,B}\mathbf{V}\right) = \left(Q_{A,B}\ \ P_{B,C}\right)\mathbf{V} = S_{A,C}\mathbf{V}$$

The elements of the product of two quaternions S = QP are calculated by

$$\begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix} = \begin{pmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{pmatrix} \begin{pmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{pmatrix} = \begin{pmatrix} p_0 & -p_1 & -p_2 & -p_3 \\ p_1 & p_0 & p_3 & -p_2 \\ p_2 & -p_3 & p_0 & p_1 \\ p_3 & p_2 & -p_1 & p_0 \end{pmatrix} \begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{pmatrix}$$

Inverse of a quaternion

The inverse of a quaternion will rotate a vector by the same but negative rotation angle. Hence

$$Q^{-1} = \left(q_0\ \ q_1\ \ q_2\ \ q_3\right)^{-1} = \left(q_0\ \ -q_1\ \ -q_2\ \ -q_3\right) = \left(-q_0\ \ q_1\ \ q_2\ \ q_3\right)$$

Difference of two quaternions

Let $Q_{A,B}$ and $P_{A,B}$ be quaternions rotating from frame A to frame B. Then the "difference" of $P_{A,B}$ from $Q_{A,B}$, expressed in the coordinate frame B, is

$$\Delta_{A,B} = (Q_{A,B})^{-1}\ P_{A,B}$$

Rotation angles of a quaternion

The rotation represented by a quaternion can be decomposed into 3 components: the rotations wrt. Z-, Y-, and X-axis, *in that order*. The respective rotation angles, ψ, θ and ϕ are related to the quaternion components by

$$\psi = \tan^{-1}\left[\frac{2(q_1q_2 + q_0q_3)}{q_0^2 + q_1^2 - q_2^2 - q_3^2}\right],\quad \begin{cases} \text{if} \quad q_0^2 + q_1^2 > q_2^2 + q_3^2 \quad , & 0 < \varphi < \pi/2 \quad \text{or} \quad 3\pi/2 < \varphi < 2\pi \\ \text{if} \quad q_0^2 + q_1^2 < q_2^2 + q_3^2 \quad , & \pi/2 < \varphi < 3\pi/2 \end{cases}$$

$$\theta = \sin^{-1}\left[2(q_0 q_2 - q_1 q_3)\right] \quad , \qquad -\pi/2 < \theta < \pi/2$$

$$\phi = \tan^{-1}\left[\frac{2(q_0 q_1 + q_2 q_3)}{q_0^2 - q_1^2 - q_2^2 + q_3^2}\right], \quad \begin{cases} \text{if} \quad q_0^2 + q_3^2 > q_1^2 + q_2^2 \quad , \quad 0 < \phi < \pi/2 \quad \text{or} \quad 3\pi/2 < \phi < 2\pi \\ \text{if} \quad q_0^2 + q_3^2 < q_1^2 + q_2^2 \quad , \quad \pi/2 < \phi < 3\pi/2 \end{cases}$$

Rotation matrix *R* corresponding to a quaternion *Q*

The rotation matrix R corresponding to a quaternion

$$Q = (q_0 \quad q_1 \quad q_2 \quad q_3)$$

can be expressed in terms of the quaternion components:

$$R = \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} \\ r_{2,1} & r_{2,2} & r_{2,3} \\ r_{3,1} & r_{3,2} & r_{3,3} \end{bmatrix} = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1 q_2 + q_0 q_3) & 2(q_1 q_3 - q_0 q_2) \\ 2(q_1 q_2 - q_0 q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2 q_3 + q_0 q_1) \\ 2(q_1 q_3 + q_0 q_2) & 2(q_2 q_3 - q_0 q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}$$

Quaternion *Q* corresponding to a rotation matrix *R*

The quaternion Q corresponding to a rotation matrix

$$R = \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} \\ r_{2,1} & r_{2,2} & r_{2,3} \\ r_{3,1} & r_{3,2} & r_{3,3} \end{bmatrix}$$

can be expressed in terms of the matrix elements:

$$Q = (q_0 \quad q_1 \quad q_2 \quad q_3)$$

with

$$q_0 = \pm\sqrt{1 + r_{1,1} + r_{2,2} + r_{3,3}}\,/2$$

$$q_1 = \left(r_{2,3} - r_{3,2}\right)/4q_0$$

$$q_2 = \left(r_{3,1} - r_{1,3}\right)/4q_0$$

$$q_3 = \left(r_{1,2} - r_{2,1}\right)/4q_0$$

Appendix G: Time Definitions Used in GRACE Data

OBDH Time

OBDH time is derived from the On Board Data Handler (OBDH) 1 Pulse Per Second (1PPS) generator, which is used for time tagging the ACC and onboard Housekeeping (HK) data. This time is normally synchronized with IPU receiver time when the IPU 1 PPS is present. The absolute time transfer occurs every 10 seconds when an IPU navigation solution is sent to the OBDH. During an IPU reboot, however, the OBDH time and Receiver time are not synchronized. During the reboot process the IPU 1PPS is restarted and synchronized with GPS time, causing the OBDH to re-sync its 1PPS with the IPU 1PPS. Each re-synchronization takes about 2-3 seconds, which are not accounted for in the OBDH counter. Only after the two re-synchronizations and an IPU navigation solution the OBDH time and Receiver time are synchronized again. In general this process takes about 3 minutes. In order to properly time tag the ACC and HK data during this period a correction needs to be applied, which is described in function **tim1a2tim1b**.

