



Gravity Recovery Object Oriented Programming System

## Documentation

GitHub repository: <https://github.com/groops-devs/groops>

# Contents

<b>1 General information</b>	<b>7</b>
1.1 Config files . . . . .	7
1.2 Parsers and variables . . . . .	10
1.3 Loops and conditions . . . . .	13
1.4 Constants and the settings file . . . . .	14
1.5 Parallelization . . . . .	16
1.6 Graphical User Interface (GUI) . . . . .	17
1.7 File formats . . . . .	20
<b>2 Mathematical fundamentals</b>	<b>34</b>
2.1 Robust least squares adjustment . . . . .	34
2.2 Basis splines . . . . .	35
2.3 Autoregressive Models . . . . .	37
<b>3 Cookbook</b>	<b>38</b>
3.1 Instrument data handling . . . . .	38
3.2 GNSS satellite orbit determination and station network analysis . . . . .	44
3.3 GNSS precise point positioning (PPP) . . . . .	50
3.4 Kinematic orbit determination of LEO satellites . . . . .	52
3.5 Gravity field determination from precise orbit data (POD) . . . . .	55
3.6 GRACE gravity field recovery . . . . .	57
3.7 Regional geoid determination . . . . .	63

<b>4 Programs</b>	<b>68</b>
4.1 Programs: Covariance . . . . .	68
4.2 Programs: DoodsonHarmonics . . . . .	74
4.3 Programs: Gnss . . . . .	82
4.4 Programs: Grace . . . . .	105
4.5 Programs: Gravityfield . . . . .	115
4.6 Programs: Grid . . . . .	135
4.7 Programs: Instrument . . . . .	155
4.8 Programs: KalmanFilter . . . . .	197
4.9 Programs: Misc . . . . .	201
4.10 Programs: NormalEquation . . . . .	229
4.11 Programs: Orbit . . . . .	244
4.12 Programs: Plot . . . . .	254
4.13 Programs: Preprocessing . . . . .	262
4.14 Programs: Simulation . . . . .	278
4.15 Programs: System . . . . .	297
4.16 Programs: Conversion . . . . .	306
4.17 Programs: Deprecated . . . . .	401
<b>5 Classes</b>	<b>406</b>
5.1 AutoregressiveModelSequence . . . . .	406
5.2 Border . . . . .	408
5.3 Condition . . . . .	410
5.4 CovariancePod . . . . .	412
5.5 CovarianceSst . . . . .	413
5.6 DigitalFilter . . . . .	414
5.7 Doodson . . . . .	421
5.8 EarthRotation . . . . .	424
5.9 Eclipse . . . . .	427
5.10 Ephemerides . . . . .	428
5.11 JPL . . . . .	428
5.12 Forces . . . . .	429
5.13 GnssAntennaDefintionList . . . . .	430

5.14 GnssParametrization . . . . .	432
5.15 GnssProcessingStep . . . . .	445
5.16 GnssReceiverGenerator . . . . .	452
5.17 GnssTransmitterGenerator . . . . .	456
5.18 GnssType . . . . .	457
5.19 Gravityfield . . . . .	458
5.20 Grid . . . . .	463
5.21 InstrumentType . . . . .	469
5.22 InterpolatorTimeSeries . . . . .	470
5.23 Kernel . . . . .	472
5.24 Loop . . . . .	477
5.25 Magnetosphere . . . . .	482
5.26 MatrixGenerator . . . . .	483
5.27 MiscAccelerations . . . . .	488
5.28 NoiseGenerator . . . . .	491
5.29 NormalEquation . . . . .	492
5.30 Observation . . . . .	495
5.31 OrbitPropagator . . . . .	508
5.32 ParameterNames . . . . .	510
5.33 ParameterSelector . . . . .	513
5.34 ParametrizationAcceleration . . . . .	515
5.35 ParametrizationGnssAntenna . . . . .	518
5.36 ParametrizationGravity . . . . .	520
5.37 ParametrizationSatelliteTracking . . . . .	523
5.38 ParametrizationTemporal . . . . .	526
5.39 Planet . . . . .	529
5.40 PlatformSelector . . . . .	530
5.41 PlotAxis . . . . .	532
5.42 PlotColor . . . . .	535
5.43 PlotColorbar . . . . .	536
5.44 PlotGraphLayer . . . . .	537
5.45 PlotLegend . . . . .	540
5.46 PlotLine . . . . .	541

5.47 PlotMapLayer . . . . .	542
5.48 PlotMapProjection . . . . .	547
5.49 PlotSymbol . . . . .	549
5.50 PodRightSide . . . . .	550
5.51 SggRightSide . . . . .	551
5.52 SphericalHarmonicsFilter . . . . .	552
5.53 SphericalHarmonicsNumbering . . . . .	553
5.54 SstRightSide . . . . .	554
5.55 Thermosphere . . . . .	555
5.56 Tides . . . . .	556
5.57 TimeSeries . . . . .	560
5.58 Troposphere . . . . .	563



# Chapter 1

## General information

The Gravity Recovery Object Oriented Programming System (GROOPS) is a software toolkit written in C++ that enables the user to perform core geodetic tasks. Key features of the software include gravity field recovery from satellite and terrestrial data, the determination of satellite orbits from global navigation satellite system (GNSS) measurements, and the processing of GNSS constellations and ground station networks. Most tasks and algorithms are (optionally) parallelized through the Message Passing Interface (MPI), thus the software enables a smooth transition from single-CPU desktop computers to large distributed computing environments for resource intensive tasks.

Parts of GROOPS originate from developments in the Astronomical, Physical and Mathematical Geodesy Group at the University of Bonn, Germany. Since 2010 it is developed and maintained at Graz University of Technology, Austria. It is licensed under GPLv3 and hosted at GitHub, <https://github.com/groops-devs/groops>.

If you use data sets computed with GROOPS in a publication or publish the data itself, please cite our reference paper:

Mayer-Guerr, T., Behzadpour, S., Eicker, A., Ellmer, M., Koch, B., Krauss, S., Pock, C., Rieser, D., Strasser, S., Suesser-Rechberger, B., Zehentner, N., Kvas, A. (2021). GROOPS: A software toolkit for gravity field recovery and GNSS processing. Computers and Geosciences, 104864. <https://doi.org/10.1016/j.cageo.2021.104864>.

GROOPS depends on data files such as Earth rotation, Love numbers, and GNSS meta information. An initial data set that is regularly updated is available on our FTP server, <https://ftp.tugraz.at/outgoing/ITSG/groops>. You can choose between downloading the data directory or a single `data.zip` with the same content.

### 1.1 Config files

GROOPS is controlled by XML configuration files. One or more configuration files must be passed as arguments to GROOPS:

```
groops config1.xml config2.xml [...]
```

These files can be created with the graphical user interface program `groopsGui` in a convenient way (see section [GUI \(1.6\)](#)). A complete formal (computer readable) description of a configuration file in the form of an XSD schema file can be created with the command

```
groops --xsd groops.xsd
```

A configuration file consists of a list of [programs \(4\)](#) that are executed in sequential order. Each program comes with its own config options and they work independently without any internal communication between programs. Data flow between programs is realized via files. An  **outputfile** of one program can serve as an  **inputfile** for the next program. Most programs are deliberately kept small and focused on a specific task. This modularity combined with the general purpose design of many programs enables the creation of complex workflows with little effort. Including [loops and conditions \(1.3\)](#) in a config file provides even more flexibility.

Individual programs (and also other optional config elements) can be disabled and are ignored during execution. Mandatory config elements are indicated by a star (\*). Empty optional elements are ignored or a meaningful default value is assumed.

The elements of a configuration file can be one of the following basic data types:

- **int**: integer number
- **uint**: unsigned integer number
- **double**: floating point number
- **angle**: given in degree
- **time**: given in modified Julian date (MJD)
- **boolean**: 0: false, 1: true
- **string**: text
- **filename**: absolute path to a file or path relative to the working directory
- **expression**: numerical expression evaluated during execution
- **doodson**: Doodson number or Darwin's name of a tidal frequency
- **gnssType**: GNSS observation type according to the RINEX 3 definition

The first 5 data types also allow numerical expressions as input in addition to pure numbers. In addition to these basic types, there are a large number of complex data types called classes, which are described in section [Classes \(5\)](#).

### 1.1.1 Variables

In addition to programs a config file can also include elements called variables. These elements are comparable to read-only variables in programming and can be referenced from any program and config element. This can be done by either linking an element directly to a variable or by using the name as a variable in an expression of an input field (see section [Parsers and variables \(1.2\)](#)). While elements can only be directly linked to variables of the same type, this also supports complex data types such as  **gravityfield**. Thus it is possible to, for example, define a reference gravity field once in the global section and use it multiple times in different programs.

Variables can be declared anywhere in the configuration file. Variables in locations other than the global section have a local scope and hide global variables or variables from a hierarchy level above. They are valid after declaration until the end of the hierarchy level is reached or a new variable with the same name is declared.

Variables are not evaluated directly when they are declared, but only later when they are used in a config element. This means, for example, that a variable **satelliteFile** with

`data/swarm_orbit_{loopTime:%D}.dat` can be declared in the global section without the variable `loopTime` having to be known at this time.

One special variable is `groopsDataDir`, which is used as a variable in most default file paths throughout many GROOPS programs. Since this variable is going to be needed in most config files, it is recommended to define it in a template file that is used when creating new config files in the GUI. See section [Graphical User Interface \(GUI\) \(1.6\)](#) for details on how to set up a template file.

Global variables can be manipulated when running a config file by passing the argument `--global <name>=<value>`. For example, running the command

```
groops --global timeStart=58849 --global satellite=swarm config.xml
```

runs the config file `config.xml` but replaces the values of the global variable `timeStart` and `satellite` with `58849` and `swarm`, respectively. If a global variable passed as an argument does not already exist in the config file, it will be added with the type `string`. Only the basic data types listed above are supported. This feature can be useful when running GROOPS from the command line or from an external script file.

## 1.2 Parsers and variables

The XML configuration file is evaluated by two parsers. In a first step a text parser is applied. In the second step mathematical expressions are resolved to a number. Variables (see section [variables \(1.1.1\)](#)) can be referenced via their name directly for the expression parser or in the form `{name}` for the text parser.

### 1.2.1 Mathematical expression parser

In all input fields that accept numbers (int, uint, double, angle, time) numerical expressions are also allowed. Declared variables can be accessed via their name. The following operations and functions are defined:

- Constants: `pi()`, `rho()=180/pi()`, `nan()`, `c()`: light velocity, `G()`: gravitational constant, `GM()`: gravitational constant of the Earth, `R()`: reference radius of the Earth
- Mathematical: `+`, `-`, `*`, `/`, `^`
- Comparison: `==`, `!=`, `<`, `<=`, `>`, `>=`, result is 1 or 0
- Logical: `not !`, and `&&`, `||`, or `isnan(x)`, result is 1 or 0
- Functions: `sqrt(x)`, `exp(x)`, `sin(x)`, `cos(x)`, `tan(x)`, `asin(x)`, `acos(x)`, `atan(x)`, `abs(x)`, `round(x)`, `ceil(x)`, `floor(x)`, `deg2rad(x)`, `rad2deg(x)`
- Functions with 2 arguments: `atan2(y,x)`, `min(x,y)`, `max(x,y)`, `mod(x,y)`
- Time functions: `now()`: local time in MJD, `date2mjd(year, month, day)`, `gps2utc(mjd)`, `utc2gps(mjd)`, `dayofyear(mjd)`, `decimalyear(mjd)`
- Condition: `if(c,x,y)`: If the first argument is true (not 0), the second argument is evaluated, otherwise the third.

### 1.2.2 Text parser

Before the mathematical expression parser evaluates the expression, a simple text parser is applied. The text parser is used for all input fields (also file names). It scans the text for terms like `{variable}` and replaces it by the text content of the `variable`. A literal '`{`' character must be escaped with '`#{`'.

The text parser allows regex replacements in the form `{text/regex/replace}`. All matches of `regex` in the `text` are replaced by `replace`. Possible `variables` in the three parts are evaluated beforehand. Capturing groups `()` can be accessed by `$1`, `$2`, ... in the replacement (`$0` is the complete match). Additional escape sequences are:

- `\l` lowercase next char,
- `\u` uppercase next char,
- `\L` lowercase until `\E`,
- `\U` uppercase until `\E`,
- `\Q` quote (disable) pattern metacharacters until `\E`,
- `\E` end either case modification or quoted section.

Examples:

- `{variable}/test/text}` replaces all occurrences of `test` by `text`.
- `{TEXT/.+/\L$0}` converts text to lower case.
- `{012345/.#{2}(.#{3}).*/$1}` extracts the substring at index 2 and length 3 resulting in 234. Note the escaping `#{`.

The text parser also evaluates terms in the form `{expression:format}` and replaces it by a formatted output. In order not to get confused with the regex replacements, the `'/'` character must be escaped with `'#/'` in the expression. The `format` contains the text to be written as output. It can contain embedded format specifiers that are replaced by the value of the expression and formatted as requested (also multiple times). In the following, the resulting formatted output is given in the brackets for an expression with the example value of 57493.8:

- `%i`: Integer [57494]
- `%f`: Decimal floating point [57493.800000]
- `%e`: Scientific notation [5.749380e+04]
- `%g`: Use the shortest representation: `%e` or `%f` [57493.8]
- `%c`: Interpret number as ASCII character
- `%%`: Write a single literal `%` character

The following specifiers interpret the value of the expression as MJD (modified Julian date):

- `%y`: Four digit year [2016]
- `%Y`: Two digit year [16]
- `%m`: Month [04]
- `%d`: Day of month [15]
- `%H`: Hour [19]
- `%M`: Minute [12]
- `%S`: Second [00]
- `%D`: Date (same as `%y-%m-%d`) [2016-04-15]
- `%T`: Time (same as `%H-%M-%S`) [19-12-00]
- `%W`: GPS week [1892]
- `%w`: Day of GPS week (0..6) [5]
- `%O`: Day of year (1..366)

The format can be specified further with `%[width][.precision]specifier`, where `[width]` is the minimum number of characters to be printed. If the value to be printed is shorter than this number, the result is padded with blank spaces (or zeros if `[width]` starts with a zero). The `[.precision]` defines the number of digits after the period (for `%g` the number of significant digits instead).

Example: Two variables `time=57493+19/24+12/1440` and `satellite swarm` are set in the global section. The `inputfile=data/{time:%y}/{satellite}_{time:%D}.dat` is expanded to "data/2016/swarm\_2016-04-15.dat".

Example: The variable `x=3+5` is set in the global section. The expression `number=2*x` is evaluated by the expression parser to `=16`. In contrast if we use brackets like in `number=2*{x}` the expression is first evaluated by the text parser to `"2*3+5"` and the expression parser now gives the result `=11`.

### 1.2.3 Variables for data

Some programs (e.g. **FunctionsCalculate**, **InstrumentArcCalculate**, **GriddedDataCalculate**, or the plot programs) read data (**matrix**) or **gridded data** and evaluate input/output expressions for each data row. For these kind of expressions additional variables are automatically defined for each data column ( $X$  stands for the data column number:  $0 \dots n$ ):

- **index**: the row number, starting with zero
- **dataX**: the value itself
- **dataXcount**: number of rows
- **dataXmin**
- **dataXmax**
- **dataXsum**
- **dataXmean**
- **dataXrms**: root mean square
- **dataXstd**: standard deviation
- **dataXmedian**
- **dataXmad**: median absolute deviation
- **dataXstep**: the minimal difference between two neighboring data points in the column

For **gridded data** input the following variables are additionally defined for each data point:

- **longitude** in degrees
- **latitude** in degrees
- **height** in meters
- **cartesianX** coordinate in meters
- **cartesianY** coordinate in meters
- **cartesianZ** coordinate in meters
- **area** of the unit sphere
- **dataXwmean**: area-weighted mean
- **dataXwrms**: area-weighted root mean square
- **dataXwstd**: area-weighted standard deviation

## 1.3 Loops and conditions

The program flow within a config file can be controlled by the classes **loop** and **condition**. The easiest way to access these classes is with the programs **LoopPrograms** and **IfPrograms**.

The programs defined in **IfPrograms** are only executed if the defined **condition** is met. A typical example is to check whether a file that should have been created in previous programs actually exists. Further options are string comparisons and checking the result of a numerical expression or the return value of an external command.

With **LoopPrograms** it is possible to repeat the programs defined inside within a loop. The class **loop** creates a sequence to loop over and defines **variables** (1.2) that contain the index and element for the current iteration.

The **loop** and **condition** can also be attributed to single config elements (including programs). Config elements with an assigned loop are repeated, with the loop variables being evaluated for each element. If a **condition** is attributed to a config element in addition to a loop, each element within the loop is only created if the condition is met. Conditions can also be attributed to optional elements without an associated loop. If the condition is not met, the optional element will be treated as if it was not provided.

Example: A program needs all files in a download directory as input. All the **inputfiles** can be selected manually of course, but it is much easier to assign a loop variable with **inputfile={loopFile}** and attribute a **loop:directoryListing**. The loop lists the content of the download directory and assigns each file name to the **variableLoopFile=loopFile**.

## 1.4 Constants and the settings file

GROOPS uses some built-in constants like DEFAULT\_GM or the definition of leap seconds, which are defined in `source/base/constants.cpp`.

A complete list of the constants can be written to an XML file with:

```
groops --write-settings <groopsDefaults.xml>
```

The built-in constants can be overwritten by a `groopsDefaults.xml` file in the working directory or by explicitly passing the file as an argument at execution:

```
groops --settings <groopsDefaults.xml> <config.xml>
```

It might also be useful to adjust the default values in the schema file used by the [GUI \(1.6\)](#):

```
groops --settings <groopsDefaults.xml> --xsd <groops.xsd>
```

Example file:

```
<?xml version="1.0" encoding="UTF-8"?>
<groops>
    <LIGHT_VELOCITY>299792458</LIGHT_VELOCITY>
    <DEFAULT_GM>3.986004415e+14</DEFAULT_GM>
    <DEFAULT_R>6378136.3</DEFAULT_R>
    <GRS80_a>6378137.0</GRS80_a>
    <GRS80_f>298.2572221010</GRS80_f>
    <GRAVITATIONALCONSTANT>6.6730e-11</GRAVITATIONALCONSTANT>
    <R_Earth>6.37813630000000e+06</R_Earth>
    <R_Moon>1.73800000000000e+06</R_Moon>
    <GM_Earth>3.9860044150000e+14</GM_Earth>
    <GM_Sun>1.32712442076000e+20</GM_Sun>
    <GM_Moon>4.90280105600000e+12</GM_Moon>
    <GM_MERCURY>2.20320808280762e+13</GM_MERCURY>
    <GM_VENUS>3.24858603864143e+14</GM_VENUS>
    <GM_MARS>4.28283149222192e+13</GM_MARS>
    <GM_JUPITER>1.26712769822770e+17</GM_JUPITER>
    <GM_SATURN>3.79406266494906e+16</GM_SATURN>
    <TIME_EPSILON>1.0e-05</TIME_EPSILON>
    <DELTA_TAI_GPS>19</DELTA_TAI_GPS>
    <DELTA_TT_GPS>51.184</DELTA_TT_GPS>
    <J2000>51544.5</J2000>
    <leapSecond>
        <MJD>57754</MJD>
        <DELTA_UTC_GPS>-18</DELTA_UTC_GPS>
    </leapSecond>
    <leapSecond>
        <MJD>57204</MJD>
        <DELTA_UTC_GPS>-17</DELTA_UTC_GPS>
    </leapSecond>
    <leapSecond>
        <MJD>56109</MJD>
```

```
<DELTA_UTC_GPS>-16</DELTA_UTC_GPS>
</leapSecond>

...
<leapSecond>
<MJD>41317</MJD>
<DELTA_UTC_GPS>9</DELTA_UTC_GPS>
</leapSecond>
<leapSecond>
<MJD>0</MJD>
<DELTA_UTC_GPS>10</DELTA_UTC_GPS>
</leapSecond>
</groops>
```

## 1.5 Parallelization

If GROOPS is compiled with the [Message Passing Interface \(MPI\)](#), most GROOPS [programs \(4\)](#) can be run in parallel on multiple processor cores. Processing on computer clusters with distributed memory is also supported.

Many loops are parallelized by computing each loop step at a different core. Usually the first node distributes the work load, assigns loop steps to different cores, and is not participating on the actual loop computation. This means running GROOPS with only two nodes has no advantages in almost all cases. Non-parallel parts and [programs \(4\)](#) without parallel support are executed at the first node only.

Large systems of [normal equations](#), which are divided into blocks, are distributed over the nodes to reduce the memory consumption on each single node.

As all nodes may read and write files (at least reading the [config files \(1.1\)](#)) the required part of the file system must be available on all participating computers.

## 1.6 Graphical User Interface (GUI)

The graphical user interface program `groopsGui` enables the convenient creation of GROOPS config files. It uses the [Qt5 framework](#) for cross-platform support.



Figure 1.1: Overview of the GUI (components mentioned below marked in red)

### 1.6.1 Settings and first-time setup

The GUI depends on an XSD schema file containing the complete formal (computer readable) description of a GROOPS config file. This schema file can be created with the command:

```
groops --xsd <groopsDir>/groops.xsd
```

At least one schema file has to be set via the menu **Settings - Default Paths and Files**. Setting more than one schema files enables the *schema selector* in the toolbar. The selected schema will be used when (re-)opening or creating a config file. This feature is useful when working with different versions of GROOPS at the same time.

It is possible to set a *template file* via the menu **Settings - Default Paths and Files**. This can be any GROOPS config file. Whenever a new config file is created via the GUI, all global elements and programs defined in the template file are automatically created in the new config file. It is highly recommended to create a template file containing at least the global element `groopsDataDir` of type `filename`. This element is used as a [variable \(1.2\)](#) in most default file paths throughout many GROOPS programs. Thus, setting the path to the base directory containing all GROOPS data once in the template file, for example as `<groopsDataDir>=/home/<user>/groops/data`, is the most convenient way to handle default paths in GROOPS. The template file can also contain other often-used global elements, for example `tmpDir` or `timeStart` and `timeEnd`.

A *working directory* can be set via **Settings - Default Paths and Files**. This directory is used as the default directory in the save dialog of new config files.

The GUI offers the option to open the GROOPS documentation for a selected program. To use this feature, the GROOPS documentation must be generated (if not already present) with the command:

```
groops --doc <groopsDir>/docs/
```

In the menu **Settings - Default Paths and Files** the path to the HTML version of the documentation must be set (i.e. `<groopsDir>/docs/html`). Selecting any program and pressing F1 opens the documentation for this program in an external browser. Pressing F1 without having any program selected opens the main page of the GROOPS documentation.

Executing a config file from the GUI requires the setup of a run command in the menu **Settings - Commands**. It is recommended for this command to open a new terminal in which GROOPS is executed with the config file given as an argument. The placeholders `%w` and `%f` are replaced by the directory and file name of the selected config file, respectively. Multiple commands can be set up, with the option to choose one of them in the run dialog.

Example commands:

- Windows: `cd /d %w && groops.exe %f`
- Linux (KDE): `konsole --workdir %w -e bash -ic "groops %f; bash"`
- Linux (GNOME): `gnome-terminal --working-directory=%w -x bash -ic "groops %f; bash"`
- Windows, MPI with 4 processes: `cd /d %w && mpiexec -n 4 groopsMPI.exe %f`
- Linux (KDE), MPI with 4 processes: `konsole --workdir %w -e bash -ic "mpiexec -n 4 groopsMPI %f; bash"`
- Linux (GNOME), MPI with 4 processes: `gnome-terminal --working-directory=%w -x bash -ic "mpiexec -n 4 groopsMPI %f; bash"`

### 1.6.2 Basic features

Most basic features used to manipulate a config element are accessible via the context menu, for example attributing [loops and conditions \(1.3\)](#) or setting an element global. Global elements automatically appear in the dropdown value list of config elements of the same type. Selecting a global element from the dropdown list as a value links this config element to the global element. In case the global element is removed, all linked elements' values are replaced by the value of the deleted global element.

The sidebar features three widgets:

- **Open Files:** An overview of all open config files (select to change current tree)
- **Program List:** A list of all programs defined in the schema of the active tree (filterable, supports drag and drop to tree, double click appends program)
- **Undo Stack:** Tracks all changes in a config file (select to change state of tree)

In case the names of programs or config elements change over time, the GUI offers a rename feature to update outdated config files. The changes must be documented in the schema using GROOPS' rename feature. Affected elements will be marked with an icon and the context menu item **Update name** will be available to change the element to the new name defined in the schema.

### 1.6.3 Additional keyboard shortcuts

- Tree navigation:
  - **Enter**: Switch focus from tree to input field of selected row
  - **Escape**: Switch focus from input field back to tree
  - **Tab**: Next sibling element (or next sibling of parent if there is no next sibling, or next child otherwise)
  - **Shift+Tab**: Previous sibling element (or parent if there is no previous sibling)
  - **Ctrl+Tab**: Next tab/tree
  - **Ctrl+Shift+Tab**: Previous tab/tree
  - **Ctrl+Space**: Interact with the element (e.g. filename/program: open dialog; time: switch focus between input fields)
  - **Ctrl+Up/Down**: Next/previous sibling element
  - **Ctrl+Left/Right**: Fold/expand (complex) element
- Tree manipulation:
  - **Ctrl+Shift+Up/Down**: Move unbounded list element (e.g. program, layer) up/down
- Drag and Drop of tabs to other programs (i.e. text editors) or other GUI windows:
  - **Drag**: Copy tab (= keep in source window)
  - **Shift+Drag**: Move tab (= remove from source window)
- Drag and Drop GROOPS config file(s) into GUI:
  - **Drag**: Open file(s) in new tab(s)
  - **Shift+Drag**: Open file in current tab (replaces current tab, only works with a single file)

## 1.7 File formats

All GROOPS files are written either in XML, binary, or ASCII format depending on the filename extension.

- .xml: XML format
  - .dat: binary format
  - .txt and all other extensions: ASCII format

With an additional extension of '.gz' files are directly compressed and uncompressed. It is also possible to directly uncompress and read (but not write) **Unix compress**'d files ('.Z').

Comments are allowed in ASCII files and all the text starting from the character '#' to the end of the line is ignored.

The program **FileConvert** can be used to convert between the different formats. This program is also useful to get some general information of files in binary format.

The following special file types are used in GROOPS:

### 1.7.1 Admittance

Interpolation matrix to create ocean minor tides from modeled major tides. The file can be created with **DoodsonHarmonicsCalculateAdmittance** and used e.g. in **doodsonHarmonicTide**.

See  [Doodson Harmonics](#) Calculate Admittance.

### 1.7.2 ArcList

With the **InstrumentSynchronize** an **Instrument** file can be divided into time intervals and within the intervals into arcs. This file provides the information about the mapping of arcs to time intervals.

This file can be used for the variational equation approach or **KalmanBuildNormals**.

58923.00000000000000000000	112
58924.00000000000000000000	120
58925.00000000000000000000	128
58926.00000000000000000000	136
58927.00000000000000000000	144
58928.00000000000000000000	153
58929.00000000000000000000	161
58930.00000000000000000000	169
58931.00000000000000000000	177
58932.00000000000000000000	185
58933.00000000000000000000	193
58934.00000000000000000000	201
58935.00000000000000000000	210
58936.00000000000000000000	218
58937.00000000000000000000	226
58938.00000000000000000000	234
58939.00000000000000000000	242
58940.00000000000000000000	250

### 1.7.3 DoodsonEarthOrientationParameter

Corrections for Earth orientation parameters (EOP) ( $x_p, y_p, UT1, LOD$ ) as cos/sin oscillations for a list of doodson tidal frequencies.

```
groops doodsonEarthOrientationParameter version=20200123
      11 # number of constituents
# dood.    xpCos [arcsec]           xpSin [arcsec]           ypCos [arcsec]           ypSin [arcsec]
# -----
155.645  2.39999999999999786e-07  1.7999999999999971e-07  1.7999999999999971e-07 -2.399999999999
155.655 -8.22000000000000920e-06 -6.28000000000000012e-06 -6.28000000000000012e-06  8.220000000000
155.665 -1.6499999999999872e-06 -1.26000000000000007e-06 -1.26000000000000007e-06  1.649999999999
157.455 -1.53999999999999867e-06 -1.1999999999999946e-06 -1.1999999999999946e-06  1.539999999999
157.465 -3.40000000000000270e-07 -2.5999999999999988e-07 -2.5999999999999988e-07  3.400000000000
161.557  9.9999999999999547e-08  8.00000000000000167e-08  8.00000000000000167e-08 -9.999999999999
162.556  2.54999999999999726e-06  2.01999999999999718e-06  2.01999999999999718e-06 -2.549999999999
163.545 -4.89999999999999672e-07 -3.79999999999999616e-07 -3.79999999999999616e-07  4.899999999999
163.555  4.27299999999999238e-05  3.01099999999999801e-05  3.01099999999999801e-05 -4.272999999999
164.554 -3.59999999999999943e-07 -2.800000000000000191e-07 -2.800000000000000191e-07  3.599999999999
164.556 -1.02999999999999879e-06 -7.99999999999999638e-07 -7.99999999999999638e-07  1.029999999999
```

### 1.7.4 DoodsonHarmonic

Ocean tides are represented as time variable gravitational potential and is given by a fourier expansion

$$V(\mathbf{x}, t) = \sum_f V_f^c(\mathbf{x}) \cos(\Theta_f(t)) + V_f^s(\mathbf{x}) \sin(\Theta_f(t)), \quad (1.1)$$

where  $V_f^c(\mathbf{x})$  and  $V_f^s(\mathbf{x})$  are spherical harmonics. The  $\Theta_f(t)$  are the arguments of the tide constituents  $f$ :

$$\Theta_f(t) = \sum_{i=1}^6 n_f^i \beta_i(t), \quad (1.2)$$

where  $\beta_i(t)$  are the Doodson's fundamental arguments ( $\tau, s, h, p, N', p_s$ ) and  $n_f^i$  are the Doodson multipliers for the term at frequency  $f$ .

To extract the potential coefficients of  $V_f^c$  and  $V_f^s$  for each frequency  $f$  use [DoodsonHarmonics2PotentialCoefficients](#).

See also [PotentialCoefficients2DoodsonHarmonics](#).

### 1.7.5 EarthOrientationParameter

Earth Orientation Parameter (EOP) as provided by the International Earth Rotation and Reference Systems Service (IERS) (e.g EOP 14 C04 (IAU2000A)).

See [IersC04IAU2000EarthOrientationParameter](#), [IersRapidIAU2000EarthOrientationParameter](#).

```
groups earthOrientationParameter version=20200123
    9641 # number of epochs
# UTC [MJD]           xp [arcsec]           yp [arcsec]           deltUT [sec]
# =====
5.8947000000000000000000e+04 5.69059999999999825e-02 4.099130000000000273e-01 -2.316246000000000138e-01
5.8948000000000000000000e+04 5.771400000000000141e-02 4.11015999999999925e-01 -2.332083000000000073e-01
5.8949000000000000000000e+04 5.813000000000000111e-02 4.12009999999999874e-01 -2.3461570000000000104e-01
5.8950000000000000000000e+04 5.854100000000000276e-02 4.12984999999999910e-01 -2.357567999999999886e-01
5.8951000000000000000000e+04 5.90859999999999963e-02 4.13986999999999939e-01 -2.36614999999999920e-01
5.8952000000000000000000e+04 5.976900000000000268e-02 4.154180000000000095e-01 -2.37210599999999937e-01
5.8953000000000000000000e+04 6.095400000000000124e-02 4.167310000000000181e-01 -2.37599499999999913e-01
5.8954000000000000000000e+04 6.210199999999999748e-02 4.18092999999999924e-01 -2.378588000000000091e-01
5.8955000000000000000000e+04 6.290999999999999370e-02 4.196619999999999795e-01 -2.38045499999999932e-01
5.8956000000000000000000e+04 6.385599999999999610e-02 4.214060000000000028e-01 -2.382557999999999898e-01
5.8957000000000000000000e+04 6.455500000000000127e-02 4.229890000000000039e-01 -2.385857000000000117e-01
5.8958000000000000000000e+04 6.440300000000000191e-02 4.24254999999999932e-01 -2.390210000000000112e-01
```

### 1.7.6 EarthTide

Containing the Love numbers together with frequency corrections to compute the gravitational potential and the geometric displacements due to solid Earth tides. It is used by [tides](#).

### 1.7.7 Ephemerides

Ephemerides of sun, moon, and planets stored as coefficients of Chebyshev polynomials. Used in [Ephemerides:jpl](#).

See also [JplAscii2Ephemerides](#).

### 1.7.8 GnssAntennaDefinition

Contains a list of GNSS antennas which are identified by its name (type), serial, and radome. Each antenna consists of antenna center offsets (ACO) and antenna center variations (ACV) for different signal [types](#) (code and phase). The ACV values for each type are stored in an elevation and azimuth dependent grid.

See also [GnssAntennaDefinitionCreate](#), [GnssAntex2AntennaDefinition](#).

```

<?xml version="1.0" encoding="UTF-8"?>
<groups type="antennaDefinition" version="20190429">
    <antennaCount>65</antennaCount>
    ...
    <antenna>
        <name>BLOCK IIIA</name>
        <serial>G074</serial>
        <radome>2018-109A</radome>
        <comment>PCO provided by the Aerospace Corporation, PV from estimations by ESA/CODE</comment>
        <pattern>
            <count>3</count>
            <cell>
                <type>*1*G**</type>
                <offset>
                    <x>-1.2333333333333e-03</x>
                    <y>4.3333333333333e-04</y>
                    <z>3.15200000000000e-01</z>
                </offset>
                <dZenit>1.0000000000000e+00</dZenit>
                <pattern>
                    <type>0</type>
                    <rows>1</rows>
                    <columns>18</columns>
                    <cell row="0" col="0">1.39000000000000e-02</cell>
                    <cell row="0" col="1">1.28000000000000e-02</cell>
                    <cell row="0" col="2">1.02000000000000e-02</cell>
                    <cell row="0" col="3">5.80000000000000e-03</cell>
                    <cell row="0" col="4">1.10000000000000e-03</cell>
                    <cell row="0" col="5">-4.50000000000000e-03</cell>
                    <cell row="0" col="6">-9.70000000000000e-03</cell>
                    <cell row="0" col="7">-1.28000000000000e-02</cell>
                    <cell row="0" col="8">-1.34000000000000e-02</cell>
                    <cell row="0" col="9">-1.18000000000000e-02</cell>
                    <cell row="0" col="10">-8.90000000000000e-03</cell>
                    <cell row="0" col="11">-4.50000000000000e-03</cell>
                    <cell row="0" col="12">1.20000000000000e-03</cell>
                    <cell row="0" col="13">7.20000000000000e-03</cell>
                    <cell row="0" col="14">1.33000000000000e-02</cell>
                    <cell row="0" col="15">1.33000000000000e-02</cell>
                    <cell row="0" col="16">1.33000000000000e-02</cell>
                </pattern>
            </cell>
        </pattern>
    </antenna>

```

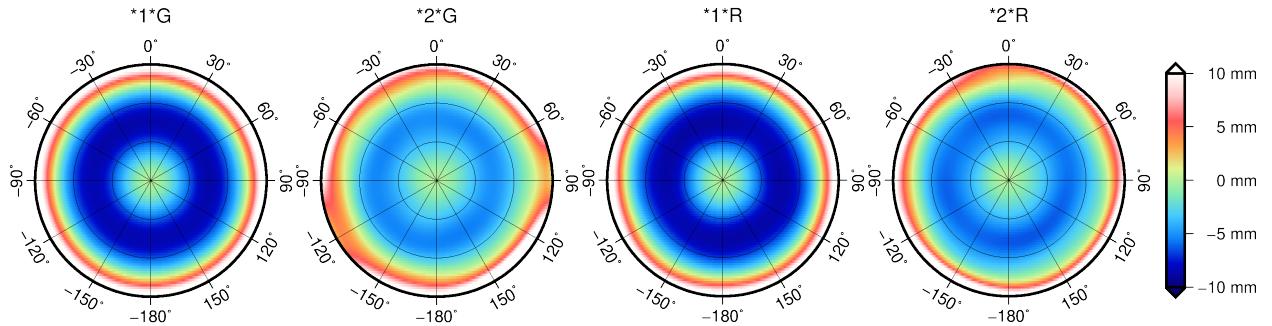


Figure 1.2: Antenna center variations of ASH701945D\_M for two frequencies of GPS and GLONASS

```

        <cell row="0" col="17">1.33000000000000e-02</cell>
    </pattern>
</cell>
...
</pattern>
</antenna>
</groops>
```

### 1.7.9 GnssReceiverDefinition

Contains a list of GNSS receivers which are identified by its name, serial, and version. Defines for each receiver a list of signal types which can be observed. Can also be used for GNSS transmitters to define a list of transmitted signal types. For GLONASS satellites the frequency number can be stored in the *version* field.

See [GnssReceiverDefinitionCreate](#).

```

<?xml version="1.0" encoding="UTF-8"?>
<groops type="receiverDefinition" version="20190429">
    <receiverCount>112</receiverCount>
    <receiver>
        <name>GLONASS</name>
        <serial>R779</serial>
        <version>2</version>
        <comment/>
        <types>
            <count>4</count>
            <cell>*1CR**J</cell>
            <cell>*1PR**J</cell>
            <cell>*2CR**J</cell>
            <cell>*2PR**J</cell>
        </types>
    </receiver>
    ...
    <receiver>
        <name>GLONASS-K1</name>
        <serial>R802</serial>
        <version>7</version>
        <comment/>
        <types>
            <count>10</count>
            <cell>*1CR**0</cell>
            <cell>*1PR**0</cell>
            <cell>*2CR**0</cell>
            <cell>*2PR**0</cell>
            <cell>*3IR**</cell>
            <cell>*3QR**</cell>
            <cell>*4AR**</cell>
            <cell>*4BR**</cell>
            <cell>*6AR**</cell>
            <cell>*6BR**</cell>
        </types>
    </receiver>
</groops>
```

### 1.7.10 GnssSignalBias

Signal biases of GNSS transmitters or receivers for different [gnssType](#).

```
groops gnssSignalBias version=20200123
      5 # number of signals
# type    bias [m]
# =====
C1CG06 -1.752461109688110974e-01
C1WG06  4.005884595055994590e-02
C2WG06  6.597469378913034532e-02
L1*G06 -2.736169875580296909e-02
L2*G06  3.422596762686257871e-02
```

See also [GnssProcessing](#), [GnssSimulateReceiver](#), [GnssSignalBias2Matrix](#), [GnssSignalBias2SinexBias](#).

### 1.7.11 GnssStationInfo

DEPRECATED. Use [Platform](#) instead.

### 1.7.12 GriddedData

List of arbitrarily distributed points defined by geographic coordinates and ellipsoidal height. Each point can also have an associated area (projected on the unit sphere with a total area of  $4\pi$ ). This file format supports multiple values per point (called `data0`, `data1` and so on).

For regular gridded data and binary format (`*.dat`) a more efficient storage scheme is used.

See also: [GriddedDataCreate](#).

```
groops griddedData version=20200123
 1 2 6.3781370000000000000e+06 6.356752314140356146e+06 72 # hasArea, data columns, ellipsoid a, ell
# longitude [deg]           latitude [deg]           height [m]           unit areas [-]
# =====
-1.6500000000000000000e+02 7.5000000000000000000e+01 0.0000000000000000000e+00 7.014893453974438420e
-1.3500000000000000000e+02 7.5000000000000000000e+01 0.0000000000000000000e+00 7.014893453974438420e
-1.0500000000000000000e+02 7.5000000000000000000e+01 0.0000000000000000000e+00 7.014893453974438420e
-7.5000000000000000000e+01 7.5000000000000000000e+01 0.0000000000000000000e+00 7.014893453974438420e
-4.5000000000000000000e+01 7.5000000000000000000e+01 0.0000000000000000000e+00 7.014893453974438420e
-1.5000000000000000000e+01 7.5000000000000000000e+01 0.0000000000000000000e+00 7.014893453974438420e
 1.49999999999997691e+01 7.5000000000000000000e+01 0.0000000000000000000e+00 7.014893453974438420e
 4.49999999999997868e+01 7.5000000000000000000e+01 0.0000000000000000000e+00 7.014893453974438420e
 7.49999999999997158e+01 7.5000000000000000000e+01 0.0000000000000000000e+00 7.014893453974438420e
 1.04999999999999574e+02 7.5000000000000000000e+01 0.0000000000000000000e+00 7.014893453974438420e
 1.34999999999999432e+02 7.5000000000000000000e+01 0.0000000000000000000e+00 7.014893453974438420e
 1.64999999999999432e+02 7.5000000000000000000e+01 0.0000000000000000000e+00 7.014893453974438420e
-1.6500000000000000000e+02 4.5000000000000000711e+01 0.0000000000000000000e+00 1.916504532594049681e
-1.3500000000000000000e+02 4.5000000000000000711e+01 0.0000000000000000000e+00 1.916504532594049681e
```

### 1.7.13 GriddedDataTimeSeries

Time series of data for arbitrarily distributed points defined by geographic coordinates and ellipsoidal height. The data can be temporal interpolated by [basis splines \(2.2\)](#). The file format consists of a [griddedData](#), a time series, and for each spatial point and spline node pair multiple values called `data0`, `data1`, ....

A `GriddedDataTimeSeries` can be generated from individual [griddedData](#) with the program [GriddedData2GriddedDataTimeSeries](#). Vice-versa, a `GriddedDataTimeSeries` can be evaluated at a specific time stamp to obtain a [griddedData](#) with [GriddedDataTimeSeries2GriddedData](#).

### 1.7.14 Instrument

This template file format can store different observations in a epoch wise manner. Each epoch consists of a time and additional data, e.g orbits, accelerometer data, star camera quaternions (see [InstrumentType](#)). The time series can be divided in several arcs (see [InstrumentSynchronize](#)).

Also a [matrix](#) file is allowed as one single arc. The first column must contain times [MJD]. Without any extra column the instrument type is INSTRUMENTTIME, with one additional column the type is MISCVALUE, and for more columns the type MISCVALUES is used.

```
groops instrument version=20200123
# SATELLITETRACKING
      -9          60 # instrument type, number of arcs
# Time [MJD]           data0: range [m]           data1: range-rate [m/s]   data2: range-acc [m/s^2]
# =====
      12 # number of epochs of 1. arc
54588.0000000000000000 -5.074649470097549492e+05 5.755440207134928654e-01 1.877605261528093308e-03
54588.000057870370255841 -5.074620458130163024e+05 5.849357691551860805e-01 1.878948916234051596e-03
54588.000115740740966430 -5.074590976427756250e+05 5.943331739937073310e-01 1.879937220634776869e-03
54588.00017361111222272 -5.074561024756557308e+05 6.037340169611068452e-01 1.880370529387525701e-03
54588.000231481481478113 -5.074530602992626373e+05 6.131368121270999172e-01 1.880680632122925426e-03
54588.000289351851733954 -5.074499711071007187e+05 6.225398878861636565e-01 1.880495369480403561e-03
54588.00034722222444543 -5.074468349029610981e+05 6.319414138081351773e-01 1.880073731783055927e-03
54588.000405092592700385 -5.074436516971451929e+05 6.413404243585696385e-01 1.879464843086203459e-03
54588.000462962962956226 -5.074404215058300761e+05 6.507353310092597320e-01 1.878578987216372124e-03
54588.000520833333212067 -5.074371443491023383e+05 6.601267978060636477e-01 1.87778184949659246e-03
54588.000578703703922656 -5.074338202460713219e+05 6.695136489207137442e-01 1.876962042758626532e-03
54588.000636574074178498 -5.074304492190054734e+05 6.788964444122400632e-01 1.876091925462087043e-03
      12 # number of epochs of 2. arc
54588.000694444444434339 -5.074270312892858055e+05 6.882748400534359767e-01 1.875376456928801432e-03
54588.000752314814690180 -5.074235664742725785e+05 6.976508178537534910e-01 1.874929898412159559e-03
54588.000810185185400769 -5.074200547868391732e+05 7.070236200716006891e-01 1.874312324351668077e-03
54588.000868055555656611 -5.074164962409950094e+05 7.163943828291452487e-01 1.873924188388115340e-03
54588.000925925925912452 -5.074128908454515622e+05 7.257639682023964145e-01 1.874025826380292404e-03
54588.000983796296168293 -5.074092386012640782e+05 7.351333608427884636e-01 1.873680487441316657e-03
54588.001041666666878882 -5.074055395130896359e+05 7.445020815182646912e-01 1.873849502509668122e-03
54588.001099537037134724 -5.074017935789784533e+05 7.538716732272922050e-01 1.873971633320137753e-03
54588.001157407407390565 -5.073980007962241652e+05 7.632414098560330595e-01 1.873984767500571974e-03
54588.001215277777646406 -5.073941611626467784e+05 7.726123093411200182e-01 1.874295246964456478e-03
54588.001273148148356995 -5.073902746728868224e+05 7.819835205798950639e-01 1.874226146744964808e-03
54588.001331018518612836 -5.073863413272026228e+05 7.913547196412918927e-01 1.874173804634685515e-03
```

### 1.7.15 Matrix

Stores matrices and vectors. Only one triangle is written for symmetric or triangular matrices.

The header (the matrix definition) is optional. Therefore a pure text with only numbers in columns are also allowed. This simplifies the handling of external data.

Instead of a matrix file also an [instrument](#) file is allowed. The first column is the time [MJD], the other columns depends on the instrument type.

```
groops matrix version=20200123
LowerSymmetricMatrix( 4 x 4 )
  1.000000000000000e+00
  0.000000000000000e+00  1.000000000000000e+00
  0.000000000000000e+00  0.000000000000000e+00  1.000000000000000e+00
  0.000000000000000e+00  0.000000000000000e+00  0.000000000000000e+00  1.000000000000000e+00
```

### 1.7.16 MeanPolarMotion

The mean pole of the Earth rotation is represented by a polynomial in a time interval.

```
<?xml version="1.0" encoding="UTF-8"?>
<groops type="meanPolarMotion" version="20200123">
  <meanPolarMotion>
    <intervalCount>2</intervalCount>
    <interval>
      <timeStart>42778.0000000000000000</timeStart>
      <degree>3</degree>
      <xp>5.5974100000000e-02</xp>
      <xp>1.8243000000000e-03</xp>
      <xp>1.8413000000000e-04</xp>
      <xp>7.0240000000000e-06</xp>
      <yp>3.4634600000000e-01</yp>
      <yp>1.7896000000000e-03</yp>
      <yp>-1.0729000000000e-04</yp>
      <yp>-9.0800000000000e-07</yp>
    </interval>
    <interval>
      <timeStart>55197.0000000000000000</timeStart>
      <degree>1</degree>
      <xp>2.3513000000000e-02</xp>
      <xp>7.6141000000000e-03</xp>
      <yp>3.5889100000000e-01</yp>
      <yp>-6.2870000000000e-04</yp>
    </interval>
  </meanPolarMotion>
</groops>
```

### 1.7.17 NormalEquation

Stores a [system of normal equations](#) (5.29)

$$\mathbf{N}\hat{\mathbf{x}} = \mathbf{n}. \quad (1.3)$$

. This file format consists of multiple files. The file name `normals.dat.gz` corresponds to the following files:

- `normals.dat.gz` or `normals.00.00.dat.gz` ... `normals.On.On.dat.gz`: the normal matrix  $\mathbf{N}$  as [matrix](#),
- `normals.rightHandSide.dat.gz`: the right hand side(s)  $\mathbf{n}$  as [matrix](#),
- `normals.parameterNames.txt`: [parameter names](#),
- `normals.info.xml`: u.a. containing the number of observations and the quadratic sum of (reduced) observations  $\mathbf{l}^T \mathbf{P} \mathbf{l}$ .

A large normal matrix may be splitted into blocks and stored in multiple files. The block row/column number is indicated in the file name. Only the upper blocks of the sysmmetric matrix are considered. Matrix in blocks can be distributed on muliple nodes in parallel mode to efficiently use distributed memory.

### 1.7.18 OceanPoleTide

Describes the reaction of the ocean mass to the change of the centrifugal potential (polar wobble) in terms spherical harmonics.

See also [Iers2OceanPoleTide](#).

### 1.7.19 ParameterName

Name of parameters of a system of [normal equations](#) or [solution vector](#).

A parameter name is a string `<object>:<type>:<temporal>:<interval>` containg four parts divided by :

1. object: Object this parameter refers to, e.g. `graceA`, `G023`, `earth`, ...
2. type: Type of this parameter, e.g. `accBias`, `position.x`, ...
3. temporal: Temporal representation of this parameter, e.g. `trend`, `polynomial.degree1`, ...
4. interval: Interval/epoch this parameter represents, e.g. `2017-01-01_00-00-00_2017-01-02_00-00-00`, `2018-01-01_00-00-00`.

In the documentation a star (\*) in the name means this part is untouched and useally set by other classes. Times are written as `yyyy-mm-dd hh-mm-ss` and intervals (if not empty) as `<timeStart>_<timeEnd>`.