Receiver Time

Receiver time is derived from the IPU 1PPS generator, which is synchronized with GPS time realization at the IPU reboot time. After the IPU reboot time, the 1PPS can freely drift until the next IPU reboot. This time is used for time tagging the onboard KBR, SCA and GPS data.

GPS Time

GPS time is the *corrected* Receiver time used in the all Level-1B data. The time corrections, which are recorded in CLK_1B data file, are based on GPS time realization solutions using GPS data from GRACE spacecraft as well as ground GPS tracking sites. It is realized by first solving for errors in the GPS constellation clocks using approximately 80 ground stations relative to a GPS receiver connected to a ground reference clock. The ground reference clock is a highly stable ground reference, typically the alternative US master clock in Colorado Springs (AMC2) Ref. 5.

Appendix H: Inertial to SRF Quaternion Based on GRACE Ephemeris

The reference attitude quaternions (Inertial to SRF) can be computed in two ways:

1) Using the inertial position and velocity vector if one GRACE spacecraft is available
2) Using the inertial position and the relative inertial position vector between the two GRACE spacecraft

Let

$\vec{R}_i$ = Inertial position vector for GRACE *i*
$\dot{\vec{R}}_i$ = Inertial velocity vector for GRACE *i*
$\vec{X}_i$ = Inertial unit vector of the SRF X-axis for GRACE *i*
$\vec{Y}_i$ = Inertial unit vector of the SRF Y-axis for GRACE *i*
$\vec{Z}_i$ = Inertial unit vector of the SRF Z-axis for GRACE *i*

In case 1) the unit vectors of the SRF axis are calculated according to:

$$\vec{Z}_i = -\vec{R}_i / \left|\vec{R}_i\right|$$

$$\vec{Y}_i = (\vec{Z}_i \times \dot{\vec{R}}_i) / \left|\vec{Z}_i \times \dot{\vec{R}}_i\right|$$

$$\vec{X}_i = (\vec{Y}_i \times \vec{Z}_i) / \left|\vec{Y}_i \times \vec{Z}_i\right|$$

The SRF unit vectors are then converted into quaternions by forming first the rotation matrix A according to:

$$A = [\vec{X}_i \quad \vec{Y}_i \quad \vec{Z}_i]$$

The rotation matrix is then converted to a quaternion according to:

$$q_0 = \sqrt{1 + A_{1,1} + A_{2,2} + A_{3,3}} / 2$$

$$q_1 = -\left(A_{2,3} - A_{3,2}\right) / 4q_0$$

$$q_2 = -\left(A_{3,1} - A_{1,3}\right) / 4q_0$$

$$q_3 = -\left(A_{1,2} - A_{2,1}\right) / 4q_0$$

For case 2) the velocity vector $\dot{\vec{R}}_i$ is replaced with the relative position vector $\vec{R}_{ji} = \vec{R}_j - \vec{R}_i$. The calculation of the reference quaternion is then identical as described above.

Appendix I: Unifying Relative Wrapping

The onboard K- and Ka-band KBR phases have secular trends of ±~500 kHz and ±670 kHz respectively. To preserve data precision, 10^8-cycle wrappings are applied on board to raw K- and Ka-band phases so that their values are always within $\pm 10^8$ cycles. The number of wrappings generally increased by ±1 every 100 to 200 sec. When the phase data are processed, the relative wrappings need to be unified in the following way:

Let ϕ_n be the phase at time point T_n, for n = 0, 1, 2, … where T_0 is the first time point in the current data arc being processed. The following process is carried out for all n involved in the current data arc, starting from T_0:

if $(\phi_n - \phi_{n-1}) / 10^8 < 0.5$, 10^8 is added to ϕ_n

if $(\phi_n - \phi_{n-1}) / 10^8 > 0.5$, 10^8 is subtracted from ϕ_n

and the process is iterated until the condition

$$-0.5 \le (\phi_n - \phi_{n-1})/10^8 \le 0.5$$

is satisfied.

KBR data arcs involved in phase break checking, filling short data gaps and timetag correction are typically short (< 2.1 sec) and the number of relative wrapping is generally within ±1 and no iteration of the above steps is needed. The added ±1 wrapping is only temporary; after the processing, the added ±1 wrapping is removed and the data will remain wrapped to maintain the precision.

KBR data involved in the digital filtering are the combined data between the two GRACE spacecraft. Hence, the number of relative wrappings of concern is that of the combined phases, which have the secular trend nearly canceled out. Hence the net number of wrappings is generally within ±1 over the entire day. The added ±1 wrapping is not removed after the combining and digital filtering processes so that the resulting data product will be wrap-free.

Appendix J: SCA Data Combination from Two Star Cameras

Each GRACE spacecraft is equipped with two star cameras, each pointing in a nominal direction 45° from body-fixed Z (upward) axis and perpendicular to the X (forward) axis. Each star camera provides a measurement of rotation (in the form of a quaternion) from inertial frame to the camera frame, with a larger uncertainty (by a factor of ~8) in the rotation wrt. the boresight than those wrt. the perpendicular directions. A detailed derivation of the scheme for the optimal combination of the attitude measurements from the two star cameras has been given in Ref. 3. In the following, only the equations used in the combining algorithm are described.

Let

$Q_{a,S}$ = fixed quaternion rotating camera frame "a" to SRF

$Q_{b,S}$ = fixed quaternion rotating camera frame "b" to SRF

$Q_{I,a}$ = measured quaternion rotating inertial frame to camera frame "a"

$Q_{I,b}$ = measured quaternion rotating inertial frame to camera frame "b"

First, rotate the measured quaternions from both star cameras into [inertial to SRF] quaternions:

$$Q_{I,S}^{(a)} = Q_{I,a} Q_{a,S}$$

$$Q_{I,S}^{(b)} = Q_{I,b} Q_{b,S}$$

Next, determine the "difference" (Appendix F), in SRF, of the above two quaternions:

$$D = \left(Q_{I,S}^{(a)}\right)^{-1} Q_{I,S}^{(b)} = (1,\ \Delta_{ab})$$

where Δ_{ab} is the "vector" part of the quaternion D, with each of the three elements << 1.

The optimally combined quaternion is then given by

$$Q_{I,S}^{opt} = Q_{I,S}^{(a)}\ (1,\ M\Delta_{ab})$$

where the matrix M is

$$M = \frac{1}{2}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -\lambda \\ 0 & -\lambda & 1 \end{bmatrix} \quad ; \quad \lambda = \frac{\kappa^2 - 1}{\kappa^2 + 1}$$

with κ = 8 being the scaling factor for the larger uncertainty in rotation wrt. the boresight.

Acronyms:

ACC	Accelerometer (data file)
CA	observables derived from Clear Acquisition (Coarse Accuracy) GPS signal at L_1 frequency
CAL	Calibration data file
CLK	Clock correction data file
CRN	A class of digital filter characterized by an **N**-th order self-**C**onvolution of **R**ectangular time-domain window function
DOWR	Dual One Way Range
ECI	Earth Centered Inertial orbit file
GIPSY	GPS Inferred Positioning System of JPL
GPI	Interim GPS data file
GPS	Global Positioning System (data file)
GRACE	Gravity Recovery And Climate Experiment
IPU	Instrument Processing Unit
ITRF2000	International Terrestrial Reference Frame with epoch 2000
KBR	K (and Ka) Band Range (data file)
L1	observables derived from GPS signal at L_1 frequency
L2	observables derived from GPS signal at L_2 frequency
LTC	Light-Time Correction
OBDH	On Board Data Handler
PCI	Phase Center correction file
PPS	Pulse Per Second
QKS	Quaternion rotating [from Star camera frame to KBR bore sight frame] recorded in SOE file
QSA	Quaternion rotating [from Star camera frame to SRF] recorded in SOE file

SCA	Star Camera (quaternion data file)
SOE	Sequence Of Events (data file)
SRF	Science Reference Frame
tdp	time dependent parameters (GIPSY data tile)
VKB	antenna phase center offset vector recorded in SOE file

References:

1. J. B. Thomas, “An Analysis of Gravity-Field Estimation Based on Intersatellite Dual-1-Way Biased Ranging,” JPL Publication 98-15, May 1999.
2. J. Kim, “Simulation Study of a Low-Low Satellite-to-Satellite Tracking Mission,” Ph.D. dissertation, Univ. of Texas at Austin, May 2000.
3. L J. Romans, “Optimal Combination of Quaternions from Multiple Star Cameras,” JPL Interoffice Memorandum, May 2003.
4. S. Bettadpur, GRACE Product Specification Document, GRACE 327-720.
5. Willy Bertiger, Charley Dunn, Ian Harris, Gerard Kruizinga, Larry Romans, Mike Watkins, and Sien Wu, "Relative Time And Frequency Alignment Between Two Low Earth Orbiters, Grace," Proceedings of IEEE, FCS, Tampa FL, May, 2003.

These references are available at:

http://podaac.jpl.nasa.gov/grace/documentation.html