See [ParameterNamesCreate](#).

```
groops parameterName version=20200123
# object:type:temporal:interval
# =====
    10080 # number of parameters
karr:position.x::2018-06-01_00-00-00_2018-06-02_00-00-00
karr:position.y::2018-06-01_00-00-00_2018-06-02_00-00-00
karr:position.z::2018-06-01_00-00-00_2018-06-02_00-00-00
karr:troposphereWet:spline.n1:2018-06-01_00-00-00_2018-06-01_02-00-00
karr:troposphereWet:spline.n1:2018-06-01_00-00-00_2018-06-01_04-00-00
karr:troposphereWet:spline.n1:2018-06-01_02-00-00_2018-06-01_06-00-00
karr:troposphereWet:spline.n1:2018-06-01_04-00-00_2018-06-01_08-00-00
karr:troposphereWet:spline.n1:2018-06-01_06-00-00_2018-06-01_10-00-00
karr:troposphereWet:spline.n1:2018-06-01_08-00-00_2018-06-01_12-00-00
```

---

```

karr:troposphereWet:spline.n1:2018-06-01_10-00-00_2018-06-01_14-00-00
karr:troposphereWet:spline.n1:2018-06-01_12-00-00_2018-06-01_16-00-00
karr:troposphereWet:spline.n1:2018-06-01_14-00-00_2018-06-01_18-00-00
karr:troposphereWet:spline.n1:2018-06-01_16-00-00_2018-06-01_20-00-00
karr:troposphereWet:spline.n1:2018-06-01_18-00-00_2018-06-01_22-00-00
karr:troposphereWet:spline.n1:2018-06-01_20-00-00_2018-06-02_00-00-00
karr:troposphereWet:spline.n1:2018-06-01_22-00-00_2018-06-02_00-00-00
karr:troposphereGradient.x:spline.n1:2018-06-01_00-00-00_2018-06-02_00-00-00
karr:troposphereGradient.y:spline.n1:2018-06-01_00-00-00_2018-06-02_00-00-00
karr:troposphereGradient.x:spline.n1:2018-06-01_00-00-00_2018-06-02_00-00-00
karr:troposphereGradient.y:spline.n1:2018-06-01_00-00-00_2018-06-02_00-00-00
karr:signalBias01(+1.00L1CG**)::
karr:signalBias02(+1.00L2WG**)::
karr:signalBias03(+1.00L2XG**)::
G01:solarRadiationPressure.ECOM.D0::
G01:solarRadiationPressure.ECOM.DC2::
G01:solarRadiationPressure.ECOM.DS2::
G01:solarRadiationPressure.ECOM.Y0::
G01:solarRadiationPressure.ECOM.B0::
G01:solarRadiationPressure.ECOM.BC1::
G01:solarRadiationPressure.ECOM.BS1::
G01:stochasticPulse.x::2018-06-01_12-00-00
G01:stochasticPulse.y::2018-06-01_12-00-00
G01:stochasticPulse.z::2018-06-01_12-00-00
G01:arc0.position0.x::
G01:arc0.position0.y::
G01:arc0.position0.z::
G01:arc0.velocity0.x::
G01:arc0.velocity0.y::
G01:arc0.velocity0.z::
G01:signalBias01(-1.00C1CG01)::
```

G01:signalBias02(+1.00L1\*G01)::  
 G01:signalBias03(+1.00L2\*G01)::

### 1.7.20 Platform

Defines a platform with a local coordinate frame equipped with instruments. The platform might be a reference station, a low Earth satellite, or a transmitting GNSS satellite and is referenced by a marker name and number. The reference point (marker or center of mass (CoM)) can change in time relative to the local frame.

Each equipped instrument is described at least by the following information

- name
- serial number
- coordinates in the local frame
- a time interval in which the instrument was active
- the orientation for antennas and reflectors.

For GNSS satellites the platform defines the PRN. The different assigned SVNs are defined by the transmitting antennas.

Platforms for GNSS stations can be created from station log files with [GnssStationLog2Platform](#). Platforms for GNSS satellites can be created from an ANTEX file with [GnssAntex2AntennaDefinition](#).

See also [PlatformCreate](#).

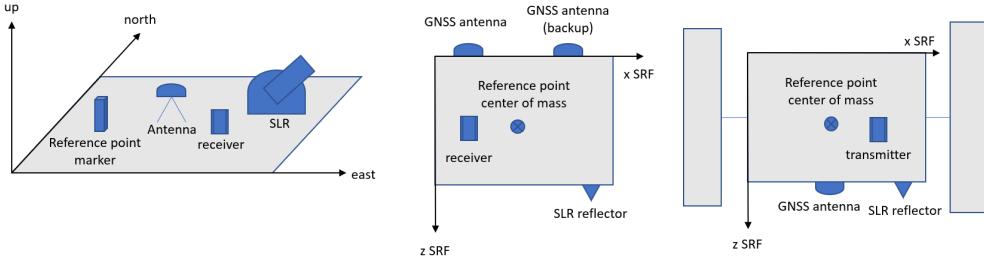


Figure 1.3: Platform for stations, LEOs, and GNSS satellites.

## 1.7.21 Polygon

List of longitude and latitudes to describe borders, e.g. river basins or continents. It is used in [border:polygon](#).

```
groops polygon version=20200123
    2 # number of polygons
    6 # number of points (1. polygon)
# longitude [deg]          latitude [deg]
# =====
-1.598200000000000216e+02  2.203000000000000114e+01
-1.59620000000000045e+02  2.1899999999999858e+01
-1.5937999999999955e+02  2.1899999999999858e+01
-1.59300000000000114e+02  2.2219999999999886e+01
-1.59580000000000125e+02  2.2219999999999886e+01
-1.598200000000000216e+02  2.203000000000000114e+01
      5 # number of points (2. polygon)
# longitude [deg]          latitude [deg]
# =====
-7.900000000000000000e+01  2.6699999999999929e+01
-7.870000000000000284e+01  2.650000000000000000e+01
-7.823000000000000398e+01  2.667000000000000171e+01
-7.793000000000000682e+01  2.667000000000000171e+01
-7.77999999999999716e+01  2.6469999999999886e+01
```

## 1.7.22 PotentialCoefficients

The standard .gfc format as defined by the ICGEM is used in ASCII the format. Only the static part is used and temporal variations (e.g. trend) are ignored. To write additional information and temporal variations use [PotentialCoefficients2Icgem](#).

### 1.7.23 SatelliteModel

Properties of a satellite to model non-conservative forces (e.g. [► miscAccelerations](#)). The file may contain surface properties, mass, drag coefficients, and antenna thrust values.

See [► SatelliteModelCreate](#) and [► SinexMetadata2SatelliteModel](#).

```
<?xml version="1.0" encoding="UTF-8"?>
<groops type="satelliteModel" version="20190429">
    <satelliteCount>1</satelliteCount>
    <satellite>
        <satelliteName>GALILEO-2</satelliteName>
        <mass>7.00000000000000e+02</mass>
        <coefficientDrag>0.00000000000000e+00</coefficientDrag>
        <surfaceCount>15</surfaceCount>
        <surface>
            <type>0</type>
            <normal>
                <x>-1.00000000000000e+00</x>
                <y>0.00000000000000e+00</y>
                <z>0.00000000000000e+00</z>
            </normal>
            <area>4.40000000000000e-01</area>
            <reflexionVisible>0.00000000000000e+00</reflexionVisible>
            <diffusionVisible>7.00000000000000e-02</diffusionVisible>
            <absorptionVisible>9.30000000000000e-01</absorptionVisible>
            <reflexionInfrared>1.00000000000000e-01</reflexionInfrared>
            <diffusionInfrared>1.00000000000000e-01</diffusionInfrared>
            <absorptionInfrared>8.00000000000000e-01</absorptionInfrared>
            <hasThermalReemission>1</hasThermalReemission>
        </surface>
        ...
        <modulCount>2</modulCount>
        <modul>
            <type>1</type>
            <rotationAxis>
                <x>0.00000000000000e+00</x>
                <y>1.00000000000000e+00</y>
                <z>0.00000000000000e+00</z>
            </rotationAxis>
            <normal>
                <x>0.00000000000000e+00</x>
                <y>0.00000000000000e+00</y>
                <z>1.00000000000000e+00</z>
            </normal>
            <surface>
                <count>4</count>
                <cell>11</cell>
                <cell>12</cell>
                <cell>13</cell>
                <cell>14</cell>
            </surface>
        </modul>
        <modul>
```

```

<type>2</type>
<antennaThrust>
  <x>0.00000000000000e+00</x>
  <y>0.00000000000000e+00</y>
  <z>2.65000000000000e+02</z>
</antennaThrust>
</modul>
</satellite>
</groops>
```

### 1.7.24 StringList

White space separated list of strings. Comments are allowed and all the text from the character '#' to the end of the line is ignored. Strings containing white spaces or the '#' character must be set in quotes ("").

```
# IGSR3 stations
abmf
abpo
ade1
adis
ajac
albh
algo
alic
alrt
amc2
aoml
areq
arev
artu
asc1
```

### 1.7.25 StringTable

White space separated table of strings in row and columns. Additional columns in a row may represent alternatives, e.g. for core stations in a GNSS network. Comments are allowed and all the text from the character '#' to the end of the line is ignored. Strings containing white spaces or the '#' character must be set in quotes ("").

```
# core network with alternative stations
artu mdvj mdvo nril
asc1 sthl
bahr bhr1 yibl nama
chat chti auck
chpi braz ufpr savo
ckis nium
coco xmis dgar dgav
cro1 scub abmf lmmf aoml
daej taej suwn osn1
darw kat1 tow2 alic
dav1 maw1
```

```

drao albh will holb nano
fair whit
glps guat
gode godz usno usn3
goug

```

### 1.7.26 TideGeneratingPotential

```

groops tideGeneratingPotential version=20200123
      3606
# Dood.   cos           sin           name
# =====
055.563 -3.122600001000726621e-06 0.0000000000000000e+00 ""
055.565  7.719644799947265879e-02 0.0000000000000000e+00 om1
055.573  1.97599999999999959e-07 0.0000000000000000e+00 ""
055.575 -7.535264999889109729e-04 0.0000000000000000e+00 om2
055.654 -4.037500000326006771e-06 0.0000000000000000e+00 ""
055.666  6.59000000000000115e-08 0.0000000000000000e+00 ""
055.753  6.02300000000000398e-07 0.0000000000000000e+00 ""
056.475  7.92000000000000775e-08 0.0000000000000000e+00 ""
056.554 -1.360322439950949897e-02 0.0000000000000000e+00 sa
056.556  7.156881999672112154e-04 0.0000000000000000e+00 ""
056.564  8.641769999583327993e-05 0.0000000000000000e+00 ""

```

### 1.7.27 TimeSplinesCovariance

Stores covariance information for [TimeSplinesGravityField](#). It can be the variances of the potential coefficients or the full covariance matrix for each temporal nodal point.

### 1.7.28 TimeSplinesGravityField

Temporal changing gravity field, parametrized as spherical harmonics in the spatial domain and parametrized as basis splines in the time domain (see [basis splines \(2.2\)](#)). It is evaluated with [gravityfield:timeSplines](#).

See also: [Gravityfield2TimeSplines](#), [PotentialCoefficients2BlockMeanTimeSplines](#).

### 1.7.29 VariationalEquation

Arcs with reference orbit and state transition matrices.

The file contains a reference orbit (position and velocity), the derivatives of the orbit with respect to the satellite state vector for each arc, transformations (rotations) between the satellite, celestial, and terrestrial frame and a satellite macro model (see [SatelliteModel](#)).

The reference orbit can be extracted with [Variational2OrbitAndStarCamera](#).

See also: [PreprocessingVariationalEquation](#).

## Chapter 2

# Mathematical fundamentals

This chapter describes the methods and mathematical fundamentals used throughout GROOPS.

## 2.1 Robust least squares adjustment

The robust least squares adjustment used in GROOPS is based on a modified Huber estimator. It down-weights observations with large outliers iteratively.

The algorithm starts with a first solution with equal weights  $\mathbf{P} = \mathbf{I}$

$$\hat{\mathbf{x}} = (\mathbf{A}^T \mathbf{P} \mathbf{A})^{-1} \mathbf{A}^T \mathbf{P} \mathbf{l}. \quad (2.1)$$

The solution is used to compute the residuals

$$\hat{e}_i = (\mathbf{l} - \mathbf{A} \hat{\mathbf{x}})_i \quad (2.2)$$

and the redundancies of all observations

$$r_i = (\mathbf{I} - \mathbf{A}(\mathbf{A}^T \mathbf{P} \mathbf{A})^{-1} \mathbf{A}^T \mathbf{P})_{ii}. \quad (2.3)$$

For observations with large residuals a new standard deviation is assigned

$$\sigma_i = \begin{cases} 1 & \text{for } \left| \frac{\hat{e}_i}{r_i} \right| \leq h \cdot \hat{\sigma} \\ \left| \frac{\hat{e}_i}{r_i h} \right|^p & \text{for } \left| \frac{\hat{e}_i}{r_i} \right| > h \cdot \hat{\sigma}, \end{cases} \quad (2.4)$$

where  $h$  is **huber**,  $p$  is **huberPower**, and  $\hat{\sigma}^2$  a robust overall variance factor computed from all residuals. The estimation is repeated **huberMaxIteration** times with a new weight matrix

$$\mathbf{P} = \text{diag} \left( \frac{1}{\sigma_1^2}, \frac{1}{\sigma_2^2}, \dots, \frac{1}{\sigma_n^2} \right) \quad (2.5)$$

or until convergence is reached.

## 2.2 Basis splines

A time variable function is given by

$$f(x, t) = \sum_i f_i(x) \Psi_i(t), \quad (2.6)$$

with the (spatial) coefficients  $f_i(x)$  as parameters and the temporal basis functions  $\Psi_i(t)$ . Basis splines are defined as polynomials of degree  $n$  in intervals between nodal points in time  $t_i$ :

- Block mean values ( $n = 0$ )

$$\Psi_i(t) = \begin{cases} 1 & \text{if } t \in [t_i, t_{i+1}), \\ 0 & \text{otherwise} \end{cases} \quad (2.7)$$

- Linear splines ( $n = 1$ )

$$\Phi_i(t) = \begin{cases} \tau_{i-1} & \text{if } t_{i-1} \leq t \leq t_i, \\ 1 - \tau_i & \text{if } t_i \leq t \leq t_{i+1}, \\ 0 & \text{otherwise.} \end{cases} \quad (2.8)$$

- Quadratic splines ( $n = 2$ )

$$\Phi_i(t) = \begin{cases} \frac{1}{2}\tau_{i-1}^2 & \text{if } t_{i-1} \leq t \leq t_i, \\ -\tau_{i-1}^2 + \tau_{i-1} + \frac{1}{2} & \text{if } t_i \leq t \leq t_{i+1}, \\ \frac{1}{2}\tau_{i-1}^2 - \tau_{i-1} + \frac{1}{2} & \text{if } t_i \leq t \leq t_{i+1}, \\ 0 & \text{otherwise.} \end{cases} \quad (2.9)$$

- Cubic splines ( $n = 3$ )

$$\Phi_i(t) = \begin{cases} \frac{1}{6}\tau^3 & \text{if } t_{i-1} \leq t \leq t_i, \\ -\frac{3}{6}\tau^3 + \frac{3}{6}\tau^2 + \frac{3}{6}\tau + \frac{1}{6} & \text{if } t_{i-1} \leq t \leq t_i, \\ \frac{3}{6}\tau^3 - \tau^2 + \frac{4}{6} & \text{if } t_{i-1} \leq t \leq t_i, \\ -\frac{1}{6}\tau^3 + \frac{3}{6}\tau^2 - \frac{3}{6}\tau + \frac{1}{6} & \text{if } t_{i-1} \leq t \leq t_i, \\ 0 & \text{otherwise.} \end{cases} \quad (2.10)$$

where  $\tau$  is the normalized time in each time interval

$$\tau_i = \frac{t - t_i}{t_{i+1} - t_i}. \quad (2.11)$$

The total number of coefficients  $f_i(x)$  is  $N = N_t + n - 1$ , where  $N_t$  is the count of nodal time points  $t_i$  and  $n$  is the degree.



Figure 2.1: Basis splines for different degrees with nodal points every 6 hours.

## 2.3 Autoregressive Models

A multivariate (or vector) autoregressive model is one possible representation of a random process. It specifies, that the output at epoch  $t$  depends on the  $p$  previous epochs, where  $p$  is denoted process order, plus a stochastic term. In the following, finite order vector autoregressive - VAR( $p$ ) in short - models as implemented in GROOPS will be described.

### 2.3.1 Definition

A finite order VAR( $p$ ) model is defined as

$$\mathbf{y}_e(t_i) = \sum_{k=1}^p \Phi_k^{(p)} \mathbf{y}_e(t_{i-k}) + \mathbf{w}(t_i), \quad \mathbf{w}(t_i) \sim \mathcal{N}(0, \Sigma_{\mathbf{w}}^{(p)}), \quad (2.12)$$

where  $\mathbf{y}_e(t_i)$  are realizations of a random vector process. Subtracting the right hand side and substituting the stochastic term  $-\mathbf{w}(t_i)$  with the residual  $\mathbf{v}(t_i)$  gives us

$$\mathbf{0} = \mathbf{y}_e(t_i) - \sum_{k=1}^p \Phi_k^{(p)} \mathbf{y}_e(t_{i-k}) + \mathbf{v}(t_i) \quad (2.13)$$

which can be used as pseudo-observation equations in the determination of the parameters  $\mathbf{y}_e(t_i)$ . In matrix notation this reads

$$0 = \begin{bmatrix} \mathbf{I} & -\Phi_1^{(p)} & \dots & -\Phi_p^{(p)} \end{bmatrix} \begin{bmatrix} \mathbf{y}_e(t_i) \\ \mathbf{y}_e(t_{i-1}) \\ \vdots \\ \mathbf{y}_e(t_{i-p}) \end{bmatrix} + \mathbf{v}(t_i). \quad (2.14)$$

After rearranging the vectors  $\mathbf{x}_t$  to have ascending time stamps

$$0 = \begin{bmatrix} -\Phi_p^{(p)} & \dots & -\Phi_1^{(p)} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{y}_e(t_{i-p}) \\ \vdots \\ \mathbf{y}_e(t_{i-1}) \\ \mathbf{y}_e(t_i) \end{bmatrix} + \mathbf{v}(t_i) \quad (2.15)$$

For practical purposes, the residuals above are further decorrelated using the inverse square root of the white noise covariance matrix, leading to

$$\bar{\mathbf{v}}(t_i) = \underbrace{\Sigma_{\mathbf{w}}^{(p)}^{-\frac{1}{2}}}_{=\mathbf{W}} \mathbf{v}(t_i), \quad \bar{\mathbf{v}}(t_i) \sim \mathcal{N}(0, \mathbf{I}). \quad (2.16)$$

The used square root is in principle arbitrary, but should satisfy  $\mathbf{W}^T \mathbf{W} = \Sigma_{\mathbf{w}}^{(p)}$ . This means that both eigendecomposition based roots and Cholesky factors can be used. After applying the matrix from the left, we arrive at the observation equations

$$0 = \begin{bmatrix} -\mathbf{W} \Phi_p^{(p)} & \dots & -\mathbf{W} \Phi_1^{(p)} & \mathbf{W} \end{bmatrix} \begin{bmatrix} \mathbf{y}_e(t_{i-p}) \\ \vdots \\ \mathbf{y}_e(t_{i-1}) \\ \mathbf{y}_e(t_i) \end{bmatrix} + \bar{\mathbf{v}}(t_i) \quad (2.17)$$

which yields fully decorrelated residuals. Currently, VAR( $p$ ) models are saved to a single file which contains this matrix.

# Chapter 3

## Cookbook

This chapter presents recipes to explore the GROOPS feature set.

### 3.1 Instrument data handling

GROOPS provides functions and programs to read/write, preprocess, analyze and visualize uniformly and non-uniformly sampled instrument data. This includes tools for filter design and analysis, re-sampling, smoothing, detrending, and power spectrum estimation. This tutorial goes through exemplary steps for data handling procedures.

#### 3.1.1 Reading data

- GROOPS is able to read and convert relevant data from various LEO and GNSS satellites. Instrument files need to be converted into the respective GROOPS format using conversion programs. Depending on the content of the input file, the data is stored with a specific  **Instrument type**. User also has the option to change the type later on with  **InstrumentSetType**.
- Multiple files can be concatenated to one file using  **InstrumentConcatenate**. Using this program, it is also possible to sort the epochs, remove the duplicates and NaN values.
- *Example: Concatenating instrument files*
  - Create three successively daily sinusoidal signals with  **TimeSeriesCreate** and set their type to MISCVALUE with  **InstrumentSetType**. In this example, each data set has an overlap of 1 hour with their following dataset.
  - Merge all datasets to one single file with  **InstrumentConcatenate**.
- Many measurements involve data collected asynchronously by multiple sensors with different sampling. Use  **InstrumentSynchronize** for a continuous harmonization of the data over time or segmentation of the data into arcs.

#### 3.1.2 Preprocessing

Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues. Following steps are usually required to be taken:



Figure 3.1: Example 1: Concatenating instrument files into one dataset.

- Gross outlier removal:
  - Create reference values to compare the input data with. Depending on the instrument type, this can be done by simulation programs such as [SimulateAccelerometer](#) or [SimulateStarCamera](#). If no reference data is available, the outlier detection is based on the data itself. If needed, synchronize the reference data file and the input data with [InstrumentSynchronize](#).
  - In case of star camera data, compute the differences between the input data and the reference data with [InstrumentStarCameraMultiply](#).
  - Set a threshold for outlier detection in [InstrumentRemoveEpochsByCriteria](#). The threshold is defined empirically according to the accuracy characteristics of each data products. If the differences exceed a predefined threshold, the corresponding epochs are removed. An arbitrary margin can be defined to additionally remove epochs before and after the identified outliers. It is also possible to remove epochs at specific times using [InstrumentRemoveEpochsByTimes](#).
  - Missing epochs can be filled by reference data with [InstrumentConcatenate](#).
  - It is also possible to interpolate the missing epochs with [InstrumentResample](#).
  - *Example: Removing outliers in a synthetic data.*
    - \* Create a sinusoidal signal with an amplitude of 1.0 using [TimeSeriesCreate](#) and set its type to MISCVALUE with [InstrumentSetType](#).
    - \* Add zero-mean, white Gaussian noise with a standard deviation of 0.1 with [NoiseInstrument](#). Interpret this data as a real measurement file.
    - \* Set the threshold criteria to 0.2 in [InstrumentRemoveEpochsByCriteria](#) and remove the outliers and their nearest epochs in 20 second interval.
    - \* Fill the data gaps with [InstrumentResample](#).
- Downsampling:
  - If the sampling is irregular use [InstrumentResample](#) to make the sampling equidistant.
  - Use [InstrumentSynchronize](#) to divide the data at gaps into arcs.
  - Apply a lowpass filter (e.g. Butterworth) with the Nyquist frequency of the target sampling as cutoff with [InstrumentFilter](#). Apply the filter in both directions to avoid phase shifts.
  - Use [InstrumentReduceSampling](#) to down-sample the data.



Figure 3.2: Example 2: Removing gross outliers.

- Calibration:

- For a general instrument file, **InstrumentDetrend** subtracts offsets or linear/nonlinear trends from the input data. This can be achieved also with **FunctionsCalculate** or **InstrumentArcCalculate** by applying determined calibration factors or solving a least-square adjustment.
- For accelerometer data, **InstrumentAccelerometerEstimateBiasScale** is designed to estimate and subtract complex biases or scales with respect to simulated accelerometer data. If a thruster file is given, the corresponding epochs are eliminated during estimation process.
- *Example: GRACE-C accelerometer calibration*
  - \* For one particular date, read and convert Level-1B GRACE-C orbit, star camera, accelerometer, and thruster data with **GraceL1b2Orbit**, **GraceL1b2StarCamera**, **GraceL1b2Accelerometer**, and **GraceL1b2Thruster** respectively. It is also required to read the macro-model data of the satellite using the related information in the official document and convert it to GROOPS format with **SatelliteModelCreate**.
  - \* Use **SimulateAccelerometer** to generate simulated accelerations due to non-gravitational force models including: **miscAccelerations:atmosphericDrag**, **miscAccelerations:solarRadiationPressure**, and **miscAccelerations:albedo**.
  - \* Calibrate the real measurements with a daily constant accelerometer bias by choosing a constant parameter per axis in **parametrizationAcceleration:accBias**.

### 3.1.3 Statistical analysis

- **InstrumentStatisticsTimeSeries** returns statistics for one or more instrument files. **InstrumentArcCalculate** is also able to generate a **statistics** file with one mid epoch per arc.

### 3.1.4 Spectral analysis

Spectral analysis studies the frequency spectrum contained in discrete, uniformly sampled data. The Fourier transform is a tool that reveals frequency components of a signal by representing it in frequency space. The Power Spectral Density (PSD) is a measurement of the energy at each frequency.



Figure 3.3: Example 3: Calibrating GRACE-C ACT1B data.

- If the sampling is irregular use **InstrumentResample** to make the sampling equidistant.
- Use **Instrument2PowerSpectralDensity** to compute PSD.
- If covariance function of a dataset is available, use **CovarianceFunction2PowerSpectralDensity**.
- *Example: Spectral analysis of a synthetic signal.*
  - Create a sinusoidal signal with an amplitude of 1.0 using **TimeSeriesCreate** and set its type to MISCVALUE with **InstrumentSetType**. Interpret this data as a simulation data file.
  - Add zero-mean, white Gaussian noise with a standard deviation of 0.1 with **NoiseInstrument**. Interpret this data as a real measurement file.
  - Compute PSD of the simulated and measurement data and represent the results with **PlotGraph**.



Figure 3.4: Example 4: Spectral analysis of a synthetic signal.

### 3.1.5 Data visualization

- Argument of latitude plot Plotting instrument data as a function of satellite position in orbit and time reveals features related to the orbit geometry or environmental conditions. For circular orbits, the position of satellite can be specified by the argument of latitude.
  - Synchronize the instrument data file with the related orbit data using [InstrumentSynchronize](#).
  - Use [Orbit2ArgumentOfLatitude](#) to compute argument of latitude at each epoch.
  - Plot the instrument data versus argument of latitude and time with [PlotGraph](#).
  - *Example: Argument of latitude representation of GRACE-C eclipse factors*
    - \* Compute eclipse factors at each epoch of GRACE-C orbit at an arbitrary time using [Orbit2EclipseFactor](#).
    - \* Synchronize the eclipse factor data file with the related orbit data using [InstrumentSynchronize](#).
    - \* Use [Orbit2ArgumentOfLatitude](#) to compute argument of latitude at each epoch and visualize the results with [PlotGraph](#).



Figure 3.5: Example 5: GRACE-C eclipse factors represented in argument of latitude plot.

- Ground-track plot Plotting instrument data with respect to the satellite ground track is useful to identify any features of geophysical origin in the data.
  - Synchronize the instrument data file with the related orbit data using [InstrumentSynchronize](#).
  - Use [Orbit2Groundtracks](#) to map instrument data to satellite ground-track.
  - Visualize the output with [PlotMap](#).
  - *Example 6: Ground-track representation of GRACE-C eclipse factors*
    - \* Compute eclipse factors at each epoch of GRACE-C orbit at an arbitrary time using [Orbit2EclipseFactor](#).
    - \* Synchronize the eclipse factor data file with the related orbit data using [InstrumentSynchronize](#).



Figure 3.6: Example 6: GRACE-C eclipse factors represented in ground-track plot.

- \* Use **Orbit2Groundtracks** to generate the gridded data. Each grid value represents the mean value of eclipse factor over the instrument time period (1 month). visualize the results with **PlotMap**.

## 3.2 GNSS satellite orbit determination and station network analysis

This cookbook chapter describes an example of global GNSS processing as done by analysis centers of the International GNSS Service (IGS). Resulting products usually comprise:

- Satellite orbits, clocks, and signal biases
- Station positions, clocks, signal biases, and troposphere estimates
- Earth orientation parameters

Scientific details about the underlying processing approach and the applied parametrizations, models, and corrections can be found in a doctoral thesis available under DOI [10.3217/978-3-85125-885-1](https://doi.org/10.3217/978-3-85125-885-1).

An example scenario for this task is available at <https://ftp.tugraz.at/outgoing/ITSG/groops/scenario/scenarioGnssNetwork.zip>. It includes GROOPS scripts and data for the example, but not the general GROOPS data and metadata found at <https://ftp.tugraz.at/outgoing/ITSG/groops> (data folder or zipped archive). The scenario generally represents what is described in this cookbook, but may slightly differ in certain settings.

*Note: Global GNSS processing can become very computationally intensive. Depending on the number of satellites and stations, the observation and processing sampling, and parametrizations it can quickly exceed the capabilities of a normal desktop computer and may require computer clusters or number crunchers (see section [Parallelization \(1.5\)](#)).*

### 3.2.1 Data preparation

Most of the required metadata files are provided in GROOPS file formats at <https://ftp.tugraz.at/outgoing/ITSG/groops>. These files are regularly updated.

Data that has to be gathered from other sources comprises:

- **Receiver observations:** GNSS measurements converted from RINEX format (see [► RinexObservation2GnssReceiver](#)).
- **Approximate orbits:** broadcast or precise orbits in CRF for orbit integration (see [► GnssRinexNavigation2OrbitClock](#) or [► Sp3Format2Orbit](#)).
- **Approximate clocks:** broadcast or precise clocks (see [► GnssRinexNavigation2OrbitClock](#) or [► GnssClockRinex2InstrumentClock](#))

Receiver observations, broadcast ephemerides, and precise satellite orbits and clocks can be downloaded from the [IGS Data Centers](#). GPS, GLONASS, and Galileo orbits and clocks for the period 1994-2020 are also available as part of [Graz University of Technology's contribution to IGS repro3](#).

The [example scenario](#) includes a small set of this data. The script `010groopsConvert.xml` can be used to convert these external formats into GROOPS formats.

Prepare a [station list file](#) that contains the stations to be processed. Each line can contain more than one station. The first station in each line that has data available is used for the processing. If your network contains more than 60-70 stations, it is recommended to start processing with a core network (see [Advanced \(3.2.4\)](#)). In this case, define an additional [core station list file](#) that can also have multiple stations per line.

### 3.2.2 Preprocessing: Orbit integration

Numerical integration of the satellite orbits is the first step in global GNSS processing. Dynamic orbits are integrated based on **force models** and then fitted to the approximate orbits by estimating their initial state and additional empirical parameters for solar radiation pressure to improve the orbit fit. The resulting **variational equations** file contains the integrated orbit, derivatives with respect to the satellite state vector, attitude, Earth rotation and satellite model.

Orbit preprocessing is covered by the script `020groopsGnssPreprocessing.xml` in the [example scenario](#).

It is recommended to perform the steps below in a **loop** (1.3) over all satellites/PRNs using **LoopPrograms**. To get the relation between `{prn}` and `{svn}` setup an additional **loop:platformEquipment** inside **loop:loop** with

- **inputfilePlatform**: the old **inputfileTransmitterInfo**
- **equipmentType** = gnssAntenna
- **variableLoopName** = block
- **variableLoopSerial** = svn
- **variableLoopTimeStart** = svnTimeStart
- **variableLoopTimeEnd** = svnTimeEnd
- **condition:expression**
  - **expression** = (`svnTimeStart <= loopTime`) **&&** (`loopTime < svnTimeEnd`)

This second loop should perform only one step. The following programs are looped over all `{prn}`:

- **InstrumentResample**: resample approximate orbits from [data preparation](#) (3.3.1) to target sampling (e.g., 1 minute) by defining a **timeSeries** based on a **method:polynomial** (**polynomialDegree**=7, **maxDataPointRange**=7200, **maxExtrapolationDistance**=900).
- **OrbitAddVelocityAndAcceleration**: add velocity via running polynomial (**polynomialDegree**=2) derivation (needed for attitude computation)
- **SimulateStarCameraGnss**
- **PreprocessingVariationalEquation**:
  - **inputfileSatelliteModel**=`{groopsDataDir}/gnss/transmitter/satelliteModel/satelliteModel_box`
  - **inputfileOrbit**: the resampled approximate orbit from **InstrumentResample**
  - **inputfileStarCamera**: the attitude file from **SimulateStarCameraGnss**
  - **forces**: see below
  - **gradientfield:potentialCoefficients**: a static gravity field (e.g. GOCO06s) with **maxDegree**=4.
- **PreprocessingVariationalEquationOrbitFit**: fit the integrated orbit (**inputfileVariational**) to the approximate orbit (**inputfileOrbit**) by least squares adjustment. Add **parametrizationAcceleration:gnssSolarRadiation** and select the ECOM2 parameters to be estimated.

Force models usually include:

- **gravityfield:potentialCoefficients**: static gravity field (e.g. GOCO06s)
- **gravityfield:trend**
  - **gravityfield:potentialCoefficients**: trend component of time-variable gravity field (e.g. GOCO06s)
- **gravityfield:oscillation**
  - **gravityfieldCos:potentialCoefficients**: annual cosine component of time-variable gravity field (e.g. GOCO06s)
  - **gravityfieldSin:potentialCoefficients**: annual sine component of time-variable gravity field (e.g. GOCO06s)
- **tides:astronomicalTide**: astronomical tides (e.g. based on JPL ephemeris)
- **tides:earthTide**: Earth tide (IERS conventions)
- **tides:doodsonHarmonicTide**: ocean tides (e.g. FES 2014b)
- **tides:poleTide**: pole tides (IERS conventions)
- **tides:poleOceanTide**: ocean pole tides (IERS conventions)
- **miscAccelerations:solarRadiationPressure**: solar radiation pressure (box-wing model)
- **miscAccelerations:albedo**: Earth radiation pressure (albedo model)
- **miscAccelerations:antennaThrust**: antenna thrust (e.g. from IGS metadata SINEX file)
- **miscAccelerations:relativisticEffect**: relativistic effects (IERS conventions)

For the spherical harmonics expansions a **maxDegree=60** is more than enough.

The result of the preprocessing should be a **variational equations** file, a **reduced dynamic orbit** file from **PreprocessingVariationalEquationOrbitFit** and an **attitude** file from **SimulateStarCameraGnss** for each satellite.

### 3.2.3 GNSS processing

The script `030groopsGnssProcessing.xml` in the [example scenario](#) implements the following steps and settings.

These are the settings for **GnssProcessing**. If not otherwise stated use the default values.

The first step is setting the processing sampling, in this example it is 30 seconds. The processing interval usually is a single 24-hour day, therefore define **timeSeries:uniformSampling** with **timeStart=<mjd>**, **timeEnd=<mjd>+1**, **sampling=30/86400** (processing sampling).

Add the appropriate **transmitters:gnss** (e.g. GPS, GLONASS, and Galileo) and provide the required files:

- **inputfileOrbit** from [preprocessing \(3.2.2\)](#)
- **inputfileAttitude** from [preprocessing \(3.2.2\)](#)
- **inputfileClock** from [data preparation \(3.3.1\)](#)

The following settings are needed in **receiver:stationNetwork**:

- **inputfileStationList**: list of all stations to be processed
- **inputfileObservations**: The converted RINEX observation files.
- **tidalDisplacement**: Use the settings described in [receiver:stationNetwork \(5.16.1\)](#).
- **excludeType**: Signals you might want to exclude are C\*?G (old unknown GPS code observations), \*3\*R (GLONASS G3 freq.), \*6\*E (Galileo E6 freq.).

Add the following **parametrizations** and define the **outputfiles** you are interested in inside each of them:

- **ionosphereSTEC**: add a constraint of **sigmaSTEC=40**
- **ionosphereMap**: add **temporal:splines** with linear (**degree=1**) 2-hourly splines
- **clocks**: optionally change **selectTransmitterZeroMeans** to **wildcard** with **name=G\*** to align clocks to mean over GPS (instead of all) satellites
- **signalBiases**
- **ambiguities**
- **codeBiases**
- **tecBiases**
- **temporalBias**: time-variable GPS L5 phase bias with **type=L5\*G** and **parametrizationTemporal:splines** with degree 3 and hourly nodes.
- **staticPositions**
- **troposphere**: select **troposphere:viennaMapping** with the appropriate vmf3grid file. Add **troposphereWetEstimation:splines** with linear (**degree=1**) 2-hourly splines and **troposphereGradientEstimation:splines** with linear daily splines.
- **transmitterDynamicOrbit**: provide **inputfileVariational=preprocessing/variational.{prn}.dat** from the preprocessing step. Add **parametrizationAcceleration:gnssSolarRadiation** and **stochasticPulse:irregular** parameter at center of day to further improve orbit fit.
- **earthRotation**
  - **estimatePole:constant**: polar motion
  - **estimatePole:trend** polar motion rate (at center of day with **timeStep=1**)
  - **estimateUT1:trend**: length of day (at center of day with **timeStep=-1** to match IGS sign convention)
- **constraints**: loose constraint on GPS L5 phase biases, **parameters:wildcard:type=signalBias.L5\***, **sigma=5** meters, and **relativeToApriori=yes**
- **constraints**: loose constraint troposphere estimates, **parameters:wildcard:type=troposphere\***, **sigma=5** meters, and **relativeToApriori=yes**
- **constraints**: constraint on stochastic pulses, **parameters:wildcard:type=stochasticPulse\***, **sigma=0.1** micrometers/second.

Finally, define the **processingSteps**. This can be overwhelming at first, but offers a lot of flexibility. The example script uses a 5-minute processing sampling with subsequent clock densification to 30 seconds.

- **selectEpochs:** with **nthEpoch=10** to reduce sampling to 5 minutes.
- **selectParametrizations:** disable `constraint.STEC`, `*VTEC`, `*.tecBiases` as the ionosphere parameters are estimated in the final steps only.
- **estimate:** with **maxIterationCount=6**
- **resolveAmbiguities**
- **selectParametrizations:** enable `*` (all) parameters
- **estimate:** with **maxIterationCount=4**: final iterations (with 5-minute sampling) and ionosphere parameters
- **group:** clock densification to 30-second sampling
  - **selectEpochs:** with **nthEpoch=1** to set full 30-second sampling
  - **selectParametrizations:** disable `*` (all) parameters and reenable `*.clock*` and `*.STEC` parameters
  - **estimate:** with **maxIterationCount=6**
  - **writeResults:** with **suffix=30s** to write 30-second clock files
- **writeResults:** write the final results

With some additional steps, the full 30-second sampling can be used to estimate all parameters (not only the clocks). These steps are disabled in the example script, as they require at least 16 GB of system memory. In this case, it is not necessary to separately write the 30-second clock files as listed above.

- **selectEpochs:** with **nthEpoch=1** to set full 30-second sampling
- **selectNormalsBlockStructure:** As the system of normal equations can be very large, the memory consumption might be reduced with **keepEpochNormalsinMemory=no**. In this case the epoch parameters are directly eliminated during the accumulation and reconstructed in the solving step. This might lead to longer computation times.
- **estimate:** with **maxIterationCount=2**: final iterations with full sampling

### 3.2.4 Advanced: Processing large station networks

Processing large station networks requires some additional steps to keep the computational load to a reasonable degree. The general processing strategy is to first process a well-distributed subset of stations (i.e. a core network) to get good estimates of all satellite parameters, which then enables integer ambiguity resolution (IAR). Once the ambiguities of the core network are resolved and stable estimates for satellite phase biases are available, all other (non-core) stations can be processed individually (including IAR) while keeping the satellite parameters fixed. At last, all stations can be processed together with all satellite parameters and ionosphere parameters.

Let's start with the **processingSteps** of the core network:

- **selectReceivers** with **selectReceivers:file** using the core network station list file from `data preparation` (3.3.1) as **inputfileStringTable**.
- **selectEpochs:** with **nthEpoch=10** to reduce sampling to 5 minutes.
- **selectParametrizations:** disable `constraint.STEC`, `*VTEC`, `*.tecBiases` as the ionosphere parameters are estimated in the final steps only.

- **estimate**: with **maxIterationCount=6**
- **resolveAmbiguities**
- **estimate**: with **maxIterationCount=4**: final iterations (with 5-minute sampling)

Now all other (non-core) stations can be processed separately:

- **forEachReceiverSeparately**: with **selectReceivers:file** inside **selectReceivers:exclude** using the station list from the core network above to process all non-core stations individually with fixed transmitter parameters
  - **processingStep:estimate**: with **maxIterationCount=6**
  - **processingStep:resolveAmbiguities**
  - **processingStep:estimate**: with **maxIterationCount=4**

Next all stations are processed together with all parameters:

- **selectReceivers**: with **selectReceivers:all**
- **selectEpochs**: with **nthEpoch=1** to set full 30-second sampling
- **group**: clock densification to 30-second sampling
  - **selectParametrizations**: disable \* (all) parameters and reenable \*.clock\* and \*.STEC parameters
  - **estimate**: with **maxIterationCount=6**
- **selectParametrizations**: enable \* (all) parameters.
- **selectNormalsBlockStructure**: with **keepEpochNormalsinMemory=no**
- **estimate**: with **maxIterationCount=4**: final iterations with full sampling and all parameters
- **writeResults**: write the final results

### 3.3 GNSS precise point positioning (PPP)

This cookbook chapter describes an example of GNSS precise point positioning (PPP) for a ground station using GPS, GLONASS, and Galileo. For information on how to generate the GNSS products (orbits, clocks, signal biases, etc.) required for PPP, see the cookbook [GNSS satellite orbit determination and station network analysis \(3.2\)](#).

Scientific details about the underlying processing approach and the applied parametrizations, models, and corrections can be found in a doctoral thesis available under DOI [10.3217/978-3-85125-885-1](https://doi.org/10.3217/978-3-85125-885-1).

An example scenario for this task is available at <https://ftp.tugraz.at/outgoing/ITSG/groops/scenario/scenarioGnssPPP.zip>. It includes GROOPS scripts and data for the example, but not the general GROOPS data and metadata found at <https://ftp.tugraz.at/outgoing/ITSG/groops> (data folder or zipped archive). The scenario generally represents what is described in this cookbook, but may slightly differ in certain settings.

#### 3.3.1 Data preparation

Most of the required metadata files are provided in GROOPS file formats at <https://ftp.tugraz.at/outgoing/ITSG/groops>. These files are regularly updated.

Data that has to be gathered from other sources comprises:

- **Receiver observations:** GNSS measurements converted from RINEX format (see ▶ [RinexObservation2GnssReceiver](#))
- **Precise orbits:** precise orbits in CRF for orbit integration (see ▶ [Sp3Format2Orbit](#))
- **Precise clocks:** precise clocks (see ▶ [GnssClockRinex2InstrumentClock](#))
- **Attitude:** rotation from body frame to CRF (see ▶ [SimulateStarCameraGnss](#) or ▶ [GnssOrbex2StarCamera](#))
- **Signal biases:** code (and phase) biases (see ▶ [GnssSinexBias2SignalBias](#))

Receiver observations, precise satellite orbits and clocks, and possibly attitude and signal biases can be downloaded from the [IGS Data Centers](#). GPS, GLONASS, and Galileo orbits, clocks, attitude, and signal biases for the period 1994-2020 are also available as part of [Graz University of Technology's contribution to IGS repro3](#).

The [example scenario](#) includes a small set of this data. The script `010groopsConvert.xml` can be used to convert these external formats into GROOPS formats.

Prepare a ▶ [station list file](#) that contains the stations (one per line) to be processed.

#### 3.3.2 Processing of a ground station

The script `02groopsGnssProcessing.xml` in the [example scenario](#) implements the following steps and settings.

These are the settings for ▶ [GnssProcessing](#). If not otherwise stated use the default values.

The first step is setting the processing sampling, in this example it is 30 seconds. The processing interval usually is a single 24-hour day, ▶ [timeSeries:uniformSampling](#) with ▶ [timeStart=<mjd>](#), ▶ [timeEnd=<mjd>+1](#), ▶ [sampling=30/86400](#) (processing sampling).

Add the appropriate ▶ [transmitters:gnss](#) (e.g. GPS, GLONASS, and Galileo) and provide the required files (from [Data preparation \(3.3.1\)](#)):

- **inputfileOrbit**
- **inputfileAttitude**
- **inputfileClock**

The following settings are needed in **receiver:stationNetwork**:

- **inputfileStationList**: list of all stations to be processed
- **inputfileObservations**: The converted RINEX observation file.
- **tidalDisplacement**: Use the settings described in [receiver:stationNetwork \(5.16.1\)](#).
- **useType**: We recommend to explicitly specify the signals to be processed and to make sure that at least transmitter code biases are provided for each of them, e.g. C1CG, C1WG, C2WG, L1\*G, L2\*G, ...).
- **excludeType**: Signals you might want to exclude are L5\*G (GPS L5 phase due to time-variable bias on block IIF satellites), \*3\*R (GLONASS G3 freq.), \*6\*E (Galileo E6 freq.)

Add the following **parametrizations** and define the **outputfiles** you are interested in inside each of them:

- **ionosphereSTEC**: add a constraint of **sigmaSTEC=40**
- **ionosphereVTEC**
- **clocks**: delete **selectTransmitters**
- **signalBiases**: provide **inputfileSignalBiasTransmitter** from [Data preparation \(3.3.1\)](#).
- **ambiguities**: if precise transmitter phase biases are available you can delete **estimateTransmitterPhaseBiases**
- **codeBiases**: delete **selectTransmitters**, set **sigmaZeroMeanConstraint=0**
- **tecBiases**: delete **selectTransmitters**, set **sigmaZeroMeanConstraint=0**
- **kinematicPositions**
- **troposphere**: select **troposphere:viennaMapping** with the appropriate vmf3grid file. Add **troposphereWetEstimation:splines** with linear (**degree=1**) 2-hourly splines and **troposphereGradientEstimation:splines** with linear daily splines.
- **constraints**: loose constraint troposphere estimates, **parameters:wildcard:type=troposphere\***, **sigma=5**, and **relativeToApriori=yes**

Add the following **processingSteps**:

- **estimate**: with **maxIterationCount=8**
- **resolveAmbiguities**
- **estimate**: with **maxIterationCount=2**
- **writeResults**

When processing multiple stations at the same time, moving **estimate** and **resolveAmbiguities** into the processing step **forEachReceiverSeparately** sets up and solves the normal equations independently for each station.

## 3.4 Kinematic orbit determination of LEO satellites

This cookbook chapter describes exemplarily the steps for determining kinematic orbits of low-Earth orbit (LEO) satellites.

An example scenario for this task is available at <https://ftp.tugraz.at/outgoing/ITSG/groops/scenario/scenarioLeoKinematicOrbit.zip>. It includes GROOPS scripts and data for the example, but not the general GROOPS data and metadata found at <https://ftp.tugraz.at/outgoing/ITSG/groops> (data folder or zipped archive). The scenario generally represents what is described in this cookbook, but may slightly differ in certain settings.

### 3.4.1 Prepare GNSS satellite data

Most of the required metadata files are provided in GROOPS file formats at <https://ftp.tugraz.at/outgoing/ITSG/groops>. These files are regularly updated.

Data that has to be gathered from other sources comprises:

- **Receiver observations:** GNSS measurements converted from RINEX format (see ▶ [RinexObservation2GnssReceiver](#))
- **Precise orbits:** precise orbits in CRF for orbit integration (see ▶ [Sp3Format2Orbit](#))
- **Precise clocks:** precise clocks (see ▶ [GnssClockRinex2InstrumentClock](#))
- **Attitude:** rotation from body frame to CRF (see ▶ [SimulateStarCameraGnss](#) or ▶ [GnssOrbex2StarCamera](#))
- **Signal biases:** code (and phase) biases (see ▶ [GnssSinexBias2SignalBias](#))

Receiver observations, precise satellite orbits and clocks, and possibly attitude and signal biases can be downloaded from the [IGS Data Centers](#). GPS, GLONASS, and Galileo orbits, clocks, attitude, and signal biases for the period 1994-2020 are also available as part of [Graz University of Technology's contribution to IGS repro3](#).

The [example scenario](#) includes a small set of this data. The script `010groopsConvert.xml` can be used to convert these external formats into GROOPS formats.

### 3.4.2 Prepare LEO metadata

Metadata for several LEO missions is available at <https://ftp.tugraz.at/outgoing/ITSG/groops/data/gnss/receiverLowEarthOrbiter>.

If you want to process another mission, you can create the necessary files with these steps:

- For creating the ▶ [GnssSatelliteInfo](#) file use ▶ [PlatformCreate](#). Note that the rotation from the satellite reference frame into the antenna reference frame, as well as the change of the center of mass due to fuel consumption, has to be considered here.
- The ▶ [GnssReceiverDefinition](#) file can be created by using ▶ [GnssReceiverDefinitionCreate](#). Here you can specify which GNSS signal types the receiver observes.
- For creating the ▶ [GnssAntennaDefinition](#) file use ▶ [GnssAntennaDefinitionCreate](#). Here you can define phase center offsets for the antenna.

- For determining the elevation dependent accuracies the program **GnssAntennaDefinitionCreate** is used again.
  - **antenna**: set to new
  - Set the **patterns** for code (**type=C\*\***) and phase (**type=L\*\***). The standard deviation is expressed e.g. with **values=0.001/cos(2\*PI/180\*zenith)**.

### 3.4.3 Prepare LEO data

The [example scenario](#) includes a small set of this data for the GRACE-FO mission. The script `02groopsConvertGracefo.xml` can be used to convert these external formats into GROOPS formats.

The data preparation steps are:

- Conversion of the approximate orbit and star camera data into GROOPS format using a conversion program.
- If no attitude data is given the star camera data can be simulated by using **SimulateStarCamera** or **SimulateStarCameraSentinel1**.
- The GNSS observation data (given in RINEX format) can be converted with **RinexObservation2GnssReceiver**.
- Suitable programs to get daily data are **InstrumentConcatenate** and **InstrumentSynchronize**.
- For interpolating the orbit and star camera data to GNSS receiver epochs use **InstrumentResample** and provide the converted RINEX observation file as input for **timeSeries:instrument**.
- For synchronizing these data use **InstrumentSynchronize**.

Detailed description of instrument data handling can be found in [Instrument data handling \(3.1\)](#).

### 3.4.4 Estimate kinematic orbits

The script `03groopsGnssProcessing.xml` in the [example scenario](#) implements the following steps and settings.

These are the settings for **GnssProcessing**. If not otherwise stated use the default values.

As we have only one receiver the processing sampling can be directly taken from the observation file: **timeSeries:instrument**.

Add the appropriate **transmitters:gnss** (e.g. GPS) and provide the required files (from [Prepare GNSS satellite data \(3.4.1\)](#)):

- **infileOrbit**
- **infileAttitude**
- **infileClock**

The following files (from [Prepare LEO metadata \(3.4.2\)](#) and [Prepare LEO data \(3.4.3\)](#)) and settings are needed in **receiver:lowEarthOrbiter**:

- **inputfileStationInfo**: The satellite info file.
- **inputfileAntennaDefinition**
- **inputfileReceiverDefinition**
- **inputfileAccuracyDefinition**
- **inputfileObservations**: The converted RINEX observation file.
- **inputfileOrbit**: The approximate orbit.
- **inputfileStarCamera**: The convered or simulated attitude.
- **useType**: We recommend to explicitly specify the signals to be processed and to make sure that at least transmitter code biases are provided for each of them, e.g. C1CG, C1WG, C2WG, L1\*G, L2\*G, ...).

Add the following **parametrizations** and define the **outputfiles** within you are interested in:

- **ionosphereSTEC**: add a constraint of **sigmaSTEC=40**
- **ionosphereVTEC**: set **mapR=6371e3+450e3** to satellite height and ionosphere height **mapH=50e3** above.
- **clocks**: delete **selectTransmitters**
- **signalBiases**: provide **inputfileSignalBiasTransmitter** created with **GnssSinexBias2SignalBias**
- **ambiguities**: if precise transmitter phase biases are available you can delete **estimateTransmitterPhaseBiases**
- **codeBiases**: delete **selectTransmitters**, set **sigmaZeroMeanConstraint=0**
- **tecBiases**: delete **selectTransmitters**, set **sigmaZeroMeanConstraint=0**
- **kinematicPositions**

Add the following **processingSteps**:

- **estimate**: with **maxIterationCount=8**
- **resolveAmbiguities**
- **estimate**: with **maxIterationCount=2**
- **computeCovarianceMatrix**
- **writeResults**

## 3.5 Gravity field determination from precise orbit data (POD)

This cookbook chapter describes exemplarily the steps for determining the monthly gravity variations from precise orbit data (POD).

### 3.5.1 Step 1: Preparation of data

Following data have to be prepared monthly with an adequate sampling, e.g. 10 s using **InstrumentConcatenate**:

- Precise (kinematic) orbit data
- 3x3 covariance matrices data
- Initial orbit data used for precise orbit determination
- Star camera data
- Accelerometer data

Reduced sampling can be achieved by **InstrumentReduceSampling**. If the satellite mission does not provide any required accelerometer data, these data can be generated via **SimulateAccelerometer**.

For satellite missions with less knowledge about the acting forces, it make sense to consider more than one state vector within an orbit revolution. Otherwise the accuracy of the estimated parameters will decrease. This implies that shorter arcs are necessary. The assignment of the kinematic orbit data as well as the 3x3 covariance matrices data to the arcs can be done with **InstrumentSynchronize**.

### 3.5.2 Step 2: Conversion of the background gravity field

**Gravityfield2SphericalHarmonicsVector** converts the static respectively background gravity field into spherical harmonics.

### 3.5.3 Step 3: Preprocessing POD

For determining the accuracies and weights of the kinematic orbits it is sufficient to make a least-square estimation with only certain parameters, due to the fact that some parameters do not influence the estimation of the accuracies and weights. This estimation is done with **PreprocessingPod**. Additional this program determines the temporal correlation of the kinematic orbit positions x,y and z. If short arcs are used the setting **observation:podIntegral** shall be used. This setting considers the frictional forces by means of a macro model as well as the conservative and non-conservative forces.

### 3.5.4 Step 4: Solving of normal equations system

**NormalsSolverVCE** sets up the observation equations and summarized them to a normal equations system. The subsequent least-square estimation delivers the parameters surcharges.

### 3.5.5 Step 5: Determination of the estimated gravity field parameters

The estimated parameters result from the re-addition of the background field, which is done in **MatrixCalculate**.

### 3.5.6 Step 6: Conversion of the gravity field parameters

► [Gravityfield2PotentialCoefficients](#) converts the gravity field parameters into spherical harmonics.

## 3.6 GRACE gravity field recovery

This cookbook chapter describes an example of estimating a gravity field solution using GRACE observation data. For the respective month a set of spherical harmonic coefficients up to a maximum degree is determined. An example scenario for this task can be found at <https://ftp.tugraz.at/outgoing/ITSG/groops/scenario/scenarioGraceGravityfieldRecovery.zip> including the required GROOPS scripts and data sets for the gravity field recovery process. The background models are provided at <https://ftp.tugraz.at/outgoing/ITSG/groops/data/>.

### 3.6.1 Background models

The following background models were used during the data processing:

- **Earth rotation:** [IERS 2010](#)
- **Moon, sun and planets ephemerides:** [JPL DE432](#)
- **Earth tide:** [IERS 2010](#)
- **Ocean tide:** [FES2014b](#)
- **Pole tide:** [IERS 2010](#)
- **Ocean pole tide:** [Desai 2004](#)
- **Atmospheric tides:** [AOD1B RL06](#)
- **Atmosphere and Ocean Dealiasing:** [AOD1B RL06](#)
- **Sub-monthly continental hydrology:** [LSDM \(ESMGFZ\)](#)
- **Relativistic corrections:** [IERS 2010](#)

These models were reduced during the analysis process and are not present in the solution. The [GOCO06s](#) model was used as the static gravity field as well as for the trend component and annual oscillation. In the script `000groopsBackgroundModels.xml` a monthly mean of the GOCO06s including the time-variable components is determined in form of time splines using  [Gravityfield2TimeSplines](#). This model is later added back to the final gravity solution.

### 3.6.2 Instrument data preparation

The ITSG gravity field solutions are computed from the official GRACE L1B [JPL \(2018\)](#) and GRACE-FO L1B [JPL \(2019\)](#) observation data. The data sets for this example are provided in GROOPS file format in the scenario folder.

The satellite-to-satellite-tracking (SST) data consists of:

- **K-band range rates**
- **Light time correction**
- **Antenna offset corrections**

Additional observation data required for the processing comprises:

- Star camera observations
- Accelerometer data
- Approximate orbits
- Thruster data

The determination of

- Kinematic orbits
- 3x3 epoch covariances

is depicted in [Kinematic orbit determination of LEO satellites \(3.4\)](#). These data sets are also provided in the scenario folder.

Data preparation is handled in the script `010groopsInstruments.xml`. The approximate orbits (initial dynamic orbits) of the satellites, the star camera observations, the accelerometer data and the thruster data are resampled with a 5s sampling and small gaps in the data are filled using  [InstrumentResample](#). Gross outliers are removed using  [InstrumentRemoveEpochsByCriteria](#) and the data is synchronized using  [InstrumentSynchronize](#).

The approximate orbits are later used as a priori information for the dynamic orbit integration. In addition to the observed orientation of the spacecrafts (star camera observations), the nominal orientation is computed using  [SimulateStarCameraGrace](#). The difference between observed and simulated orientation is determined using  [InstrumentStarCameraMultiply](#) and is employed in the outlier detection.

The accelerometer data is initially calibrated by estimating a bias using  [InstrumentAccelerometerEstimateBiasScale](#) with respect to simulated data created with  [SimulateAccelerometer](#). For simulating accelerometer data a satellite model implying the satellite's mass and surfaces is required. Such a model can be created with  [SatelliteModelCreate](#). Models for the GRACE and GRACE-FO satellites are also provided at <https://ftp.tugraz.at/outgoing/ITSG/groops/data/satelliteModel/>. Non-gravitational forces comprising atmospheric drag, solar radiation pressure and albedo have to be modeled when simulating the accelerometer data. The acceleration bias parameters are determined as degree 3 time splines with 6h nodes. When determining these parameters the thruster events are excluded from the estimation.

The SST observations, the light time corrections and the antenna center corrections are synchronized with a 5s sampling together with simulated SST data created with  [SimulateSatelliteTracking](#). Simulated data is used for the outlier detection of the original SST observations.

The sampling of the kinematic orbits is reduced to 60s using  [InstrumentReduceSampling](#) and an outlier detection is performed using the approximate dynamic orbits.

The approximate orbits, the star camera observations and the accelerometer data are divided into 24h arcs (variational arcs). The kinematic orbits, its 3x3 epoch covariances, KBR observations, light time corrections, antenna center corrections and star camera observations are divided into 3h arcs per day (short arcs). Additionally the approximate orbits and the star camera observations are also synchronized to short arcs.

Further information on instrument data preparation can be found in [Instrument data handling \(3.1\)](#).

### 3.6.3 Variational equations

In this processing step dynamic orbits are computed for a complete 24h orbit arc by integrating the forces acting on the GRACE/GRACE-FO satellites. Additionally, the state transition matrix is set up. The dynamic orbits are then fitted to kinematic orbits and SST observations in a least squares adjustment by co-estimating additional accelerometer calibration parameters together with the initial state vector. The newly estimated parameters are then used to re-estimate the dynamic orbits and setting up the new state transition matrix.

The script `020groopsVariational.xml` in the scenario folder implements the required processing steps. Time splines from a time-variable gravity field are estimated using **Gravityfield2TimeSplines**. In this step the static gravity field (GOCO06s) is combined with the following time-variable components:

- **gravityfield:potentialCoefficients**: static gravity field
- **gravityfield:trend**
  - **gravityfield:potentialCoefficients**: trend component of gravity field
- **gravityfield:oscillation**
  - **gravityfieldCos:potentialCoefficients**: annual cosine component of gravity field
  - **gravityfieldSin:potentialCoefficients**: annual sine component of gravity field
- **gravityfield:timeSplines**: atmosphere and ocean dealiasing (AOD1B RL06)
- **gravityfield:timeSplines**: hydrology dealiasing (LSDM)
- **doodsonHarmonicTide**: ocean tides (FES2014b)
- **doodsonHarmonicTide**: atmospheric tides (AOD1B RL06)
- **tides:poleTide**: pole tides (IERS 2010)
- **tides:poleOceanTide**: ocean pole tides (IERS 2010)

**maxDegree=220** and **sampling=10/1440** is sufficient.

In **PreprocessingVariationalEquation** the **Variational equations** comprising the integrated orbit together with the state transition matrix are stored in **outputfileVariational**.

This program has to be executed for both GRACE or GRACE-FO satellites and it is recommended to use **LoopPrograms**.

- **infileSatelliteModel**: satellite model from `020groopsInstruments.xml`
- **infileOrbit**: the approximate orbits from `020groopsInstruments.xml`
- **infileStarCamera**: the attitude file from `020groopsInstruments.xml`
- **infileAccelerometer**: the accelerometer data from `020groopsInstruments.xml`
- **forces**: see below
- **ephemerides**: JPL DE432
- **gradientfield:potentialCoefficients**: a static gravity field (GOCO06s) with **maxDegree=10** is more than sufficient.

The **force models** include:

- **gravityfield:timeSplines**: the previously estimated time-variable gravity field
- **tides:astronomicalTide**: astronomical tides (based on JPL DE432 ephemerides)
- **tides:earthTide**: Earth tide (IERS conventions)
- **miscAccelerations:relativisticEffect**: relativistic effects (IERS conventions)

In **PreprocessingVariationalEquationOrbitFit** the integrated orbit (**inputfileVariational**) is fitted to the kinematic orbit (**inputfileOrbit**) by least squares adjustment. The additional accelerometer calibration parameters can be defined by

- **parametrizationAcceleration**: accelerometer scale factor (once per day)
- **parametrizationAcceleration**: accelerometer bias (time spline with 6h nodes)

The observation equations (parameter sensitivity matrix) are computed by integration of the variational equations (**inputfileVariational**) using a polynomial with **integrationDegree=7**. **PreprocessingVariationalEquationOrbitFit** has to be executed per satellite.

**PreprocessingVariationalEquationSstFit** fits two dynamic orbits **inputfileVariational1/2** to the SST observations and the kinematic orbits.

- **rightHandSide**: input for observation vectors
  - **inputfileSatelliteTracking**: K-band range rate observations
  - **inputfileSatelliteTracking**: light time correction
  - **inputfileSatelliteTracking**: antenna offset corrections
  - **inputfileOrbit1**: kinematic orbit of satellite 1
  - **inputfileOrbit2**: kinematic orbit of satellite 2
- **sstType**: rangeRate
- **inputfileVariational1**: dynamic orbit and integrated state matrix of satellite 1
- **inputfileVariational2**: dynamic orbit and integrated state matrix of satellite 2
- **parametrizationAcceleration1**: same as in In **PreprocessingVariationalEquationOrbitFit**
- **parametrizationAcceleration2**: same as in In **PreprocessingVariationalEquationOrbitFit**
- **integrationDegree**: 7
- **interpolationDegree**: 7
- **covarianceSst**
  - **sigma**: 1
- **covariancePod1**
  - **sigma**: 1
  - **inputfileCovariancePodEpoch**: 3x3 epoch covariances
- **covariancePod2**
  - **sigma**: 1
  - **inputfileCovariancePodEpoch**: 3x3 epoch covariances

The estimated accelerometer calibration parameters from **PreprocessingVariationalEquationOrbitFit** and **PreprocessingVariationalEquationSstFit** are determined as corrections and stored in **outputfileSolution**. Both correction estimates have to be summed up using **FunctionsCalculate**.

The dynamic orbit and the resulting accelerometer calibration parameters are now used to re-integrate the orbit once more using **PreprocessingVariationalEquation** and introducing **parametrizationAcceleration** as **estimatedParameters**. This step usually ensures convergence. If the maximum orbit difference is still not sufficient this step can be repeated again.

### 3.6.4 Preprocessing

The script `030groupsPreprocessing.xml` implements the following steps and settings. The program **PreprocessingSst** processes SST observations and kinematic orbit data and performs a complete least squares adjustment for gravity field determination by computing the observations equations. Force model parameters (gravitational potential coefficients and accelerometer calibration parameters) are computed by integrating the parameter sensitivity matrix from the variational equations. Parameters describing effects due to the SST observation system and geometry (KBR antenna phase center variations) are computed using the dynamic orbits as a Taylor point. Short time gravity variations can be co-estimated together with the monthly mean gravity field. The autoregressive model sequence constraining the short time parameters is provided in the data folder. See [Kvas 2019](#) for more information about this co-estimation.

- **observation:** sstVariational
  - **rightHandSide:**
    - \* **inputfileSatelliteTracking:** KBR range rates
    - \* **inputfileSatelliteTracking:** light time correction
    - \* **inputfileSatelliteTracking:** antenna offset corrections
    - \* **inputfileOrbit1:** kinematic orbit of satellite 1
    - \* **inputfileOrbit2:** kinematic orbit of satellite 2
  - **sstType:** rangeRate
  - **inputfileVariational1:** dynamic orbit and integrated state matrix of satellite 1
  - **inputfileVariational2:** dynamic orbit and integrated state matrix of satellite 2
  - **ephemerides:** JPL DE432
  - **parametrizationGravity:** spherical harmonics from **minDegree=2** to **maxDegree=60**
  - **parametrizationGravity:** high frequency parametrization
  - **parametrizationAcceleration1:** same as in In **PreprocessingVariationalEquationOrbitFit**
  - **parametrizationAcceleration2:** same as in In **PreprocessingVariationalEquationOrbitFit**
  - **parametrizationSst:** antenna phase center variations (y and z for both satellites)
  - **integrationDegree:** 7
  - **interpolationDegree:** 7
- **covarianceSst**
  - **sigma:** 1e-7
  - **sampling:** 5 [seconds]
- **covariancePod1/2**
  - **sigma:** 2
  - **inputfileCovariancePodEpoch:** 3x3 epoch covariances

- **sampling:** 60 [seconds]
- **estimateShortTimeVariations**
  - **autoregressiveModelSequence:** AR models
  - **parameterSelection:** names
    - \* **parameterName:** parametrizationGravity
    - **parametrization:** high frequency parametrization

**ParameterSelection2IndexVector** and **MatrixCalculate** with **matrix:reorder** can be used to extract the desired spherical harmonic coefficients from **outputfileSolution** and the respective standard deviations from **outputfileSigmax** up to a certain degree.

In the program **Gravityfield2PotentialCoefficients** the estimated spherical harmonics coefficients are read with **gravityfield:fromParametrization**. The monthly mean gravity field can be added back by additionally selecting the time splines created in `000groopsBackgroundModels.xml` using **gravityfield:timeSplines**. The preprocessing solution is saved as a **spherical harmonics file**.

### 3.6.5 Setting up normal equations

Normal equations are set up in the script `040groopsMonthlyNormals120.xml` using the program **NormalsBuildShortTimeStaticLongTime**. The time intervals which the normal equations are divided into are defined in **inputfileArcList**. The normal equations are based on **observation** including the SST data, the kinematic orbits and the variational equations. The parametrization of the gravity field can be set with **observation:parametrizationGravity** (e.g. spherical harmonics up to degree and order 120). Accelerometer calibration parameters and KBR antenna phase center variations can be parameterized using **parametrizationAcceleration** and **parametrizationSst**. With **estimateShortTimeVariations** short time variations of the gravity field can be co-estimated. The parameters selected by **parameterSelection** (e.g. linear splines with 6h nodes) are constrained by an **autoregressiveModelSequence**. Additional temporal variations (e.g. trend and annual oscillation) could be estimated with **estimateLongTimeVariations**.

### 3.6.6 Solving normal equations

The desired spherical harmonic coefficients are determined in the script `050groopsMonthlySolve.xml`. **NormalsSolverVCE** accumulates **normalEquation** and solves the total combined system. Variance component estimation is used to determine the relative weighting of the individual normals. The estimated parameter vector (**outputfileSolution**) the estimated accuracies (**outputfileSigmax**) and the full covariance matrix (**outputfileCovariance**) can be saved. Using **Gravityfield2PotentialCoefficients** the final solution can be saved as a **spherical harmonics file** by adding back the monthly mean gravity field to the estimated spherical harmonic coefficients.

## 3.7 Regional geoid determination

This shows exemplary the computation of a regional geoid using terrestrial gravimetric observations in combination with a global satellite model such as GOCO06s. The geoid is estimated in a least squares adjustment with a parametrization using radial basis functions. A detailed description of the method is given in Christian Pock (2017), Consistent Combination of Satellite and Terrestrial Gravity Field Observations in Regional Geoid Modeling, Dissertation TU Graz.

### 3.7.1 Gravimetric data

Here it is assumed that the measured absolute gravity data is given at points in ellipsoidal coordinates. The observed values should be converted to SI units  $m/s^2$ .

- **Matrix2GriddedData** to convert data from text file in tabular form.



Figure 3.7: Distribution of gravimetric observations

### 3.7.2 Topography

A high resolution topography model is needed to reduce the observations. As the model heights are usually given in physical heights a reference geoid is needed to compute the correct ellipsoidal height.

- **NetCdf2GridRectangular** convert into groops format.
- **Gravityfield2GriddedData**: Compute geoid heights using the GOCO06s model with
  - **grid:file** The topography grid
  - **kernel:geoidHeight**
  - **gravityfield:potentialCoefficients** the GOCO06s model
  - **gravityfield:potentialCoefficients** Subtract ( **factor**=-1) GRS680 normal field.
- **GriddedDataCalculate**: Generate a new combined griddedData file with the orthometric height (**data0**) and the geoid height (**data1**).
- **GriddedTopography2PotentialCoefficients**: Compute the gravitational potential in terms of spherical harmonics up to a maximum degree of the global satellite model. This is the part of the topography, which is already included in the global satellite model. The integration boundaries are **radialUpperBound**=**data0+data1** and **radialLowerBound**=**data1**.

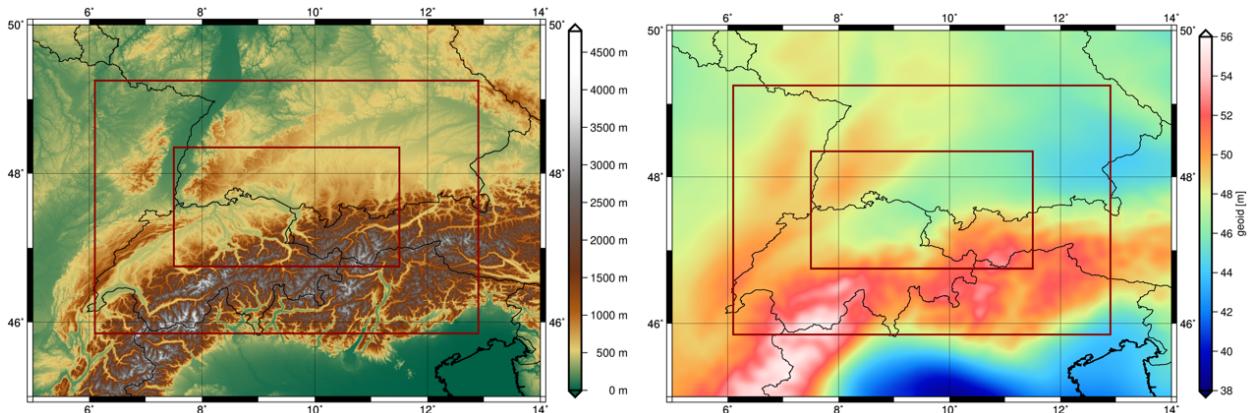


Figure 3.8: Topography and geoid heights

### 3.7.3 Reduce

Calculate approximate reference gravity to reduce it from the observations.

- **Gravityfield2AbsoluteGravity**
  - **grid:File** at observation positions
  - **gravityfield:tides** Centrifugal potential
  - **gravityfield:potentialCoefficients** full GOCO06s model
  - **gravityfield:topography** (**radialUpperBound**=**data0+data1**, **radialLowerBound**=**data1**)
  - **gravityfield:potentialCoefficients** Subtract (**factor**=-1) the potential part of the topography already included in the GOCO06s model.
- **GriddedDataCalculate** to calculate observed minus computed.
- Large outliers can be removed in **GriddedDataCalculate** with **removalCriteria**.

### 3.7.4 Radial Basis Functions (RBF)

The residual gravity is parametrized in terms of Radial Basis Functions **parametrizationGravity:radialBasis**. The basis functions should be distributed on a regular **grid** covering a somewhat larger area than the observations, see **border**. The shape of the functions **kernel:coefficients** should reflect the signal content of reduced observations and are defined by the coefficients.

- **RadialBasisSplines2KernelCoefficients**
  - **gravityfield:potentialCoefficients** accuracies of GOCO06s model
  - **maxDegree**=7000. Complemented by Kaula's rule of thumb

The maximum degree should correspond to the spatial resolution. Rule of thumb: the number of spherical harmonic coefficients ( $\text{maxDegree} + 1$ )<sup>2</sup> should roughly agree to the number of grid points if they would cover the complete Earth.

### 3.7.5 Compute: Estimate parameters in a least squares adjustment

Setup the observation equations and accumulate the system of normal equations.

- **NormalsBuild**
  - **normalEquation:design** with **observation:terrestrial**
    - \* **kernel:disturbance**
    - \* **parametrizationGravity:radialBasis** with **kernel:coefficients**
- **NormalsSolverVCE**
  - **normalEquation:file** from **NormalsBuild**
  - **normalEquation:regularization** towards zero means regularization towards the GOCO06s, which is reduced from the data.



Figure 3.9: Gravity disturbances: observed minus computed



Figure 3.10: Degree amplitudes for the shape of the radial basis functions

### 3.7.6 Restore: Calculate the geoid solution

Evaluate the estimated parameters and add back the reduced reference models.

- **Gravityfield2GriddedData:** Compute approximate geoid heights using the GOCO06s model with
  - **grid** select a grid with target resolution at ellipsoid
  - **kernel:geoidHeight**
  - **gravityfield:potentialCoefficients** GOCO06s model
  - **gravityfield:potentialCoefficients** Subtract (**factor**=-1) GRS80 normal field.
- **GriddedDataCalculate:** Move points from ellipsoid to geoid with **height**=**data0**
- **Gravityfield2GriddedData**
  - **grid:file** from above.
  - **kernel:geoidHeight**
  - **gravityfield:fromParametrization** The solution vector **x** together with the RBF parametrization
  - **gravityfield:potentialCoefficients** GOCO06s model
  - **gravityfield:topography** (**radialUpperBound**=**data0+data1**, **radialLowerBound**=**data1**)
  - **gravityfield:potentialCoefficients** Subtract (**factor**=-1) the potential part of the topography already included in the GOCO06s model.
  - **gravityfield:potentialCoefficients** Subtract (**factor**=-1) GRS80 normal field.
  - **convertToHarmonics**=no, otherwise the RBF are converted to harmonics up to degree 7000.
- **GriddedDataCalculate:** Set **height**=0 of the computed geoid grid.
- **GridRectangular2NetCdf**



Figure 3.11: Estimated geoid

# Chapter 4

## Programs

This chapter details programs included in GROOPS, describing what they are and what they do. For usage examples see the cookbook in Chapter 3.

### 4.1 Programs: Covariance

#### 4.1.1 AutoregressiveModel2CovarianceMatrix

This program computes the covariance structure of a random process represented by an AR model sequence. The covariance matrix is determined by accumulating the normal equations of all AR models in **autoregressiveModelSequence** and inverting the combined normal equation matrix. For each output file in **outputfileCovarianceMatrix**, the covariance matrix of appropriate time lag is saved (the first file contains the auto-covariance, second file cross covariance and so on). The matrix for lag  $h$  describes the covariance between  $x_{t-h}$  and  $x_t$ , i.e.  $\Sigma(t-h, t)$ .

Name	Type	Annotation
outputfileCovarianceMatrix	filename	covariance matrix for each lag
autoregressiveModelSequence	autoregressiveModelSequence	AR model sequence

### 4.1.2 CovarianceFunction2DigitalFilter

Computes digital filter coefficients for a **digital filter** of given degree and order. The filter coefficients are computed by fitting them to an approximated impulse response represented by the cholesky factor of the covariance matrix.

The parameter **warmup** determines from which element of the cholesky matrix the coefficients (default: half the covariance length) are fitted.

Per default, the program computes filter coefficients which generate colored noise when applied to a white noise sequence. When **decorrelationFilter** is set, a decorrelation filter is computed which yields white noise when applied to colored noise.

Name	Type	Annotation
outputfileFilter	filename	filter coefficients
inputfileCovariance	filename	first column: time steps, following columns: covariance functions
column	uint	Column with covariance function to be fitted
warmup	uint	number of samples until diagonal of Cholesky factor is flat (default: half covariance length)
numeratorDegree	uint	Maximum degree of numerator polynomial (MA constituent)
denominatorDegree	uint	Maximum degree of denominator polynomial (AR constituent)
decorrelationFilter	boolean	compute a decorrelation filter

### 4.1.3 CovarianceFunction2PowerSpectralDensity

One sided Power Spectral Density (PSD) from a covariance function. The first column of **inputfileCovarianceFunction** should contain the time lag in seconds. Multiple covariance functions (in the following column)s are supported. The output is a **matrix** with first column contains the frequency [Hz] and the other columns the PSD [ $\text{unit}^2/\text{Hz}$ ].

Conversion between covariance function  $c_j$  and PSD  $p_k$  is performed by discrete cosine transformation:

$$p_k = 2\Delta t \left( c_0 + c_{n-1}(-1)^k + \sum_{j=1}^{n-2} 2c_j \cos(\pi j k / (n-1)) \right). \quad (4.1)$$

See also **PowerSpectralDensity2CovarianceFunction**.

---

Name	Type	Annotation
<b>outputfilePSD</b>	filename	first column: frequency [Hz], other columns PSD [ $\text{unit}^2/\text{Hz}$ ]
<b>inputfileCovarianceFunction</b>	filename	first column: time steps, following columns: covariance functions

---

#### 4.1.4 CovarianceMatrix2AutoregressiveModel

This program computes a VAR(p) model from empirical covariance matrices. The **inputfileCovarianceMatrix** represent the covariance structure of the process: the first file should contain the auto-covariance, the second the cross-covariance of lag one, the next cross-covariance of lag two and so on.

Cross-covariance matrices  $\Sigma_{\Delta_k}$  are defined as the cross-covariance between epoch  $t - k$  and  $t$ . If the process realizations  $x_t$  are arrange by ascending time stamps ( $\{ \dots, x_{t-2}, x_{t-1}, x_t, x_{t+1}, x_{t+2}, \dots \}$ ), the covariance structure of the (stationary) process is therefore given by

$$\begin{bmatrix} \Sigma & \Sigma_{\Delta_1} & \Sigma_{\Delta_2} & \dots \\ \Sigma^T_{\Delta_1} & \Sigma & \Sigma_{\Delta_1} & \dots \\ \Sigma^T_{\Delta_2} & \Sigma^T_{\Delta_1} & \Sigma & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \quad (4.2)$$

The estimate AR model is saved as single matrix **outputfileAutoregressiveModel** according to the GROOPS AR model conventions.

Name	Type	Annotation
☞ <b>outputfileAutoregressiveModel</b>	filename	coefficients and white noise covariance of AR(p) model
☞ <b>inputfileCovarianceMatrix</b>	filename	file name of covariance matrix

### 4.1.5 CovarianceMatrix2Correlation

This program computes the pearson correlation coefficient

$$\rho_{ij} = \frac{\sigma_{ij}}{\sigma_i \sigma_j} \quad (4.3)$$

from a given covariance matrix stored in **inputfileCovarianceMatrix**. The result is stored in **outputfileCorrelationMatrix**.

Name	Type	Annotation
<b>outputfileCorrelationMatrix</b>	filename	correlation matrix
<b>inputfileCovarianceMatrix</b>	filename	covariance matrix

This program is parallelized (1.5).

#### 4.1.6 PowerSpectralDensity2CovarianceFunction

Covariance function from Power Spectral Density (PSD). The **inputfilePSD** contains in the first column the frequency [Hz], followed by (possibly multiple) PSDs [ $\text{unit}^2/\text{Hz}$ ]. The output is a **matrix**, the first column containing time lag [s] and the other columns the covariance functions [ $\text{unit}^2$ ]. Conversion between PSD  $p_j$  and covariance function  $c_k$  is performed by discrete cosine transformation:

$$c_k = \frac{1}{4\Delta t(n-1)} \left( p_0 + p_{n-1}(-1)^k + \sum_{j=1}^{n-2} 2p_j \cos(\pi j k / (n-1)) \right). \quad (4.4)$$

See also **CovarianceFunction2PowerSpectralDensity**.

Name	Type	Annotation
<b>outputfileCovarianceFunction</b>	filename	first column: time steps [seconds], following columns: covariance functions
<b>inputfilePSD</b>	filename	first column: frequency [Hz], following columns PSD [ $\text{unit}^2/\text{Hz}$ ]

## 4.2 Programs: DoodsonHarmonics

### 4.2.1 DoodsonAdmittanceInterpolation

To visualize the interpolation of the minor tides. The output is a [matrix](#) with the first column containing the tidal frequency, the second column is the tide generating amplitude (from [inputfileTideGeneratingPotential](#)), and the following columns the contribution of the major tides to the this tidal frequency as defined in [inputfileAdmittance](#).



Figure 4.1: Linear interpolation of minor tides in the diurnal band.

Name	Type	Annotation
<a href="#">outputfile</a>	filename	
<a href="#">inputfileAdmittance</a>	filename	interpolation of minor constituents
<a href="#">inputfileTideGeneratingPotential</a>	filename	

### 4.2.2 DoodsonAdmittanceTimeSeries

To visualize the interpolation of the minor tides it computes cosine multipliers of all major tides. Without admittance this would be a simple cos oscillation. The  **outputfileTimeSeries** is an  **instrument** file (MISCVALUES) containing the cos of all the major tides.



Figure 4.2: Cosine of the Mf tidal frequency with modulation from the interpolated minor tides.

Name	Type	Annotation
 <b>outputfileTimeSeries</b>	filename	MISCVALUES (cos of major tides, ...)
 <b>inputfileAdmittance</b>	filename	cos/sin multipliers of the major tides
 <b>timeSeries</b>	timeSeries	

### 4.2.3 DoodsonArguments2TimeSeries

Time series of doodson/fundamental arguments. The **outfileTimeSeries** contains the six Doodson arguments, followed by the five fundamental arguments in radians.

Name	Type	Annotation
outfileTimeSeries	filename	each epoch: 6 doodson args, 5 fundamental args [rad]
timeSeries	timeSeries	

#### 4.2.4 DoodsonHarmonics2GriddedAmplitudeAndPhase

This program reads a **inputfileDoodsonHarmonics** and evaluates a single tidal constituent selected by **doodson** (Doodson number or Darwin's name, e.g. 255.555 or M2). This program computes the amplitude and phase from the cos and sin coefficients on a given **grid**. The type of functional (e.g gravity anomalies or geoid heights) can be chosen with **kernel**. The values will be saved together with points expressed as ellipsoidal coordinates (longitude, latitude, height) based on a reference ellipsoid with parameters **R** and **inverseFlattening**. To visualize the results use **PlotMap**.

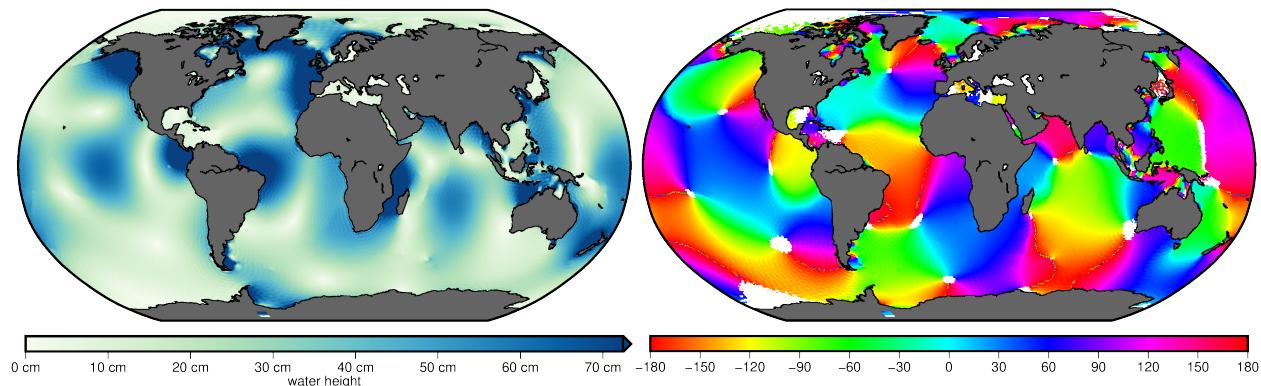


Figure 4.3: M2 amplitude and phase of FES2014b.

Name	Type	Annotation
outputfileGrid	filename	ampl, phase [-pi,pi], cos, sin
inputfileDoodsonHarmonics	filename	
doodson	doodson	tidal constituent
filter	sphericalHarmonicsFilter	
grid	grid	
kernel	kernel	
minDegree	uint	
maxDegree	uint	
factor	double	the values on grid are multiplied by this factor
R	double	reference radius for ellipsoidal coordinates on output
inverseFlattening	double	reference flattening for ellipsoidal coordinates on output, 0: spherical coordinates

This program is parallelized (1.5).

#### 4.2.5 DoodsonHarmonics2PotentialCoefficients

The **inputfileDoodsonHarmonics** contains a Fourier series of a time variable gravitational potential at specific tidal frequencies (tides)

$$V(\mathbf{x}, t) = \sum_f V_f^c(\mathbf{x}) \cos(\theta_f(t)) + V_f^s(\mathbf{x}) \sin(\theta_f(t)), \quad (4.5)$$

where  $V_f^c(\mathbf{x})$  and  $V_f^s(\mathbf{x})$  are spherical harmonics expansions. If set the expansions are limited in the range between **minDegree** and **maxDegree** inclusively. The coefficients are related to the reference radius **R** and the Earth gravitational constant **GM**.

The **outputfilePotentialCoefficients** is not a single file but a series of files. For each spherical harmonics expansion  $V_f^c(\mathbf{x})$  and  $V_f^s(\mathbf{x})$  a separate file is created where the variables **variableLoopName**, **variableLoopDoodson**, **variableLoopCosSin** are set accordingly. The file name should contain these variables, e.g. `coeff.{name}.{doodson}.{cossin}.gfc`.

If **applyXi** the Doodson-Warburg phase correction (see IERS conventions) is applied to the cos/sin potentialCoefficients before.

Name	Type	Annotation
<b>outputfilePotentialCoefficients</b>	filename	
<b>variableLoopName</b>	string	variable with darwins's name of each constituent
<b>variableLoopDoodson</b>	string	variable with doodson code of each constituent
<b>variableLoopCosSin</b>	string	variable with 'cos' or 'sin' of each constituent
<b>variableLoopIndex</b>	string	variable with index of each constituent (starts with zero)
<b>variableLoopCount</b>	string	variable with total number of constituents
<b>inputfileDoodsonHarmonics</b>	filename	
<b>inputfileTideGeneratingPotential</b>	filename	to compute Xi phase correction
<b>minDegree</b>	uint	
<b>maxDegree</b>	uint	
<b>GM</b>	double	Geocentric gravitational constant
<b>R</b>	double	reference radius
<b>applyXi</b>	boolean	apply Doodson-Warburg phase correction (see IERS conventions)

#### 4.2.6 DoodsonHarmonicsCalculateAdmittance

Computes the admittance function to interpolate minor tides from tides given in **inputfileDoodsonHarmonics** using **inputfileTideGeneratingPotential**.

Name	Type	Annotation
outputfileAdmittance	filename	
inputfileDoodsonHarmonics	filename	
inputfileTideGeneratingPotential	filename	TGP
threshold	double	[m^2/s^2] only interpolate tides with TGP greater than threshold
degreeInterpolation	uint	polynomial degree for interpolation
degreeExtrapolation	uint	polynomial degree for extrapolation
excludeDoodsonForInterpolation	doodson	major tides not used for interpolation

### 4.2.7 ModelEquilibriumTide

Computes the equilibrium ocean tide of the long periodic **tideGeneratingPotential**.

The ocean surface is represented by **gridOcean** and the gravitational effect is numerically integrated to spherical harmonics using **maxDegree**, **GM**, and **R**.

It takes self attraction and loading into account using the Love numbers **infilePotentialLoadLoveNumber** and **infileDeformationLoadLoveNumber**.

Additionally the effects of the solid Earth tide are considered, both the gravitational (Love numbers **k20**, **k20plus**) and the geometrical (Love numbers **h20\_0**, **h20\_2**) effect.

See also **PotentialCoefficients2DoodsonHarmonics**.

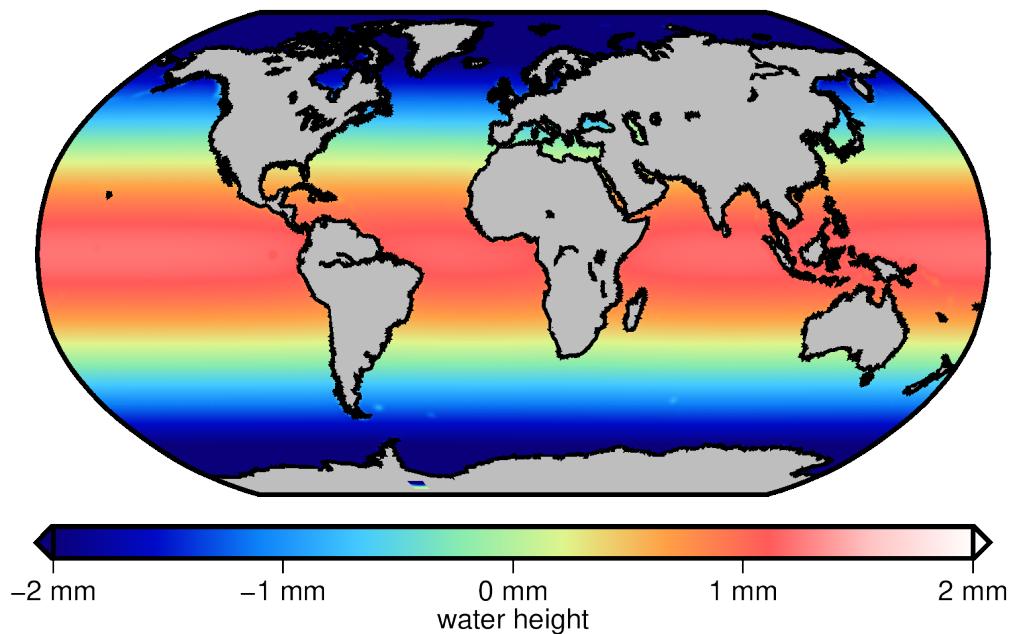


Figure 4.4: Equilibrium tide of SA constituent

Name	Type	Annotation
<b>outputfilePotentialCoefficients</b>	filename	includes the loading
<b>gridOcean</b>	grid	
<b>maxDegree</b>	uint	
<b>GM</b>	double	Geocentric gravitational constant
<b>R</b>	double	reference radius
<b>density</b>	double	[kg/m <sup>3</sup> ] density of sea water
<b>tideGeneratingPotential</b>	double	[m <sup>2</sup> /s <sup>2</sup> ]
<b>k20</b>	double	earth tide love number
<b>k20plus</b>	double	earth tide love number
<b>h20_0</b>	double	earth tide love number
<b>h20_2</b>	double	earth tide love number
<b>infilePotentialLoadLoveNumber</b>	filename	
<b>infileDeformationLoadLoveNumber</b>	filename	

This program is parallelized (1.5).

#### 4.2.8 PotentialCoefficients2DoodsonHarmonics

Create a [DoodsonHarmonic file](#) from a list of cos/sin [potentialCoefficients](#) for given [doodson](#) (Doodson number or Darwin's name, e.g. 255.555 or M2) tidal constituents. If [applyXi](#) the Doodson-Warburg phase correction (see IERS conventions) is applied before.

Name	Type	Annotation
<a href="#">outputfileDoodsonHarmonics</a>	filename	
<a href="#">inputfileTideGeneratingPotential</a>	filename	to compute Xi phase correction
<a href="#">constituent</a>	sequence	
<a href="#">doodson</a>	<a href="#">doodson</a>	
<a href="#">inputfileCosPotentialCoefficients</a>	filename	
<a href="#">inputfileSinPotentialCoefficients</a>	filename	
<a href="#">minDegree</a>	uint	
<a href="#">maxDegree</a>	uint	
<a href="#">GM</a>	double	Geocentric gravitational constant
<a href="#">R</a>	double	reference radius
<a href="#">applyXi</a>	boolean	apply Doodson-Warburg phase correction (see IERS conventions)

## 4.3 Programs: Gnss

### 4.3.1 GnssAntennaDefinition2ParameterVector

Estimates parameters of a parametrization of **► antennaCenterVariations**, which represents all antennas from **► inputFileAntennaDefinition** matching the wildcard patterns of **► name**, **► serial**, **► radome**.

The provided values at the grid points of the pattern of each gnssType are used as pseudo-observations. A subset of patterns can be selected with **► types**.

The **► GnssAntennaDefinition file** can be modified to the demands before with **► GnssAntennaDefinitionCreate**.

See also **► ParameterVector2GnssAntennaDefinition**.

Name	Type	Annotation
<b>► outputfileSolution</b>	filename	
<b>► outputfileParameterNames</b>	filename	
<b>►* antennaCenterVariations</b>	<a href="#">parametrizationGnssAntenna</a>	
<b>►* inputFileAntennaDefinition</b>	filename	
<b>► name</b>	string	
<b>► serial</b>	string	
<b>► radome</b>	string	
<b>► types</b>	<a href="#">gnssType</a>	if not set, all types in the file are used

### 4.3.2 GnssAntennaDefinition2Skyplot

Produce a **skypplot** of antenna center variations which can be plotted with **PlotMap**.

The first antenna from **inputfileAntennaDefinition** matching the wildcard patterns of **name**, **serial**, **radome** is used.

For each antenna pattern (**gnssType**) a separate data column is computed. A subset of patterns can be selected with **types**.

Azimuth and elevation are written as ellipsoidal longitude and latitude in a **griddedData file**. The chosen ellipsoid parameters **R** and **inverseFlattening** are arbitrary but should be the same as in **grid** and **PlotMap**.

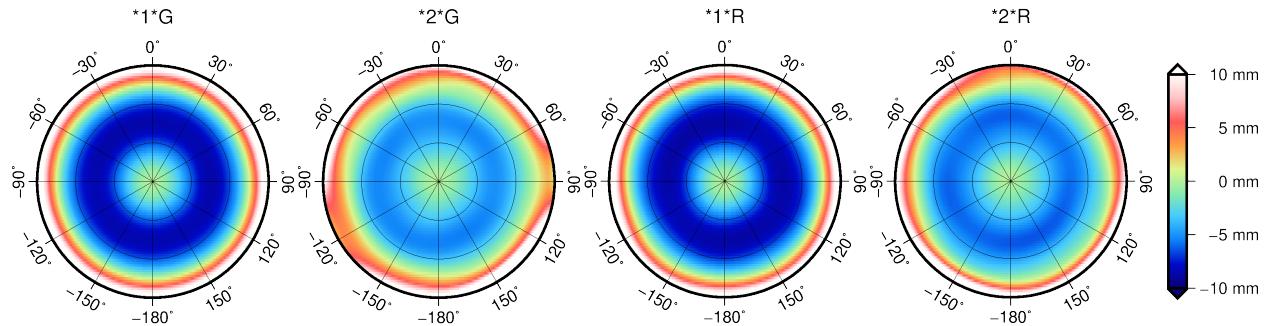


Figure 4.5: Antenna Center Variations of ASH701945D\_M for two frequencies of GPS and GLONASS

Name	Type	Annotation
<b>outputfileGriddedData</b>	filename	data column for each gnssType
<b>inputfileAntennaDefinition</b>	filename	
<b>grid</b>	grid	
<b>name</b>	string	
<b>serial</b>	string	
<b>radome</b>	string	
<b>types</b>	gnssType	if not set, all types in the file are used
<b>R</b>	double	reference radius for ellipsoidal coordinates
<b>inverseFlattening</b>	double	reference flattening for ellipsoidal coordinates

### 4.3.3 GnssAntennaDefinitionCreate

Create a **GNSS antenna definition file** (Antenna Center Variations, ACV) consisting of multiple antennas. The antennas can be created from scratch or can be selected from existing files. This program can also be used to modify existing files.

Furthermore it can be used to create accuracy definition files containing azimuth and elevation dependent accuracy values for antennas. To create an accuracy pattern for phase observations with 1 mm accuracy at zenith and no azimuth dependency, define a pattern with **type=L**, **values=0.001/cos(zenith/rho)**.

The antennas in **outputfileAntennaDefinition** are sorted by names and duplicates are removed (first one is kept).

Name	Type	Annotation
<b>outputfileAntennaDefinition</b>	filename	
<b>antenna</b>	<b>gnssAntennaDefintionList</b>	

#### 4.3.4 GnssAntennaNormalsConstraint

Apply constraints to **normal equations** containing **antennaCenterVariations**. Usually the antenna center variations are estimated together with other parameters like station coordinates, signal biases and slant TEC in **GnssProcessing**. This results in a rank deficient matrix as not all parameters can be separated. The deficient can be solved by adding pseudo observation equations as constraints.

To separate antenna center variations and signal biases apply **constraint:mean** for each GNSS **type**. The observation equation for the integral mean of antenna center variations (ACV) in all azimuth  $A$  and elevation  $E$  dependent directions

$$0 = \iint ACV(A, E) d\Phi \approx \sum_i ACV(A_i, E_i) \Delta\Phi_i \quad (4.6)$$

is approximated by a grid defined by **deltaAzimuth**, **deltaZenith**, and **maxZenith**.

To separate from station coordinates use **constraint:centerMean** and from slant TEC parameters use **constraint:TEC**.

The constraints are applied separately to all antennas matching the wildcard patterns of **name**, **serial**, **radome**.

See also **ParameterVector2GnssAntennaDefinition**.

Name	Type	Annotation
<b>outputfileNormalEquation</b>	filename	with applied constraints
<b>inputfileNormalEquation</b>	filename	
<b>constraint</b>	choice	
<b>center</b>	sequence	zero center (x,y,z) of a single pattern
<b>type</b>	<b>gnssType</b>	applied for each matching types
<b>applyWeight</b>	boolean	from normal equations
<b>sigma</b>	double	[m]
<b>centerMean</b>	sequence	zero center (x,y,z) as (weighted) mean of all patterns
<b>applyWeight</b>	boolean	from normal equations
<b>sigma</b>	double	[m]
<b>constant</b>	sequence	zero constant (mean of all directions) of a single pattern
<b>type</b>	<b>gnssType</b>	applied for each matching types
<b>applyWeight</b>	boolean	from normal equations
<b>sigma</b>	double	[m]
<b>constantMean</b>	sequence	zero constant (mean of all directions) as (weighted) mean of all patterns
<b>applyWeight</b>	boolean	from normal equations
<b>sigma</b>	double	[m]
<b>TEC</b>	sequence	zero TEC computed as (weighted) least squares from all types
<b>applyWeight</b>	boolean	from normal equations
<b>sigma</b>	double	[TECU]
<b>antennaCenterVariations</b>	<b>parametrizationGnssAntenna</b>	apply constraints to all matching antennas
<b>antennaName</b>	string	
<b>antennaSerial</b>	string	apply constraints to all matching antennas
<b>antennaRadome</b>	string	apply constraints to all matching antennas

▶ deltaAzimuth	angle	[degree] sampling of pattern to estimate center/constant
▶ deltaZenith	angle	[degree] sampling of pattern to estimate center/constant
▶ maxZenith	angle	[degree] sampling of pattern to estimate center/constant

---

### 4.3.5 GnssAttitudeCreateInfoCreate

Creates attitude info file (`Instrument(MISCVALUES)`) used by `SimulateStarCameraGnss`. One or more `attitudeInfos` can be specified. They are valid from `timeStart` until the start of the subsequent `attitudeInfo`. `maxManeuverTime` is used by `SimulateStarCameraGnss` to look for ongoing orbit maneuvers before/after the given orbit that might affect the attitude at the beginning or end of a given orbit.

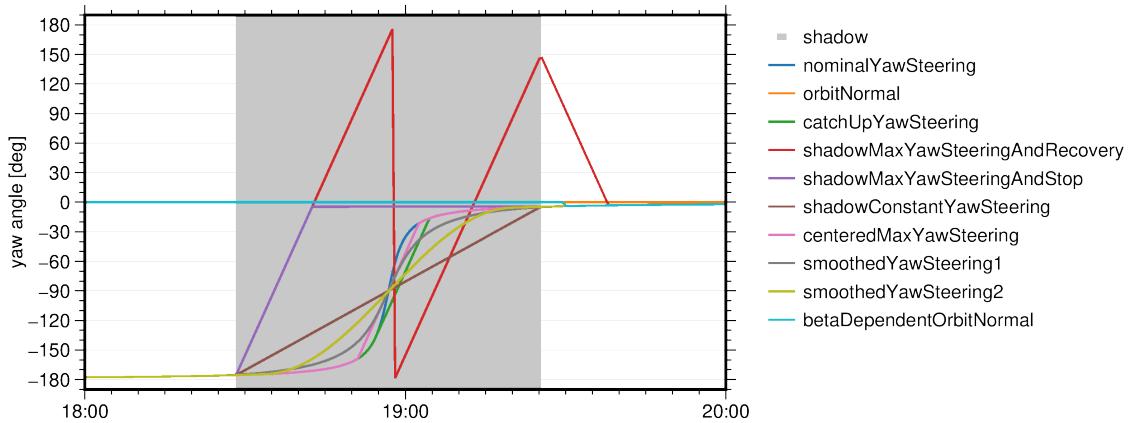


Figure 4.6: Overview of attitude modes used by GNSS satellites

Here is a list of GNSS satellite types for which the attitude behavior is known and their respective attitude modes and required parameters:

- **GPS-II/IIA** [1]
  - `defaultMode`: nominalYawSteering
  - `midnightMode`: shadowMaxYawSteeringAndRecovery
  - `noonMode`: catchUpYawSteering
  - `maxYawRate`: 0.12 deg/s
  - `yawBias`: 0.5 deg
  - `maxManeuverTime`: 2 h
- **GPS-IIR/IIR-M** [1]
  - `defaultMode`: nominalYawSteering
  - `midnightMode`: catchUpYawSteering
  - `noonMode`: catchUpYawSteering
  - `maxYawRate`: 0.2 deg/s
  - `maxManeuverTime`: 30 min
- **GPS-IIF** [2]
  - `defaultMode`: nominalYawSteering
  - `midnightMode`: shadowConstantYawSteering
  - `noonMode`: catchUpYawSteering
  - `maxYawRate`: 0.11 deg/s
  - `yawBias`: -0.7 deg

- **maxManeuverTime:** 1.5 h
- **GLO-M** [3]
  - **defaultMode:** nominalYawSteering
  - **midnightMode:** shadowMaxYawSteeringAndStop
  - **noonMode:** centeredMaxYawSteering
  - **maxYawRate:** 0.25 deg/s
  - **noonBetaThreshold:** 2 deg
  - **maxManeuverTime:** 1.5 h
- **GAL-1** [4]
  - **defaultMode:** nominalYawSteering
  - **midnightMode:** smoothedYawSteering1
  - **noonMode:** smoothedYawSteering1
  - **maxManeuverTime:** 1.5 h
- **GAL-2** [4]
  - **defaultMode:** nominalYawSteering
  - **midnightMode:** smoothedYawSteering2
  - **noonMode:** smoothedYawSteering2
  - **midnightBetaThreshold:** 4.1 deg
  - **noonBetaThreshold:** 4.1 deg
  - **activationThreshold:** 10 deg
  - **maxManeuverTime:** 5656 s
- **BDS-2G/3G** [5, 6]
  - **defaultMode:** orbitNormal
  - **midnightMode:** orbitNormal
  - **noonMode:** orbitNormal
- **BDS-2I** [5]
  - **defaultMode:** nominalYawSteering
  - **midnightMode:** betaDependentOrbitNormal
  - **noonMode:** betaDependentOrbitNormal
  - **maxYawRate:** 0.085 deg/s
  - **midnightBetaThreshold:** 4 deg
  - **noonBetaThreshold:** 4 deg
  - **activationThreshold:** 5 deg
  - **maxManeuverTime:** 24 h
- **BDS-2M** [5]
  - **defaultMode:** nominalYawSteering
  - **midnightMode:** betaDependentOrbitNormal
  - **noonMode:** betaDependentOrbitNormal

- **maxYawRate**: 0.159 deg/s
- **midnightBetaThreshold**: 4 deg
- **noonBetaThreshold**: 4 deg
- **activationThreshold**: 5 deg
- **maxManeuverTime**: 13 h
- **BDS-3I/3SI** [6]
  - **defaultMode**: nominalYawSteering
  - **midnightMode**: smoothedYawSteering2
  - **noonMode**: smoothedYawSteering2
  - **midnightBetaThreshold**: 3 deg
  - **noonBetaThreshold**: 3 deg
  - **activationThreshold**: 6 deg
  - **maxManeuverTime**: 5740 s
- **BDS-3M/3SM** [6]
  - **defaultMode**: nominalYawSteering
  - **midnightMode**: smoothedYawSteering2
  - **noonMode**: smoothedYawSteering2
  - **midnightBetaThreshold**: 3 deg
  - **noonBetaThreshold**: 3 deg
  - **activationThreshold**: 6 deg
  - **maxManeuverTime**: 3090 s
- **QZS-1** [7]
  - **defaultMode**: nominalYawSteering
  - **midnightMode**: betaDependentOrbitNormal
  - **noonMode**: betaDependentOrbitNormal
  - **maxYawRate**: 0.01 deg/s
  - **yawBias**: 180 deg
  - **midnightBetaThreshold**: 20 deg
  - **noonBetaThreshold**: 20 deg
  - **activationThreshold**: 18.5 deg
  - **maxManeuverTime**: 24 h
- **QZS-2G** [7]
  - **defaultMode**: orbitNormal
  - **midnightMode**: orbitNormal
  - **noonMode**: orbitNormal
  - **yawBias**: 180 deg
- **QZS-2I** [7]
  - **defaultMode**: nominalYawSteering
  - **midnightMode**: centeredMaxYawSteering

- **noonMode**: centeredMaxYawSteering
- **maxYawRate**: 0.055 deg/s
- **midnightBetaThreshold**: 5 deg
- **noonBetaThreshold**: 5 deg
- **maxManeuverTime**: 1.5 h

Some specific satellites may deviate in their attitude behavior or parameters (e.g. G013-G040, R713, C005, C015, C017, J001).

References for the attitude behavior information:

1. Kouba (2009)
2. Kuang et al. (2017)
3. Dilssner et al. (2011)
4. <https://www.gsc-europa.eu/support-to-developers/galileo-satellite-metadata#3>
5. Wang et al. (2018)
6. Li et al. (2018)
7. <https://qzss.go.jp/en/technical/qzssinfo/index.html>

Name	Type	Annotation
outputfileAttitudeInfo	filename	
attitudeInfo	sequence	
timeStart	time	
defaultMode	choice	default attitude mode
nominalYawSteering		yaw to keep solar panels aligned to Sun (e.g. most GNSS satellites outside eclipse)
orbitNormal		keep fixed yaw angle, for example point X-axis in flight direction (e.g. BDS-2G, BDS-3G, QZS-2G)
midnightMode	choice	attitude mode for maneuvers around orbit midnight
nominalYawSteering		yaw to keep solar panels aligned to Sun (e.g. most GNSS satellites outside eclipse)
orbitNormal		keep fixed yaw angle, for example point X-axis in flight direction (e.g. BDS-2G, BDS-3G, QZS-2G)
catchUpYawSteering		yaw at maximum yaw rate to catch up to nominal yaw angle (e.g. GPS-* (noon), GPS-IIR (midnight))
shadowMaxYawSteeringAndRecovery		yaw at maximum yaw rate from shadow start to end, recover after shadow (e.g. GPS-IIA (midnight))
shadowMaxYawSteeringAndStop		yaw at maximum yaw rate from shadow start until nominal yaw angle at shadow end is reached, then stop (e.g. GLO-M (midnight))
shadowConstantYawSteering		yaw at constant yaw rate from shadow start to end (e.g. GPS-IIF (midnight))
centeredMaxYawSteering		yaw at maximum yaw rate centered around noon/midnight (e.g. QZS-2I, GLO-M (noon))
smoothedYawSteering1		yaw based on an auxiliary Sun vector for a smooth yaw maneuver (e.g. GAL-1)
smoothedYawSteering2		yaw based on a modified yaw-steering law for a smooth yaw maneuver (e.g. GAL-2, BDS-3M, BDS-3I)

 betaDependentOrbitNormal		switch to orbit normal mode if below beta angle threshold (e.g. BDS-2M, BDS-2I, QZS-1)
 noonMode	choice	attitude mode for maneuvers around orbit noon
 nominalYawSteering		yaw to keep solar panels aligned to Sun (e.g. most GNSS satellites outside eclipse)
 orbitNormal		keep fixed yaw angle, for example point X-axis in flight direction (e.g. BDS-2G, BDS-3G, QZS-2G)
 catchUpYawSteering		yaw at maximum yaw rate to catch up to nominal yaw angle (e.g. GPS-* (noon), GPS-IIR (midnight))
 centeredMaxYawSteering		yaw at maximum yaw rate centered around noon/midnight (e.g. QZS-2I, GLO-M (noon))
 smoothedYawSteering1		yaw based on an auxiliary Sun vector for a smooth yaw maneuver (e.g. GAL-1)
 smoothedYawSteering2		yaw based on a modified yaw-steering law for a smooth yaw maneuver (e.g. GAL-2, BDS-3M, BDS-3I)
 betaDependentOrbitNormal		switch to orbit normal mode if below beta angle threshold (e.g. BDS-2M, BDS-2I, QZS-1)
 maxYawRate	double	[degree/s] maximum yaw rate of the satellite
 yawBias	double	[degree] yaw bias applied in satellite attitude control system
 midnightBetaThreshold	double	[degree] limit midnight maneuver to this absolute angle of the Sun above/below the satellite orbital plane
 noonBetaThreshold	double	[degree] limit noon maneuver to this absolute angle of the Sun above/below the satellite orbital plane
 activationThreshold	double	[degree] limit maneuver to this yaw/Earth-spacecraft-Sun angle (depending on mode)
 maxManeuverTime	double	[s] maximum duration of maneuver or maximum maneuver lookup time before/after orbit start/end

### 4.3.6 GnssBiasClockAlignment

This program can be used to absolutely align GNSS transmitter clocks to reference clocks (i.e. broadcast clocks). Each 'group' of **transmitters**, usually a system like GPS or Galileo, is aligned individually by a constant shift over all transmitters. If **alignClocksByFreqNo** is set, GLONASS transmitters will be divided by frequency number into groups of nominally two transmitters. The offset between clocks and reference clocks will be shifted into receiver code biases, if **receiver** is provided."

By setting **alignFreqNoBiasesAtReceiver** and providing **receiver**, this program can further align GLONASS transmitter signal biases so that the differences between frequency number-dependent receiver signal biases are minimal, which helps if PPP users don't set up individual signal biases per frequency number at the receiver. Alignment is done by computing signal bias residuals to the mean over all frequency numbers of a signal type at each receiver and then computing the means over all receivers for each frequency number and shifting those from the receiver signal biases to the transmitter signal biases. Internal consistency of the biases is not affected by this.

If you only want to align GLONASS frequency numbers, provide the same clocks in **infileClock** and **infileReferenceClock**.

Name	Type	Annotation
<b>transmitter</b>	sequence	one element per satellite
<b>outfileClock</b>	filename	aligned clock instrument file
<b>outfileSignalBias</b>	filename	(GLONASS only) aligned signal bias file
<b>infileClock</b>	filename	clock instrument file
<b>infileReferenceClock</b>	filename	reference clock instrument file
<b>infileSignalBias</b>	filename	(GLONASS only) signal bias file
<b>infileTransmitterInfo</b>	filename	transmitter platform file
<b>receiver</b>	sequence	one element per station
<b>outfileSignalBias</b>	filename	aligned signal bias file
<b>infileSignalBias</b>	filename	signal bias file
<b>alignClocksByFreqNo</b>	boolean	align clocks for each GLONASS frequency number separately
<b>alignFreqNoBiasesAtReceiver</b>	boolean	align frequency number-dependent code biases for each receiver

### 4.3.7 GnssEstimateClockShift

This program estimates an epoch-wise clock shift in a constellation of GNSS satellites. Each separate **data** represents a satellite... (e.g. 32 GPS satellites). The shift to reference clocks can be estimated by providing **inputfileInstrumentRef**. Clock shifts are estimated for each epoch given by **timeSeries**.

Name	Type	Annotation
outputfileShiftTimeSeries	filename	columns: mjd, clock shift
data	sequence	e.g. satellite
outputfileInstrument	filename	corrected clocks
outputfileInstrumentDiff	filename	clock difference after correction
inputfileInstrument	filename	input clocks
inputfileInstrumentRef	filename	reference clocks (subtracted from input clocks)
timeSeries	timeSeries	clock epochs
margin	double	[s] margin for time comparison

### 4.3.8 GnssGlonassFrequencyNumberUpdate

Update/set GLONASS frequency number in **inputfileTransmitterInfo** files.

PRN/SVN to frequency number source: <http://semisys.gfz-potsdam.de/semisys/api/?symname=2002&format=json&satellite=GLO>.

See also **GnssAntex2AntennaDefinition**.

Name	Type	Annotation
▶ outputfileTransmitterInfo	filename	templated for PRN list (variableNamePrn)
▶* inputfileTransmitterInfo	filename	templated for PRN list (variableNamePrn)
▶* inputfilePrn2FrequencyNumber	filename	GROOPS matrix with columns: GLONASS PRN, SVN, mjdStart, mjdEnd, frequencyNumber
…▶ prn	string	PRN (e.g. R01) for transmitter info files
▶ variableNamePrn	string	variable name for PRN in transmitter info files

### 4.3.9 GnssProcessing

This program processes GNSS observations. It calculates the linearized observation equations, accumulates them into a system of normal equations and solves it.

The primary use cases of this program are:

- GNSS satellite orbit determination and station network analysis (3.2)
- Kinematic orbit determination of LEO satellites (3.4)
- GNSS precise point positioning (PPP) (3.3)

The observation epochs are defined by **timeSeries** and only observations at these epochs (within a **timeMargin**) are considered.

To calculate observation equations from the tracks, the model parameters or unknown parameters need to be defined beforehand. These unknown parameters can be chosen arbitrarily by the user with an adequate list of defined **parametrization**. Some of the **parametrization** also include a priori models.

Lastly it is required to define the process flow of the gnssProcessing. This is accomplished with a list of **processingSteps**. Each step is processed consecutively. Some steps allow the selection of parameters, epochs, or the normal equation structure, which affects all subsequent steps. A minimal example consists of following steps:

- **estimate**: iterative float solution with outlier downweighting
- **resolveAmbiguities**: fix ambiguities to integer and remove them from the normals
- **estimate**: few iteration for final outlier downweighting
- **writeResults**: write the output files defined in **parametrization**

If the program is run on multiple processes the **receivers** (stations or LEO satellites) are distributed over the processes.

See also **GnssSimulateReceiver**.

Name	Type	Annotation
timeSeries	timeSeries	defines observation epochs
timeMargin	double	[seconds] margin to consider two times identical
transmitter	gnssTransmitterGenerator	constellation of GNSS satellites
receiver	gnssReceiverGenerator	ground station network or LEO satellite
earthRotation	earthRotation	apriori earth rotation
parametrization	gnssParametrization	models and parameters
processingStep	gnssProcessingStep	steps are processed consecutively

This program is **parallelized** (1.5).

#### 4.3.10 GnssReceiverDefinitionCreate

Create a [GNSS receiver definition file](#).

Name	Type	Annotation
outfileGnssReceiverDefinition	filename	
receiverDefinition	sequence	
name	string	
serial	string	
version	string	
comment	string	
gnssType	gnssType	

### 4.3.11 GnssResiduals2AccuracyDefinition

Compute antenna accuracies from observation **inputfileResiduals**. The **inputfileStationInfo** is needed to assign the residuals to the equipped antenna at observation times.

The **outputfileAccuracyDefinition** contains at first step the same accuracy information for all antennas as the input file. Only the azimuth  $A$  and elevation  $E$  dependent grid points of the patterns where enough residuals are available ( $> \text{minRedundancy}$ ) are replaced by estimated accuracy

$$\sigma(A, E) = \sqrt{\frac{\sum_i e_i^2(A, E)}{\sum_i r_i(A, E)}}, \quad (4.7)$$

where  $e_i$  are the azimuth and elevation dependent residuals and  $r_i$  the corresponding redundancies (number of observations minus the contribution to the estimated parameters).

The **inputfileAccuracyDefinition** can be modified to the demands before with **GnssAntennaDefinitionCreate** (e.g. with **antenna:resample**).

To verify the results the **outputfileAntennaMean** and the accumulated **outputfileAntennaRedundancy** of the computed pattern grid points can be written.

Example: Analysis of TerraSAR-X residuals of one month shows that low elevation GPS satellites are not tracked by the onboard receiver. An estimation of accuracies for these directions is not possible from the residuals and the apriori accuracies are left untouched. The other directions show very low phase noise hardly elevation and azimuth dependent for L2W. A nearly zero mean indicates the use of adequate antenna center variations in the processing.

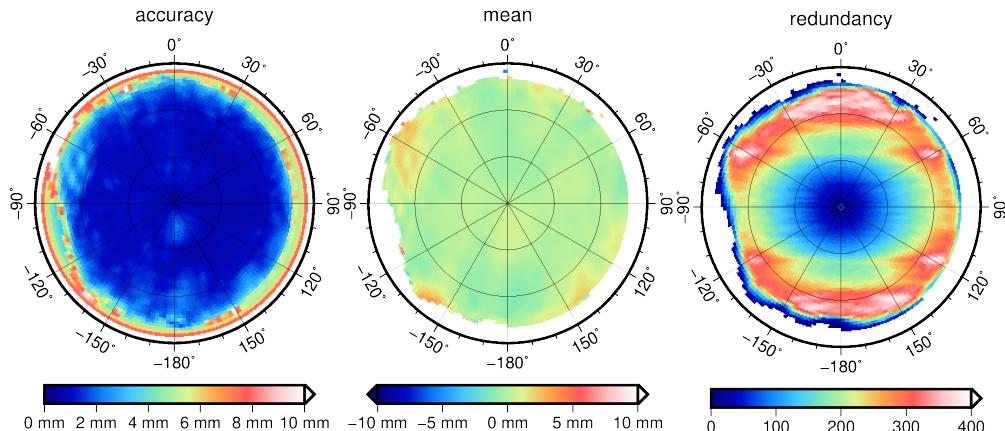


Figure 4.7: L2W accuracies of TerraSAR-X determined from residuals of one month

Name	Type	Annotation
<b>outputfileAccuracyDefinition</b>	filename	elevation and azimuth dependent accuracy
<b>outputfileAntennaMean</b>	filename	weighted mean of the residuals
<b>outputfileAntennaRedundancy</b>	filename	redundancy of adjustment
<b>inputfileAccuracyDefinition</b>	filename	apriori accuracies
<b>inputfileStationInfo</b>	filename	to assign residuals to antennas
<b>isTransmitter</b>	boolean	stationInfo is of a transmitter
<b>thresholdOutlier</b>	double	ignore residuals with sigma/sigma0 greater than threshold
<b>minRedundancy</b>	double	min number of residuals. to estimate sigma
<b>inputfileResiduals</b>	filename	GNSS receiver residuals

### 4.3.12 GnssResiduals2Skyplot

Write GNSS residuals together with azimuth and elevation to be plotted with **PlotMap**. Azimuth and elevation are written as ellipsoidal longitude and latitude in a **griddedData** file. The chosen ellipsoid parameters **R** and **inverseFlattening** are arbitrary but should be the same as in **PlotMap**. If with **typeTransmitter** (e.g. '\*\*\*G18') a single transmitter is selected the azimuth and elevation are computed from the transmitter point of view.

For each GNSS **type** an extra data column is created.

A **GNSS residual file** includes additional information besides the residuals, which can also be selected with **type**

- A1\*, E1\*: azimuth and elevation at receiver
- A2\*, E2\*: azimuth and elevation at transmitter
- I\*\*: Estimated slant total electron content (STEC)

Furthermore these files may include for each residual **type** information about the redundancy and the accuracy relation  $\sigma/\sigma_0$  of the estimated  $\sigma$  versus the a priori  $\sigma_0$  from the least squares adjustment. The 3 values (residuals, redundancy,  $\sigma/\sigma_0$ ) are coded with the same type. To get access to all values the corresponding type must be repeated in **type**.

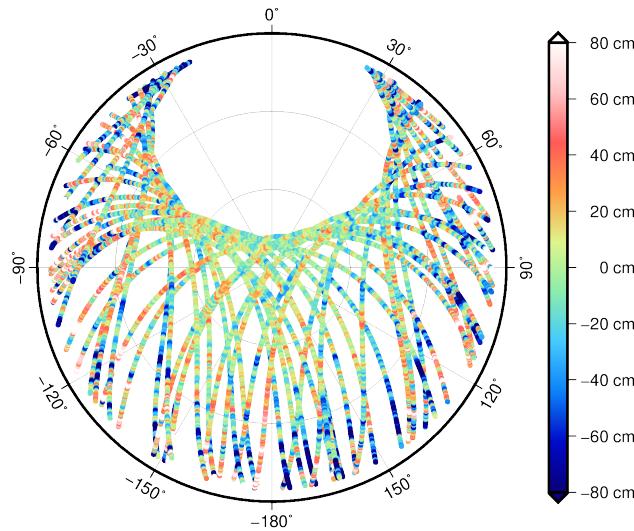


Figure 4.8: GPS C2W residuals of GRAZ station at 2012-01-01

Name	Type	Annotation
<b>outputfileGriddedData</b>	filename	
<b>type</b>	<b>gnssType</b>	
<b>typeTransmitter</b>	<b>gnssType</b>	choose transmitter view, e.g. '***G18'
<b>inputfileResiduals</b>	filename	GNSS receiver residuals
<b>R</b>	double	reference radius for ellipsoidal coordinates
<b>inverseFlattening</b>	double	reference flattening for ellipsoidal coordinates

### 4.3.13 GnssSignalBias2Matrix

Computes signal biases for a given list of **types**. If the type list is empty, all types contained in **inputfileSignalBias** are used. The resulting **outputfileMatrix** contains a vector with an entry for each type.

Name	Type	Annotation
outputfileMatrix	filename	
outputfileTypes	filename	ASCII list of types
inputfileSignalBias	filename	
types	gnssType	if not set, all types in the file are used

### 4.3.14 GnssSimulateReceiver

This program simulates observations from receivers to GNSS satellites. These simulated observations can then be used in **GnssProcessing**, for example to conduct closed-loop simulations.

One or more GNSS constellations must be defined via **transmitter**. Receivers such as ground station networks or Low Earth Orbit (LEO) satellites can be defined via **receiver**.

If multiple receivers defined an **outputfileGnssReceiver** and **outputfileClock** are written for each single receiver with the [variable \(1.2\)](#) `{station}` being replaced by the receiver name.

A list of simulated observation types can be defined via **observationType**. Noise can be added to both observations and clock errors via **noiseObservation** and **noiseClockReceiver**, respectively. Observation noise is interpreted as a factor that is multiplied to the accuracy derived from the accuracy pattern of the respective observation type (see **inputfileAccuracyDefinition** in **receiver**).

The **parametrization** are used to simulate a priori models (e.g. troposphere, signal biases). Parameter settings and outputfiles are ignored.

If the program is run on multiple processes the **receivers** (stations or LEO satellites) are distributed over the processes.

Name	Type	Annotation
<b>outputfileGnssReceiver</b>	filename	variable <code>{station}</code> available, simulated observations
<b>outputfileClock</b>	filename	variable <code>{station}</code> available, simulated receiver clock errors
<b>timeSeries</b>	<b>timeSeries</b>	defines observation epochs
<b>timeMargin</b>	double	[seconds] margin to consider two times identical constellation of GNSS satellites
<b>transmitter</b>	<b>gnssTransmitterGenerator</b>	ground station network or LEO satellite
<b>receiver</b>	<b>gnssReceiverGenerator</b>	apriori earth rotation
<b>earthRotation</b>	<b>earthRotation</b>	models and parameters
<b>parametrization</b>	<b>gnssParametrization</b>	simulated observation types
<b>observationType</b>	<b>gnssType</b>	<code>[-]</code> noise is multiplied with type accuracy pattern of receiver
<b>noiseObservation</b>	<b>noiseGenerator</b>	[m] noise added to the simulated receiver clock
<b>noiseClockReceiver</b>	<b>noiseGenerator</b>	

This program is [parallelized \(1.5\)](#).

### 4.3.15 GnssStationInfoCreate

Create a **GnssStationInfo** file from scratch by defining attributes such as **markerName**, **markerNumber**, **comment**, **approxPosition**, **antenna** and **receiver**.

See also **GnssAntex2AntennaDefinition** and **GnssStationLog2Platform**.

Name	Type	Annotation
outputfileStationInfo	filename	
markerName	string	
markerNumber	string	
comment	string	
approxPositionX	double	[m] in TRF
approxPositionY	double	[m] in TRF
approxPositionZ	double	[m] in TRF
antenna	sequence	
name	string	
serial	string	
radome	string	
comment	string	
timeStart	time	
timeEnd	time	
positionX	double	[m] ARP in north, east, up or vehicle system
positionY	double	[m] ARP in north, east, up or vehicle system
positionZ	double	[m] ARP in north, east, up or vehicle system
rotationX	angle	[degree] from local/vehicle to left-handed antenna system
rotationY	angle	[degree] from local/vehicle to left-handed antenna system
rotationZ	angle	[degree] from local/vehicle to left-handed antenna system
flipX	boolean	flip x-axis (after rotation)
flipY	boolean	flip y-axis (after rotation)
flipZ	boolean	flip z-axis (after rotation)
receiver	sequence	
name	string	
serial	string	
version	string	
comment	string	
timeStart	time	
timeEnd	time	
referencePoint	sequence	e.g. center of mass in satellite frame
comment	string	
xStart	double	[m] in north, east, up or vehicle system
yStart	double	linear motion between start and end
zStart	double	
xEnd	double	[m] in north, east, up or vehicle system
yEnd	double	linear motion between start and end
zEnd	double	
timeStart	time	
timeEnd	time	

### 4.3.16 InstrumentGnssReceiver2TimeSeries

Convert selected GNSS observations or residuals into a simpler time series format. The **outputfileTimeSeries** is an **Instrument file** (MISCVALUES). For each epoch the first data column contains the PRN, the second the satellite system, followed by a column for each GNSS **type**. As normally more than one GNSS transmitter is tracked per epoch, the output file has several lines per observed epoch (epochs with the same time, one for each transmitter).

The second data column of the output contains a number representing the system

- 71: 'G', GPS
- 82: 'R', GLONASS
- 69: 'E', GALILEO
- 67: 'C', BDS
- 83: 'S', SBAS
- 74: 'J', QZSS
- 73: 'I', IRNSS .

A **GNSS residual file** includes additional information besides the residuals, which can also be selected with **type**

- A1\*, E1\*: azimuth and elevation at receiver
- A2\*, E2\*: azimuth and elevation at transmitter
- I\*\*: Estimated slant total electron content (STEC)

Furthermore these files may include for each residual **type** information about the redundancy and the accuracy relation  $\sigma/\sigma_0$  of the estimated  $\sigma$  versus the apriori  $\sigma_0$  from the least squares adjustment. The three values (residuals, redundancy,  $\sigma/\sigma_0$ ) are coded with the same type. To get access to all values the corresponding type must be repeated in **type**.

Example: Selected GPS phase residuals (**type**='L1\*G' and **type**='L2\*G'). Plotted with **PlotGraph** with two **layer:linesAndPoints** (**valueX**='data0', **valueY**='100\*data3+data1' and **valueY**='100\*data4+data1' respectively).

Name	Type	Annotation
<b>outputfileTimeSeries</b>	filename	Instrument (MISCVALUES): prn, system, values for each type
<b>inputfileGnssReceiver</b>	filename	GNSS receiver observations or residuals
<b>type</b>	<b>gnssType</b>	

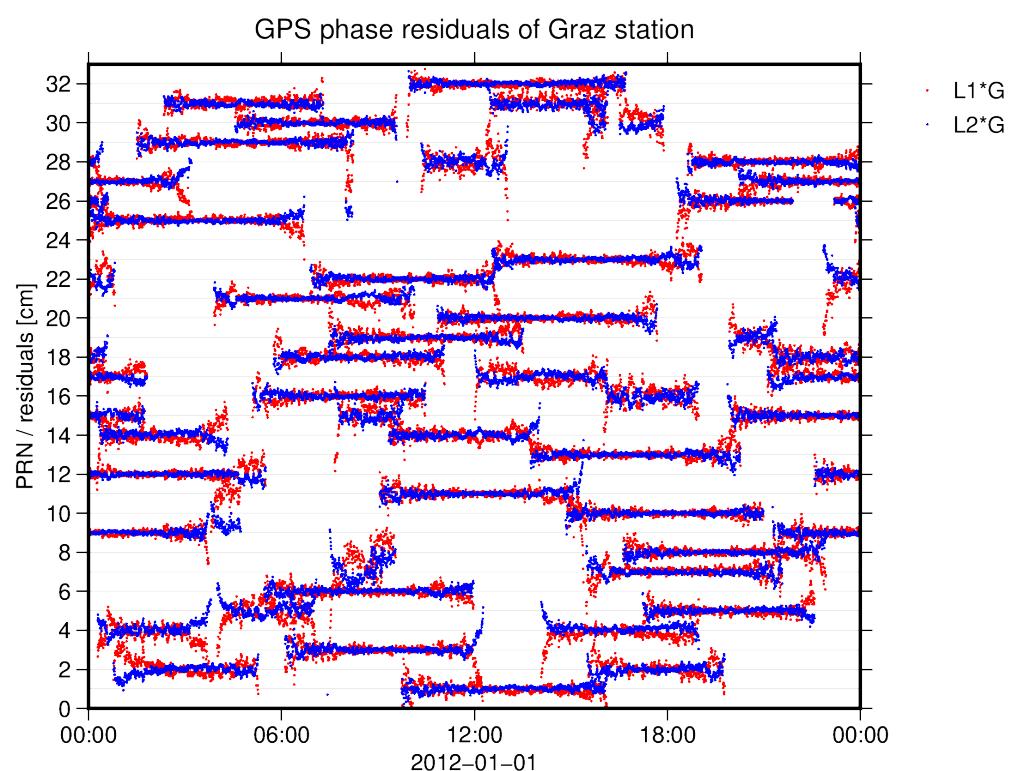


Figure 4.9: GPS residuals in cm, shifted by PRN

### 4.3.17 ParameterVector2GnssAntennaDefinition

Updates an [GnssAntennaDefinition file](#) with estimated parameters which belongs to the parametrization [antennaCenterVariations](#). The [outfileAntennaDefinition](#) contains all antennas from [infileAntennaDefinition](#). The antenna center variations representend by the [infileSolution](#) are added to the matching antennas.

The [GnssAntennaDefinition file](#) can be modified to the demands before with [GnssAntennaDefinitionCreate](#).

The following steps are used to estimate antenna center variations:

- [GnssAntennaDefinitionCreate](#) or [GnssAntex2AntennaDefinition](#)
- [GnssProcessing](#) with [infileAntennaDefinition](#) as apriori and writing [normal equations](#) with parametrization of [antennaCenterVariations](#)
- [NormalsEliminate](#): eliminate all other than antenna parameters
- [NormalsAccumulate](#): accumulate normals over a sufficient long period
- [GnssAntennaNormalsConstraint](#): constrain unsolvable parameter linear combinations
- [NormalsSolverVCE](#): estimate the parameter vector
- [ParameterVector2GnssAntennaDefinition](#): update [infileAntennaDefinition](#)

See also [ParameterVector2GnssAntennaDefinition](#), [GnssAntennaNormalsConstraint](#).

Name	Type	Annotation
<a href="#">outfileAntennaDefinition</a>	filename	all apriori antennas
<a href="#">infileAntennaDefinition</a>	filename	apriori antennas
<a href="#">antennaCenterVariations</a>	<a href="#">parametrizationGnssAntenna</a>	
<a href="#">infileSolution</a>	filename	
<a href="#">infileParameterNames</a>	filename	

## 4.4 Programs: Grace

### 4.4.1 EnsembleAveragingScaleModel

This programs estimate satellite-to-satellite-tracking (SST) deterministic signals due to eclipse transits from residuals. The ensemble averaging method is used to characterize the average properties of signal shapes across all transit events. Each shape is assigned to one arc of 3 hours (default). This can be modefied by enabling  **averagingInterval**.

Name	Type	Annotation
 <b>outputfileScaleModel</b>	filename	
 <b>inputfileGrace1EclipseFactor</b>	filename	GRACE-A eclipse factors computed with integrated orbit
 <b>inputfileGrace2EclipseFactor</b>	filename	GRACE-B eclipse factors computed with integrated orbit
 <b>inputfileGraceResiduals</b>	filename	SST Residuals
 <b>timeMargin</b>	uint	epochs before eclipse mode
 <b>waveLength</b>	uint	length of the sample wave
 <b>averagingInterval</b>	sequence	
 <b>nearestNeighborNumber</b>	uint	

#### 4.4.2 GraceAntennaCenterCorrectionArcCovariance

This program computes covariance information for the non-stationary noise of the KBR antenna offset correction (AOC) from the orientation covariance matrices provided in Level-1B products via variance propagation. By using the output **outputfileSatelliteTrackingCovariance** in **PreprocessingSst**, noise model distinguishes between the stationary noise of ranging observations and the nonstationary AOC noise.

The covariances are derived from the partial derivative of the AOC w.r.t. the roll/pitch/yaw rotations and star camera covariances **inputfileScaCovariance1** and **inputfileScaCovariance2**.

The covariances for the range-rates and range-acceleration are computed by differentiating an interpolation polynomial of degree **interpolationDegree**.

Name	Type	Annotation
► outputfileSatelliteTrackingCovariance	filename	corrections for range, range-rate, and range-accelerations
► sstType	choice	
└► range		
└► rangeRate		
└► rangeAcceleration		
└► inputFileOrbit1	filename	
└► inputFileOrbit2	filename	
└► inputFileStarCamera1	filename	
└► inputFileStarCamera2	filename	
└► inputFileScaCovariance1	filename	
└► inputFileScaCovariance2	filename	
└► sigmaAccelerometerX	double	[rad/s^2]
└► sigmaAccelerometerY	double	[rad/s^2]
└► sigmaAccelerometerZ	double	[rad/s^2]
└► antennaCenters	choice	KBR antenna phase center
└► value	sequence	
└► center1X	double	x-coordinate of antenna position in SRF [m] for GRACEA
└► center1Y	double	y-coordinate of antenna position in SRF [m] for GRACEA
└► center1Z	double	z-coordinate of antenna position in SRF [m] for GRACEA
└► center2X	double	x-coordinate of antenna position in SRF [m] for GRACEB
└► center2Y	double	y-coordinate of antenna position in SRF [m] for GRACEB
└► center2Z	double	z-coordinate of antenna position in SRF [m] for GRACEB
└► file	sequence	
└► inputAntennaCenters	filename	
└► interpolationDegree	uint	differentiation by polynomial approximation of degree n

This program is parallelized (1.5).

### 4.4.3 GraceOrbit2TransplantTimeOffset

This program computes the time shift between two co-orbiting satellites based on dynamic orbit data. When applied to data of the first satellite, the computed time shift virtually shifts data of first satellite into the location of the second satellite. Note that **inputfileOrbit1** and **inputfileOrbit2** need velocity and acceleration data, which can be computed with **OrbitAddVelocityAndAcceleration**. The program tries to find a minimum of the objective function

$$f(\Delta t) = \|r_1(t) - r_2(t + \Delta t)\|^2, \quad (4.8)$$

by applying Newton's method to the first derivative, thus iteratively computing

$$\Delta t_{k+1} = \Delta t_k + \frac{f'(\Delta t_k)}{f''(\Delta t_k)}. \quad (4.9)$$

This iteration is stopped when the difference between consecutive time shift values falls below **threshold** or **maximumIterations** is reached. An **initialGuess** of the time shift can speed up convergence.

See also **OrbitAddVelocityAndAcceleration** and **InstrumentApplyTimeOffset**.

Name	Type	Annotation
<b>outputfileTimeOffset</b>	filename	estimated time offset in seconds (MISCVALUE)
<b>inputfileOrbit1</b>	filename	orbit data of satellite 1
<b>inputfileOrbit2</b>	filename	orbit data of satellite 2
<b>interpolationDegree</b>	uint	polynomial degree for the interpolation of position, velocity and acceleration
<b>initialGuess</b>	double	initial guess for the time shift [seconds]
<b>maximumIterations</b>	uint	maximum number of iterations
<b>threshold</b>	double	when the maximum difference between two iterations is below this value, stop [seconds]

This program is [parallelized \(1.5\)](#).

#### 4.4.4 GraceSstResidualAnalysis

This program applies the Multi-Resolution Analysis (MRA) using Discrete Wavelet Transform (DWT) to the monthly GRACE SST post-fit residuals. First, the residuals are transferred into wavelet domain by applying an 8 level Daubechies wavelet transform (default). In the next step, detail coefficients are merged into three major groups due to their approximate frequency subbands:

- Low scale details, corresponding to the frequency band above 10 mHz;
- Intermediate scale details, corresponding to the approximate frequency range above 3 mHz up to 10 mHz;
- High scale details, corresponding to the approximate frequency range above 0.5 mHz up to 10 mHz.

In the last step, each group is reconstructed back into time domain.

Name	Type	Annotation
outfileInstrumentHighScale	filename	High scale details
outfileInstrumentMidScale	filename	Intermediate scale details
outfileInstrumentLowScale	filename	Low scale details
infileInstrument	filename	GRACE SST Residuals
infileWavelet	filename	wavelet coefficients

#### 4.4.5 GraceSstScaleModel

This programs estimate satellite-to-satellite-tracking (SST) deterministic signals due to eclipse transits and low-SNR values from post-fit residuals. The low-SNR effects are estimated by directly using the residual values. The ensemble averaging method is used to characterize the average properties of eclipse transit signal shapes across all transit events. Each shape is assigned to one arc of 3 hours (default). This can be modefied by enabling **▷ averagingInterval**.

Name	Type	Annotation
▷ inputFileGraceResiduals	filename	SST Residuals
▷ timeMargin	uint	epochs before instrumental events
▷ waveLength	uint	length of the sample wave
▷ estimateEclipseTransitScale	sequence	
▷ outputFileScaleModel	filename	
▷ inputFileGrace1EclipseFactor	filename	GRACE-A eclipse factors computed with integrated orbit
▷ inputFileGrace2EclipseFactor	filename	GRACE-B eclipse factors computed with integrated orbit
▷ averagingInterval	sequence	
▷ nearestNeighborNumber	uint	
▷ estimateLowSnrScale	sequence	
▷ outputFileScaleModel	filename	
▷ inputFileGraceSstSNR	filename	GRACE SNR values

#### 4.4.6 GraceSstSpecialEvents

Time-indexing deterministic signals in the GRACE K-Band measurements caused by Sun intrusions into the star camera baffles of GRACE-A and eclipse transits of the satellites. The events are determined by satellites' position ( [inputfileOrbit1/2](#)) and orientation ( [inputfileStarCamera1/2](#)). Each type of event is represented by its mid-interval point per orbit revolution and is reported in [outputfileEvents](#).

The waveform of each event is nearly constant within one month and can be approximated by a polynomial. For the purpose of gravity field recovery, each waveform is parameterized by a polynomial and the coefficients of this polynomial are estimated as additional instrument calibration parameters in a common adjustment with all other instrument, satellite, and gravity field parameters, see [parametrizationSatelliteTracking:specialEffect](#).

Name	Type	Annotation
<a href="#">outputfileEvents</a>	filename	
<a href="#">outputfileIntervals</a>	filename	
<a href="#">inputfileOrbit1</a>	filename	
<a href="#">inputfileOrbit2</a>	filename	
<a href="#">inputfileStarCamera1</a>	filename	
<a href="#">inputfileStarCamera2</a>	filename	
<a href="#">ephemerides</a>	ephemerides	
<a href="#">eclipse</a>	eclipse	
<a href="#">marginLeft</a>	double	margin size (on both sides) [seconds]
<a href="#">marginRight</a>	double	margin size (on both sides) [seconds]

#### 4.4.7 GraceThrusterResponse2Accelerometer

Add modeled thruster responses to accelerometer data. The epochs and durations are given in the **► inputfileThruster** (THRUSTER).

The **► inputfileThrusterResponse** is a  $(6 \times 3)$  matrix with the linear accelerations in the SRF ( $x, y, z$ ) in one line per pair:

1. Negative Yaw,
2. Positive Pitch,
3. Positive Yaw,
4. Negative Pitch,
5. Negative Roll,
6. Positive Roll.

Name	Type	Annotation
► outputfileAccelerometer	filename	ACCELEROMETER
► inputfileAccelerometer	filename	ACCELEROMETER
► inputfileThruster	filename	THRUSTER
► inputfileThrusterResponse	filename	thruster model (matrix with one line per pair)

#### 4.4.8 InstrumentSatelliteTrackingAntennaCenterCorrection

This program computes the correction due to offset of the antenna center relative the center of mass. The offsets  $\mathbf{c}_A$  and  $\mathbf{c}_B$  in **inputfileAntennaCenters** are given in the satellite reference frame. These offsets are rotated into the the inertial frame with  $\mathbf{D}_A$  and  $\mathbf{D}_B$  from **inputfileStarCamera** and projected onto the line of sight (LOS)

$$\rho_{AOC} = \mathbf{e}_{AB} \cdot (\mathbf{D}_A \mathbf{c}_A - \mathbf{D}_B \mathbf{c}_B), \quad (4.10)$$

with the unit vector in line of sight direction

$$\mathbf{e}_{AB} = \frac{\mathbf{r}_B - \mathbf{r}_A}{\|\mathbf{r}_B - \mathbf{r}_A\|}. \quad (4.11)$$

The corrections for the range-rates and range-acceleration are computed by differentiating an interpolation polynomial of degree **interpolationDegree**.

Name	Type	Annotation
▶ outputfileSatelliteTracking	filename	corrections for range, range-rate, and range-accelerations
▶ inputfileOrbit1	filename	
▶ inputfileOrbit2	filename	
▶ inputfileStarCamera1	filename	
▶ inputfileStarCamera2	filename	
▶ antennaCenters	choice	KBR antenna phase center
└▶ value	sequence	
└▶ center1X	double	x-coordinate of antenna position in SRF [m] for GRACEA
└▶ center1Y	double	y-coordinate of antenna position in SRF [m] for GRACEA
└▶ center1Z	double	z-coordinate of antenna position in SRF [m] for GRACEA
└▶ center2X	double	x-coordinate of antenna position in SRF [m] for GRACEB
└▶ center2Y	double	y-coordinate of antenna position in SRF [m] for GRACEB
└▶ center2Z	double	z-coordinate of antenna position in SRF [m] for GRACEB
└▶ file	sequence	
└▶ inputAntennaCenters	filename	
▶ interpolationDegree	uint	differentiation by polynomial approximation of degree n

This program is parallelized (1.5).

#### 4.4.9 InstrumentStarCameraAngularAccelerometerFusion

This program estimates the satellites orientation from star camera data  $\blacktriangleright \text{inputfileStarCamera}$  and angular accelerometer data  $\blacktriangleright \text{inputfileAngularAcc}$ . The combination of both observation types is achieved in a least square adjustment. The optimal weighting between the two different observation groups is achieved by means of VCE in combination with a robust estimator. The system of linearized observation equations within the sensor fusion approach can be formulated as:

$$\begin{bmatrix} \mathbf{l}_{ACC1B} \\ \mathbf{l}_{SCA1B} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{ACC1B} & \mathbf{B}_{ACC1B} \\ \mathbf{A}_{SCA1B} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{q} \\ \mathbf{b} \end{bmatrix} = \begin{bmatrix} \frac{\partial \omega}{\partial \mathbf{q}} & \frac{\partial \omega}{\partial \mathbf{b}} \\ \mathbf{I} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{q} \\ \mathbf{b} \end{bmatrix} \quad (4.12)$$

with

$$\begin{aligned} \mathbf{l}_{ACC1B} &= \dot{\omega}_{ACC1B} - \dot{\omega}_0, \\ \mathbf{l}_{SCA1B} &= \mathbf{q}_{SCA1B} - \mathbf{q}_0, \\ \mathbf{q}_{Fusion} &= \mathbf{q} + \mathbf{q}_0. \end{aligned} \quad (4.13)$$

The reference values  $\mathbf{q}_0$  and  $\dot{\omega}_0$  are derived from  $\blacktriangleright \text{inputfileStarCameraReference}$ . In the course of the estimation, the accelerometer data is calibrated, by setting a bias factor  $\mathbf{b}$  with  $\blacktriangleright \text{accBias}$ .

Name	Type	Annotation
$\blacktriangleright \text{outputfileStarCamera}$	filename	combined quaternions
$\blacktriangleright \text{outputfileCovariance}$	filename	epoch-wise covariance matrix
$\blacktriangleright \text{outputfileCovarianceMatrix}$	filename	full arc-wise covariance matrix per arc. arc number is appended to filename
$\blacktriangleright \text{outputfileEpochSigmaStarCamera}$	filename	from vce and outlier detection
$\blacktriangleright \text{outputfileEpochSigmaAccelerometer}$	filename	from vce and outlier detection
$\blacktriangleright \text{outputfileAngularAcc}$	filename	angular acceleration observations (bias removed)
$\blacktriangleright \text{outputfileSolution}$	filename	estimated parameter (one column for each arc)
$\blacktriangleright \text{inputfileStarCameraReference}$	filename	quaternions as taylor point
$\blacktriangleright \text{inputfileStarCamera}$	filename	star camera observations
$\blacktriangleright \text{inputfileStarCameraCovariance}$	filename	star camera observations
$\blacktriangleright \text{inputfileAngularAcc}$	filename	angular acceleration observations
$\blacktriangleright \text{correctAccNonQuadratic}$	boolean	apply correction (non-square proof mass)
$\blacktriangleright \text{accBias}$	parametrizationTemporal	accelerometer bias per interval and axis
$\blacktriangleright \text{accScale}$	parametrizationTemporal	accelerometer scale per interval and axis
$\blacktriangleright \text{sigmaStarcamera}$	double	[rad]
$\blacktriangleright \text{sigmaAccelerometerX}$	double	[rad/s^2]
$\blacktriangleright \text{sigmaAccelerometerY}$	double	[rad/s^2]
$\blacktriangleright \text{sigmaAccelerometerZ}$	double	[rad/s^2]
$\blacktriangleright \text{estimateSigmaScaPerAxis}$	boolean	separate variance factor for roll, pitch, yaw, instead of one common factor.
$\blacktriangleright \text{estimateSigmaAccPerAxis}$	boolean	separate variance factor for each accelerometer axis, instead of one common factor.
$\blacktriangleright \text{huber}$	double	residuals $\$;\$$ huber*sigma0 are downweighted

▶ huberPower	double	residuals \$i\$ huber: sigma=(e/huber)^power*sigma0
▶ interpolationDegree	uint	
▶ iterationCount	uint	non linear equation solved iteratively

---

This program is parallelized (1.5).

## 4.5 Programs: Gravityfield

### 4.5.1 Gravityfield2AbsoluteGravity

This program computes the absolute value of gravity  $\|\mathbf{g}\|$  of a **gravityfield** on a given **grid**. The result is multiplicated with **factor**. To get the full gravity vector in a terrestrial frame add the centrifugal part, see **gravityfield:tides:centrifugal**.

The values will be saved together with points expressed as ellipsoidal coordinates (longitude, latitude, height) based on a reference ellipsoid with parameters **R** and **inverseFlattening**.

It is intended to compute gravity anomalies from absolute gravity observations. To visualize the results use **PlotMap**.

Name	Type	Annotation
<b>outputfileGriddedData</b>	filename	
<b>grid</b>	grid	
<b>gravityfield</b>	gravityfield	
<b>factor</b>	double	the result is multiplied by this factor, set -1 to subtract the field
<b>time</b>	time	at this time the gravity field will be evaluated
<b>R</b>	double	reference radius for ellipsoidal coordinates on output
<b>inverseFlattening</b>	double	reference flattening for ellipsoidal coordinates on output, 0: spherical coordinates

This program is [parallelized \(1.5\)](#).

### 4.5.2 Gravityfield2AreaMeanTimeSeries

This program computes a time series of time variable **gravityfield** functionals averaged over a given area, e.g. equivalent water heights in the amazon basin. The type of functional (e.g gravity anomalies or geoid heights) can be choosen with **kernel**. The average is performed at each time step by a weighted average over all **grid** points where the weight is the associated area at each point. If **removeMean** is set the temporal mean is removed from the time series. To speed up the computation the gravity field can be converted to spherical harmonics before the computation with **convertToHarmonics**.

Additionally the root mean square of the values in the area at each time step can be computed if **computeRms** is set.

Additionally the accuracy of the value at each time step can be computed if **computeSigma** is set.

The **outputfileTimeSeries** is an instrument file with one, two, or three data columns. First data column contains the computed functionals and the following columns contain the RMS and the accuracies (optionally).

To visualize the results use **PlotGraph**.

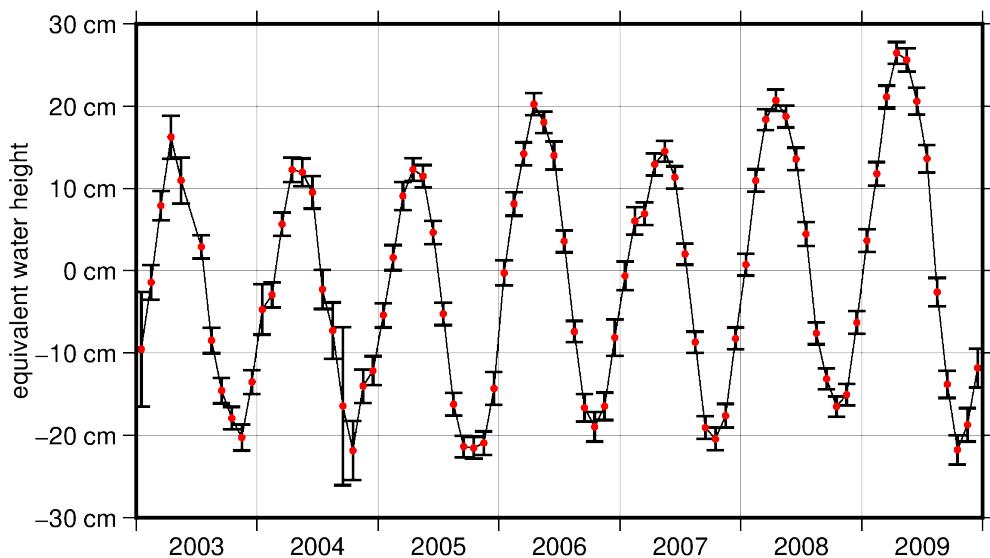


Figure 4.10: Amazon basin: Area mean values of ITSG-Grace2016 monthly solutions with error bars from computed sigma.

Name	Type	Annotation
outputfileTimeSeries	filename	
grid	grid	
timeSeries	timeSeries	
kernel	kernel	
gravityfield	gravityfield	
convertToHarmonics	boolean	gravityfield is converted to spherical harmonics before evaluation, may accelerate the computation
multiplyWithArea	boolean	multiply time series with total area (useful for mass estimates)
removeMean	boolean	remove the temporal mean of the series
computeRms	boolean	additional rms each time step
computeSigma	boolean	additional error bars at each time step

This program is [parallelized](#) (1.5).

### 4.5.3 Gravityfield2Deflections

This program computes the deflections of the vertical  $\xi$  in north direction and  $\eta$  in east direction in radian according to

$$\xi = g_x/\gamma \quad \text{and} \quad \eta = g_y/\gamma, \quad (4.14)$$

where  $\mathbf{g} = \nabla V$  is the gravity vector from **gravityfield** in the local ellipsoidal system (north, east, up) and  $\gamma$  is the normal gravity at that point.

The values will be saved together with points expressed as ellipsoidal coordinates (longitude, latitude, height) based on a reference ellipsoid with parameters **R** and **inverseFlattening**.

Name	Type	Annotation
outputfileGriddedData	filename	xi (north), eta (east) [rad]
grid	grid	
gravityfield	gravityfield	
time	time	at this time the gravity field will be evaluated
R	double	reference radius for ellipsoidal coordinates on output
inverseFlattening	double	reference flattening for ellipsoidal coordinates on output, 0: spherical coordinates

This program is parallelized (1.5).

#### 4.5.4 Gravityfield2DegreeAmplitudes

This program computes degree amplitudes from a **gravityfield** and saves them to a **matrix** file with three columns: the degree, the degree amplitude, and the formal errors.

The coefficients can be converted to different functionals with **kernel**. The gravity field can be evaluated at different altitudes by specifying **evaluationRadius**. Polar regions can be excluded by setting **polarGap**. If set the expansion is limited in the range between **minDegree** and **maxDegree** inclusively. The coefficients are related to the reference radius **R** and the Earth gravitational constant **GM**.

See also **PotentialCoefficients2DegreeAmplitudes**.

Name	Type	Annotation
<b>outputfileMatrix</b>	filename	three column matrix with degree, signal amplitude, formal error
<b>gravityfield</b>	<b>gravityfield</b>	
<b>kernel</b>	<b>kernel</b>	
<b>type</b>	choice	type of variances
<b>rms</b>		degree amplitudes (square root of degree variances)
<b>accumulation</b>		cumulate variances over degrees
<b>median</b>		meadian of absolute values per degree
<b>time</b>	time	at this time the gravity field will be evaluated
<b>evaluationRadius</b>	double	evaluate the gravity field at this radius (default: evaluate at surface
<b>polarGap</b>	angle	exclude polar regions (aperture angle in degrees)
<b>minDegree</b>	uint	
<b>maxDegree</b>	uint	
<b>GM</b>	double	Geocentric gravitational constant
<b>R</b>	double	reference radius

#### 4.5.5 Gravityfield2DegreeAmplitudesPlotGrid

This program computes a **timeSeries** of a time variable **gravityfield** and saves it as degree amplitudes. The expansion is limited in the range between **minDegree** and **maxDegree** inclusivly

$$\sigma_n = \frac{GM}{R} \left( \frac{R}{r} \right)^{n+1} k_n \sqrt{\sum_{m=0}^n c_{nm}^2 + s_{nm}^2}. \quad (4.15)$$

The **outputfileTimeSeries** is a matrix with every row containing the time, degree, degree amplitude, and the formal error.

To visualize the results use **PlotGraph**.

See also **Gravityfield2DegreeAmplitudes**.

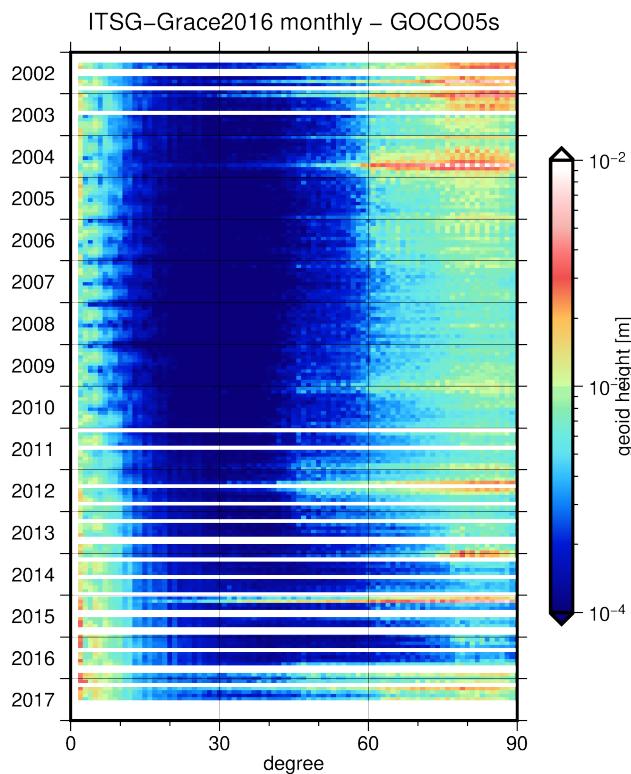


Figure 4.11: Degree amplitudes of monthly ITSG-Grace2016 solutions relative to GOCO05s.

Name	Type	Annotation
<b>outputfileTimeSeries</b>	filename	each row: mjd, degree, amplitude, formal error
<b>gravityfield</b>	gravityfield	
<b>kernel</b>	kernel	
<b>timeSeries</b>	timeSeries	
<b>evaluationRadius</b>	double	evaluate the gravity field at this radius (default: evaluate at surface)
<b>polarGap</b>	angle	exclude polar regions (aperture angle in degrees)
<b>minDegree</b>	uint	minimal degree
<b>maxDegree</b>	uint	maximal degree
<b>GM</b>	double	Geocentric gravitational constant
<b>R</b>	double	reference radius

### 4.5.6 Gravityfield2DisplacementTimeSeries

This program computes a time series of displacements of a list of stations ( **grid**) due to the effect of time variable loading masses. The displacement **u** of a station is calculated according to

$$\mathbf{u}(\mathbf{r}) = \frac{1}{\gamma} \sum_{n=0}^{\infty} \left[ \frac{h_n}{1+k_n} V_n(\mathbf{r}) \mathbf{e}_{up} + R \frac{l_n}{1+k_n} \left( \frac{\partial V_n(\mathbf{r})}{\partial \mathbf{e}_{north}} \mathbf{e}_{north} + \frac{\partial V_n(\mathbf{r})}{\partial \mathbf{e}_{east}} \mathbf{e}_{east} \right) \right], \quad (4.16)$$

where  $\gamma$  is the normal gravity, the load Love and Shida numbers  $h_n, l_n$  are given by **inputfileDeformationLoadLoveNumber** and the load Love numbers  $k_n$  are given by **inputfilePotentialLoadLoveNumber**. The  $V_n$  are the spherical harmonics expansion of the full time variable gravitational potential (potential of the loading mass + deformation potential):

$$V(\mathbf{r}) = \sum_{n=0}^{\infty} V_n(\mathbf{r}). \quad (4.17)$$

Deformations due to Earth tide and due to polar tides are computed using the IERS conventions. Eq. (4.16) is not used in these cases.

The **outputfileTimeSeries** is an **instrument file**, MISCVALUES. The data columns contain the deformation of each station in  $x, y, z$  in a global terrestrial reference frame or alternatively in a local ellipsoidal frame (north, east, up) if **localReferenceFrame** is set.

Name	Type	Annotation
<b>outputfileTimeSeries</b>	filename	x,y,z [m] per station
<b>grid</b>	grid	station list
<b>timeSeries</b>	timeSeries	
<b>gravityfield</b>	gravityfield	
<b>tides</b>	tides	
<b>earthRotation</b>	earthRotation	
<b>ephemerides</b>	ephemerides	
<b>inputfileDeformationLoadLoveNumber</b>	filename	
<b>inputfilePotentialLoadLoveNumber</b>	filename	if full potential is given and not only loading potential
<b>removeMean</b>	boolean	remove the temporal mean of each coordinate
<b>localReferenceFrame</b>	boolean	local left handed reference frame (north, east, up)

### 4.5.7 Gravityfield2EmpiricalCovariance

This program estimates an spatial and temporal covariance matrix from a time series of gravity fields.

Firstly for every time step  $t_i$  a spherical harmonics vector  $\mathbf{x}_i$  from the time variable gravity field is generated. The coefficients of the spherical harmonics expansion are in the sequence given by **numbering**. If set the expansion is limited in the range between **minDegree** and **maxDegree** inclusivly. The coefficients are related to the reference radius **R** and the Earth gravitational constant **GM**.

In the next step the empirical covariance matrix is estimated according to

$$\Sigma(\Delta i)_{full} = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i \mathbf{x}_{i+\Delta i}^T, \quad (4.18)$$

where  $\Delta i$  is given by **differenceStep**.

From the diagonal elements of  $\Sigma(\Delta i)$  the isotropic accuracies are computed

$$\sigma_n^2 = \frac{1}{2n+1} \sum_{m=0}^n \sigma_{cnm}^2 + \sigma_{snm}^2, \quad (4.19)$$

and a diagonal matrix is constructed  $\Sigma_{iso} = \text{diag}(\sigma_2^2, \dots, \sigma_N^2)$ . The result is computed:

$$\Sigma(\Delta i) = \alpha_{full} \Sigma(\Delta i)_{full} + \alpha_{iso} \Sigma(\Delta i)_{iso}. \quad (4.20)$$

Name	Type	Annotation
<b>outputfileCovarianceMatrix</b>	filename	
<b>outputfilePotentialCoefficients</b>	filename	
<b>gravityfield</b>	<b>gravityfield</b>	
<b>minDegree</b>	uint	
<b>maxDegree</b>	uint	
<b>GM</b>	double	Geocentric gravitational constant
<b>R</b>	double	reference radius
<b>numbering</b>	<b>sphericalHarmonicsNumbering</b>	numbering scheme for solution vector
<b>removeMean</b>	boolean	
<b>timeSeries</b>	<b>timeSeries</b>	sampling of the gravityfield
<b>differenceStep</b>	uint	choose dt for: $x,i(t) - x,j(t+dt)$
<b>factorFullMatrixPart</b>	double	
<b>factorIsotropicPart</b>	double	
<b>intervals</b>	<b>timeSeries</b>	

### 4.5.8 Gravityfield2Gradients

This program computes gravity gradients from **gravityfield** on a **grid** in a global terrestrial reference frame or alternatively in a local ellipsoidal frame (north, east, up) if **localReferenceFrame** is set. In **outputfileGriddedData** the values [ $V_{xx}, V_{yy}, V_{zz}, V_{xy}, V_{xz}, V_{yz}$ ] will be saved together with points expressed as ellipsoidal coordinates (longitude, latitude, height) based on a reference ellipsoid with parameters **R** and **inverseFlattening**.

Name	Type	Annotation
<b>outputfileGriddedData</b>	filename	$V_{xx} \ V_{yy} \ V_{zz} \ V_{xy} \ V_{xz} \ V_{yz}$
<b>grid</b>	grid	
<b>gravityfield</b>	gravityfield	
<b>localReferenceFrame</b>	boolean	local left handed reference frame (north, east, up)
<b>time</b>	time	at this time the gravity field will be evaluated
<b>R</b>	double	reference radius for ellipsoidal coordinates on output
<b>inverseFlattening</b>	double	reference flattening for ellipsoidal coordinates on output, 0: spherical coordinates

This program is [parallelized \(1.5\)](#).

### 4.5.9 Gravityfield2GridCovarianceMatrix

This program propagates the covariance matrix of a **gravityfield** evaluated at **time** to a **grid**. The full variance-covariance matrix is computed and written to a **matrix file**:

$$\Sigma_y = F \Sigma_x F^T \quad (4.21)$$

The **kernel** determines the quantity of the grid values, for example, **kernel:waterHeight**.

See also **GravityfieldCovariancesPropagation2GriddedData**, **GravityfieldVariancesPropagation2GriddedD**

Name	Type	Annotation
<b>outputfileMatrix</b>	filename	symmetric grid covariance matrix
<b>grid</b>	grid	
<b>kernel</b>	kernel	
<b>gravityfield</b>	gravityfield	
<b>time</b>	time	at this time the gravity field will be evaluated

### 4.5.10 Gravityfield2GriddedData

This program computes values of a **gravityfield** on a given **grid**. The type of value (e.g gravity anomalies or geoid heights) can be chosen with **kernel**. If a time is given the gravity field will be evaluated at this point of time otherwise only the static part will be used. The values will be saved together with points expressed as ellipsoidal coordinates (longitude, latitude, height) based on a reference ellipsoid with parameters **R** and **inverseFlattening**. To speed up the computation the gravity field can be converted to spherical harmonics before the computation with **convertToHarmonics**.

To visualize the results use **PlotMap**.

Name	Type	Annotation
outfileGriddedData	filename	
grid	grid	
kernel	kernel	
gravityfield	gravityfield	
convertToHarmonics	boolean	gravityfield is converted to spherical harmonics before evaluation, may accelerate the computation
time	time	at this time the gravity field will be evaluated
R	double	reference radius for ellipsoidal coordinates on output
inverseFlattening	double	reference flattening for ellipsoidal coordinates on output, 0: spherical coordinates

This program is parallelized (1.5).

#### 4.5.11 Gravityfield2GriddedDataTimeSeries

This program computes values of a **gravityfield** on a given **grid** for each time step of **timeSeries**. The type of value (e.g gravity anomalies or geoid heights) can be chosen with **kernel**. To speed up the computation the gravity field can be converted to spherical harmonics before the computation with **convertToHarmonics**. The **outputfileTimeSeries** is an instrument (MISCVALUES) file with a data column for each grid point per epoch.

This program enables the use of all instrument programs like **InstrumentFilter**, **InstrumentArcStatistics** or **InstrumentDetrend** to analyze and manipulate time series of gridded data.

See also **TimeSeries2GriddedData**, **Gravityfield2GriddedData**

Name	Type	Annotation
<b>outputfileTimeSeries</b>	filename	each epoch: data of grid points (MISCVALUES)
<b>grid</b>	grid	
<b>kernel</b>	kernel	
<b>gravityfield</b>	gravityfield	
<b>convertToHarmonics</b>	boolean	gravityfield is converted to spherical harmonics before evaluation, may accelerate the computation
<b>timeSeries</b>	timeSeries	

This program is parallelized (1.5).

### 4.5.12 Gravityfield2PotentialCoefficients

This program evaluates a time variable **gravityfield** at a given **time** and saves it as a **spherical harmonics file**. If set the expansion is limited in the range between **minDegree** and **maxDegree** inclusivly. The coefficients are related to the reference radius **R** and the Earth gravitational constant **GM**.

Name	Type	Annotation
outputfilePotentialCoefficients	filename	
gravityfield	gravityfield	
minDegree	uint	
maxDegree	uint	
GM	double	Geocentric gravitational constant
R	double	reference radius
time	time	at this time the gravity field will be evaluated

### 4.5.13 Gravityfield2PotentialCoefficientsTimeSeries

This program computes a **timeSeries** of a time variable **gravityfield** and converts to coefficients of a spherical harmonics expansion. The expansion is limited in the range between **minDegree** and **maxDegree** inclusivly. The coefficients are related to the reference radius **R** and the Earth gravitational constant **GM**.

The **outputfileTimeSeries** contains the potential coefficients as data columns for each epoch in the sequence given by **numbering**.

Name	Type	Annotation
<b>outputfileTimeSeries</b>	filename	instrument file (MISCVALUES)
<b>gravityfield</b>	<b>gravityfield</b>	
<b>timeSeries</b>	<b>timeSeries</b>	
<b>minDegree</b>	uint	
<b>maxDegree</b>	uint	
<b>GM</b>	double	Geocentric gravitational constant
<b>R</b>	double	reference radius
<b>numbering</b>	<b>sphericalHarmonicsNumbering</b>	numbering scheme

#### 4.5.14 Gravityfield2SphericalHarmonicsVector

This program evaluates a time variable **gravityfield** at a given **time** and saves a **vector** with the coefficients of a spherical harmonics expansion in the sequence given by **numbering**. If set the expansion is limited in the range between **minDegree** and **maxDegree** inclusively. The coefficients are related to the reference radius **R** and the Earth gravitational constant **GM**.

This coefficients vector can be used as a approximate solution, see **NormalsMultiplyAdd**, or as pseudo oberservations for regularization, see **normalEquation:regularization**.

For back transformation use **Gravityfield2PotentialCoefficients** with **gravityfield:fromParametrization**.

Name	Type	Annotation
<b>outfileVector</b>	filename	
<b>gravityfield</b>	<b>gravityfield</b>	
<b>startIndex</b>	uint	start index to put the coefficients in the solution vector
<b>minDegree</b>	uint	
<b>maxDegree</b>	uint	
<b>GM</b>	double	Geocentric gravitational constant
<b>R</b>	double	reference radius
<b>numbering</b>	<b>sphericalHarmonicsNumbering</b>	numbering scheme for solution vector
<b>time</b>	time	at this time the gravity field will be evaluated
<b>useSigma</b>	boolean	use formal errors instead of coefficients

#### 4.5.15 Gravityfield2TimeSplines

This program estimates splines in time domain from a time variable gravity field and writes **outputfileTimeSplines**. The **gravityfield** is sampled at **sampling**, converted to potential coefficients in the range between **minDegree** and **maxDegree** inclusively. The time series of spherical harmonics can be temporal filtered with **temporalFilter**.

In the next step temporal splines with **splineDegree** and nodal points given at **splineTimeSeries** are adjusted to the time series in a least squares sense. This is very fast for block means (**splineDegree = 0**) but for other splines a large systems of equations must be solved. In the adjustment process the time series of gravity fields can be interpreted as samples at the given times or as continuous function with linear behaviour between sampled points (**linearInterpolation**).

To combine a series of potential coefficients to a spline file with block means (**splineDegree = 0**) use the fast **PotentialCoefficients2BlockMeanTimeSplines** instead.

Name	Type	Annotation
<b>outputfileTimeSplines</b>	filename	
<b>gravityfield</b>	<b>gravityfield</b>	
<b>temporalFilter</b>	<b>digitalFilter</b>	filter sampled gravity field in time
<b>minDegree</b>	uint	
<b>maxDegree</b>	uint	
<b>GM</b>	double	Geocentric gravitational constant
<b>R</b>	double	reference radius
<b>sampling</b>	<b>timeSeries</b>	gravity field is sampled at these times
<b>removeMean</b>	boolean	remove the temporal mean of the series before estimating the splines
<b>linearInterpolation</b>	boolean	assume linear behavior between sampled points
<b>splineDegree</b>	uint	degree of splines
<b>splineTimeSeries</b>	<b>timeSeries</b>	nodal points of splines in time domain

#### 4.5.16 Gravityfield2TrendPotentialCoefficients

This program estimates **parametrizationTemporal** (e.g. mean, trend, annual) from a time variable gravity field.

In a first step a time variable **gravityfield** is sampled at **timeSeries** and converted to coefficients of a spherical harmonics expansion. The expansion is limited in the range between **minDegree** and **maxDegree** inclusively. The coefficients are related to the reference radius **R** and the Earth gravitational constant **GM**.

These coefficients serve as observations of a robust least squares adjustment (2.1) to estimate **parametrizationTemporal** parameters. For each temporal parameter an **outputfilePotentialCoefficients** is generated.

Name	Type	Annotation
<b>outputfilePotentialCoefficients</b>	filename	for each temporal parameter
<b>gravityfield</b>	<b>gravityfield</b>	
<b>timeSeries</b>	<b>timeSeries</b>	
<b>parametrizationTemporal</b>	<b>parametrizationTemporal</b>	
<b>minDegree</b>	uint	
<b>maxDegree</b>	uint	
<b>GM</b>	double	Geocentric gravitational constant
<b>R</b>	double	reference radius
<b>huber</b>	double	for robust least squares
<b>huberPower</b>	double	for robust least squares
<b>huberMaxIteration</b>	uint	(maximum) number of iterations for robust estimation

#### 4.5.17 GravityfieldCovariancesPropagation2GriddedData

This program computes the covariance between a source point given by longitude/latitude ( $\mathbf{L}$ ,  $\mathbf{B}$ ) and the points of a  $\mathbf{grid}$  in terms of the functional given by  $\mathbf{kernel}$  from the variance-covariance matrix of a  $\mathbf{gravityfield}$  evaluated at  $\mathbf{time}$ .

If  $\mathbf{computeCorrelation}$  is set, the program returns the correlation according to

$$r_{ij} = \frac{\sigma_{ij}}{\sigma_i \sigma_j} \quad (4.22)$$

in the range of  $[-1, 1]$  instead of the covariance.

See also [Gravityfield2GridCovarianceMatrix](#), [GravityfieldVariancesPropagation2GriddedData](#).

Name	Type	Annotation
$\mathbf{outputfileGriddedData}$	filename	gridded data file containing the covariance between source point and grid points
$\mathbf{grid}$	grid	
$\mathbf{kernel}$	kernel	functional
$\mathbf{gravityfield}$	gravityfield	
$\mathbf{time}$	time	at this time the gravity field will be evaluated
$\mathbf{L}$	angle	longitude of variance point
$\mathbf{B}$	angle	latitude of variance point
$\mathbf{height}$	double	ellipsoidal height of source point
$\mathbf{computeCorrelation}$	boolean	compute correlations instead of covariances
$\mathbf{R}$	double	reference radius for ellipsoidal coordinates on output
$\mathbf{inverseFlattening}$	double	reference flattening for ellipsoidal coordinates on output, 0: spherical coordinates

This program is parallelized (1.5).

### 4.5.18 GravityfieldReplacePotentialCoefficients

Replaces single potential coefficients in a gravity field. Both **gravityfield** and **gravityfieldReplacement** are evaluated at **time** and converted to spherical harmonic coefficients. Single **coefficients** are then replaced in **gravityfield** by the values from **gravityfieldReplacement** and the result is written to **outputfilePotentialCoefficients** from **minDegree** to **maxDegree**,

Name	Type	Annotation
<b>outputfilePotentialCoefficients</b>	filename	
<b>gravityfield</b>	<b>gravityfield</b>	
<b>gravityfieldReplacement</b>	<b>gravityfield</b>	single coefficients are replaced by the other gravityfield contains the coefficients for replacement
<b>coefficients</b>	choice	
└ <b>cnm</b>	sequence	
└ <b>degree</b>	uint	
└ <b>order</b>	uint	
└ <b>snm</b>	sequence	
└ <b>degree</b>	uint	
└ <b>order</b>	uint	
<b>minDegree</b>	uint	
<b>maxDegree</b>	uint	
<b>GM</b>	double	Geocentric gravitational constant
<b>R</b>	double	reference radius
<b>time</b>	time	at this time the gravity field will be evaluated

#### 4.5.19 GravityfieldVariancesPropagation2GriddedData

This program propagates variance-covariance matrix of a **gravityfield** evaluated at **time** to the points of a **grid** in terms of the functional given by **kernel**. The resulting **outputfileGriddedData** contains the standard deviations of the grid points.

See also **Gravityfield2GridCovarianceMatrix**, **GravityfieldCovariancesPropagation2GriddedData**.

Name	Type	Annotation
<b>outputfileGriddedData</b>	filename	standard deviation at each grid point
<b>grid</b>	grid	
<b>kernel</b>	kernel	functional
<b>gravityfield</b>	gravityfield	
<b>time</b>	time	at this time the gravity field will be evaluated
<b>R</b>	double	reference radius for ellipsoidal coordinates on output
<b>inverseFlattening</b>	double	reference flattening for ellipsoidal coordinates on output, 0: spherical coordinates

This program is parallelized (1.5).

## 4.6 Programs: Grid

### 4.6.1 GriddedData2AreaMeanTimeSeries

This program computes a time series of area mean values in a basin represented by **► border** from a sequence of grid files. If a file is not found, the epoch is skipped. Per default the weighted average of all points in the given border is computed where the points are weighted by their area element.

If **► computeMean** is set, the time average of each grid points is subtracted before the computation. If **► multiplyWithArea** is set, the weighted average is multiplied with the total basin area. This is useful for computing the total mass in the basin.

The **► outputFileTimeSeries** is an instrument file, where the first columns are the mean value each data column in the grid files, followed by the the weighted RMS for each data column in the grid files if **► computeRms** is set. If the number of data columns differs between the grid files, the remaining columns are padded with zeros.

See also **► Gravityfield2AreaMeanTimeSeries**.

Name	Type	Annotation
<b>► outputFileTimeSeries</b>	filename	
<b>► inputFileGriddedData</b>	filename	
<b>► border</b>	border	
<b>► timeSeries</b>	timeSeries	
<b>► multiplyWithArea</b>	boolean	multiply time series with total area (useful for mass estimates)
<b>► removeMean</b>	boolean	remove the temporal mean of the series
<b>► computeRms</b>	boolean	additional rms each time step

### 4.6.2 GriddedData2GriddedDataStatistics

This program assigns values **inputfileGriddedData** to the nearest points of a new **grid**. If some of the new points are not filled in with data **emptyValue** is used instead. If multiple points of the input fall on the same node the result can be selected with **statistics** (e.g. mean, root mean square, min, max, ...). It also is possible to simply count the number of data points that were assigned to each point.

Be aware in case borders are given within **grid**, the **outputfileGriddedData** will have points excluded before the assignment of old points to the new points. The data from **inputfileGriddedData** will not be limited by the given borders! See [GriddedDataConcatenate \(4.6.9\)](#) to limit the **inputfileGriddedData** to given borders.

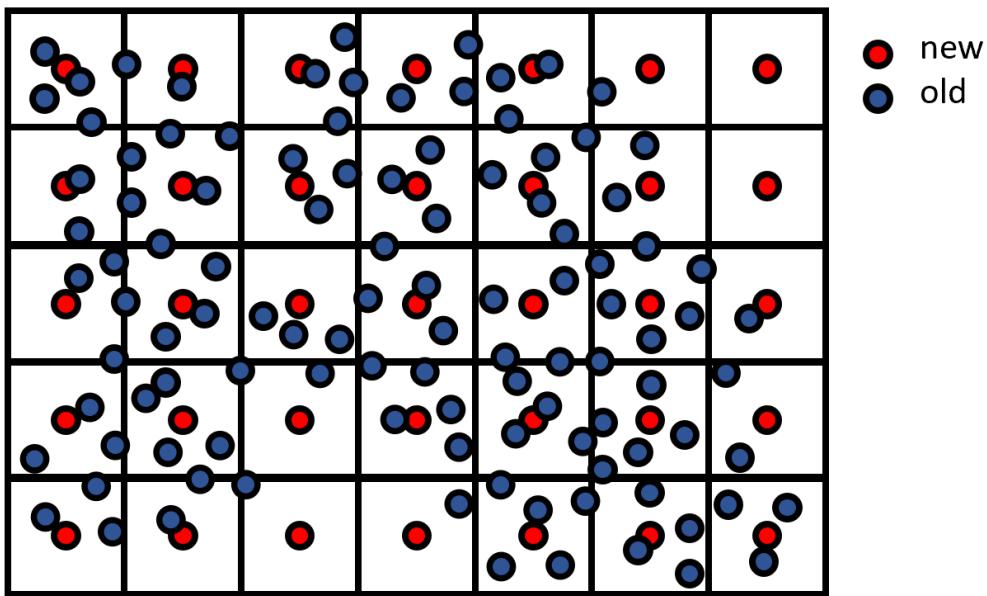


Figure 4.12: Assignment of irregular distributed data to grid.

Name	Type	Annotation
<b>outputfileGriddedData</b>	filename	
<b>inputfileGriddedData</b>	filename	
<b>grid</b>	grid	
<b>statistic</b>	choice	statistic used if multiple values fall on the same cell
<b>mean</b>		mean
<b>wmean</b>		area weighted mean
<b>rms</b>		root mean square
<b>wrms</b>		area weighted root mean square
<b>std</b>		standard deviation
<b>wstd</b>		area weighted standard deviation
<b>sum</b>		sum
<b>min</b>		minimum value
<b>max</b>		maximum value
<b>count</b>		number of values
<b>first</b>		first value
<b>last</b>		last value
<b>emptyValue</b>	double	value for nodes without data
<b>R</b>	double	reference radius for ellipsoidal coordinates
<b>inverseFlattening</b>	double	reference flattening for ellipsoidal coordinates

### 4.6.3 GriddedData2GriddedDataTableSeries

Write a series of **inputfileGriddedData** with the corresponding **timeSeries** as a single **gridded data time series file**. The **splineDegree** defines the possible temporal interpolation of data in the output file. For a file with spline degree 0 (temporal block means) the time intervals in which the grids are valid are defined between adjacent points in time. Therefore one more point in time is needed than the number of input grid files for degree 0.

See also **GriddedDataTableSeries2GriddedData**.

Name	Type	Annotation
outputfileGriddedDataTableSeries	filename	
inputfileGriddedData	filename	file count must agree with number of times+splineDegree-1
timeSeries	timeSeries	
splineDegree	uint	degree of splines

#### 4.6.4 GriddedData2Matrix

This program converts **inputfileGriddedData** to **outputfileMatrix** with data columns. The grid is expressed as ellipsoidal coordinates (longitude, latitude, height) based on a reference ellipsoid with parameters **R** and **inverseFlattening**. The content of the output matrix can be controlled by **outColumn** expressions applied to every grid point. The common data variables for grids are available, see [dataVariables \(1.2.3\)](#).

Name	Type	Annotation
outputfileMatrix	filename	point list as matrix with longitude and latitude values in columns and possible additional columns
inputfileGriddedData	filename	
R	double	reference radius for ellipsoidal coordinates on output
inverseFlattening	double	reference flattening for ellipsoidal coordinates on output, 0: spherical coordinates
outColumn	expression	expression (variables: longitude, latitude, height, area, data0, data1, ...)

### 4.6.5 GriddedData2PotentialCoefficients

This program estimate potential coefficients from **inputfileGriddedData** gravity field functionals. It used a simple quadrature formular

$$c_{nm} = \frac{1}{4\pi} \frac{R}{GM} \sum_i f_i \left( \frac{r_i}{R} \right)^{n+1} k_n C_{nm}(\lambda_i, \vartheta_i) \Delta\Phi_i \quad (4.23)$$

or a **leastSquares** adjustment with block diagonal normal matrix (order by order). For the latter one the data must be regular distributed.

The **values**  $f_i$  and the **weights**  $\Delta\Phi_i$  are expressions using the common data variables for grids, see [dataVariables \(1.2.3\)](#). Multiple **outputfilePotentialCoefficients** can be estimated in one step. For each an individual **value** must be specified. The type of the gridded data (e.g gravity anomalies or geoid heights) must be set with **kernel**  $k_n$ .

The expansion is limited in the range between **minDegree** and **maxDegree** inclusively. The coefficients are related to the reference radius **R** and the Earth gravitational constant **GM**.

For irregular distributed data and using the full variance covariance matrix use **NormalsSolverVCE** together with **oberservation:terrestrial** and **parametrizationGravity:sphericalHarmonics**.

See also **GriddedDataTimeSeries2PotentialCoefficients**.

Name	Type	Annotation
<b>outputfilePotentialCoefficients</b>	filename	one file for each value expression
<b>inputfileGriddedData</b>	filename	
<b>value</b>	expression	expression to compute values (input columns are named data0, data1, ...)
<b>weight</b>	expression	expression to compute values (input columns are named data0, data1, ...)
<b>kernel</b>	kernel	data type of input values
<b>minDegree</b>	uint	
<b>maxDegree</b>	uint	
<b>GM</b>	double	Geocentric gravitational constant
<b>R</b>	double	reference radius
<b>leastSquares</b>	boolean	false: quadrature formular, true: least squares adjustment order by order

This program is [parallelized \(1.5\)](#).

#### 4.6.6 GriddedData2SphericalDistance

Compute the spherical distance on the unit sphere in radians between all point pairs of two grids. The spherical distance is computed by

$$\psi_{12} = \arccos(\mathbf{n}_1 \cdot \mathbf{n}_2), \quad (4.24)$$

where  $\mathbf{n}_i$  is the (normalized) position. This implies that all points are projected onto the unit sphere.

Name	Type	Annotation
outputfileMatrix	filename	matrix containing the spherical distance between all point pairs [rad]
grid1	grid	
grid2	grid	

### 4.6.7 GriddedData2TimeSeries

Write a series of **inputfileGriddedData** with the corresponding **timeSeries** as a single time series file (**instrument**, MISCVALUES).

If **groupDataByPoints** is true the **outputfileTimeSeries** starts for each epoch with all data (data0, data1...) for the first point, followed by all data of the second point and so on. If **groupDataByPoints** is false, the file starts with data0 for all points, followed by all data1 and so on.

This enables the use of all instrument programs like **InstrumentFilter** or **InstrumentDetrend** to analyze and manipulate time series of gridded data.

See also **TimeSeries2GriddedData**.

Name	Type	Annotation
<b>outputfileTimeSeries</b>	filename	each epoch: multiple data for points (MISCVALUES)
<b>inputfileGriddedData</b>	filename	file count must agree with number of times
<b>timeSeries</b>	<b>timeSeries</b>	
<b>groupDataByPoints</b>	boolean	multiple data are given point by point, otherwise: data0 for all points, followed by all data1

### 4.6.8 GriddedDataCalculate

This program manipulates **grid files** with data in columns similar to **FunctionsCalculate**, see there for more details. If several **inputfiles** are given the data columns are copied side by side. All **inputfiles** must contain the same grid points. The columns are enumerated by **data0**, **data1**, ... .

The content of **outfileGriddedData** is controlled by **outColumn**. The algorithm to compute the output is as follows: The expressions in **outColumn** are evaluated once for each grid point of the input. The variables **data0**, **data1**, ... are replaced by the according values from the input columns before. Additional variables are available, e.g. **index**, **data0rms**, see **dataVariables (1.2.3)**.

For a simplified handling **constants** can be defined by **name=value**. It is also possible to estimate **parameters** in a least squares adjustment. The **leastSquares** serves as template for observation equations for every point. The expression **leastSquares** is evaluated for each grid point. The variables **data0**, **data1**, ... are replaced by the according values from the input columns before. In the next step the parameters are estimated in order to minimize the expressions in **leastSquares** in the sense of least squares.

Afterwards grid points are removed if one of the **removalCriteria** expressions for this grid point evaluates true (not zero).

An extra **statistics:outfile** can be generated with one row of data. For the computation of the **outColumn** values all **dataVariables (1.2.3)** are available (e.g. **data3mean**, **data4std**) inclusively the **constants** and estimated **parameters** but without the **data0**, **data1**, ... itself. The variables and the numbering of the columns refers to the **outfileGriddedData**.

See also **FunctionsCalculate**, **InstrumentArcCalculate**, **MatrixCalculate**.

Name	Type	Annotation
<b>outfileGriddedData</b>	filename	
<b>inputfileGriddedData</b>	filename	
<b>constant</b>	expression	define a constant by <b>name=value</b>
<b>parameter</b>	expression	define a parameter by <b>name[=value]</b>
<b>leastSquares</b>	expression	try to minimize the expression by adjustment of the parameters
<b>removalCriteria</b>	expression	points are removed if one criterion evaluates true. <b>data0</b> is the first data field.
<b>longitude</b>	expression	
<b>latitude</b>	expression	
<b>height</b>	expression	
<b>area</b>	expression	expression: e.g. $\text{deltaL} * 2.0 * \sin(\text{deltaB}/2.0) * \cos(\text{latitude}/\rho)$
<b>value</b>	expression	expression to compute values (input columns are named <b>data0</b> , <b>data1</b> , ...)
<b>computeArea</b>	boolean	automatically area computation of rectangular grids (overwrite area)
<b>R</b>	double	reference radius for ellipsoidal coordinates
<b>inverseFlattening</b>	double	reference flattening for ellipsoidal coordinates
<b>statistics</b>	sequence	
<b>outfile</b>	filename	matrix with one row, columns are user defined
<b>outColumn</b>	expression	expression to compute statistics columns, <b>data*</b> are the output-Columns

### 4.6.9 GriddedDataConcatenate

This program concatenate grid from several **inputfileGriddedData** and write it to a new **outputfileGriddedData**. Input files must have the same number of data columns. If **sort** is enabled, the points are sorted by latitudes starting from north/west to south east. Identical points (within a **margin**) can be removed with **removeDuplicates**.

Name	Type	Annotation
outputfileGriddedData	filename	
inputfileGriddedData	filename	
border	border	
sortPoints	boolean	sort from north/west to south east
removeDuplicates	choice	remove duplicate points
keepFirst	sequence	keep first point, remove all other identicals
margin	double	margin distance for identical points [m]
keepLast	sequence	keep last point, remove all other identicals
margin	double	margin distance for identical points [m]
R	double	reference radius for ellipsoidal coordinates
inverseFlattening	double	reference flattening for ellipsoidal coordinates

#### 4.6.10 GriddedDataCreate

This program creates a **grid** and writes it to **outputfileGrid**. The grid is expressed as ellipsoidal coordinates (longitude, latitude, height) based on a reference ellipsoid with parameters **R** and **inverseFlattening**. Extra **value** columns can be appended using expressions with the common **data variables** (1.2.3) for gridded data.

Name	Type	Annotation
outputfileGrid	filename	
grid	grid	
R	double	reference radius for ellipsoidal coordinates on output
inverseFlattening	double	reference flattening for ellipsoidal coordinates on output, 0: spherical coordinates
value	expression	expression (variables as 'longitude', 'height', 'area' are taken from the gridded data)

### 4.6.11 GriddedDataInterpolate

Interpolate values of a regular rectangular **inputfileGriddedData** to new points given by **grid** and write as **outputfileGriddedData**. Only longitude and latitude of points are considered; the height is ignored for interpolation.

(Only nearest neighbor method is implemented at the moment.)

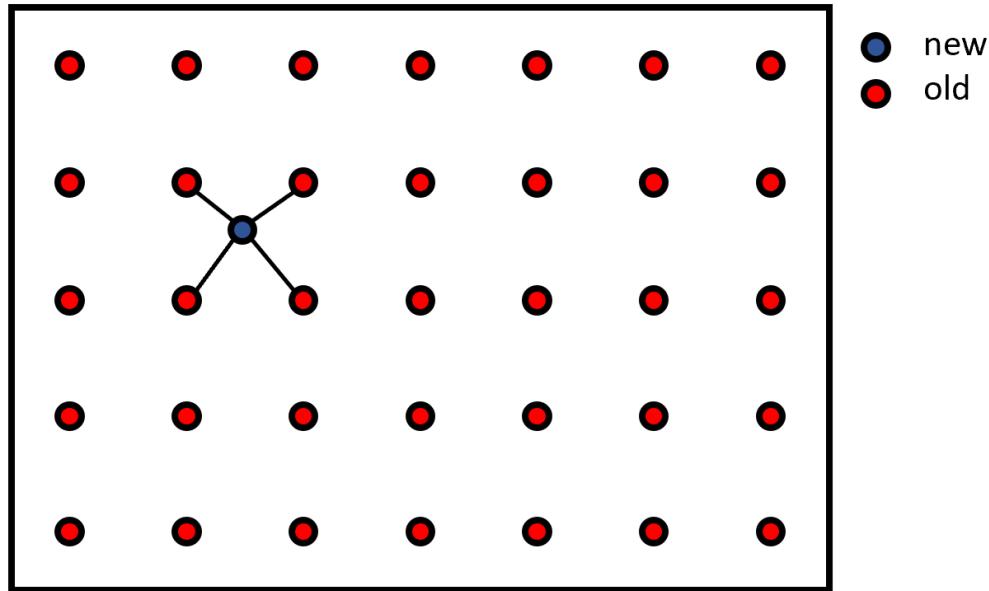


Figure 4.13: Interpolation of point data from rectangular gridded data.

Name	Type	Annotation
outputfileGriddedData	filename	
inputfileGriddedData	filename	must be rectangular
grid	grid	
method	choice	
nearestNeighbor		

### 4.6.12 GriddedDataReduceSampling

Generate coarse grid by computing area weighted mean values. The number of points is decimated by averaging integer multiples of grid points ( $\blacktriangleright$  **multiplierLongitude**,  $\blacktriangleright$  **multiplierLatitude**).

if  $\blacktriangleright$  **volumeConserving** is set, data are interpreted as heights above ellipsoid and the tesseroid volume

$$V = \int_r^{r+H} \int_{\varphi_1}^{\varphi_2} \int_{\lambda_1}^{\lambda_2} r^2 \cos \varphi d\varphi d\lambda dr \quad (4.25)$$

is conserved, where  $r$  is the radius of the ellipsoid at grid center and  $(\varphi_1 - \varphi_2) \times (\lambda_1 - \lambda_2)$  are the grid cell boundaries. This is meaningful for Digital Elevation Models (DEM).

The fine grid can be written, where the first coarse grid values (data0) are additionally appended.

Name	Type	Annotation
$\blacktriangleright$ <b>outputfileCoarseGridRectangular</b>	filename	coarse grid
$\blacktriangleright$ <b>outputfileFineGridRectangular</b>	filename	fine grid with additional coarse grid values
$\blacktriangleright$ <b>inputfileFineGridRectangular</b>	filename	Digital Terrain Model
$\blacktriangleright$ <b>multiplierLongitude</b>	uint	Generalizing factor
$\blacktriangleright$ <b>multiplierLatitude</b>	uint	Generalizing factor
$\blacktriangleright$ <b>volumeConserving</b>	boolean	data are interpreted as heights above ellipsoid

### 4.6.13 GriddedDataTimeSeries2GriddedData

Read a **inputfileGriddedDataTimeSeries** and write for each epoch a **gridded data** file where the **variableLoopTime** and **variableLoopIndex** are expanded for each point of the given **timeSeries** to create the file name for this epoch (see [text parser \(1.2.2\)](#)).

If **timeSeries** is not set the temporal nodal points from the inputfile are used.

See also **GriddedData2GriddedDataTimeSeries**.

Name	Type	Annotation
outputfilesGriddedData	filename	for each epoch
variableLoopTime	string	variable with time of each epoch
variableLoopIndex	string	variable with index of current epoch (starts with zero)
variableLoopCount	string	variable with total number of epochs
inputfileGriddedDataTimeSeries	filename	
timeSeries	timeSeries	otherwise times from inputfile are used

#### 4.6.14 GriddedDataTimeSeries2PotentialCoefficients

This program estimate potential coefficients from **inputfileGriddedDataTimeSeries** in the same way as **GriddedData2PotentialCoefficients** but not only for one grid but for each epoch of **timeSeries** of if not set for the temporal nodal points from the inputfile. The **outputfilePotentialCoefficients** (one for each **value**) are written for each epoch with the expansion of **variableLoopTime** and **variableLoopIndex** (see [text parser \(1.2.2\)](#)).

See also **GriddedData2PotentialCoefficients**.

Name	Type	Annotation
...* outputfilesPotentialCoefficients	filename	for each epoch
▶ variableLoopTime	string	variable with time of each epoch
▶ variableLoopIndex	string	variable with index of current epoch (starts with zero)
▶ variableLoopCount	string	variable with total number of epochs
▶ inputfileGriddedDataTimeSeries	filename	
▶ timeSeries	timeSeries	otherwise times from inputfile are used
...* value	expression	expression (variables: longitude, latitude, height, area, data0, data1, ...)
▶ weight	expression	expression to compute values (input columns are named data0, data1, ...)
▶ kernel	kernel	kernel in which the grid values are given
▶ minDegree	uint	
...* maxDegree	uint	
▶ GM	double	Geocentric gravitational constant
▶ R	double	reference radius for potential coefficients
▶ leastSquares	boolean	false: quadrature formular, true: least squares adjustment order by order

This program is [parallelized \(1.5\)](#).

### 4.6.15 GriddedTopography2AtmospherePotentialCoefficients

Estimate interior and exterior potential coefficients for atmosphere above digital terrain models. Coefficients for interior  $(1/r)^{n+1}$  and exterior  $(r^n)$  are computed. The density of the atmosphere is assumed to be (Sjöberg, 1998)

$$\rho_0 \left( \frac{R}{R+h} \right)^\nu, \quad (4.26)$$

where  $R$  is the radial distance of the ellipsoid at each point,  $h$  the radial height above the ellipsoid,  $\rho_0$  is **densitySeaLevel** and **nu**  $\nu$  is a constant factor. The density is integrated from **radialLowerBound** and **upperAtmosphericBoundary** above the ellipsoid. The **radialLowerBound** is typically the topography and can be computed as expression at every point from **inputfileGriddedData**.

Name	Type	Annotation
outputfilePotentialCoefficientsExterior	filename	
outputfilePotentialCoefficientsInterior	filename	
inputfileGriddedData	filename	Digital Terrain Model
densitySeaLevel	double	[kg/m**3]
ny	double	Constant for Atmosphere
radialLowerBound	expression	expression (variables 'L', 'B', 'height', 'data', and 'area' are taken from the gridded data)
upperAtmosphericBoundary	double	constant upper bound [m]
factor	double	the result is multiplied by this factor, set -1 to subtract the field
minDegree	uint	
maxDegree	uint	
GM	double	Geocentric gravitational constant
R	double	reference radius

This program is [parallelized \(1.5\)](#).

#### 4.6.16 GriddedTopography2PotentialCoefficients

Estimate potential coefficients from digital terrain models. Coefficients for interior  $(1/r)^{n+1}$  and exterior  $(r^n)$  are computed.

Name	Type	Annotation
▶ outputfilePotentialCoefficients	filename	
▶ outputfilePotentialCoefficientsInterior	filename	
▶ inputfileGriddedData	filename	Digital Terrain Model
▶ density	expression	expression [ $\text{kg}/\text{m}^3$ ]
▶ radialUpperBound	expression	expression (variables 'L', 'B', 'height', 'data', and 'area' are taken from the gridded data)
▶ radialLowerBound	expression	expression (variables 'L', 'B', 'height', 'data', and 'area' are taken from the gridded data)
▶ factor	double	the result is multiplied by this factor
▶ minDegree	uint	
▶ maxDegree	uint	
▶ GM	double	Geocentric gravitational constant
▶ R	double	reference radius

This program is parallelized (1.5).

#### 4.6.17 GriddedTopographyEllipsoidal2Radial

Interpolate digital terrain models from ellipsoidal heights to radial heights.

Name	Type	Annotation
▶ outputfileGriddedData	filename	
◀ inputfileGriddedData	filename	Digital Terrain Model

#### 4.6.18 Matrix2GriddedData

This program reads a [matrix file](#) with data in columns and convert into [gridded data](#). The input columns are enumerated by `data0`, `data1`, ..., see [dataVariables \(1.2.3\)](#).

Name	Type	Annotation
▶ <code>outputfileGriddedData</code>	filename	
▶ <code>inputfileMatrix</code>	filename	
▶ <code>points</code>	choice	
▶ <code>ellipsoidal</code>	sequence	
▶ <code>longitude</code>	expression	expression
▶ <code>latitude</code>	expression	expression
▶ <code>height</code>	expression	expression
▶ <code>cartesian</code>	sequence	
▶ <code>x</code>	expression	expression
▶ <code>y</code>	expression	expression
▶ <code>z</code>	expression	expression
▶ <code>area</code>	expression	expression (e.g. $\delta L * 2 * \sin(\delta B / 2) * \cos(data1 / RHO)$ )
▶ <code>value</code>	expression	expression
▶ <code>sortPoints</code>	boolean	sort from north/west to south east
▶ <code>computeArea</code>	boolean	the area can be computed automatically for rectangular grids
▶ <code>R</code>	double	reference radius for ellipsoidal coordinates
▶ <code>inverseFlattening</code>	double	reference flattening for ellipsoidal coordinates

#### 4.6.19 MatrixRectangular2GriddedData

Read gridded data (matrix).

Name	Type	Annotation
outputfileGriddedData	filename	
inputfileMatrix	filename	
rowMajor	boolean	true: data is ordered row by row, false: columnwise
startLongitude	angle	longitude of upper left corner of the grid
startLatitude	angle	latitude of upper left corner of the grid
deltaLongitude	angle	sampling, negative for east to west data
deltaLatitude	angle	sampling, negative for south to north data
R	double	reference radius for ellipsoidal coordinates
inverseFlattening	double	reference flattening for ellipsoidal coordinates

#### 4.6.20 TimeSeries2GriddedData

Interpret the data columns of **inputfileTimeSeries** as data points of a corresponding **grid**.

For each epoch a **gridded data file** is written where the **variableLoopTime** and **variableLoopIndex** are expanded for each point of the given time series to create the file name for this epoch (see [text parser \(1.2.2\)](#)).

The number of input data columns must be a multiple of the number  $n$  of grid points. If **isGroupedDataByPoint** is true the **inputfileTimeSeries** starts with all data (`data0, data1...`) for the first point, followed by all data of the second point and so on. If **isGroupedDataByPoint** is false, the file starts with `data0` for all points, followed by all `data1` and so on.

See also **GriddedData2TimeSeries**.

Name	Type	Annotation
outputfilesGriddedData	filename	for each epoch
variableLoopTime	string	variable with time of each epoch
variableLoopIndex	string	variable with index of current epoch (starts with zero)
variableLoopCount	string	variable with total number of epochs
inputfileTimeSeries	filename	each epoch: multiple data for points (MISCVALUES)
grid	grid	corresponding grid points
isDataGroupedByPoint	boolean	multiple data are given point by point, otherwise: first data0 for all points, followed by all data1
R	double	reference radius for ellipsoidal coordinates on output
inverseFlattening	double	reference flattening for ellipsoidal coordinates on output, 0: spherical coordinates

## 4.7 Programs: Instrument

### 4.7.1 Instrument2AllanVariance

This program computes the overlapping Allan variance from an  **inputfileInstrument**. The estimate is averaged over all arcs (arcs are assumed to contain no data gaps).

The overlapping Allan variance is defined as

$$\sigma^2(m\tau_0) = \frac{1}{2(m\tau_0)^2(N - 2m)} \sum_{n=1}^{N-2m} (x_{n+2m} - 2x_{n+m} + x_n)^2, \quad (4.27)$$

where  $m\tau_0$  is the averaging interval defined by the median sampling  $\tau_0$ .

---

Name	Type	Annotation
 <b>outputfileAllanVariance</b>	filename	column 0: averaging interval [seconds], column 1-(n-1): Allan variance for each data column
 <b>inputfileInstrument</b>	filename	

---

This program is [parallelized \(1.5\)](#).

### 4.7.2 Instrument2CovarianceFunctionVCE

This estimates a covariance function of **inputfileInstrument** for all selected columns with **startDataFields** and **countDataFields**. The estimation is performed robustly via variance component estimation. Bad arcs are downweigthed and the accuracies can be written with **outputfileSigmasPerArc**. The length of the covariance functions are determined by the longest arc. Additionally the data can be detrended with **parameter** and **parameterPerArc**.

Name	Type	Annotation
outputfileCovarianceFunction	filename	covariance functions
outputfileSigmasPerArc	filename	accuracies of each arc
outputfileResiduals	filename	
outputfileSolution	filename	estimated parameter vector (global part only)
inputfileInstrument	filename	
startDataFields	uint	start
countDataFields	uint	number of data fields (default: all after start)
parameter	parametrizationTemporal	data is reduced by temporal representation
parameterPerArc	parametrizationTemporal	data is reduced by temporal representation
iterationCount	uint	number of iterations for the estimation

This program is parallelized (1.5).

### 4.7.3 Instrument2CrossCorrelationFunction

This program computes the cross correlation between all corresponding data columns in two **Instrument files**. The instrument files must be synchronized ( **InstrumentSynchronize**). The **outputfileCorrelation** is a matrix with the first column containing the time lag followed by cross-correlation function for each data column. The maximum lag is defined by the maximum arc length.

The correlation is based on the unbiased estimate of the cross-covariance between data columns  $x$  and  $y$ ,

$$\sigma_{xy}(h) = \frac{1}{N} \sum_{k=1} x_{k+h} y_k, \quad (4.28)$$

which is averaged over all arcs. From this estimate, the correlation for each lag is then computed via

$$r_{xy}(h) = \frac{\sigma_{xy}(h)}{\sigma_x(0)\sigma_y(0)}, \quad (4.29)$$

which is the ratio between the biased estimates of the cross-covariance at lag  $h$  and the auto-covariance of the individual data columns.

For instrument with data gaps, lag bins without any data are set to NAN.

Name	Type	Annotation
<b>outputfileCorrelation</b>	filename	column 1: time lag, column 2..n cross-correlation
<b>inputfileInstrument</b>	filename	
<b>inputfileInstrumentReference</b>	filename	

This program is [parallelized \(1.5\)](#).

#### 4.7.4 Instrument2Histogram

This program computes the arc-wise histogram from an [Instrument file](#). The output is a [matrix](#) with the first column containing the lower bound of each bin. The other columns contain the histograms for each arc.

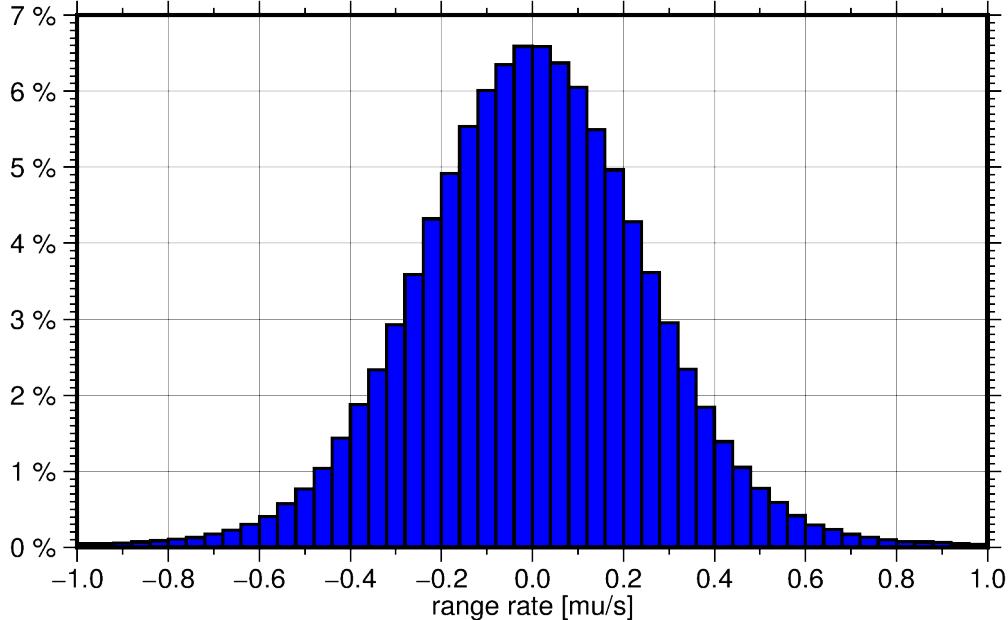


Figure 4.14: GRACE range-rate residuals of one month (one arc) divided into 50 bins.

Name	Type	Annotation
outputfileMatrix	filename	column 1: lower bin bound; columns 2 to N: histogram of each arc
inputfileInstrument	filename	
selectDataField	uint	select channel for histogram computation
binCount	uint	(default: Freedman-Diaconis' choice, maximum of all channels)
lowerBound	expression	lower bound for bins (default: global minimum, data values outside are ignored)
upperBound	expression	upper bound for bins (default: global maximum, data values outside are ignored)
relative	boolean	output relative frequencies
cumulative	boolean	accumulate frequencies

This program is parallelized (1.5).

### 4.7.5 Instrument2PowerSpectralDensity

This program computes the power spectral density (PSD) for all data fields in an **Instrument file**. The PSD is computed using Lomb's method. For each arc and each frequency  $f$ , a sinusoid is fit to the data

$$l_i = a \cos(2\pi f t_i) + b \sin(2\pi f t_i) + e_i \quad (4.30)$$

The PSD for this frequency is then computed by forming the square sum of adjusted observations:

$$P(f) = \sum_i \hat{l}_i^2. \quad (4.31)$$

The resulting PSD is the average over all arcs. For regularly sampled time series, this method yields the same results as FFT based PSD estimates.

A regular frequency grid based on the longest arc and the median sampling is computed. The maximum number of epochs per arc is determined by

$$N = \frac{t_{\text{end}} - t_{\text{start}}}{\Delta t_{\text{median}}} + 1, \quad (4.32)$$

the Nyquist frequency is given by

$$f_{\text{nyq}} = \frac{1}{2\Delta t_{\text{median}}}. \quad (4.33)$$

If it is suspected that **Instrument** contains secular variations, the input should be detrended using **InstrumentDetrend**.

See also **Instrument2CovarianceFunctionVCE**, **CovarianceFunction2PowerSpectralDensity**, **PowerSpectralDensity2CovarianceFunction**.

Name	Type	Annotation
<b>outputfilePSD</b>	filename	estimated PSD: column 0: frequency vector, column 1-(n-1): PSD estimate for each channel
<b>inputfileInstrument</b>	filename	

This program is **parallelized** (1.5).

#### 4.7.6 Instrument2RmsPlotGrid

This program computes an RMS plot grid from one or more **inputfileInstrument** containing 3D data (e.g. orbits or station positions), which can then be plotted as gridded data in **PlotGraph**. The RMS is computed from the difference between **inputfileInstrument** and **inputfileInstrumentReference**. All instrument files must be synchronized (see **InstrumentSynchronize**).

Each separate **inputfileInstrument** represents an entry (e.g. a satellite or station) in the resulting grid. Therefore, providing, for example, 32 orbit files of GPS satellites results in a grid with columns: mjd, id (0-31), rms.

The first three data columns of the instrument data are considered for computation of the RMS values. The **factor** can be set to, for example,  $\text{sqrt}(3)$  to get 3D instead of 1D RMS values.

If **timeIntervals** are provided, each **inputfileInstrument** and **inputfileInstrumentReference** serves as a template with variable **loopTime**. This allows concatenation of instrument files, for example to create a month-long RMS plot grid from daily GPS orbit files (see below).

Helmer parameters between the two frames can be estimated each epoch optionally if **estimateShift**, **estimateScale**, or **estimateRotation** are set. It uses a [robust least squares adjustment \(2.1\)](#).

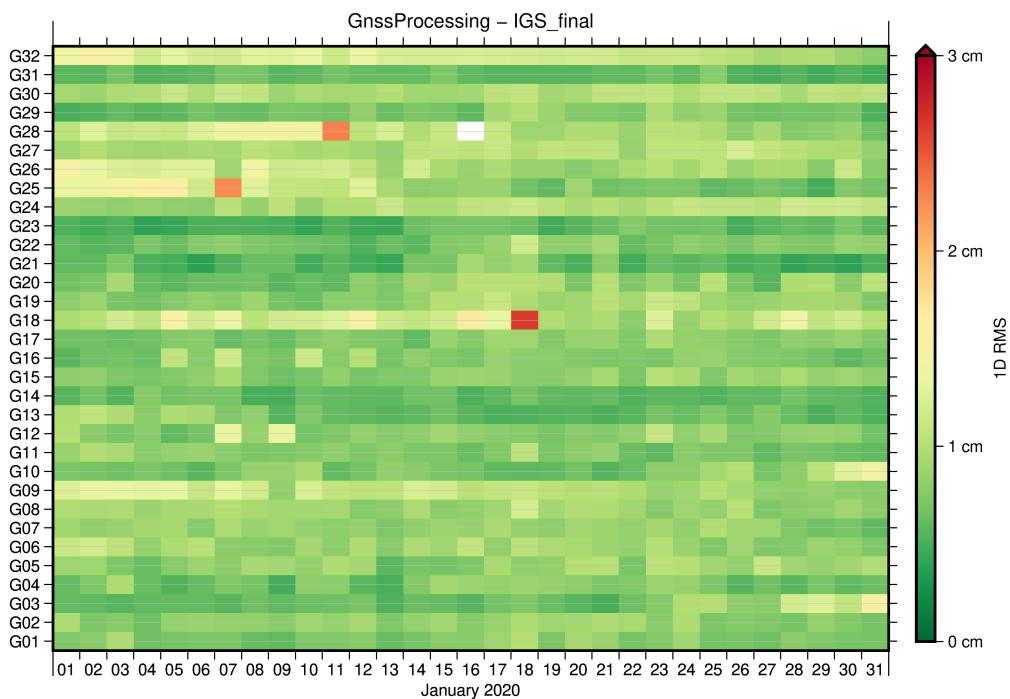


Figure 4.15: Comparison of estimated GPS orbits with IGS final solution.

Name	Type	Annotation
<b>outputfileRmsPlotGrid</b>	filename	columns: mjd, id, rms
<b>outputfileHelmertTimeSeries</b>	filename	columns: mjd, tx, ty, tz, scale, rx, ry, rz
<b>inputfileInstrument</b>	filename	one file per satellite/station
<b>inputfileInstrumentReference</b>	filename	one file per satellite/station, same order as above
<b>timeIntervals</b>	timeSeries	for <b>{loopTime}</b> variable in inputfile
<b>factor</b>	double	e.g. $\text{sqrt}(3)$ for 3D RMS
<b>estimateShift</b>	boolean	coordinate center every epoch
<b>estimateScale</b>	boolean	scale factor of position every epoch
<b>estimateRotation</b>	boolean	rotation every epoch

▶ huber	double	for robust least squares
▶ huberPower	double	for robust least squares
▶ huberMaxIteration	uint	(maximum) number of iterations for robust estimation

---

#### 4.7.7 Instrument2Scaleogram

This program computes the wavelet transform of a time series up to a **maxLevel**. The scalogram is written to a matrix which can be plotted by using a gridded layer in **PlotGraph**. Individual detail levels can be written to matrix files by setting **outputfileLevels**. The data column to be decomposed must be set by **selectDataField**.

The wavelet transform is implemented as a filter bank, so care should be taken when the input contains data gaps. Low/highpass wavelet filters are applied in forward and backward direction, input is padded symmetric. See **digitalFilter** for details.

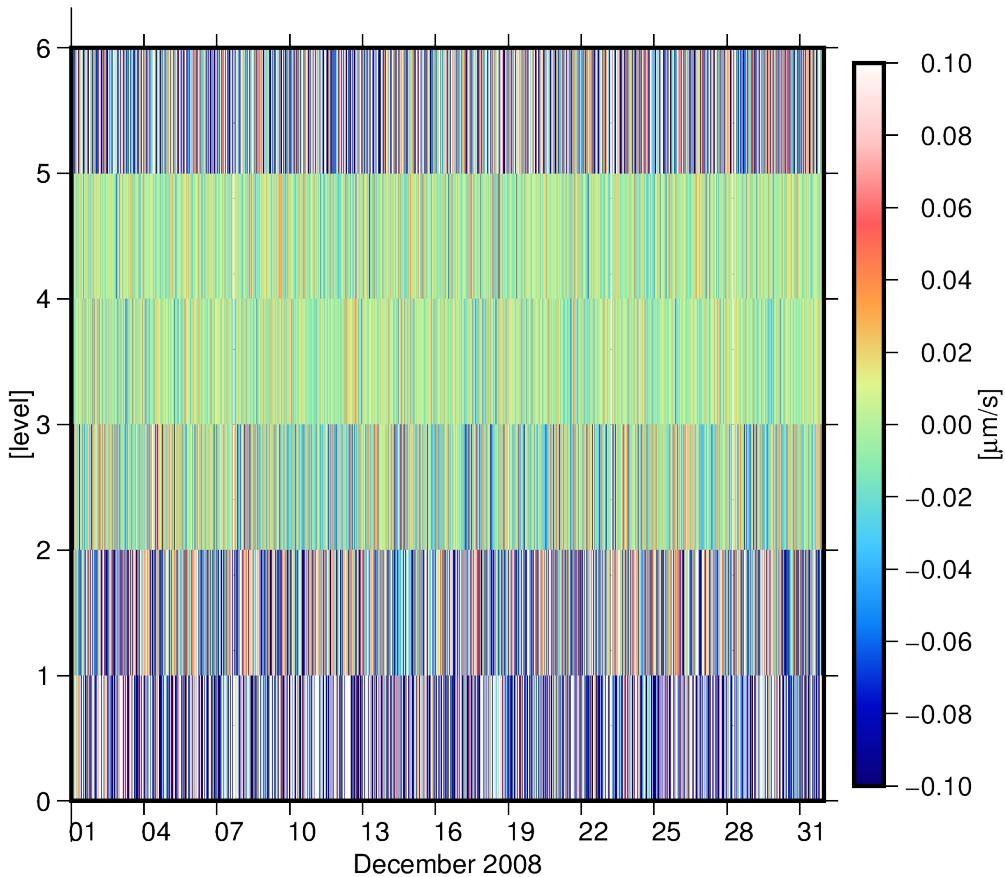


Figure 4.16: GRACE range-rate residuals of one month.

Name	Type	Annotation
outputfileScaleogram	filename	matrix columns: mjd, level, value
outputfileLevels	filename	use loopLevel as variable
inputfileInstrument	filename	
inputfileWavelet	filename	wavelet coefficients
selectDataField	uint	data column to transform
maxLevel	uint	maximum level of decomposition (default: full)

### 4.7.8 Instrument2SpectralCoherence

This program computes the spectral coherence between two **Instrument** files.

The (magnitude-squared) coherence is defined as

$$C_{xy}(f) = \frac{|P_{xy}(f)|^2}{P_{xx}(f)P_{yy}(f)} \quad (4.34)$$

and is a measure in the range [0, 1] for the similarity of the signals  $x$  and  $y$  in frequency domain.  $P_{xy}$  is the cross-spectral density between  $x$  and  $y$  and  $P_{xx}$ ,  $P_{yy}$  are auto-spectral densities. Auto- and cross-spectral densities are computed using Lomb's method (see **Instrument2PowerSpectralDensity** for details).

The resulting PSD is the average over all arcs. For regularly sampled time series, this method yields the same results as FFT based PSD estimates.

A regular frequency grid based on the longest arc and the median sampling is computed. The maximum number of epochs per arc is determined by

$$N = \frac{t_{\text{end}} - t_{\text{start}}}{\Delta t_{\text{median}}} + 1, \quad (4.35)$$

the Nyquist frequency is given by

$$f_{\text{nyq}} = \frac{1}{2\Delta t_{\text{median}}}. \quad (4.36)$$

If it is suspected that **inputfileInstrument** contains secular variations, the input should be detrended using **InstrumentDetrend**.

The **outputfileCoherence** contains a matrix with the frequency vector as first column, the coherence for each instrument channel is saved in the following columns.

Name	Type	Annotation
outputfileCoherence	filename	column 1: frequency, column 2-n coherence
inputfileInstrument	filename	
inputfileInstrumentReference	filename	

This program is [parallelized \(1.5\)](#).

### 4.7.9 Instrument2Spectrogram

This program applies the Short Time Fourier Transform (STFT) to selected data columns of **inputfileInstrument** and computes the spectrogram. The STFT is computed at centered **timeSeries** with an (possible overlapping) rectangular window with **windowLength** seconds. Data gaps are zero padded within the window.

The **outputfileSpectrogram** is a matrix with each row the time (MJD), the frequency [Hz], and the amplitudes [ $\text{unit}/\sqrt{\text{Hz}}$ ] for the selected data columns. It can be plotted with **PlotGraph**.

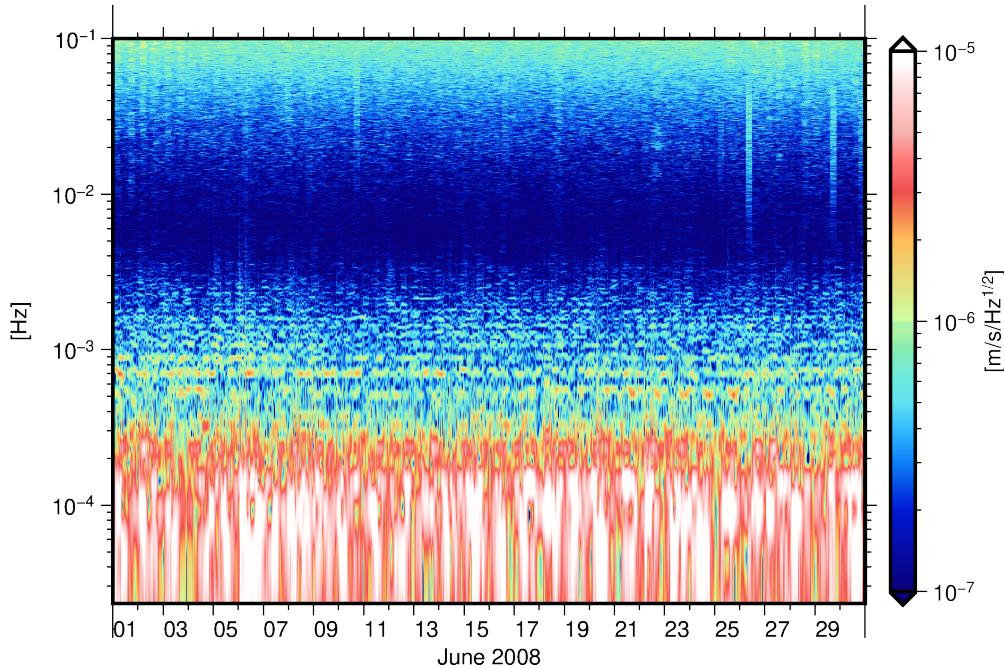


Figure 4.17: GRACE range-rate residuals of one month (window of 6 hours).

Name	Type	Annotation
<b>outputfileSpectrogram</b>	filename	mjd, freq, ampl0, ampl1, ...
<b>inputfileInstrument</b>	filename	
<b>timeSeries</b>	<b>timeSeries</b>	center of SFFT window
<b>windowLength</b>	double	[seconds]
<b>startDataFields</b>	uint	start
<b>countDataFields</b>	uint	number of data fields (default: all)

This program is parallelized (1.5).

#### 4.7.10 InstrumentAccelerometer2ThermosphericDensity

This program estimates neutral mass densities along the satellite trajectory based on [Accelerometer data](#). In order to determine the neutral mass density the accelerometer input should only reflect the accelerations due to drag (e.g. [miscAccelerations:atmosphericDrag](#)). Thus, influences from solar and Earth radiation pressure must be reduced beforehand.

Name	Type	Annotation
<code>outputfileDensity</code>	filename	MISCVOLUME (kg/m^3)
<code>satelliteModel</code>	filename	satellite macro model
<code>inputfileOrbit</code>	filename	
<code>inputfileStarCamera</code>	filename	
<code>inputfileAccelerometer</code>	filename	add non-gravitational forces in satellite reference frame
<code>thermosphere</code>	<a href="#">thermosphere</a>	used to compute temperature and wind
<code>considerTemperature</code>	boolean	compute drag and lift, otherwise simple drag coefficient is used
<code>considerWind</code>	boolean	
<code>earthRotation</code>	<a href="#">earthRotation</a>	
<code>ephemerides</code>	<a href="#">ephemerides</a>	

This program is [parallelized \(1.5\)](#).

#### 4.7.11 InstrumentAccelerometerApplyEstimatedParameters

This program evaluates estimated satellite parameters and writes the result to an accelerometer file.

Name	Type	Annotation
outputfileAccelerometer	filename	
inputfileSatelliteModel	filename	satellite macro model
inputfileOrbit	filename	
inputfileStarCamera	filename	
inputfileAccelerometer	filename	add non-gravitational forces in satellite reference frame
earthRotation	earthRotation	
ephemerides	ephemerides	may be needed by parametrizationAcceleration
parametrizationAcceleration	parametrizationAcceleration	orbit force parameters
inputfileParameter	filename	estimated orbit force parameters
indexStart	int	position in the solution vector
rightSide	int	if solution contains several right hand sides, select one
factor	double	the result is multiplied by this factor

This program is parallelized (1.5).

### 4.7.12 InstrumentAccelerometerEstimateBiasScale

This program calibrates **inputfileAccelerometer** with respect to simulated accelerometer data, see **SimulateAccelerometer**. The parameters **outputfileSolution** of **parametrizationAcceleration** are estimated and the effect is reduced to calibrate the **accelerometer** data.

If **inputfileThruster** is given, the corresponding epochs (within **marginThruster**) are not used for the parameter estimation, but the accelerometer epochs are still calibrated afterwards. An arbitrary instrument file is allowed here.

The **inputfileOrbit**, **inputfileStarCamera**, **earthRotation**, **ephemerides**, and **satelliteModel** are only needed for some special parametrizations.

Name	Type	Annotation
<b>outputfileAccelerometer</b>	filename	
<b>outputfileSolution</b>	filename	
<b>inputfileAccelerometer</b>	filename	
<b>inputfileAccelerometerSim</b>	filename	
<b>inputfileThruster</b>	filename	remove thruster events
<b>marginThruster</b>	double	margin size (on both sides) [seconds]
<b>inputfileOrbit</b>	filename	
<b>inputfileStarCamera</b>	filename	
<b>earthRotation</b>	<b>earthRotation</b>	
<b>ephemerides</b>	<b>ephemerides</b>	may be needed by parametrizationAcceleration
<b>inputfileSatelliteModel</b>	filename	satellite macro model
<b>parametrizationAcceleration</b>	<b>parametrizationAcceleration</b>	

This program is [parallelized \(1.5\)](#).

### 4.7.13 InstrumentAccelerometerEstimateParameters

This program estimates calibration parameters for acceleration data given an optional reference acceleration. Specifically, the program solves the equation

$$\mathbf{a} - \mathbf{a}_{\text{ref}} = \mathbf{f}(\mathbf{x}) + \mathbf{e} \quad (4.37)$$

for the unknown parameters  $\mathbf{x}$ , where  $\mathbf{a}$  is given in **inputfileAccelerometer** and  $\mathbf{a}_{\text{ref}}$  is given in **inputfileAccelerometerReference**. The parametrization of  $\mathbf{x}$  can be set via **parametrizationAcceleration**. Optionally, the empirical covariance functions for the accelerations  $\mathbf{a}$  can be estimated by enabling **estimateCovarianceFunctions**.

The estimated parameters are written to the file **outputfileSolution** and can be used by **InstrumentAccelerometerApplyEstimatedParameters** to calibrate accelerometer measurements.

Name	Type	Annotation
outputfileSolution	filename	values for estimated parameters
outputfileParameterNames	filename	names of the estimated parameters
estimateArcSigmas	sequence	
outputfileArcSigmas	filename	accuracies of each arc
estimateEpochSigmas	sequence	
outputfileEpochSigmas	filename	estimated epoch-wise sigmas
estimateCovarianceFunctions	sequence	
outputfileCovarianceFunction	filename	covariance functions for x, y, z direction
inputfileAccelerometer	filename	
inputfileAccelerometerReference	filename	if not given, reference acceleration is assumed zero
inputfileOrbit	filename	may be needed by parametrizationAcceleration
inputfileStarCamera	filename	may be needed by parametrizationAcceleration
inputfileSatelliteModel	filename	satellite macro model (may be needed by parametrizationAcceleration)
earthRotation	earthRotation	may be needed by parametrizationAcceleration
ephemerides	ephemerides	may be needed by parametrizationAcceleration
parametrizationAcceleration	parametrizationAcceleration	
sigmaX	double	apriori accuracy in x-axis
sigmaY	double	apriori accuracy in y-axis
sigmaZ	double	apriori accuracy in z-axis
iterationCount	uint	iteration count for determining the covariance function

This program is parallelized (1.5).

#### 4.7.14 InstrumentApplyTimeOffset

This program applies a **inputfileTimeOffset** (MISCVALUE) to an **inputfileInstrument**. The time offsets in seconds are multiplied with a **factor**. The instrument files must be synchronized (see **InstrumentSynchronize**).

Name	Type	Annotation
outfileInstrument	filename	
infileInstrument	filename	
infileTimeOffset	filename	MISCVALUE with time offset in seconds
factor	double	applied to time offset

### 4.7.15 InstrumentArcCalculate

This program manipulates the data columns every arc of an **Instrument** file similar to **FunctionsCalculate**, see there for more details. If several **inputfileInstruments** are given the data columns are copied side by side. For this the instrument files must be synchronized (see **InstrumentSynchronize**). For the data columns the standard data variables are available, see **dataVariables (1.2.3)**. For the time column (MJD) a variable epoch (together with epochmean, epochmin, ...) is defined additionally.

The content of **outfileInstrument** is controlled by **outColumn**. The number of **outColumn** must agree with the selected **outType**. The algorithm to compute the output is as follows: The expressions in **outColumn** are evaluated once for each epoch of the input. The variables **data0**, **data1**, ... are replaced by the according values from the input columns before. If no **outColumn** are specified all input columns are used instead directly. The **instrument type** can be specified with **outType** and must be agree with the number of columns.

An extra **statistics** file can be generated with one mid epoch per arc. For the computation of the **outColumn** values all **dataVariables (1.2.3)** are available (e.g. **epochmin**, **data0mean**, **data1std**, ...) inclusively the **constants** and estimated **parameters** but without the **data0**, **data1**, ... itself. The variables and the numbering of the columns refers to the **outfileInstrument**.

See also **FunctionsCalculate**, **MatrixCalculate**.

Name	Type	Annotation
<b>outfileInstrument</b>	filename	
<b>inputfileInstrument</b>	filename	data columns are appended to the right
<b>constant</b>	expression	define a constant by name=value
<b>parameter</b>	expression	define a parameter by name[=value]
<b>leastSquares</b>	expression	try to minimize the expression by adjustment of the parameters
<b>removalCriteria</b>	expression	row is removed if one criterion evaluates true.
<b>outType</b>	<b>instrumentType</b>	
<b>outColumn</b>	expression	expression of output columns, extra 'epoch' variable
<b>statistics</b>	sequence	
<b>outfileInstrument</b>	filename	instrument file with mid epoch per arc, data columns are user defined
<b>outColumn</b>	expression	expression to compute statistics columns, data* are from outColumn

This program is parallelized (1.5).

### 4.7.16 InstrumentArcCrossStatistics

Computes statistics of selected data columns between two **Instrument** files arc wise. The **outputfileStatisticsTimeSeries** contains for every arc one (mid) epoch with statistics column(s). Possible statistics are

- Correlation

$$\rho = \frac{\sum_i x_i y_i}{\sqrt{(\sum_i x_i^2)(\sum_i y_i^2)}}, \quad (4.38)$$

- Error RMS

$$rms = \sqrt{\frac{1}{N} \sum_i (x_i - y_i)^2}, \quad (4.39)$$

- Nash-Sutcliffe coefficient (NSC)

$$nsc = 1 - \frac{\sum_i (x_i - y_i)^2}{\sum_i (y_i - \bar{y})^2}. \quad (4.40)$$

With **removeArcMean** the mean of each data column of each arc is reduced before.

With **perColumn** separate statistics for each selected data column are computed, otherwise an overall value is computed.

See also **InstrumentArcStatistics**, **InstrumentStatisticsTimeSeries**.

Name	Type	Annotation
<b>outputfileStatisticsTimeSeries</b>	filename	statistics column(s) per arc, MISCVALUES
<b>inputfileInstrument</b>	filename	
<b>inputfileInstrumentReference</b>	filename	
<b>statistics</b>	choice	
<b>correlation</b>		rms of differences
<b>errorRMS</b>		with respect to reference field
<b>nashSutcliffe</b>		
<b>removeArcMean</b>	boolean	
<b>startDataFields</b>	uint	start
<b>countDataFields</b>	uint	number of data fields (default: all)
<b>perColumn</b>	boolean	compute statistic per column

This program is [parallelized \(1.5\)](#).

#### 4.7.17 InstrumentArcStatistics

Computes statistics of selected data columns of **inputfileInstrument** arc wise. The **outputfileStatisticsTimeSeries** contains for every arc one (mid) epoch with statistics column(s). Possible statistics are root mean square, standard deviation, mean, median, min, and max.

With **perColumn** separate statistics for each selected data column are computed, otherwise an overall value is computed.

See also **InstrumentArcCrossStatistics**, **InstrumentStatisticsTimeSeries**.

Name	Type	Annotation
outputfileStatisticsTimeSeries	filename	columns: mjd, statistics column(s) per instrument file
inputfileInstrument	filename	
statistics	choice	
rootMeanSquare		
standardDeviation		
mean		
median		
min		
max		
epochCount		
startDataFields	uint	start
countDataFields	uint	number of data fields (default: all)
perColumn	boolean	compute statistic per column
ignoreNan	boolean	ignore NaN values in input

This program is parallelized (1.5).

### 4.7.18 InstrumentConcatenate

This program concatenate the arcs from several **Instrument files** and write it to a new **file**. Input files must be of the same type. The arcs are merged to one arc even though there is a gap inbetween. To split the data into arcs use **InstrumentSynchronize**. Three options are available: **sort**, **removeDuplicates** and **checkForNaNs**. If **sort** is enabled, the program reads all files, no matter if they are sorted correctly in time, and then sorts the epochs. If **removeDuplicates** is enabled, the program checks the whole data set for epochs that are contained twice. And if **checkForNaNs** is enabled the data set is checked for invalid epochs containing NaNs.

Name	Type	Annotation
outfile	filename	
infile	filename	
sort	boolean	sort epochs with increasing time
removeDuplicates	choice	remove duplicate epochs
keepFirst	sequence	keep first epoch with the same time stamp, remove all others
margin	double	margin for identical times [seconds]
keepLast	sequence	keep last epoch with the same time stamp, remove all others
margin	double	margin for identical times [seconds]
checkForNaNs	boolean	remove epochs with NaN values in one of the data fields

#### 4.7.19 InstrumentCovarianceCheck

This program checks **inputfileCovariance3d** 3x3 covariance matrices if they are invertible or not and removes the invalid epochs.

Name	Type	Annotation
outputfileCovariance3d	filename	
inputfileCovariance3d	filename	

This program is parallelized (1.5).

### 4.7.20 InstrumentDetrend

Reduces **parametrizationTemporal** (e.g. const, trend, polynomial) per arc from selected data columns of **inputfileInstrument** using a robust **robust least squares adjustment** (2.1).

The **outputfileTimeSeriesArcParameters** contains for every arc one (mid) epoch with the estimated parameters. The order is: first all data (`data0`, `data1`, ...) of first temporal parameter, followed by all data of the second temporal parameter and so on.

Name	Type	Annotation
outputfileInstrument	filename	detrended instrument time series
outputfileTimeSeriesArcParameters	filename	time series of estimated parameters per arc
inputfileInstrument	filename	
parametrizationTemporal	parametrizationTemporal	per arc, data is reduced by temporal representation
startDataFields	uint	start
countDataFields	uint	number of data fields (default: all after start)
huber	double	for robust least squares
huberPower	double	for robust least squares
huberMaxIteration	uint	(maximum) number of iterations for robust estimation

This program is [parallelized](#) (1.5).

#### 4.7.21 InstrumentEarthRotation

Precompute Earth rotation matrix from celestial to terrestrial frame and save as  [StarCamera file](#).

Name	Type	Annotation
 <code>outputfileStarCamera</code>	<code>filename</code>	rotation from CRF to TRF
 <code>earthRotation</code>	<code>earthRotation</code>	
 <code>timeSeries</code>	<code>timeSeries</code>	

### 4.7.22 InstrumentEstimateEmpiricalCovariance

This program estimates the empirical auto- and cross-covariance of selected data columns per arc of **inputfileInstrument**. The maximum computed lag is determined by the number of **outputfileCovarianceMatrix** specified (for a single output file only the auto-covariance is determined, for two output files auto- and cross-covariance is computed and so on).

Stationarity is assumed for the input time series, which means the temporal covariance matrix has Toeplitz structure.

$$\begin{bmatrix} \Sigma & \Sigma_{\Delta_1} & \Sigma_{\Delta_2} & \Sigma_{\Delta_3} & \Sigma_{\Delta_4} \\ & \Sigma & \Sigma_{\Delta_1} & \Sigma_{\Delta_2} & \Sigma_{\Delta_3} \\ & & \Sigma & \Sigma_{\Delta_1} & \Sigma_{\Delta_2} \\ & & & \Sigma & \Sigma_{\Delta_1} \\ & & & & \Sigma \end{bmatrix} \quad (4.41)$$

The matrix for lag  $h$  describes the covariance between  $x_{t-h}$  and  $x_t$ , i.e.  $\Sigma(t-h, t)$ .

To get a reliable estimate, **InstrumentDetrend** should be called first.

Name	Type	Annotation
<b>outputfileCovarianceMatrix</b>	filename	
<b>inputfileInstrument</b>	filename	
<b>startDataFields</b>	uint	start
<b>countDataFields</b>	uint	number of data fields (default: all after start)

This program is [parallelized \(1.5\)](#).

#### 4.7.23 InstrumentEstimateHelmertTransformation

This program estimates a 3D Helmert transformation between two networks (frame realizations, e.g. GNSS satellite or station network). Each separate **data** represents a satellite/station/... (e.g. 32 GPS satellites). The instrument data (x,y,z position) considered can be set with **startData**. The Helmert parameters are set up according to **parametrizationTemporal** for each **timeIntervals** and are estimated using a robust least squares adjustment (2.1).

Name	Type	Annotation
outputfileHelmertTimeSeries	filename	columns: mjd, Tx,Ty,Tz,s,Rx,Ry,Rz according to temporal parametrization e.g. satellite, station
data	sequence	
outputfileInstrument	filename	transformed positions as instrument type Vector3d
outputfileInstrumentDiff	filename	position difference as instrument type Vector3d
inputfileInstrument	filename	
inputfileInstrumentReference	filename	
startDataFields	uint	
timeIntervals	timeSeries	start index of position (x,y,z) columns parameters are estimated per interval
parametrizationTemporal	parametrizationTemporal	temporal parametrization
estimateShift	boolean	coordinate center
estimateScale	boolean	scale factor of position
estimateRotation	boolean	rotation
huber	double	for robust least squares
huberPower	double	for robust least squares
huberMaxIteration	uint	(maximum) number of iterations for robust estimation

#### 4.7.24 InstrumentFilter

This program filter selected data columns of **inputfileInstrument** with **digitalFilter** arc wise.

Name	Type	Annotation
outputfileInstrument	filename	
inputfileInstrument	filename	
digitalFilter	<a href="#">digitalFilter</a>	
startDataFields	uint	start
countDataFields	uint	number of data fields (default: all after start)

This program is [parallelized \(1.5\)](#).

#### 4.7.25 InstrumentInsertNAN

This program inserts NAN epochs into **inputfileInstrument** files, either at specific **times** or where gaps in the instrument are detected.

Name	Type	Annotation
outfileInstrument	filename	
infileInstrument	filename	
times	timeSeries	Insert NAN at specific times.
atGaps	boolean	Insert NAN where epochs are more than 1.5 times the median sampling apart.
atArcEnds	boolean	Insert one epoch with data NAN at arc ends

This program is parallelized (1.5).

### 4.7.26 InstrumentMultiplyAdd

This program multiply **Instrument** data with a factor and add them together. Afterwards the mean of each arc and data column can be removed with **removeArcMean**. The instrument files must be synchronized (**InstrumentSynchronize**).

See also **InstrumentArcCalculate**.

Name	Type	Annotation
outputfileInstrument	filename	
instrument	sequence	
inputfileInstrument	filename	
factor	double	
removeArcMean	boolean	remove mean value of each arc

This program is **parallelized** (1.5).

#### 4.7.27 InstrumentReduceSampling

This program reduce the sampling of a instrument file. Only epochs with a time stamp with a division by **sampling** without remainder are kept (inside **margin**).

Name	Type	Annotation
outfileInstrument	filename	
infileInstrument	filename	
sampling	double	new sampling in seconds
margin	double	margin around the new sampling in seconds
relative2FirstEpoch	boolean	compute sampling relative to time of first epoch

### 4.7.28 InstrumentRemoveEpochsByCriteria

This program removes epochs from **inputfileInstrument** by evaluating a set of **removalCriteria** expressions. For the data columns the standard data variables are available, see [dataVariables \(1.2.3\)](#).

The instrument data can be reduced by data from **inputfileInstrumentReference** prior to evaluation of the expressions.

To reduce the data by its median, use an expression like `data1-data1mean`. To remove epochs that deviate by more than 3 sigma use `abs(data1)>3*data1std` or `abs(data0-data0median)>3*1.4826*data0mad`.

All arcs in the input instrument file are concatenated, meaning expressions like `data1mean` refer to the complete dataset. The removed epochs can be saved in a separate **outfileInstrumentRemovedEpochs**.

Name	Type	Annotation
outfileInstrument	filename	all data is stored in one arc
outfileInstrumentRemovedEpochs	filename	all data is stored in one arc
inputfileInstrument	filename	arcs are concatenated for processing
inputfileInstrumentReference	filename	if given, the reference data is reduced prior to the expressions being evaluated
removalCriteria	expression	epochs are removed if one criterion evaluates true. data0 is the first data field.
margin	double	remove data around identified epochs (on both sides) [seconds]

#### 4.7.29 InstrumentRemoveEpochsByTimes

This program compares an **Instrument** file with a **time series**. Epochs contained within the time series (including a defined margin) are removed from the instrument file. The margin is added on both sides of the epochs. The arcs of the instrument file are concatenated to one arc. The removed epochs can be saved in a separate instrument file.

Name	Type	Annotation
outfileInstrument	filename	all epochs are concatenated in one arc
outfileInstrumentRemovedEpochs	filename	all epochs are concatenated in one arc
infileInstrument	filename	
timePoints	timeSeries	
margin	double	margin size (on both sides) [seconds]

### 4.7.30 InstrumentRemoveEpochsThruster

This program remove epochs from an [Instrument file](#). The epochs are defined by a [Thruster file](#) plus a defined margin before and after the thruster firings. The arcs of the instrument file are concatenated to one arc. The removed epochs can be saved in a separate instrument file.

Name	Type	Annotation
▶ outfileInstrument	filename	all epochs are concatenated in one arc
▶ outfileInstrumentRemovedEpochs	filename	all epochs are concatenated in one arc
▶ inputfileInstrument	filename	
▶ inputfileThruster	filename	THRUSTER
▶ marginBefore	double	margin before start of firing [seconds]
▶ marginAfter	double	margin after end of firing [seconds]

#### 4.7.31 InstrumentResample

This program resamples **Instrument** data to a given **timeSeries** using a resampling **method**.

This program can also be used to reduce the sampling of an instrument file, but a better way to reduce the sampling of noisy data with regular sampling is to use a low pass filter first with **InstrumentFilter** and then thin out the data with **InstrumentReduceSampling**.

Name	Type	Annotation
outfileInstrument	filename	
infileInstrument	filename	
method	interpolatorTimeSeries	resampling method
timeSeries	timeSeries	resampled points in time

### 4.7.32 InstrumentRotate

This program rotates **Instrument data** into a new reference frame (using **inputfileStarCamera**). The rotation is usually done from satellite frame into inertial frame.

To apply Earth rotation to orbits use **Orbit2EarthFixedOrbit**. For other instrument data use **InstrumentEarthRotation** before.

Name	Type	Annotation
outputfileInstrument	filename	
inputfileInstrument	filename	
inputfileStarCamera	filename	
inverseRotate	boolean	

This program is [parallelized \(1.5\)](#).

#### 4.7.33 InstrumentSetType

Convert **Instrument** data into instrument data with new **Type**. The selected number of data columns must agree with the **Type**.

Name	Type	Annotation
outfileInstrument	filename	
infileInstrument	filename	
type	instrumentType	
startDataFields	uint	start
countDataFields	uint	number of data fields (default: all after start)

#### 4.7.34 InstrumentStarCamera2AccAngularRate

This program derive from a time series of quaternions a series of angular rates and angular accelerations. The derivatives are computed by a polynomial interpolation with  **interpolationDegree** of the quaternions.

Name	Type	Annotation
 <b>outputfileAngularRate</b>	filename	[rad/s], VECTOR3D
 <b>outputfileAngularAcc</b>	filename	[rad/s**2], VECTOR3D
 <b>inputfileStarCamera</b>	filename	
 <b>interpolationDegree</b>	uint	derivation by polynomial interpolation of degree n

#### 4.7.35 InstrumentStarCamera2RollPitchYaw

Compute roll, pitch, yaw angles from **inputfileStarCamera** data. Optional the angles are computed relative to a **inputfileStarCameraReference**.

See also **SimulateStarCamera**.

Name	Type	Annotation
outputfileInstrument	filename	roll, pitch, yaw [rad], VECTOR3D
inputfileStarCamera	filename	
inputfileStarCameraReference	filename	nominal orientation

This program is parallelized (1.5).

#### 4.7.36 InstrumentStarCamera2RotaryMatrix

Write **inputfileStarCamera** rotations as **outputfileInstrument** rotary matrices (for each epoch  $xx, xy, xz, yx, yy, yz, zx, zy, zz$ ).

Name	Type	Annotation
outputfileInstrument	filename	xx, xy, xz, yx, yy, yz, zx, zy, zz (MISCVALUES)
inputfileStarCamera	filename	

This program is [parallelized \(1.5\)](#).

#### 4.7.37 InstrumentStarCameraMultiply

This program applies several rotations given by ▶ **inputfileStarCamera**. The resulting rotation is written as ▶ **outputfileStarCamera**. All instrument files must be synchronized (▶ **InstrumentSynchronize**).

Name	Type	Annotation
▶ <b>outputfileStarCamera</b>	filename	
▶ <b>instrument</b>	sequence	
└▶ <b>inputfileStarCamera</b>	filename	
└▶ <b>inverse</b>	boolean	

### 4.7.38 InstrumentStatisticsTimeSeries

This program computes a time series of statistics for one or more instrument files. Possible statistics are root mean square, standard deviation, mean, median, min, and max. The columns of the output time series are defined either as one per **inputfileInstrument** or, if **perColumn** is true, statistics are computed per column for each file. Providing e.g. 32 orbit files of GPS satellites results in a time series matrix with columns: mjd, statisticsG01, statisticsG02, ..., statisticsG32. If **intervals** are provided, the input data is split into these intervals and one statistic is computed per interval. Otherwise, overall statistics are computed. The instrument data considered for computation of the component-wise statistics can be set with **startDataFields** and **countDataFields**. The **factor** can be set to e.g.  $\sqrt{3}$  to get 3D instead of 1D RMS values.

See also **InstrumentArcStatistics**, **InstrumentArcCrossStatistics**.

Name	Type	Annotation
<b>outputfileStatisticsTimeSeries</b>	filename	columns: mjd, statistics column(s) per instrument file
<b>inputfileInstrument</b>	filename	
<b>statistics</b>	choice	
<b>rootMeanSquare</b>		
<b>standardDeviation</b>		
<b>mean</b>		
<b>median</b>		
<b>sum</b>		
<b>min</b>		
<b>max</b>		
<b>epochCount</b>		
<b>startDataFields</b>	uint	start
<b>countDataFields</b>	uint	number of data fields (default: all)
<b>perColumn</b>	boolean	compute statistic per column
<b>ignoreNaN</b>	boolean	ignore NaN values in statistic computation
<b>intervals</b>	timeSeries	intervals for statistics computation (one statistic per interval)
<b>factor</b>	double	e.g. $\sqrt{3}$ for 3D RMS

### 4.7.39 InstrumentSynchronize

This program reads several **Instrument** files and synchronize the data. Every epoch with some missing data will be deleted so the remaining epochs have data from every instrument.

In a second step the epochs are divided into arcs with maximal epochs (or **maxArcLen**) without having a gap inside an arc. A Gap is defined by a time step with at least **minGap** seconds between consecutive epochs or if not set the 1.5 of the median sampling. Arc with an epoch count less than **minArcLen** will be rejected.

A specific region can be selected with **border**. In this case one of the instrument data must an orbit.

If **timeIntervals** is given the data are also divided into time bins. The assignment of arcs to the bins can be saved in **outputfileArcList**. This file can be used for the variational equation approach or **KalmanBuildNormals**.

Instrument files from **irregularData** are not synchronized but divided into the same number of arcs within the same time intervals. Data outside the defined arcs will be deleted.

Name	Type	Annotation
data	sequence	
outputfileInstrument	filename	
inputfileInstrument	filename	
margin	double	margin for identical times [seconds]
minGap	double	minimal time to define a gap and to begin a new arc, 0: no dividing [seconds], if not set 1.5*median sampling is used
minArcLength	uint	minimal number of epochs of an arc
maxArcLength	uint	maximal number of epochs of an arc
arcType	choice	all arcs or only ascending or descending arcs are selected
ascending		
descending		
border	border	only data in a specific region is selected
timeIntervals	timeSeries	divide data into time bins
outputfileArcList	filename	arc and time bin mapping
irregularData	sequence	instrument files with irregular sampling
outputfileInstrument	filename	
inputfileInstrument	filename	
minArcLength	uint	minimal number of epochs in an arc

#### 4.7.40 InstrumentWaveletDecomposition

This program performs a multilevel one-dimensional wavelet analysis on one **selectDataField** data column of **infileInstrument**. The **outfileInstrument** contains the decomposed levels in time domain  $a_J, d_J, \dots, d_1$

Name	Type	Annotation
outfileInstrument	filename	MISCVALUES, decomposed levels in time domain $a_J, d_J, \dots, d_1$
infileInstrument	filename	
selectDataField	uint	select a data column for decomposition
infileWavelet	filename	wavelet coefficients
level	uint	level of decomposition

#### 4.7.41 LocalLevelFrame2StarCamera

Compute rotation ( `StarCamera file`) from local level frame (ellipsoidal north, east, down) to TRF for positions given in `inputfileInstrument` (first 3 data columns).

Name	Type	Annotation
<code>outputfileStarCamera</code>	filename	rotation matrix from local level frame (ellipsoidal north, east, down) to TRF
<code>inputfileInstrument</code>	filename	origin of local level frame
<code>constantOriginPerArc</code>	boolean	use constant origin for all epochs of an arc (median position)
<code>R</code>	double	reference radius for ellipsoidal coordinates
<code>inverseFlattening</code>	double	reference flattening for ellipsoidal coordinates, 0: spherical coordinates

This program is parallelized (1.5).

## 4.8 Programs: KalmanFilter

### 4.8.1 KalmanBuildNormals

This program sets up normal equations based on **observation** for short-term gravity field variations. It computes the normal equations based on the intervals  $i \in \{1, \dots, N\}$  given in the **arcList**. It sets up the least squares adjustment

$$\begin{bmatrix} l_1 \\ l_2 \\ \vdots \\ l_N \end{bmatrix} = \begin{bmatrix} \mathbf{A}_1 & & & \\ & \mathbf{A}_2 & & \\ & & \ddots & \\ & & & \mathbf{A}_N \end{bmatrix} \begin{bmatrix} \mathbf{x}^{(1)} \\ \mathbf{x}^{(2)} \\ \vdots \\ \mathbf{x}^{(N)} \end{bmatrix} + \begin{bmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \\ \vdots \\ \mathbf{e}_N \end{bmatrix}, \quad (4.42)$$

and subsequently computes the normal equations  $\mathbf{N}_i, \mathbf{n}_i$  for each interval. If **eliminateNonGravityParameters** is true, all non-gravity parameters are eliminated before the normals are written to **outputfileNormalEquation**. For each time interval in **arcList** a single **normal equation file** is written.

This program computes the input normals for **KalmanFilter** and **KalmanSmootherLeastSquares**.

Name	Type	Annotation
<b>outputfileNormalEquation</b>	filename	outputfile for normal equations
<b>observation</b>	<b>observation</b>	
<b>inputfileArcList</b>	filename	list to correspond points of time to arc numbers
<b>eliminateNonGravityParameters</b>	boolean	eliminate additional parameters from normals, 0: all parameter are saved

This program is **parallelized** (1.5).

## 4.8.2 KalmanFilter

The program computes time variable gravity fields using the Kalman filter approach of

Kurtenbach, E., Eicker, A., Mayer-Gürr, T., Holschneider, M., Hayn, M., Fuhrmann, M., and Kusche, J. (2012). Improved daily GRACE gravity field solutions using a Kalman smoother. *Journal of Geodynamics*, 59–60, 39–48. <https://doi.org/10.1016/j.jog.2012.02.006>.

The updated state  $\mathbf{x}_t^+$  is determined by solving the least squares adjustment

$$\mathbf{l}_t = \mathbf{A}_t \mathbf{x}_t + \mathbf{e}_t \quad \mathbf{e}_t \sim \mathcal{N}(0, \mathbf{R}_t) \quad \mathbf{B} \mathbf{x}_{t-1}^+ = \mathbf{I} \mathbf{x}_t + \mathbf{v}_t \quad \mathbf{v} \sim \mathcal{N}(0, \mathbf{Q} + \mathbf{B} \mathbf{P}_{t-1}^+ \mathbf{B}^T). \quad (4.43)$$

In normal equation form this can be written as

$$\hat{\mathbf{x}}_t = \mathbf{x}_t^+ = (\mathbf{N}_t + \mathbf{P}_t^{-1})^{-1}(\mathbf{n}_t + \mathbf{P}_t^{-1} \mathbf{x}_t^-), \quad (4.44)$$

where  $\mathbf{x}_t^- = \mathbf{B} \mathbf{x}_{t-1}^+$  and  $\mathbf{P}_t^- = \mathbf{Q} + \mathbf{B} \mathbf{P}_{t-1}^+ \mathbf{B}^T$  are the predicted state and its covariance matrix.

The process dynamic  $\mathbf{B}, \mathbf{Q}$  is represented as an [autoregressive model \(2.3\)](#), and passed to the program through **inputfileAutoregressiveModel**. The sequence of normal equations  $\mathbf{N}_t, \mathbf{n}_t$  are given as list of **inputfileNormalEquations**, which can be generated using **loops**. In the same way, the **matrix files** for **outputfileUpdatedState** and **inputfileUpdatedStateCovariance** can also be specified using **loops**.

If no **inputfileInitialState** is set, a zero vector with appropriate dimensions is used. The **inputfileInitialStateCovarianceMatrix** however must be given.

See also **KalmanBuildNormals**, **KalmanSmoother**.

Name	Type	Annotation
☞ outputfileUpdatedState	filename	estimated state $\mathbf{x}+$ (nx1-matrix)
☞ outputfileUpdatedStateCovarianceMatrix	filename	estimated state's covariance matrix $\text{Cov}(\mathbf{x}+)$
☞ inputfileNormalEquations	filename	normal equations input file
☞ inputfileInitialState	filename	initial state $\mathbf{x}0$
☞ inputfileInitialStateCovarianceMatrix	filename	initial state's covariance matrix $\text{Cov}(\mathbf{x}0)$
☞ inputfileAutoregressiveModel	filename	file name of autoregressive model

### 4.8.3 KalmanSmoothen

Apply the Rauch-Tung-Striebel smoother to a gravity field time series computed by **KalmanFilter**. This is the implementation of the approach presented in

Kurtenbach, E., Eicker, A., Mayer-Gürr, T., Holschneider, M., Hayn, M., Fuhrmann, M., and Kusche, J. (2012). Improved daily GRACE gravity field solutions using a Kalman smoother. *Journal of Geodynamics*, 59–60, 39–48. <https://doi.org/10.1016/j.jog.2012.02.006>.

The result has zero phase and the squared magnitude response of **inputfileAutoregressiveModel** (see **autoregressiveModel** (2.3) for details). **inputfileUpdatedState** and **inputfileUpdatedStateCovariance** are the output of a **KalmanFilter** forward sweep. The matrix files for **outputfileUpdatedState**, **inputfileUpdatedState** and **inputfileUpdatedStateCovariance** can also be specified using **loops**.

See also **KalmanBuildNormals**, **KalmanFilter** and **KalmanSmoothenLeastSquares**.

Name	Type	Annotation
<b>outputfileState</b>	filename	estimated parameters (nx1-matrix)
<b>outputfileStateCovarianceMatrix</b>	filename	estimated parameters' covariance matrix
<b>inputfileUpdatedState</b>	filename	
<b>inputfileUpdatedStateCovariance- Matrix</b>	filename	
<b>inputfileAutoregressiveModel</b>	filename	file name of autoregressive model

#### 4.8.4 KalmanSmootherLeastSquares

This program estimates temporal gravity field variations with a constraint least squares adjustment. Prior information is introduced by means of a **autoregressiveModelSequence** which represent a stationary random process (see the [autoregressive model description \(2.3\)](#)) for details.

The output files for the estimated gravity field (**outputfileSolution**), the corresponding standard deviations (**outputfileSigmax**) and the full covariance matrix (**outputfileCovariance**) can be specified using **loops**. Similarly, the **infileNormalEquations** can also be specified using **loops**.

See also **KalmanBuildNormals**, **KalmanFilter** and **KalmanSmoother**

Name	Type	Annotation
☞ <b>outputfileSolution</b>	filename	file name of solution vector (use time tags)
☞ <b>outputfileSigmax</b>	filename	file name of sigma vector (use time tags)
☞ <b>outputfileCovariance</b>	filename	file name of full covariance matrix (use time tags)
☞ <b>infileNormalEquations</b>	filename	input normal equations (loopTime will be expanded)
☞ <b>autoregressiveModelSequence</b>	autoregressiveModelSequence	file containing AR model for spatiotemporal constraint

This program is parallelized (1.5).

## 4.9 Programs: Misc

### 4.9.1 DigitalFilter2FrequencyResponse

Compute amplitude-, phase-, group delay and frequency response of a **digitalFilter** cascade. The **outputfileResponse** is a matrix with following columns: freq [Hz], ampl, phase [rad], group delay [-], real, imag.

When **unwrapPhase** is set to true,  $2\pi$  jumps of the phase response are removed before writing the output to file.

The response of the filter cascade is given by the product of each individual frequency response:

$$H(f) = \prod_f H_j(f). \quad (4.45)$$

Amplitude and phase response are computed from the frequency response via

$$A(f) = |H(f)| \text{ and } \Phi(f) = \arctan \frac{\mathcal{I}(H(f))}{\mathcal{R}(H(f))}. \quad (4.46)$$

The group delay is computed by numerically differentiating the phase response

$$\tau_g(f_k) = \frac{1}{2} \left[ \frac{\Phi(f_k) - \Phi(f_{k-1})}{2\pi(f_k - f_{k-1})} + \frac{\Phi(f_{k+1}) - \Phi(f_k)}{2\pi(f_{k+1} - f_k)} \right] \approx \frac{d\Phi}{df} \frac{df}{d\omega}. \quad (4.47)$$

The frequency vector for a **length**  $N$  and a **sampling**  $\Delta t$  is given by

$$f_k = \frac{k}{N\Delta t}, \quad k \in \{0, \dots, \left\lfloor \frac{N+2}{2} \right\rfloor - 1\}. \quad (4.48)$$

See also **DigitalFilter2ImpulseResponse**.

Name	Type	Annotation
<b>outputfileResponse</b>	filename	columns: freq [Hz], ampl, phase [rad], group delay [-], real, imag
<b>digitalFilter</b>	<b>digitalFilter</b>	
<b>length</b>	uint	length of the data series in time domain
<b>sampling</b>	double	sampling to determine frequency [seconds]
<b>skipZeroFrequency</b>	boolean	omit zero frequency when writing to file
<b>unwrapPhase</b>	boolean	unwrap phase response

### 4.9.2 DigitalFilter2ImpulseResponse

Impulse response of a **digitalFilter** cascade. The impulse response is computed by filtering a sequence with **length** samples and a unit impulse at index **pulseLag**.

The **outputfileResponse** is a matrix with the time stamp (zero at **pulseLag**) in the first column and the impulse response  $h_k$  in the second column.

See also **DigitalFilter2FrequencyResponse**.

Name	Type	Annotation
<b>outputfileResponse</b>	filename	columns: time [seconds], response
<b>digitalFilter</b>	<b>digitalFilter</b>	
<b>length</b>	uint	length of the impulse response
<b>pulseLag</b>	uint	start of the pulse in the data series
<b>sampling</b>	double	[seconds]

### 4.9.3 EarthOrientationParameterTimeSeries

Computes a **timeSeries** (GPS time) of Earth Orientation Parameter (EOP). The **instrument file** (MIS-CVALUES) contains the elements at each epoch in the following order:

- $x_p$  [rad]
- $y_p$  [rad]
- $s_p$  [rad]
- $UT1 - UTC$  [seconds]
- length of day (LOD) [seconds]
- $X$  [rad]
- $Y$  [rad]
- $S$  [rad]

The values are in situ values with all corrections and models applied. The time series can be used to precompute Earth rotation with a low temporal resolution (e.g. 10 min) and reuse the file in **earthRotation:file** to interpolate the data to the needed epochs (e.g. to rotate orbit data). As some Earth rotation models are quite slow this can accelerate the computation.

Name	Type	Annotation
outputfileEOP	filename	each row: mjd(GPS), xp, yp, sp, dUT1, LOD, X, Y, S
earthRotation	earthRotation	
timeSeries	timeSeries	

This program is [parallelized \(1.5\)](#).

#### 4.9.4 EarthRotaryVectorTimeSeries

Computes a **»outputfileTimeSeries** of Earth's rotary axis and its temporal derivative at **»timeSeries** (GPS time). The **»instrument** file (MISCVALUES) contains the elements at each epoch in the following order:

- $\omega_x[\text{rad/s}]$
- $\omega_y[\text{rad/s}]$
- $\omega_z[\text{rad/s}]$
- $\dot{\omega}_x[\text{rad/s}^2]$
- $\dot{\omega}_y[\text{rad/s}^2]$
- $\dot{\omega}_z[\text{rad/s}^2]$ .

Name	Type	Annotation
<b>»outputfileTimeSeries</b>	filename	wx, wy, wz [rad], dwx, dwy, dwz [rad/s <sup>2</sup> ]
<b>»earthRotation</b>	<b>earthRotation</b>	
<b>»timeSeries</b>	<b>timeSeries</b>	
<b>»inTRF</b>	boolean	terrestrial reference frame, otherwise celestial

This program is parallelized (1.5).

#### 4.9.5 EclipseFactor2GriddedData

This program converts the output of a **eclipse** model on a given **grid**. The time for the evaluation can be specified in **time**. The values will be saved together with points expressed as ellipsoidal coordinates (longitude, latitude, height) based on a reference ellipsoid with parameters **R** and **inverseFlattening**.

Name	Type	Annotation
outputfileGriddedData	filename	eclipse factor
grid	grid	
eclipse	eclipse	
ephemerides	ephemerides	
earthRotation	earthRotation	
time	time	
R	double	reference radius for ellipsoidal coordinates on output
inverseFlattening	double	reference flattening for ellipsoidal coordinates on output, 0: spherical coordinates

This program is [parallelized \(1.5\)](#).

#### 4.9.6 FilterMatrixWindowedPotentialCoefficients

Create a spherical harmonic window matrix. The window matrix  $\mathbf{W}$  is generated in space domain through spherical harmonic synthesis and analysis matrices. The resulting linear operator can be written as

$$\mathbf{W} = \mathbf{K}\mathbf{A}\Omega\mathbf{S}\mathbf{K}^{-1}. \quad (4.49)$$

Here,  $\mathbf{K}$  is a diagonal matrix with the **kernel** coefficients on the main diagonal,  $\mathbf{S}$  is the spherical harmonic synthesis matrix,  $\Omega$  is defined by the values in **inputfileGriddedData** and the expression **value**,  $\mathbf{A}$  is the spherical harmonic analysis matrix. The resulting window matrix is written to a **matrix** file.

The spherical harmonic degree range, and coefficient numbering are defined by **minDegree**, **maxDegree**, and **numbering**.

Note that a proper window function  $\Omega$  should contain values in the range  $[0, 1]$ . The window function  $\Omega$  can feature a smooth transition between 0 and 1 to avoid ringing effects.

Name	Type	Annotation
<b>outputfileWindowMatrix</b>	filename	
<b>inputfileGriddedData</b>	filename	gridded data which defines the window function in space domain
<b>value</b>	expression	expression to compute the window function (input columns are named data0, data1, ...)
<b>kernel</b>	kernel	kernel for windowing
<b>minDegree</b>	uint	
<b>maxDegree</b>	uint	
<b>GM</b>	double	Geocentric gravitational constant
<b>R</b>	double	reference radius
<b>numbering</b>	sphericalHarmonicsNumbering	numbering scheme for solution vector

This program is parallelized (1.5).

### 4.9.7 FunctionsCalculate

This program manipulates **matrix files** with data in columns. If several **inputfiles** are given the data columns are copied side by side. All **inputfiles** must contain the same number of rows. The columns are enumerated by `data0, data1, ...`.

The content of **outfile** is controlled by **outColumn**. The algorithm to compute the output is as follows: The expressions in **outColumn** are evaluated once for each row of the input. The variables `data0, data1, ...` are replaced by the according values from the input columns before. Additional variables are available, e.g. `index, dataOrms`, see [dataVariables \(1.2.3\)](#). If no **outColumn** are specified all input columns are used instead directly.

For a simplified handling **constants** can be defined by `name=value`, e.g. `annual=365.25`. It is also possible to estimate **parameters** in a least squares adjustment. The **leastSquares** serves as template for observation equations for every row. The expression **leastSquares** is evaluated for each row in the **inputfile**. The variables `data0, data1, ...` are replaced by the according values from the input columns before. In the next step the parameters are estimated in order to minimize the expressions in **leastSquares** in the sense of least squares.

Afterwards complete rows are removed if one of the **removalCriteria** expressions for this row evaluates true (not zero).

An extra **statistics** file can be generated with one row of data. For the computation of the **outColumn** values all [dataVariables \(1.2.3\)](#) are available (e.g. `data3mean, data4std`) inclusively the **constants** and estimated **parameters** but without the `data0, data1, ...` itself. The variables and the numbering of the columns refers to the **outfile**.

First example: To calculate the mean of two values at each row set **outColumn** to `0.5*(data1+data0)`.

Second example: An input file contain a column with times and a column with values. To remove a trend from the values define the **parameters trend and bias**. The observation equation in **leastSquares** is `data1 - (trend*data0+bias)`. For output you can define the following columns for example:

- **outColumn**=`data0`: points in time.
- **outColumn**=`data1`: the values itself.
- **outColumn**=`trend*data0+bias`: the linear fit.
- **outColumn**=`data1-trend*data0-bias`: the residuals.

The extra statistics file could contain in this case:

- **outColumn**=`data0max-data0min`: time span.
- **outColumn**=`bias`: estimated parameter.
- **outColumn**=`trend`: estimated parameter.
- **outColumn**=`data3rms`: root mean square of the residuals.

See also **InstrumentArcCalculate**, **GriddedDataCalculate**, **MatrixCalculate**.

Name	Type	Annotation
------	------	------------

outputfile	filename	
inputfile	filename	
constant	expression	define a constant by name=value
parameter	expression	define a parameter by name[=value]
leastSquares	expression	try to minimize the expression by adjustment of the parameters
removalCriteria	expression	row is removed if one criterion evaluates true.
outColumn	expression	expression to compute output columns (input columns are named data0, data1, ...)
statistics	sequence	
outputfile	filename	matrix with one row, columns are user defined
outColumn	expression	expression to compute statistics columns, data* are the outputColumns

### 4.9.8 Grs2PotentialCoefficients

This program creates potential coefficients from the defining constants of a Geodetic Reference System (GRS). The potential coefficients excludes the centrifugal part. The form of the reference ellipsoid is either determined by the dynamical form factor **J2**, or the geometric **inverseFlattening**. One of those form parameters must be specified.

The default values create the GRS80.

Name	Type	Annotation
<b>outputfilePotentialCoefficients</b>	filename	
<b>maxDegree</b>	uint	
<b>GM</b>	double	Geocentric gravitational constant
<b>R</b>	double	reference radius
<b>omega</b>	double	Angular velocity of rotation
<b>J2</b>	double	Dynamical form factor
<b>inverseFlattening</b>	double	Geometric inverse flattening of reference ellipsoid (0: sphere, ignored when J2 is set)

### 4.9.9 Kaula2SigmaPotentialCoefficients

Create signal standard deviations of potential coefficients according Kaula's rule of thumb

$$\sigma_n = \frac{f}{n^p}, \quad (4.50)$$

with the degree  $n$ , the **factor**  $f$ , and the **power**  $p$ .

The standard deviations are written as formal errors of **outputfilePotentialCoefficients**.

Name	Type	Annotation
outputfilePotentialCoefficients	filename	
minDegree	uint	
maxDegree	uint	
GM	double	Geocentric gravitational constant
R	double	reference radius
power	double	$\text{sigma} = \text{factor}/\text{degree}^{\text{power}}$
factor	double	$\text{sigma} = \text{factor}/\text{degree}^{\text{power}}$

### 4.9.10 Kernel2Coefficients

This program computes and returns the coefficients and inverse coefficients of a **kernel** from from **minDegree** to **maxDegree** at a given **height**.

The main purpose is for visualization with **PlotGraph**.

Name	Type	Annotation
outfileMatrix	filename	matrix with columns degree, coefficients and inverse coefficients
kernel	kernel	
minDegree	uint	minimum degree of returned coefficients
maxDegree	uint	compute coefficients up to maxDegree
height	double	evaluate kernel at R+height [m]
R	double	reference radius

#### 4.9.11 Kernel2SigmaPotentialCoefficients

Create variances of spherical harmonics by convolution a kernel with white noise, e.g. to display filter coefficients of a Gaussian filter. The coefficients are written as formal errors of **outfilePotentialCoefficients**.

Name	Type	Annotation
outfilePotentialCoefficients	filename	
kernel	kernel	
minDegree	uint	
maxDegree	uint	
GM	double	Geocentric gravitational constant
R	double	reference radius
factor	double	

### 4.9.12 KernelEvaluate

Compute **kernel** values for distant angles. The main purpose is for visualization with **PlotGraph**.

Name	Type	Annotation
outputfileMatrix	filename	matrix with first column the angle [degree], second the kernel value
kernel	kernel	
minAngle	angle	[degree]
maxAngle	angle	[degree]
sampling	angle	[degree]
height	double	evaluate at R+height [m]
R	double	reference radius

#### 4.9.13 MagneticField2GriddedData

Computes x, y, z of the magentic field vector.

Name	Type	Annotation
outputfileGriddedData	filename	x, y, z [Tesla = kg/A/s**2]
magnetosphere	magnetosphere	
grid	grid	
time	time	
localReferenceFrame	boolean	local left handed reference frame (north, east, up)
R	double	reference radius for ellipsoidal coordinates on output
inverseFlattening	double	reference flattening for ellipsoidal coordinates on output, 0: spherical coordinates

This program is parallelized (1.5).

#### 4.9.14 MatrixCalculate

This program creates a `matrix` from multiple matrices. All `matrices` are summed up. The size of the resulting matrix is expanded to fit all matrices. The class `matrixGenerator` allows complex matrix operations before.

Name	Type	Annotation
<code>outputfileMatrix</code>	filename	
<code>matrix</code>	<code>matrixGenerator</code>	

#### 4.9.15 ObservationEquations2Files

This program computes the linearized and decorrelated equation system for each arc  $i$ :

$$\mathbf{l}_i = \mathbf{A}_i \mathbf{x} + \mathbf{B}_i \mathbf{y}_i + \mathbf{e}_i \quad (4.51)$$

using class `observation` and writes  $\mathbf{A}_i$ ,  $\mathbf{B}_i$  and  $\mathbf{l}_i$  as `matrix` files.

Name	Type	Annotation
<code>outputfileObservationVector</code>	filename	one file for each arc
<code>outputfileDesignMatrix</code>	filename	one file for each arc, without arc related parameters
<code>outputfileDesignMatrixArc</code>	filename	one file for each arc, arc related parameters
<code>variableArc</code>	string	variable with arc number
<code>outputfileParameterNames</code>	filename	without arc related parameters
<code>observation</code>	<code>observation</code>	

This program is parallelized (1.5).

### 4.9.16 PlatformCreate

Create a **Platform** file from scratch by defining attributes such as **markerName**, **markerNumber**, **comment**, **approxPosition**, **equipment**.

See also **GnssAntex2AntennaDefinition** and **GnssStationLog2Platform**.

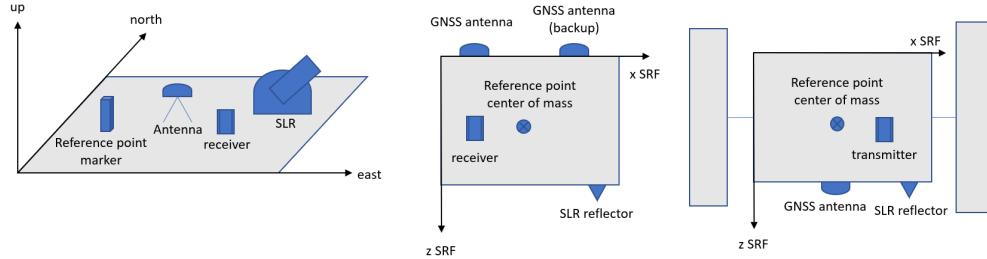


Figure 4.18: Platform for stations, LEOs, and GNSS satellites.

Name	Type	Annotation
outputfilePlatform	filename	
markerName	string	
markerNumber	string	
comment	string	
approxPositionX	double	[m] in TRF
approxPositionY	double	[m] in TRF
approxPositionZ	double	[m] in TRF
equipment	choice	
gnssAntenna	sequence	
name	string	
serial	string	
radome	string	
comment	string	
timeStart	time	
timeEnd	time	
positionX	double	[m] ARP in north, east, up or vehicle system
positionY	double	[m] ARP in north, east, up or vehicle system
positionZ	double	[m] ARP in north, east, up or vehicle system
rotationX	angle	[degree] from local/vehicle to left-handed antenna system
rotationY	angle	[degree] from local/vehicle to left-handed antenna system
rotationZ	angle	[degree] from local/vehicle to left-handed antenna system
flipX	boolean	flip x-axis (after rotation)
flipY	boolean	flip y-axis (after rotation)
flipZ	boolean	flip z-axis (after rotation)
gnssReceiver	sequence	
name	string	
serial	string	
version	string	
comment	string	
timeStart	time	
timeEnd	time	
laserRetroReflector	sequence	e.g. GFZ, ITE, IPIE
name	string	
serial	string	
comment	string	

timeStart	time	
timeEnd	time	
positionX	double	[m] optional reference point RP in satellite system
positionY	double	[m] optional reference point RP in satellite system
positionZ	double	[m] optional reference point RP in satellite system
rotationX	angle	[degree] from local/vehicle to LRR system
rotationY	angle	[degree] from local/vehicle to LRR system
rotationZ	angle	[degree] from local/vehicle to LRR system
flipX	boolean	flip x-axis (after rotation)
flipY	boolean	flip y-axis (after rotation)
flipZ	boolean	flip z-axis (after rotation)
range	double	[m] range bias (only without range matrix)
inputfileRangeMatrix	filename	[m] (azimuth(0..360) x zenith(0..dZenit*rows)
dZenit	angle	[degree] increment of range matrix
geodeticSatellite	sequence	e.g. LAGEOS
name	string	
serial	string	
comment	string	
timeStart	time	
timeEnd	time	
range	double	[m] standard center-of-mass correction
slrStation	sequence	
name	string	CDP SOD 8-digit No.
serial	string	IERS DOMES
comment	string	
timeStart	time	
timeEnd	time	
positionX	double	[m] exccentricity in north
positionY	double	[m] exccentricity in east
positionZ	double	[m] exccentricity in up
satelliteIdentifier	sequence	
name	string	Satellite COSPAR ID
serial	string	Satellite Catalog (NORAD) Number
cospar	string	
norad	string	
sic	string	SIC Code
sp3	string	SP3
comment	string	
timeStart	time	
timeEnd	time	
other	sequence	
name	string	
serial	string	
comment	string	
timeStart	time	
timeEnd	time	
positionX	double	[m] in north, east, up or vehicle system
positionY	double	[m] in north, east, up or vehicle system
positionZ	double	[m] in north, east, up or vehicle system
referencePoint	sequence	e.g. center of mass in satellite frame
comment	string	
xStart	double	[m] in north, east, up or vehicle system
yStart	double	linear motion between start and end
zStart	double	

---

	xEnd	double	[m] in north, east, up or vehicle system
	yEnd	double	linear motion between start and end
	zEnd	double	
	timeStart	time	
	timeEnd	time	

---

#### 4.9.17 PotentialCoefficients2BlockMeanTimeSplines

This program is a simplified version of **Gravityfield2TimeSplines**. It reads a series of potential coefficient files (**inputfilePotentialCoefficients**) and creates a time splines file with spline degree 0 (temporal block means) or degree 1 (linear splines). The time intervals in which the potential coefficients are valid are defined between adjacent points in time given by **splineTimeSeries**. Therefore one more point in time is needed than the number of potential coefficient files for degree 0.

The coefficients can be filtered with **filter**. If set the expansion is limited in the range between **minDegree** and **maxDegree** inclusivly. The coefficients are related to the reference radius **R** and the Earth gravitational constant **GM**.

This program is useful e.g. to combine monthly GRACE solutions to one file.

Name	Type	Annotation
outputfileTimeSplines	filename	
outputfileTimeSplinesCovariance	filename	only the variances are saved
inputfilePotentialCoefficients	filename	
filter	sphericalHarmonicsFilter	
minDegree	uint	
maxDegree	uint	
GM	double	Geocentric gravitational constant
R	double	reference radius
removeMean	boolean	remove the temporal mean of the series before estimating the splines
interpolate	boolean	interpolate missing files
splineTimeSeries	timeSeries	input files must be between points in time
splineDegree	uint	degree of splines

### 4.9.18 PotentialCoefficients2DegreeAmplitudes

This program computes degree amplitudes from `potentialCoefficients` files and saves them to a `matrix` file.

The coefficients can be filtered with `filter` and converted to different functionals with `kernel`. The gravity field can be evaluated at different altitudes by specifying `evaluationRadius`. Polar regions can be excluded by setting `polarGap`. If set the expansion is limited in the range between `minDegree` and `maxDegree` inclusivly. The coefficients are related to the reference radius `R` and the Earth gravitational constant `GM`.

The `outputfileMatrix` contains in the first 3 columns the degree, the degree amplitude, and the formal errors. For each additional `inputfilePotentialCoefficients` three columns are appended: the degree amplitude, the formal errors, and the difference to the first file.

For example the data columns for 4 `inputfilePotentialCoefficients` are

- `degree=data0`
- `PotentialCoefficients0: signal=data1, error=data2,`
- `PotentialCoefficients1: signal=data3, error=data4, difference=data5,`
- `PotentialCoefficients2: signal=data6, error=data7, difference=data8,`
- `PotentialCoefficients3: signal=data9, error=data10, difference=data11.`

See also `Gravityfield2DegreeAmplitudes`.

Name	Type	Annotation
<code>outputfileMatrix</code>	filename	matrix with degree, signal amplitude, formal error, differences
<code>inputfilePotentialCoefficients</code>	filename	
<code>kernel</code>	kernel	
<code>filter</code>	sphericalHarmonicsFilter	filter the coefficients
<code>type</code>	choice	type of variances
<code>rms</code>		degree amplitudes (square root of degree variances)
<code>accumulation</code>		cumulate variances over degrees
<code>evaluationRadius</code>	double	evaluate the gravity field at this radius (default: evaluate at surface)
<code>polarGap</code>	angle	exclude polar regions (aperture angle in degrees)
<code>minDegree</code>	uint	
<code>maxDegree</code>	uint	
<code>GM</code>	double	Geocentric gravitational constant
<code>R</code>	double	reference radius

#### 4.9.19 RadialBasisSplines2KernelCoefficients

This program calculates the coefficients  $k_n$  of a **kernel:coefficients** according to

$$k_n = \frac{GM}{4\pi R} \frac{\sigma_n}{\sqrt{2n+1}}. \quad (4.52)$$

from a given **gravityfield**, with **R** and **GM** describing the reference radius and the geocentric constant, respectively. The  $\sigma_n$  stand for the gravity field accuracies (from degree **minDegree** to **maxDegree**), if they are given. If no accuracies are provided, the  $\sigma_n$  represent the square root of the degree variances of the gravity field. If **maxDegree** exceeds the maximum degree given by **gravityfield**, the higher degrees are complemented by Kaula's rule. The output of the coefficients is given in the file **outputfileCoefficients**.

Name	Type	Annotation
<b>outputfileCoefficients</b>	filename	
<b>gravityfield</b>	<b>gravityfield</b>	use sigmas, if not given use signal (cnm,snm), if not given use kaulas rule
<b>minDegree</b>	uint	
<b>maxDegree</b>	uint	
<b>GM</b>	double	Geocentric gravitational constant
<b>R</b>	double	reference radius
<b>kaulaPower</b>	double	sigma = kaulaFactor/degree^kaulaPower
<b>kaulaFactor</b>	double	sigma = kaulaFactor/degree^kaulaPower

### 4.9.20 SatelliteModelCreate

This program creates a satellite macro model for the estimation of non-gravitational accelerations acting on a satellite. Mandatory input values are the **satelliteName**, **mass**, **coefficientDrag** and information about the satellite **surfaces**. For low Earth orbiting satellites, like GRACE for instance, a good guess for the drag coefficient could be 2.3. Apart from that, it is possible to estimate a more precise variable drag coefficient (e.g. **miscAccelerations:atmosphericDrag**), which will override this initial guess. Concerning the satellite surfaces an external file must be imported which must contain information about each single satellite plate in terms of plate **area**, the associated plate normal and re-radiation properties (reflexion, diffusion and absorption) properties in the visible and IR part. Exemplarily, a description of the macro model for GRACE can be found under: [https://podaac-tools.jpl.nasa.gov/drive/files/allData/grace/docs/ProdSpecDoc\\_v4.6.pdf](https://podaac-tools.jpl.nasa.gov/drive/files/allData/grace/docs/ProdSpecDoc_v4.6.pdf) Additionally, it is possible to add further information like antennaThrust, solar panel, temporal mass changes and massInstrument using the modules option.

Name	Type	Annotation
outputfileSatelliteModel	filename	
satellite	sequence	
satelliteName	string	
mass	double	
coefficientDrag	double	
surfaces	sequence	
inputfile	filename	each line must contain one surface element
type	expression	0: plate, 1: sphere, 2: cylinder
area	expression	[m <sup>2</sup> ]
normalX	expression	
normalY	expression	
normalZ	expression	
reflexionVisible	expression	
diffusionVisible	expression	
absorptionVisible	expression	
reflexionInfrared	expression	
diffusionInfrared	expression	
absorptionInfrared	expression	
specificHeatCapacity	expression	0: no thermal radiation, -1: direct reemission [Ws/K/m <sup>2</sup> ]
module	choice	
antennaThrust	sequence	
thrustX	double	
thrustY	double	
thrustZ	double	
solarPanel	sequence	
rotationAxisX	double	Direction to sun
rotationAxisY	double	Direction to sun
rotationAxisZ	double	Direction to sun
normalX	double	indexSurface
normalY	double	indexSurface
normalZ	double	indexSurface
indexSurface	uint	index of solar panel surfaces
massChange	sequence	
time	time	
mass	double	
massInstrument	sequence	
inputfileInstrument	filename	

#### 4.9.21 TemporalRepresentation2TimeSeries

This program computes the design matrix of temporal representation at a given time series. The output matrix contains the time steps in MJD in the first column, the other columns contain the design matrix. The intention of this program is to visualize the parametrization together with [PlotGraph](#).

Name	Type	Annotation
 <code>outputfileMatrix</code>	filename	Time (MJD) in first column, design matrix follows
 <code>timeSeries</code>	<a href="#">timeSeries</a>	
 <code>temporal</code>	<a href="#">parametrizationTemporal</a>	

### 4.9.22 ThermosphericState2GriddedData

This program converts the output (neutral mass density, temperature) of an empirical thermosphere model (e.g. JB2008) on a given **grid**. Additionally, also the thermospheric winds estimated by using the horizontal wind model HWM 2014 can be assessed. The time for the evaluation can be specified in **time**. The values will be saved together with points expressed as ellipsoidal coordinates (longitude, latitude, height) based on a reference ellipsoid with parameters **R** and **inverseFlattening**.

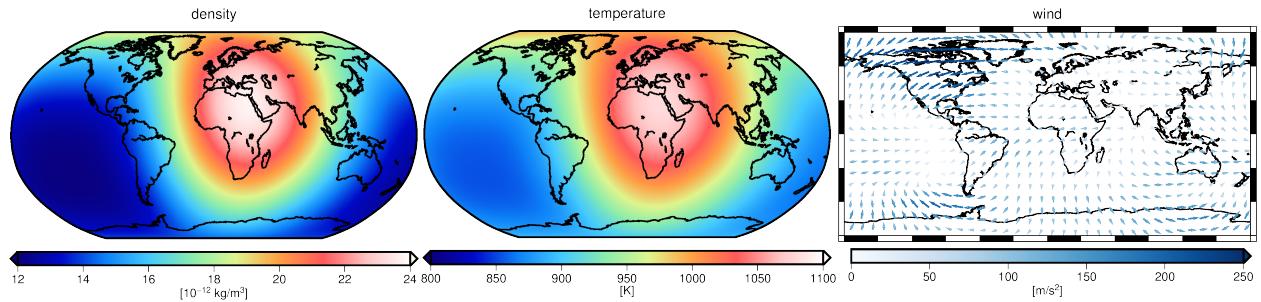


Figure 4.19: JB2008 model in 300 km height at 2003-07-01 12:00.

Name	Type	Annotation
<b>outputfileGriddedData</b>	filename	density [ $\text{kg/m}^{**3}$ ], temperature [K], wind (x, y, z) [ $\text{m/s}^{**2}$ ]
<b>thermosphere</b>	<b>thermosphere</b>	
<b>grid</b>	<b>grid</b>	
<b>time</b>	time	
<b>localReferenceFrame</b>	boolean	wind in local north, east, up, otherwise global terrestrial
<b>R</b>	double	reference radius for ellipsoidal coordinates on output
<b>inverseFlattening</b>	double	reference flattening for ellipsoidal coordinates on output, 0: spherical coordinates

This program is [parallelized \(1.5\)](#).

### 4.9.23 TimeSeries2PotentialCoefficients

Interpret the data columns of **inputfileTimeSeries** as potential coefficients. The sequence of coefficients is given by **numbering** starting from data column **startDataFields**.

For each epoch a **outputfilesPotentialCoefficients** is written where the **variableLoopTime** and **variableLoopIndex** are expanded for each point of the given time series to create the file name for this epoch, see [text parser \(1.2.2\)](#).

See also **Gravityfield2PotentialCoefficientsTimeSeries**.

Name	Type	Annotation
outputfilesPotentialCoefficients	filename	for each epoch
variableLoopTime	string	variable with time of each epoch
variableLoopIndex	string	variable with index of current epoch (starts with zero)
variableLoopCount	string	variable with total number of epochs
inputfileTimeSeries	filename	each epoch: multiple data for points (MISCVALUES)
startDataFields	uint	first data column
minDegree	uint	minimal degree
maxDegree	uint	maximal degree
GM	double	Geocentric gravitational constant
R	double	reference radius
numbering	sphericalHarmonicsNumbering	numbering scheme

#### 4.9.24 TimeSeriesCreate

This program generates an [instrument file](#), containing a time series.

Name	Type	Annotation
outputfileTimeSeries	filename	instrument file
timeSeries	timeSeries	time series to be created
data	expression	expression of output columns, extra 'epoch' variable

#### 4.9.25 Variational2OrbitAndStarCamera

Extracts the reference ▶ **outputfileOrbit**, ▶ **outputfileStarCamera**, and ▶ **outputfileEarthRotation** from ▶ **inputfileVariational**.

Name	Type	Annotation
▶ <b>outputfileOrbit</b>	filename	output orbit (instrument) file
▶ <b>outputfileStarCamera</b>	filename	output satellite attitude as star camera (instrument) file
▶ <b>outputfileEarthRotation</b>	filename	output Earth rotation as star camera (instrument) file
▶ <b>inputfileVariational</b>	filename	input variational file

## 4.10 Programs: NormalEquation

### 4.10.1 NormalsAccumulate

This program accumulates normal equations and writes the total combined system to **outputfileNormalequation**. The **inputfileNormalEquations** must have all the same size and the same block structure. This program is the simplified and fast version of the more general program **NormalsBuild**.

Name	Type	Annotation
outputfileNormalEquation	filename	
inputfileNormalEquation	filename	

### 4.10.2 NormalsBuild

This program accumulates **►normalEquations** and writes the total combined system to **►outfileNormalequation**. For a detailed description of the used algorithm see **►normalEquation**. Large normal equation systems can be divided into blocks with **►normalsBlockSize**.

A simplified and fast version of this program is **►NormalsAccumulate**. To solve the system of normal equations use **►NormalsSolverVCE**.

Name	Type	Annotation
<b>►outfileNormalEquation</b>	filename	
<b>►normalEquation</b>	<b>normalEquation</b>	
<b>►normalsBlockSize</b>	uint	block size for distributing the normal equations, 0: one block

This program is parallelized (1.5).

### 4.10.3 NormalsBuildShortTimeStaticLongTime

This program sets up normal equations based on **observation**. Additionally short time and long time variations can be parametrized based on the static parameters in **observation** in an efficient way. The observation equations are divided into time intervals  $i \in \{1, \dots, N\}$  (e.g. daily) as defined in **inputfileArcList**.

With **estimateLongTimeVariations** additional temporal variations can be co-estimated for a subset of the parameters selected by **parameterSelection**. These parameters might be spherical harmonic coefficients with a limited maximum degree. The temporal variations are represented by base functions  $\Phi_k(t_i)$  (e.g. trend and annual oscillation) given by **parametrizationTemporal**. The temporal base functions are evaluated at the mid time  $t_i$  of each interval  $i$ , multiplicatively with the design matrix  $\mathbf{A}_i$  of the selected parameters, and the design matrix is extended accordingly.

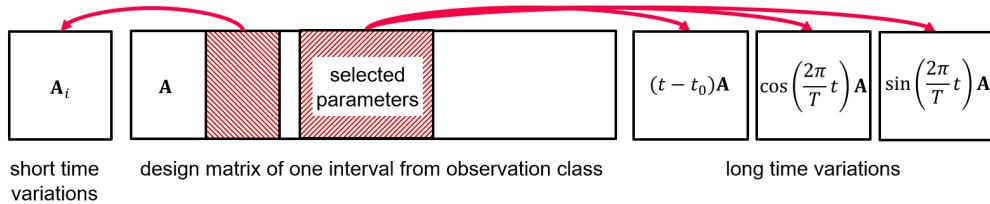


Figure 4.20: Schema of the extended design matrix.

With **estimateShortTimeVariations** short time variations of the gravity field can be co-estimated. Their purpose is to mitigate temporal aliasing. The short time parameters selected by **parameterSelection** (e.g. daily constant or linear splines every 6 hour) are constrained by an **autoregressiveModelSequence**. If only a static parameter set is selected the corresponding part of the design matrix is copied and modeled as a constant value per interval in **inputfileArcList** additionally so the corresponding temporal factor can be expressed as

$$\Phi_i(t) = \begin{cases} 1 & \text{if } t \in [t_i, t_{i+1}) \\ 0 & \text{otherwise} \end{cases}. \quad (4.53)$$

Before writing the normal equations to **outputfileNormalEquation** short time gravity and satellite specific parameters can be eliminated with **eliminateParameter**.

Example: For the computation of the mean gravity field ITSG-Grace2018s with additional trend and annual signal the normal equations are computed month by month and accumulated afterwards (see **NormalsAccumulate**). The observations were divided into daily intervals with **inputfileArcList**. The static gravity field has been parametrized as spherical harmonics up to degree  $n = 200$  in **observation:parametrizationGravity**. The trend and annual signals defined by **estimateLongTimeVariations:parametrizationTemporal** were estimated for selected parameters up to degree  $n = 120$ . To mitigate temporal aliasing daily gravity fields up to degree  $n = 40$  were setup and constrained with an **autoregressiveModelSequence** up to order three.

A detailed description of the approach is given in: Kvas, A., Mayer-Gürr, T. GRACE gravity field recovery with background model uncertainties. J Geod 93, 2543–2552 (2019). <https://doi.org/10.1007/s00190-019-01314-1>.

Name	Type	Annotation
outputfileNormalEquation	filename	outputfile for normal equations
observation	observation	
estimateShortTimeVariations	sequence	co-estimate short time gravity field variations

 autoregressiveModelSequence	autoregressiveModelSequence	AR model sequence for constraining short time gravity variations
 parameterSelection	parameterSelector	parameters describing the short time gravity field
 estimateLongTimeVariations	sequence	co-estimate long time gravity field variations
 parametrizationTemporal	parametrizationTemporal	parametrization of time variations (trend, annual, ...)
 parameterSelection	parameterSelector	parameters describing the long time gravity field
 inputFileArcList	filename	list to correspond points of time to arc numbers
 defaultBlockSize	uint	block size for distributing the normal equations, 0: one block
 eliminateParameter	boolean	eliminate short time and state parameter

---

This program is parallelized (1.5).

#### 4.10.4 NormalsCreate

Create **normal equations** from calculated matrices (**matrixGenerator**).

The **inputfileParameterNames** can be created with **ParameterNamesCreate**.

The **normalMatrix** must be symmetric. The **rightHandSide** must have the same number of rows and can contain multiple columns for multiple solutions.

The Vector  $\mathbf{l}^T \mathbf{P} \hat{\mathbf{e}}$  is the quadratic sum of observations for each column of the right hand side. It is used to determine the aposteriori accuracy

$$\hat{\sigma}^2 = \frac{\hat{\mathbf{e}}^T \mathbf{P} \hat{\mathbf{e}}}{n - m} = \frac{\mathbf{l}^T \mathbf{P} \mathbf{l} - \mathbf{n}^T \hat{\mathbf{x}}}{n - m}. \quad (4.54)$$

If the vector is not given, it is automatically determined by assuming  $\hat{\sigma}^2 = 1$ .

The number of observations  $n$  is given by the expression **observationCount**. The variable **observationCount** can be used, if it is set by a normal equation file **inputfileNormalEquationObsCount**.

Name	Type	Annotation
<b>outputfileNormalEquation</b>	filename	
<b>inputfileParameterNames</b>	filename	
<b>normalMatrix</b>	<b>matrixGenerator</b>	
<b>rightHandSide</b>	<b>matrixGenerator</b>	
<b>lP1</b>	<b>matrixGenerator</b>	vector with size of rhs columns
<b>inputfileNormalEquationObsCount</b>	filename	sets the variable <b>observationCount</b> (variables: rows, columns (rhs), observationCount)
<b>observationCount</b>	expression	

### 4.10.5 NormalsEliminate

This program eliminates parameters from a system of **inputfileNormalEquations**. To just remove (cutting out) parameters use **NormalsReorder**.

The **remainingParameters** allows the selection of parameters that will remain, all others will be eliminated. The order of remaining parameters can be modified via the parameter selection. Block size of the output normal matrix can be adjusted with **outBlockSize**. If it is set to zero, the **outputfileNormalEquation** is written to a single block file.

For example the normal equations are divided into two groups of parameters  $\hat{\mathbf{x}}_1$  and  $\hat{\mathbf{x}}_2$  according to

$$\begin{pmatrix} \mathbf{N}_{11} & \mathbf{N}_{12} \\ \mathbf{N}_{21} & \mathbf{N}_{22} \end{pmatrix} \begin{pmatrix} \hat{\mathbf{x}}_1 \\ \hat{\mathbf{x}}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{n}_1 \\ \mathbf{n}_2 \end{pmatrix}. \quad (4.55)$$

and  $\hat{\mathbf{x}}_2$  shall be eliminated, the reduced system of normal equations is given by

$$\bar{\mathbf{N}}\hat{\mathbf{x}} = \bar{\mathbf{n}} \quad \text{with} \quad \bar{\mathbf{N}} = \mathbf{N}_{11} - \mathbf{N}_{12}\mathbf{N}_{22}^{-1}\mathbf{N}_{12}^T \quad \text{and} \quad \bar{\mathbf{n}} = \mathbf{n}_1 - \mathbf{N}_{12}\mathbf{N}_{22}^{-1}\mathbf{n}_2. \quad (4.56)$$

See also **NormalsReorder**.

Name	Type	Annotation
<b>outputfileNormalEquation</b>	filename	
<b>inputfileNormalEquation</b>	filename	
<b>remainingParameters</b>	parameterSelector	parameter order/selection of output normal equations
<b>outBlockSize</b>	uint	block size for distributing the normal equations, 0: one block

This program is parallelized (1.5).

### 4.10.6 NormalsMultiplyAdd

This program modifies **inputfileNormalEquation** in a way that  $\bar{\mathbf{x}}$  is estimated instead of  $\mathbf{x}$ .

$$\bar{\mathbf{x}} := \mathbf{x} + \alpha \mathbf{x}_0, \quad (4.57)$$

where  $\mathbf{x}_0$  is **inputfileParameter** and  $\alpha$  is **factor**. This can be used to re-add reduced reference fields before a combined estimation at normal equation level. Therefore the right hand side of the normal equations is modified by

$$\bar{\mathbf{n}} := \mathbf{n} + \alpha \mathbf{N}\mathbf{x}_0, \quad (4.58)$$

and the quadratic sum of observations by

$$\bar{\mathbf{l}}^T \bar{\mathbf{P}} \bar{\mathbf{l}} := \mathbf{l}^T \mathbf{P} \mathbf{l} + \alpha^2 \mathbf{x}_0^T \mathbf{N} \mathbf{x}_0 + 2\alpha \mathbf{x}_0^T \mathbf{n} \quad (4.59)$$

As the normal matrix itself is not modified, rewriting of the matrix can be disabled by setting **writeNormalMatrix** to false.

Name	Type	Annotation
<b>outputfileNormalEquation</b>	filename	
<b>inputfileNormalEquation</b>	filename	
<b>inputfileParameter</b>	filename	x
<b>factor</b>	double	alpha
<b>writeNormalMatrix</b>	boolean	write full coefficient matrix, right hand sides and info files

This program is [parallelized \(1.5\)](#).

#### 4.10.7 NormalsRegularizationBorders

This program sets up two regularization matrices for two different regional areas. For a given set of points defined by **grid** it is evaluated, whether each point (corresponding to an unknown parameter of a respective parameterization by space localizing basis functions) is inside or outside a certain area given by **border**. Each regularization matrix is a diagonal matrix, one of them features a one if the point is inside, and a zero if the point lies outside the area. The other matrix features a zero if the point is inside, and a one if the point lies outside the area. This results in two regularization matrices with

$$\mathbf{R}_1 + \mathbf{R}_2 = \mathbf{I}. \quad (4.60)$$

The two matrices are provided as vectors of the diagonal in the output files **outputfileOutside** and **outputfileInside**. The regularization matrices are then used by **normalEquation:regularization**. As an example, the two different areas could be oceanic regions on the one hand and continental areas on the other hand.

Name	Type	Annotation
<b>outputfileInside</b>	filename	
<b>outputfileOutside</b>	filename	
<b>grid</b>	grid	nodal point distribution of parameters, e.g harmonics splines
<b>border</b>	border	regularization areas, e.g land and ocean

#### 4.10.8 NormalsRegularizationSphericalHarmonics

Diagonal regularization matrix from gravity field accuracies, if not given from signal (cnm,snm), if not given from kaulas rule. The inverse accuracies  $1/\sigma_n^2$  are used as weights in the regularization matrix. The diagonal is saved as Vector.

The corresponding pseudo observations can be computed with [Gravityfield2SphericalHarmonicsVector](#).

Name	Type	Annotation
outputfileDiagonalmatrix	filename	
gravityfield	gravityfield	use sigmas, if not given use signal (cnm,snm), if not given use kaulas rule
minRegularizationDegree	uint	
maxRegularizationDegree	uint	
minDegree	uint	
maxDegree	uint	
numbering	sphericalHarmonicsNumbering	numbering scheme for regul matrix
GM	double	Geocentric gravitational constant
R	double	reference radius
makeIsotropic	boolean	
kaulaPower	double	sigma = kaulaFactor*degree**kaulaPower
kaulaFactor	double	sigma = kaulaFactor*degree**kaulaPower

#### 4.10.9 NormalsReorder

Reorder **► inputFileNormalEquation** by selecting parameters in a specific order. The **► parameterSelection** also allows one to change dimension of the normal equations, either by cutting parameters or by inserting zero rows/columns for additional parameters. Without **► parameterSelection** the order of parameters remains the same. Additionally the block sizes of the files can be adjusted. If **► outBlockSize** is set to zero, the normal matrix is written to a single block file, which is needed by some programs.

To eliminate parameters without changing the result of the other parameters use **► NormalsEliminate**.

Name	Type	Annotation
<b>► outputFileNormalEquation</b>	filename	
<b>► inputFileNormalEquation</b>	filename	
<b>► parameterSelection</b>	<b>parameterSelector</b>	parameter order/selection of output normal equations
<b>► outBlockSize</b>	uint	block size for distributing the normal equations, 0: one block

This program is parallelized (1.5).

### 4.10.10 NormalsScale

Scales rows and columns of a system of **inputfileNormalEquation** given by a diagonal matrix **inputfileFactorVector S**

$$\bar{\mathbf{N}} := \mathbf{S} \mathbf{N} \mathbf{S} \quad \text{and} \quad \bar{\mathbf{n}} := \mathbf{S} \mathbf{n}. \quad (4.61)$$

The estimated solution is now

$$\bar{\mathbf{x}} := \mathbf{S}^{-1} \mathbf{x}. \quad (4.62)$$

This is effectively the same as rescaling columns of the design matrix. This program is useful when combining normal equations from different sources, for example in case the units of certain parameters don't match.

Name	Type	Annotation
outputfileNormalEquation	filename	
inputfileNormalEquation	filename	
inputfileFactorVector	filename	Vector containing the factors

This program is [parallelized \(1.5\)](#).

### 4.10.11 NormalsSolverVCE

This program accumulates [normalEquation](#) and solves the total combined system. The relative weighting between the individual normals is determined iteratively by means of variance component estimation (VCE). For a detailed description of the used algorithm see [normalEquation](#).

Besides the estimated parameter vector ([outputfileSolution](#)) the estimated accuracies ([outputfileSigmax](#)) and the full covariance matrix ([outputfileCovariance](#)) can be saved. Also the combined normal system can be written to [outputfileNormalEquation](#).

The [outputfileContribution](#) is a matrix with rows for each estimated parameter and columns for each [normalEquation](#) and indicates the contribution of the individual normals to the estimated parameters. Each row sum up to one.

See also [NormalsBuild](#).

Name	Type	Annotation
<a href="#">outputfileSolution</a>	filename	parameter vector
<a href="#">outputfileSigmax</a>	filename	standard deviations of the parameters (sqrt of the diagonal of the inverse normal equation)
<a href="#">outputfileCovariance</a>	filename	full covariance matrix
<a href="#">outputfileContribution</a>	filename	contribution of normal system components to the solution vector
<a href="#">outputfileVarianceFactors</a>	filename	estimated variance factors as vector
<a href="#">outputfileNormalEquation</a>	filename	the combined normal equation system
<a href="#">normalEquation</a>	<a href="#">normalEquation</a>	
<a href="#">inputfileApproxSolution</a>	filename	to accelerate convergence
<a href="#">rightHandSideNumberVCE</a>	uint	the right hand side number for estimation of variance factors
<a href="#">normalsBlockSize</a>	uint	block size for distributing the normal equations, 0: one block
<a href="#">maxIterationCount</a>	uint	maximum number of iterations for variance component estimation

This program is parallelized (1.5).

### 4.10.12 NormalsTemporalCombination

This program reads a times series of **inputfileNormalequation** with associated **timeSeries** and setup a new combined normal equation system. For each parameter a **parametrizationTemporal** is used.

It can be used to estimate trend and annual spherical harmonic coefficients from monthly GRACE normal equations.

Name	Type	Annotation
outputfileNormalEquation	filename	
inputfileNormalEquation	filename	normal equations for each point in time
timeSeries	timeSeries	times of each normal equations
parametrizationTemporal	parametrizationTemporal	

This program is [parallelized \(1.5\)](#).

#### 4.10.13 ParameterNamesCreate

Generate a **outfileParameterNames** by **parameterName**. This file can be used in **NormalsCreate** or in the class **parameterSelector**.

Name	Type	Annotation
<b>outfileParameterNames</b>	filename	output parameter names file
<b>parameterName</b>	<b>parameterNames</b>	

#### 4.10.14 ParameterSelection2IndexVector

Generate index vector from parameter selection in [matrix format](#). This vector can be used in **MatrixCalculate** with **matrix:reorder** to reorder arbitrary vectors and matrices similar to **NormalsReorder**.

The **parameterSelection** allows reordering and dimension changes, either by cutting parameters or by inserting additional parameters. **outputfileIndexVector** contains indices of parameters in **inputfileParameterNames** or -1 for newly added parameters. **outputfileParameterNames** contains the selected parameter names.

Name	Type	Annotation
<b>outputfileIndexVector</b>	filename	indices of source parameters in target normal equations
<b>outputfileParameterNames</b>	filename	output parameter names file
<b>inputfileParameterNames</b>	filename	parameter names file of source normal equations
<b>parameterSelection</b>	<b>parameterSelector</b>	parameter order/selection of target normal equations

## 4.11 Programs: Orbit

### 4.11.1 Orbit2ArgumentOfLatitude

This program computes the argument of latitude of an `orbit` and writes it as `instrument file` (MISCVALUE(S)). The data of `infileInstrument` are appended as values to each epoch.

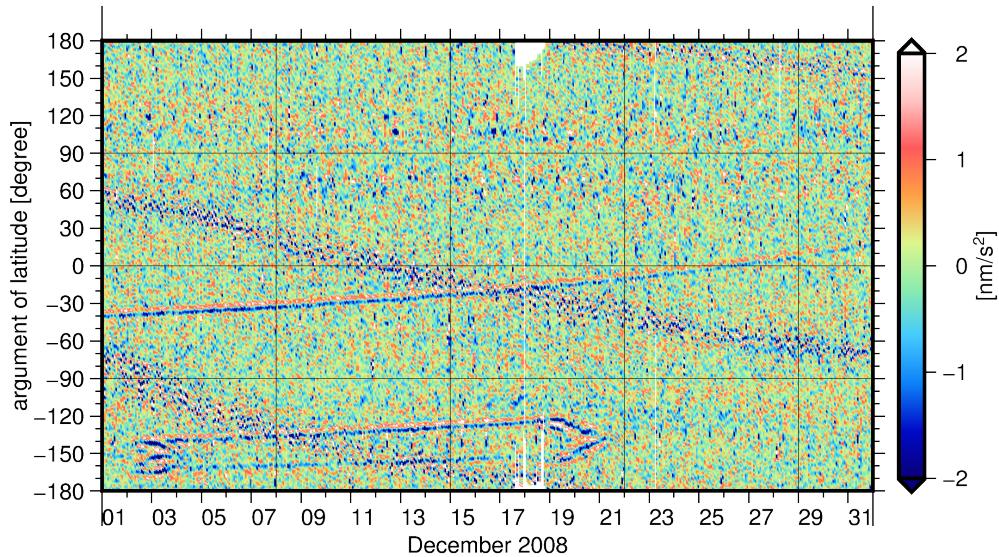


Figure 4.21: Derivation filtered GRACE range-rate residuals.

Name	Type	Annotation
<code>outputfileArgOfLatitude</code>	filename	instrument file (MISCVALUE(S): argLat, ...)
<code>infileOrbit</code>	filename	
<code>infileInstrument</code>	filename	data are appended

This program is parallelized (1.5).

### 4.11.2 Orbit2BetaPrimeAngle

This program computes the beta prime angle (between the orbital plane and earth-sun direction) and writes it as MISCVALUE(S)  instrument file. The angle is calculated w.r.t the sun (per default), but can be changed. The data of  **inputfileInstrument** are appended as values to each epoch.

Name	Type	Annotation
 <b>outputfileBetaAngle</b>	filename	instrument file (MISCVALUE(S): beta', ...)
 <b>inputfileOrbit</b>	filename	
 <b>inputfileInstrument</b>	filename	data are appended
 <b>ephemerides</b>	ephemerides	
 <b>planet</b>	planet	

This program is [parallelized \(1.5\)](#).

### 4.11.3 Orbit2EarthFixedOrbit

Normally the orbits in GROOPS are given in the celestial reference frame (CRF) with the origin in the center of mass (CoM). This program rotates the orbit with **earthRotation** from CRF to the TRF.

To additionally transform into the center of solid Earth (CE) frame (or center of Figure (CF)), a correction can be applied by providing degree one coefficients of a **gravityfield** (e.g. ocean tides).

If **celestial2terrestrial** is set to no, the inverse transformation is applied.

See also **InstrumentRotate**.

Name	Type	Annotation
<b>outputfileOrbit</b>	filename	
<b>inputfileOrbit</b>	filename	
<b>earthRotation</b>	<b>earthRotation</b>	transformation from CRF to TRF
<b>gravityfield</b>	<b>gravityfield</b>	degree 1 fluid mantle for CM2CE correction
<b>celestial2terrestrial</b>	boolean	yes: crf-\$;strf, no: trf-\$;crf

This program is parallelized (1.5).

#### 4.11.4 Orbit2EclipseFactor

This program generates an **instrument file** (MISCVALUE(S)) containing the eclipse factor for a given set of orbit. The data of **inputfileInstrument** are appended as values to each epoch.

Name	Type	Annotation
outputfileEclipseFactor	filename	instrument file (MISCVALUE(S): eclipse, ...)
inputfileOrbit	filename	
inputfileInstrument	filename	data are appended
ephemerides	ephemerides	
eclipse	eclipse	

This program is [parallelized \(1.5\)](#).

#### 4.11.5 Orbit2Groundtracks

This program write **satellites positions** as **gridded data** (**outputfileTrackGriddedData**) in a terrestrial reference frame. The points are expressed as ellipsoidal coordinates (longitude, latitude, height) based on a reference ellipsoid with parameters **R** and **inverseFlattening**. The orbit data are given in the celestial frame so **earthRotation** is needed to transform the data into the terrestrial frame. The data of **inputfileInstrument** are appended as values to each point.

Name	Type	Annotation
<b>outputfileGriddedData</b>	filename	positions as gridded data
<b>inputfileOrbit</b>	filename	
<b>inputfileInstrument</b>	filename	values at grid points
<b>earthRotation</b>	<b>earthRotation</b>	transformation from CRF to TRF
<b>R</b>	double	reference radius for ellipsoidal coordinates
<b>inverseFlattening</b>	double	reference flattening for ellipsoidal coordinates

### 4.11.6 Orbit2Kepler

This program computes the osculating Keplerian elements from position and velocity of a given **inputfileOrbit**. The **inputfileOrbit** must contain positions and velocities (see **OrbitAddVelocityAndAcceleration**).

The **outputfileKepler** is an **instrument file** (MISCVALUES) with the Keplerian elements at each epoch in the following order

- Ascending Node  $\Omega$  [degree]
- Inclination  $i$  [degree]
- Argument of perigee  $\omega$  [degree]
- major axis  $a$  [m]
- eccentricity  $e$
- mean anomaly  $M$  [degree]
- transit time of perigee  $\tau$  [mjd]

The data of **inputfileInstrument** are appended as values to each epoch.

Name	Type	Annotation
<b>outputfileKepler</b>	filename	instrument file (MISCVALUES: Omega, i, omega [degree], a [m], e, M [degree], tau [mjd], ...)
<b>inputfileOrbit</b>	filename	position and velocity at each epoch define the kepler orbit
<b>inputfileInstrument</b>	filename	data is appended
<b>GM</b>	double	Geocentric gravitational constant

This program is [parallelized \(1.5\)](#).

#### 4.11.7 Orbit2MagneticField

This program computes the magentic field vector( $x, y, z$  [ $Tesla = kg/A/s^2$ ] in CRF)) along an **Orbit** and writes it as **instrument file** (MISCVALUES). The data of **inputfileInstrument** are appended as data columns to each epoch.

Name	Type	Annotation
outfileMagneticField	filename	instrument file (x,y,z in CRF [ $Tesla = kg/A/s^2$ ]), ...)
infileOrbit	filename	
infileInstrument	filename	data are appended to output file
magnetosphere	magnetosphere	
earthRotation	earthRotation	

This program is parallelized (1.5).

#### 4.11.8 Orbit2ThermosphericState

This program computes the thermospheric state (density, temperature, wind (x,y,z in CRF)) based on empirical models along an [Orbit](#) and writes it as [Instrument file](#) (MISCVALUES). The wind is given in an celestial reference frame (CRF). The data of [inputfileInstrument](#) are appended as values to each epoch.

Name	Type	Annotation
<a href="#">outputfileThermosphericState</a>	filename	instrument file (MISCVALUES: density, temperature, wind (x,y,z in CRF), ...)
<a href="#">inputfileOrbit</a>	filename	
<a href="#">inputfileInstrument</a>	filename	data are appended to output file
<a href="#">thermosphere</a>	<a href="#">thermosphere</a>	
<a href="#">earthRotation</a>	<a href="#">earthRotation</a>	

This program is [parallelized](#) (1.5).

#### 4.11.9 OrbitAddVelocityAndAcceleration

This program computes velocities and accelerations from a given `Orbit` by differentiating a moving polynomial. The values are saved in one output file which then contains orbit, velocity and acceleration.

Name	Type	Annotation
outputfileOrbit	filename	
inputfileOrbit	filename	
polynomialDegree	uint	Polynomial degree, must be even!

This program is parallelized (1.5).

### 4.11.10 PlanetOrbit

Creates an **orbit file** of sun, moon, or planets. The orbit is given in the celestial reference frame (CRF) or alternatively in the terrestrial reference frame (TRF) if **earthRotation** is provided.

Name	Type	Annotation
outfileOrbit	filename	
planet	planet	
timeSeries	timeSeries	
ephemerides	ephemerides	
earthRotation	earthRotation	transform orbits into TRF

## 4.12 Programs: Plot

### 4.12.1 PlotDegreeAmplitudes

Plot degree amplitudes of potential coefficients computed by **Gravityfield2DegreeAmplitudes** or **PotentialCoefficients2DegreeAmplitudes** using the GMT Generic Mapping Tools (<https://www.generic-mapping-tools.org>). A variety of image file formats are supported (e.g. png, jpg, eps) determined by the extension of **outputfile**. This is a convenience program with meaningful default values. The same plots can be generated with the more general **PlotGraph**.

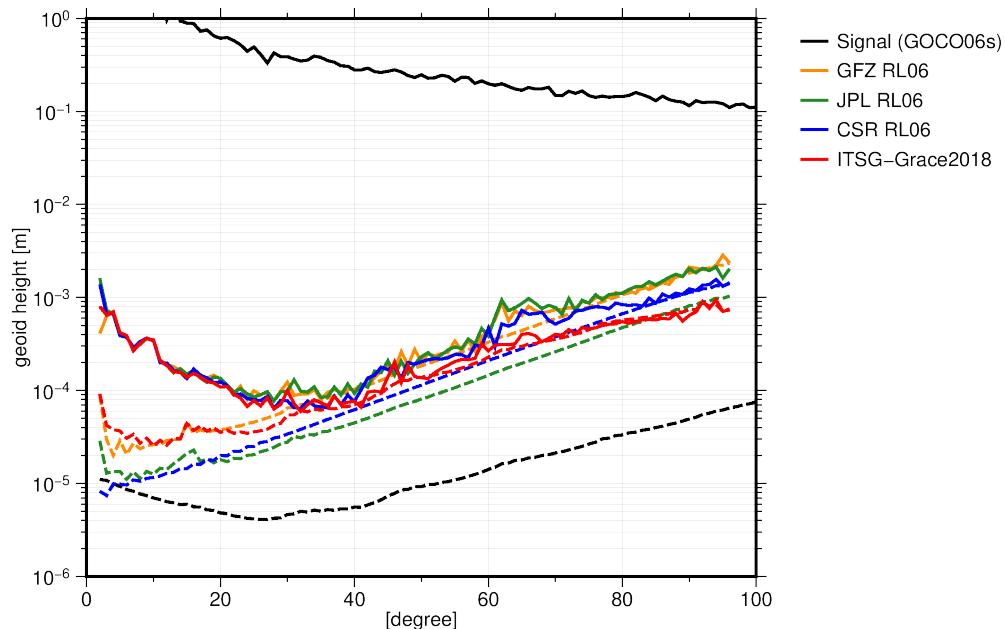


Figure 4.22: Comparison of GRACE solutions (2008-06) with GOCO06s.

Name	Type	Annotation
outputfile	filename	*.png, *.jpg, *.eps, ...
title	string	
layer	plotGraphLayer	
minDegree	uint	
maxDegree	uint	
majorTickSpacingDegree	double	boundary annotation
minorTickSpacingDegree	double	frame tick spacing
gridLineSpacingDegree	double	gridline spacing
labelDegree	string	description of the x-axis
logarithmicDegree	boolean	use logarithmic scale for the x-axis
minY	double	
maxY	double	
majorTickSpacingY	double	boundary annotation
minorTickSpacingY	double	frame tick spacing
gridLineSpacingY	double	gridline spacing
unitY	string	appended to axis values
labelY	string	description of the y-axis
logarithmicY	boolean	use logarithmic scale for the y-axis
gridLine	plotLine	The style of the grid lines.
legend	plotLegend	

---

options	sequence	further options...
width	double	in cm
height	double	in cm
titleFontSize	uint	in pt
marginTitle	double	between title and figure [cm]
drawGridOnTop	boolean	grid lines above all other lines/points
options	string	
transparent	boolean	make background transparent
dpi	uint	use this resolution when rasterizing postscript file
removeFiles	boolean	remove .gmt and script files

---

### 4.12.2 PlotGraph

Generates a two dimensional xy plot using the GMT Generic Mapping Tools (<https://www.generic-mapping-tools.org>). A variety of image file formats are supported (e.g. png, jpg, eps) determined by the extension of **outfile**.

The plotting area is defined by the two axes **axisX/Y**. An alternative **axisY2** on the right hand side can be added. The content of the graph itself is defined by one or more **layers**.

The plot programs create a temporary directory in the path of **outfile**, writes all needed data into it, generates a batch/shell script with the GMT commands, execute it, and remove the temporary directory. With setting **options:removeFiles=false** the last step is skipped and it is possible to adjust the plot manually to specific publication needs. Individual GMT settings are adjusted with **options:options="FORMAT=value"**, see <https://docs.generic-mapping-tools.org/latest/gmt.conf.html>.

See also: **PlotDegreeAmplitudes**, **PlotMap**, **PlotMatrix**, **PlotSphericalHarmonicsTriangle**.

Name	Type	Annotation
<b>outfile</b>	filename	*.png, *.jpg, *.eps, ...
<b>title</b>	string	
<b>layer</b>	plotGraphLayer	
<b>axisX</b>	plotAxis	
<b>axisY</b>	plotAxis	
<b>axisY2</b>	plotAxis	Second y-axis on right hand side
<b>colorbar</b>	plotColorbar	
<b>legend</b>	plotLegend	
<b>options</b>	sequence	further options...
<b>width</b>	double	in cm
<b>height</b>	double	in cm
<b>titleFontSize</b>	uint	in pt
<b>marginTitle</b>	double	between title and figure [cm]
<b>drawGridOnTop</b>	boolean	grid lines above all other lines/points
<b>options</b>	string	
<b>transparent</b>	boolean	make background transparent
<b>dpi</b>	uint	use this resolution when rasterizing postscript file
<b>removeFiles</b>	boolean	remove .gmt and script files

### 4.12.3 PlotMap

Generates a map using the GMT Generic Mapping Tools (<https://www.generic-mapping-tools.org>). A variety of image file formats are supported (e.g. png, jpg, eps) determined by the extension of **outfile**.

The base map is defined by a **projection** of an ellipsoid (**R**, **inverseFlattening**). The content of the map itself is defined by one or more **layers**.

The plot programs create a temporary directory in the path of **outfile**, writes all needed data into it, generates a batch/shell script with the GMT commands, execute it, and remove the temporary directory. With setting **options:removeFiles=false** the last step is skipped and it is possible to adjust the plot manually to specific publication needs. Individual GMT settings are adjusted with **options:options="FORMAT=value"**, see <https://docs.generic-mapping-tools.org/latest/gmt.conf.html>.

See also: **PlotDegreeAmplitudes**, **PlotGraph**, **PlotMatrix**,  
**PlotSphericalHarmonicsTriangle**.

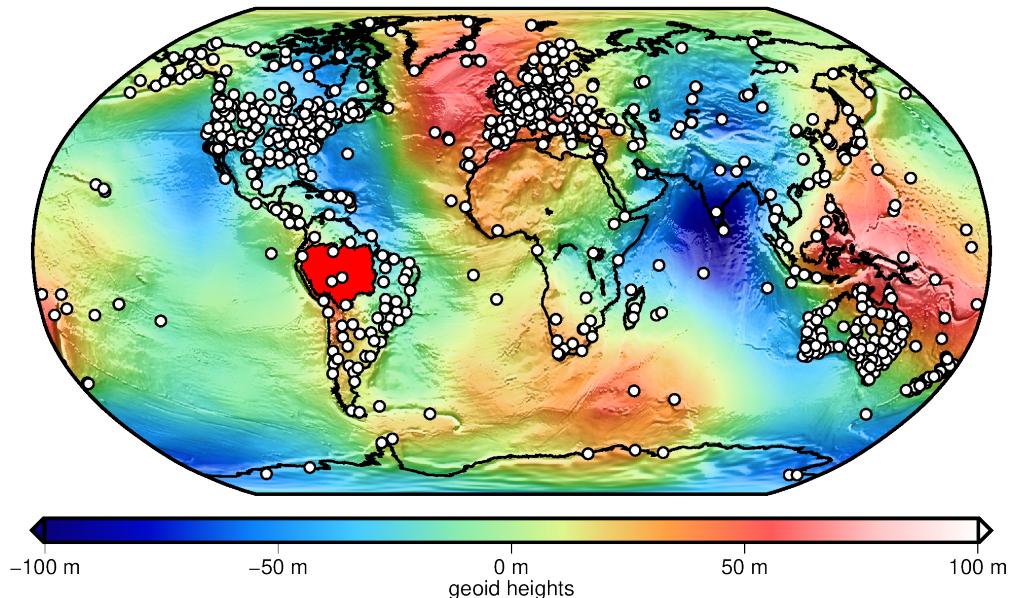


Figure 4.23: A Robinson projection with griddedData (geoid), coast, polygon (amazon), and points (IGS stations) layer.

Name	Type	Annotation
<b>outfile</b>	filename	*.png, *.jpg, *.eps, ...
<b>title</b>	string	
<b>statisticInfos</b>	boolean	
<b>layer</b>	<b>plotMapLayer</b>	
<b>R</b>	double	reference radius for ellipsoidal coordinates on output
<b>inverseFlattening</b>	double	reference flattening for ellipsoidal coordinates on output, 0: spherical coordinates
<b>minLambda</b>	angle	min. longitude (default: compute from input data)
<b>maxLambda</b>	angle	max. longitude (default: compute from input data)
<b>minPhi</b>	angle	min. latitude (default: compute from input data)
<b>maxPhi</b>	angle	max. latitude (default: compute from input data)
<b>majorTickSpacing</b>	angle	boundary annotation
<b>minorTickSpacing</b>	angle	frame tick spacing

gridLineSpacing	angle	gridline spacing
colorbar	plotColorbar	
projection	plotMapProjection	map projection
options	sequence	further options...
width	double	in cm
height	double	in cm
titleFontSize	uint	in pt
marginTitle	double	between title and figure [cm]
drawGridOnTop	boolean	grid lines above all other lines/points
options	string	
transparent	boolean	make background transparent
dpi	uint	use this resolution when rasterizing postscript file
removeFiles	boolean	remove .gmt and script files

---

#### 4.12.4 PlotMatrix

Plot the coefficients of a **inputfileMatrix** using the GMT Generic Mapping Tools (<https://www.generic-mapping-tools.org>). A variety of image file formats are supported (e.g. png, jpg, eps) determined by the extension of **outfile**.

The plot programs create a temporary directory in the path of **outfile**, writes all needed data into it, generates a batch/shell script with the GMT commands, execute it, and remove the temporary directory. With setting **options:removeFiles=false** the last step is skipped and it is possible to adjust the plot manually to specific publication needs. Individual GMT settings are adjusted with **options:options="FORMAT=value"**, see <https://docs.generic-mapping-tools.org/latest/gmt.conf.html>.

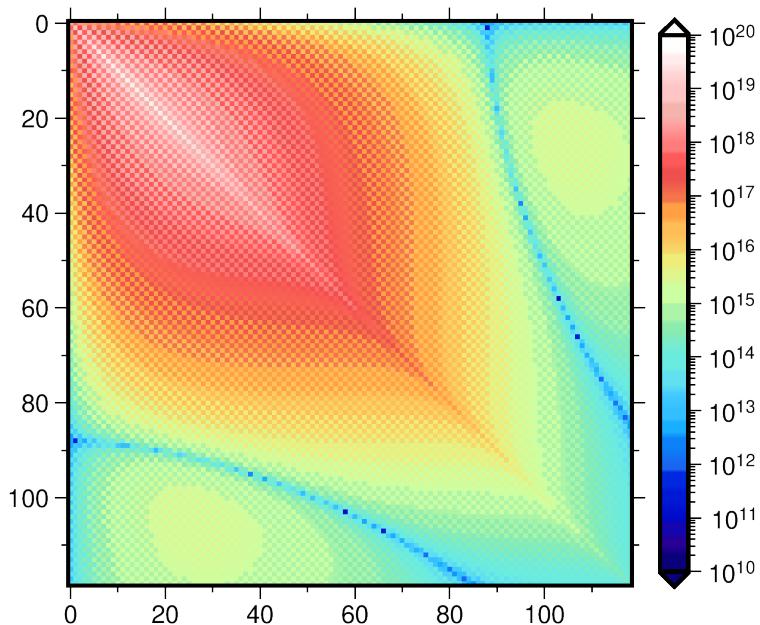


Figure 4.24: Upper left part of the DDK filter matrix.

Name	Type	Annotation
<b>outfile</b>	filename	*.png, *.jpg, *.eps, ...
<b>title</b>	string	
<b>inputfileMatrix</b>	filename	
<b>minColumn</b>	uint	minimum column index to plot
<b>maxColumn</b>	uint	maximum column index to plot
<b>majorTickSpacingX</b>	double	boundary annotation
<b>minorTickSpacingX</b>	double	frame tick spacing
<b>gridLineSpacingX</b>	double	gridline spacing
<b>minRow</b>	uint	minimum row index to plot
<b>maxRow</b>	uint	maximum row index to plot
<b>majorTickSpacingY</b>	double	boundary annotation
<b>minorTickSpacingY</b>	double	frame tick spacing
<b>gridLineSpacingY</b>	double	gridline spacing
<b>gridLine</b>	plotLine	The style of the grid lines.
<b>colorbar</b>	plotColorbar	
<b>options</b>	sequence	further options...
<b>width</b>	double	in cm
<b>height</b>	double	in cm

titleFontSize	uint	in pt
marginTitle	double	between title and figure [cm]
drawGridOnTop	boolean	grid lines above all other lines/points
options	string	
transparent	boolean	make background transparent
dpi	uint	use this resolution when rasterizing postscript file
removeFiles	boolean	remove .gmt and script files

---

### 4.12.5 PlotSphericalHarmonicsTriangle

Plot the potential coefficients of a spherical harmonic expansion using the GMT Generic Mapping Tools (<https://www.generic-mapping-tools.org>). A variety of image file formats are supported (e.g. png, jpg, eps) determined by the extension of **outfile**.

This program plots the formal errors (sigmas). If **gravityfield** provides no sigmas e.g. with **setSigmasToZero** in **gravityfield:potentialCoefficients** the coefficients itself are plotted instead.

The plot programs create a temporary directory in the path of **outfile**, writes all needed data into it, generates a batch/shell script with the GMT commands, execute it, and remove the temporary directory. With setting **options:removeFiles=false** the last step is skipped and it is possible to adjust the plot manually to specific publication needs. Individual GMT settings are adjusted with **options:options="FORMAT=value"**, see <https://docs.generic-mapping-tools.org/latest/gmt.conf.html>.

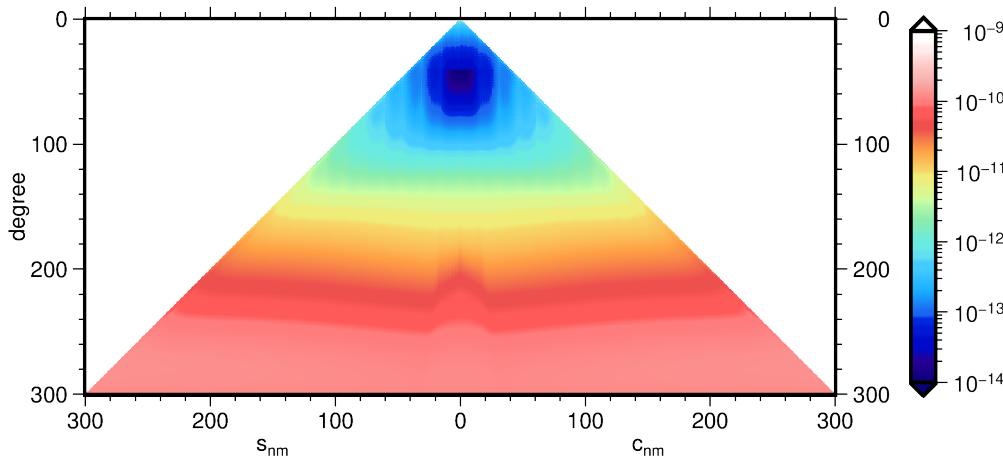


Figure 4.25: Formal errors of GOCO06s.

Name	Type	Annotation
<b>outfile</b>	filename	*.png, *.jpg, *.eps, ...
<b>title</b>	string	
<b>gravityfield</b>	<b>gravityfield</b>	use sigmas, if not given use signal (cnm,snm) at this time the gravity field will be evaluated
<b>time</b>	time	
<b>minDegree</b>	uint	
<b>maxDegree</b>	uint	
<b>majorTickSpacing</b>	double	boundary annotation
<b>minorTickSpacing</b>	double	frame tick spacing
<b>gridLineSpacing</b>	double	gridline spacing
<b>gridLine</b>	<b>plotLine</b>	The style of the grid lines.
<b>colorbar</b>	<b>plotColorbar</b>	
<b>options</b>	sequence	further options...
<b>width</b>	double	in cm
<b>height</b>	double	in cm
<b>titleFontSize</b>	uint	in pt
<b>marginTitle</b>	double	between title and figure [cm]
<b>drawGridOnTop</b>	boolean	grid lines above all other lines/points
<b>options</b>	string	
<b>transparent</b>	boolean	make background transparent
<b>dpi</b>	uint	use this resolution when rasterizing postscript file
<b>removeFiles</b>	boolean	remove .gmt and script files

## 4.13 Programs: Preprocessing

### 4.13.1 PreprocessingDualSst

This program processes satellite-to-satellite-tracking (SST) and orbit observations in a GRACE like configuration. Four different observation groups are considered separately: two types of SST and POD1/POD2 for the two satellites. This program works similar to [PreprocessingSst](#), see there for details. Here only the settings explained, which are different.

Both SST observation types are reduced by the same background models and the same impact of accelerometer measurements. The covariance matrix of the reduced observations should not consider the instrument noise only ([covarianceSst1/2](#)) but must take the cross correlations [covarianceAcc](#) into account. The covariance matrix of the reduced observations is given by

$$\Sigma \begin{pmatrix} \Delta l_{SST1} \\ \Delta l_{SST2} \end{pmatrix} = \begin{bmatrix} \Sigma_{SST1} + \Sigma_{ACC} & \Sigma_{ACC} \\ \Sigma_{ACC} & \Sigma_{SST2} + \Sigma_{ACC} \end{bmatrix}. \quad (4.63)$$

Name	Type	Annotation
▶ outputfileSolution	filename	estimated parameter vector (static part only)
▶ outputfileSigmax	filename	standard deviations of the parameters (sqrt of the diagonal of the inverse normal equation)
▶ outputfileParameterName	filename	estimated signal parameters (index is appended)
▶ estimateArcSigmas	sequence	
└▶ outputfileSigmasPerArcSst1	filename	accuracies of each arc (SST1)
└▶ outputfileSigmasPerArcSst2	filename	accuracies of each arc (SST2)
└▶ outputfileSigmasPerArcAcc	filename	accuracies of each arc (ACC)
└▶ outputfileSigmasPerArcPod1	filename	accuracies of each arc (POD1)
└▶ outputfileSigmasPerArcPod2	filename	accuracies of each arc (POD2)
▶ estimateEpochSigmas	sequence	
└▶ outputfileSigmasPerEpochSst1	filename	accuracies of each epoch (SST1)
└▶ outputfileSigmasPerEpochSst2	filename	accuracies of each epoch (SST2)
└▶ outputfileSigmasPerEpochAcc	filename	accuracies of each epoch (ACC)
└▶ outputfileSigmasPerEpochPod1	filename	accuracies of each epoch (POD1)
└▶ outputfileSigmasPerEpochPod2	filename	accuracies of each epoch (POD2)
▶ estimateCovarianceFunctions	sequence	
└▶ outputfileCovarianceFunctionSst1	filename	covariance function
└▶ outputfileCovarianceFunctionSst2	filename	covariance function
└▶ outputfileCovarianceFunctionAcc	filename	covariance function

 <code>outputfileCovarianceFunction-Pod1</code>	<code>filename</code>	covariance functions for along, cross, radial direction
 <code>outputfileCovarianceFunction-Pod2</code>	<code>filename</code>	covariance functions for along, cross, radial direction
 <code>computeResiduals</code>	<code>sequence</code>	
 <code>outputfileSst1Residuals</code>	<code>filename</code>	
 <code>outputfileSst2Residuals</code>	<code>filename</code>	
 <code>outputfileAccResiduals</code>	<code>filename</code>	
 <code>outputfilePod1Residuals</code>	<code>filename</code>	
 <code>outputfilePod2Residuals</code>	<code>filename</code>	
 <code>observation</code>	<code>choice</code>	observation equations (Sst)
 <code>dualSstVariational</code>	<code>sequence</code>	two SST observations
 <code>rightHandSide</code>	<code>sequence</code>	input for observation vectors
 <code>inputfileSatelliteTracking1</code>	<code>filename</code>	ranging observations and corrections
 <code>inputfileSatelliteTracking2</code>	<code>filename</code>	ranging observations and corrections
 <code>inputfileOrbit1</code>	<code>filename</code>	kinematic positions of satellite A as observations
 <code>inputfileOrbit2</code>	<code>filename</code>	kinematic positions of satellite B as observations
 <code>sstType</code>	<code>choice</code>	
 <code>range</code>		
 <code>rangeRate</code>		
 <code>none</code>		
 <code>inputfileVariational1</code>	<code>filename</code>	approximate position and integrated state matrix
 <code>inputfileVariational2</code>	<code>filename</code>	approximate position and integrated state matrix
 <code>ephemerides</code>	<code>ephemerides</code>	gravity field parametrization
 <code>parametrizationGravity</code>	<code>parametrizationGravity</code>	
 <code>parametrizationAcceleration1</code>	<code>parametrizationAcceleration</code>	orbit1 force parameters
 <code>parametrizationAcceleration2</code>	<code>parametrizationAcceleration</code>	orbit2 force parameters
 <code>parametrizationSst1</code>	<code>parametrizationSatelliteTracking</code>	satellite tracking parameter for first ranging observations
 <code>parametrizationSst2</code>	<code>parametrizationSatelliteTracking</code>	satellite tracking parameter for second ranging observations
 <code>integrationDegree</code>	<code>uint</code>	integration of forces by polynomial approximation of degree n
 <code>interpolationDegree</code>	<code>uint</code>	orbit interpolation by polynomial approximation of degree n
 <code>covarianceSst1</code>	<code>sequence</code>	apriori factor of covariance function
 <code>sigma</code>	<code>double</code>	

 <b>inputfileSigmasPerArc</b>	filename	apriori different accuracies for each arc (multiplicated with sigma)
 <b>inputfileSigmasPerEpoch</b>	filename	apriori different accuracies for each epoch
 <b>inputfileCovarianceFunction</b>	filename	approximate covariances in time
 <b>inputfileCovarianceMatrixArc</b>	filename	Must be given per sst arc with correct dimensions.
 <b>inputfileSigmasCovarianceMatrixArc</b>	filename	Vector with one sigma for each <code>\$ \$inputfileCovarianceMatrixArc\$; \$</code> [seconds] sampling of the covariance function
 <b>sampling</b>	double	
 <b>covarianceSst2</b>	sequence	apriori factor of covariance function
 <b>sigma</b>	double	apriori different accuracies for each arc (multiplicated with sigma)
 <b>inputfileSigmasPerArc</b>	filename	apriori different accuracies for each epoch
 <b>inputfileSigmasPerEpoch</b>	filename	approximate covariances in time
 <b>inputfileCovarianceFunction</b>	filename	Must be given per sst arc with correct dimensions.
 <b>inputfileCovarianceMatrixArc</b>	filename	Vector with one sigma for each <code>\$ \$inputfileCovarianceMatrixArc\$; \$</code> [seconds] sampling of the covariance function
 <b>inputfileSigmasCovarianceMatrixArc</b>	filename	apriori factor of covariance function
 <b>sampling</b>	double	apriori different accuracies for each arc (multiplicated with sigma)
 <b>covarianceAcc</b>	sequence	apriori different accuracies for each epoch
 <b>sigma</b>	double	approximate covariances in time
 <b>inputfileSigmasPerArc</b>	filename	Must be given per sst arc with correct dimensions.
 <b>inputfileSigmasPerEpoch</b>	filename	Vector with one sigma for each <code>\$ \$inputfileCovarianceMatrixArc\$; \$</code> [seconds] sampling of the covariance function
 <b>inputfileCovarianceFunction</b>	filename	apriori factor of covariance function
 <b>inputfileCovarianceMatrixArc</b>	filename	apriori different accuracies for each epoch
 <b>inputfileSigmasCovarianceMatrixArc</b>	filename	approximate covariances in time
 <b>sampling</b>	double	Must be given per sst arc with correct dimensions.
 <b>covariancePod1</b>	sequence	Vector with one sigma for each <code>\$ \$inputfileCovarianceMatrixArc\$; \$</code> [seconds] sampling of the covariance function
 <b>sigma</b>	double	apriori factor of covariance function

 <code>inputfileSigmasPerArc</code>	filename	apriori different accuracies for each arc (multiplied with sigma)
 <code>inputfileSigmasPerEpoch</code>	filename	apriori different accuracies for each epoch
 <code>inputfileCovarianceFunction</code>	filename	approximate covariances in time
 <code>inputfileCovariancePodEpoch</code>	filename	3x3 epoch covariances
 <code>sampling</code>	double	[seconds] sampling of the covariance function
 <code>covariancePod2</code>	sequence	apriori factor of covariance function
 <code>sigma</code>	double	
 <code>inputfileSigmasPerArc</code>	filename	apriori different accuracies for each arc (multiplied with sigma)
 <code>inputfileSigmasPerEpoch</code>	filename	apriori different accuracies for each epoch
 <code>inputfileCovarianceFunction</code>	filename	approximate covariances in time
 <code>inputfileCovariancePodEpoch</code>	filename	3x3 epoch covariances
 <code>sampling</code>	double	[seconds] sampling of the covariance function
 <code>estimateShortTimeVariations</code>	sequence	co-estimate short time gravity field variations
 <code>estimateSigma</code>	boolean	estimate standard deviation via VCE
 <code>autoregressiveModelSequence</code>	autoregressiveModelSequence	AR model sequence for constraining short time gravity variations
 <code>parameterSelection</code>	parameterSelector	parameters describing the short time gravity field
 <code>downweightPod</code>	double	downweight factor for POD
 <code>inputfileArcList</code>	filename	list to correspond points of time to arc numbers
 <code>iterationCount</code>	uint	(maximum) number of iterations for the estimation of calibration parameter and error PSD
 <code>variableNameIterations</code>	string	All output fileNames in pre-processing iteration are expanded with this variable prior to writing to disk
 <code>defaultBlockSize</code>	uint	block size of static normal equation blocks

This program is [parallelized \(1.5\)](#).

### 4.13.2 PreprocessingGradiometer

This program estimates empirical covariance functions of the gradiometer instrument noise and determine arc wise variances to downweight arcs with outliers. This program works similar to [PreprocessingPod](#), see there for details. Here only the settings explained, which are different.

...

Name	Type	Annotation
▶ outputFileCovarianceFunction	filename	
▶ outputFileSigmasPerArc	filename	accuracies of each arc
▶ outputFileSggResiduals	filename	
▶ rightHandSide	sggRightSide	input for the observation vector
▶ inputFileOrbit	filename	
▶ inputFileStarCamera	filename	
▶ earthRotation	earthRotation	
▶ ephemerides	ephemerides	
▶ parametrizationBias	parametrizationTemporal	per arc
▶ covarianceSgg	sequence	
└▶ inputFileCovarianceFunction	filename	approximate covariances in time
└▶ covarianceLength	uint	counts observation epochs
└▶ sampling	double	[seconds] sampling of the covariance function
▶ iterationCount	uint	for the estimation of calibration parameter and error PSD

This program is parallelized (1.5).

### 4.13.3 PreprocessingPod

This program estimates empirical covariance functions of the instrument noise and determine arc wise variances to downweight arc with outliers.

A complete least squares adjustment for gravity field determination is performed by computing the **observation** equations, see **observation:podIntegral** or **observation:podVariational** for details. The normal equations are accumulated and solved to **outputfileSolution** together with the estimated accuracies **outputfileSigmax**. The estimated residuals  $\hat{e} = \mathbf{l} - \mathbf{A}\hat{\mathbf{x}}$  can be computed with **computeResiduals**.

For each component (along, cross, radial) of the kinematic orbit positions a noise covariance function is estimated

$$\text{cov}(\Delta t_i) = \sum_{n=0}^{N-1} a_n^2 \cos\left(\frac{\pi}{T} n \Delta t_i\right). \quad (4.64)$$

The covariance matrix is composed of the sum of matrices  $F_n$  and unknown variance factors

$$\Sigma = a_1^2 \mathbf{F}_1 + a_2^2 \mathbf{F}_2 + \cdots + a_N^2 \mathbf{F}_N, \quad (4.65)$$

with the cosine transformation matrices

$$\mathbf{F}_n = \left( \cos\left(\frac{\pi}{T} n(t_i - t_k)\right) \right)_{ik}. \quad (4.66)$$

An additional variance factor can be computed (**estimateArcSigmas**) for each arc  $k$  according to

$$\hat{\sigma}_k^2 = \frac{\hat{\mathbf{e}}_k^T \Sigma^{-1} \hat{\mathbf{e}}_k}{r_k}, \quad (4.67)$$

where  $r_k$  is the redundancy. This variance factor should be around one for normal behaving arcs as the noise characteristics is already considered by the covariance matrix but bad arcs get a much larger variance. By applying this factor bad arcs or arcs with large outliers are downweighted.

Name	Type	Annotation
<b>outputfileSolution</b>	filename	estimated parameter vector (static part only)
<b>outputfileSigmax</b>	filename	standard deviations of the parameters (sqrt of the diagonal of the inverse normal equation)
<b>outputfileParameterName</b>	filename	names of estimated parameters (static part only)
<b>estimateArcSigmas</b>	sequence	
<b>outputfileSigmasPerArcPod</b>	filename	accuracies of each arc (POD2)
<b>estimateCovarianceFunctions</b>	sequence	
<b>outputfileCovarianceFunctionPod</b>	filename	covariance functions for along, cross, radial direction
<b>computeResiduals</b>	sequence	
<b>outputfilePodResiduals</b>	filename	
<b>observation</b>	choice	observation equations (POD)
<b>podIntegral</b>	sequence	precise orbit data (integral approach)
<b>inputfileSatelliteModel</b>	filename	satellite macro model
<b>rightHandSide</b>	<b>podRightSide</b>	input for the reduced observation vector

  <b>inputfileOrbit</b>	filename	used to evaluate the observation equations, not used as observations
  <b>inputfileStarCamera</b>   <b>earthRotation</b>   <b>ephemerides</b>   <b>gradientfield</b>	filename <b>earthRotation</b> <b>ephemerides</b> <b>gravityfield</b>	low order field to estimate the change of the gravity by position adjustement
  <b>parametrizationGravity</b>   <b>parametrizationAcceleration</b>   <b>keepSatelliteStates</b>	<b>parametrizationGravity</b> <b>parametrizationAcceleration</b> boolean	gravity field parametrization orbit force parameters set boundary values of each arc global
  <b>integrationDegree</b>	uint	integration of forces by polynomial approximation of degree n
  <b>interpolationDegree</b>	uint	orbit interpolation by polynomial approximation of degree n
  <b>accelerateComputation</b>	boolean	acceleration of computation by transforming the observations
  <b>podVariational</b>	sequence	precise orbit data (variational equations)
  <b>rightHandSide</b>   <b>inputfileOrbit</b>	sequence filename	input for observation vectors kinematic positions as observations
  <b>inputfileVariational</b>	filename	approximate position and integrated state matrix
  <b>ephemerides</b>   <b>parametrizationGravity</b>   <b>parametrizationAcceleration</b>   <b>integrationDegree</b>	<b>ephemerides</b> <b>parametrizationGravity</b> <b>parametrizationAcceleration</b> uint	gravity field parametrization orbit force parameters integration of forces by polynomial approximation of degree n orbit interpolation by polynomial approximation of degree n
  <b>interpolationDegree</b>	uint	acceleration of computation by transforming the observations
  <b>accelerateComputation</b>	boolean	
  <b>covariancePod</b>   <b>sigma</b>	sequence double	apriori factor of covariance function
  <b>inputfileSigmasPerArc</b>	filename	apriori different accuracies for each arc (multipllicated with sigma)
  <b>inputfileCovarianceFunction</b>   <b>inputfileCovariancePodEpoch</b>   <b>sampling</b>	filename filename double	approximate covariances in time 3x3 epoch covariances [seconds] sampling of the covariance function
  <b>inputfileArcList</b>	filename	list to correspond points of time to arc numbers
  <b>adjustmentThreshold</b>	double	Adjustment factor threshold: Iteration will be stopped once both SST and POD adjustment factors are under this threshold
  <b>iterationCount</b>	uint	(maximum) number of iterations for the estimation of calibration parameter and error PSD

This program is [parallelized \(1.5\)](#).

#### 4.13.4 PreprocessingSst

This program processes satellite-to-satellite-tracking (SST) and kinematic orbit observations in a GRACE like configuration. Three different observation groups are considered separately: SST and POD1/POD2 for the two satellites. This program works similar to [PreprocessingPod](#), see there for details. Here only deviations in the settings are explained.

Precise orbit data (POD) often contains systematic errors in addition to stochastic noise. In this case the variance component estimation fails and assigns too much weight to the POD data. Therefore an additional **downweightPod** factor can be applied to the standard deviation of POD for the next least squares adjustment in the iteration. This factor should also be applied as **sigma** in **observation** for computation of the final solution e.g. with [NormalsSolverVCE](#).

Short time variations of the gravity field can be co-estimated together with the static/monthly mean gravity field. The short time parameters must also be set in **observation:parametrizationGravity** and can then be selected by **estimateShortTimeVariations:parameterSelection**. If these parameters are not time variable, for example when a range of static parameters is selected, they are set up as constant for each time interval defined in **inputfileArcList**. The parameters are constrained by an **estimateShortTimeVariations:autoregressiveModelSequence**. The weight of the constrain equations in terms of the standard deviation can be estimated by means of Variance Component Estimation (VCE) if **estimateShortTimeVariations:estimateSigma** is set. The mathematical background of this co-estimation can be found in:

Kvas, A., Mayer-Gürr, T. GRACE gravity field recovery with background model uncertainties. *J Geod* 93, 2543–2552 (2019). <https://doi.org/10.1007/s00190-019-01314-1>.

Name	Type	Annotation
<b>outputfileSolution</b>	filename	estimated parameter vector (static part only)
<b>outputfileSigmax</b>	filename	standard deviations of the parameters (sqrt of the diagonal of the inverse normal equation)
<b>outputfileParameterName</b>	filename	estimated signal parameters (index is appended)
<b>estimateArcSigmas</b>	sequence	accuracies of each arc (SST)
<b>outputfileSigmasPerArcSst</b>	filename	accuracies of each arc (POD1)
<b>outputfileSigmasPerArcPod1</b>	filename	accuracies of each arc (POD2)
<b>outputfileSigmasPerArcPod2</b>	filename	
<b>estimateEpochSigmas</b>	sequence	accuracies of each epoch (SST)
<b>outputfileSigmasPerEpochSst</b>	filename	accuracies of each epoch (POD1)
<b>outputfileSigmasPerEpochPod1</b>	filename	accuracies of each epoch (POD2)
<b>outputfileSigmasPerEpochPod2</b>	filename	
<b>estimateCovarianceFunctions</b>	sequence	covariance function
<b>outputfileCovarianceFunctionSst</b>	filename	covariance functions for along, cross, radial direction
<b>outputfileCovarianceFunctionPod1</b>	filename	covariance functions for along, cross, radial direction
<b>outputfileCovarianceFunctionPod2</b>	filename	

 estimateSstArcCovarianceSigmas	sequence	
 outputfileSigmasCovarianceMatrixArc	filename	one variance factor per matrix
 computeResiduals	sequence	
 outputfileSstResiduals	filename	
 outputfilePod1Residuals	filename	
 outputfilePod2Residuals	filename	
 observation	choice	observation equations (Sst)
 sstIntegral	sequence	integral approach
 inputfileSatelliteModel1	filename	satellite macro model
 inputfileSatelliteModel2	filename	satellite macro model
 rightHandSide	sstRightSide	input for the reduced observation vector
  sstType	choice	
  range		
  rangeRate		
  rangeAcceleration		
  none		
  inputfileOrbit1	filename	used to evaluate the observation equations, not used as observations
  inputfileOrbit2	filename	used to evaluate the observation equations, not used as observations
  inputfileStarCamera1	filename	
  inputfileStarCamera2	filename	
  earthRotation	earthRotation	
  ephemerides	ephemerides	
  gradientfield	gravityfield	
  parametrizationGravity	parametrizationGravity	low order field to estimate the change of the gravity by position adjustement
  parametrizationAcceleration1	parametrizationAcceleration	gravity field parametrization
  parametrizationAcceleration2	parametrizationAcceleration	orbit1 force parameters
  parametrizationSst	parametrizationSatelliteTracking	orbit2 force parameters
  keepSatelliteStates	boolean	satellite tracking parameter set boundary values of each arc global
  integrationDegree	uint	integration of forces by polynomial approximation of degree n
  interpolationDegree	uint	orbit interpolation by polynomial approximation of degree n
  sstVariational	sequence	variational equations
  rightHandSide	sequence	input for observation vectors ranging observations and corrections
  inputfileSatelliteTracking	filename	kinematic positions of satellite A as observations
  inputfileOrbit1	filename	kinematic positions of satellite B as observations
  inputfileOrbit2	filename	
  sstType	choice	
  range		

 rangeRate		
 none		
 inputFileVariational1	filename	approximate position and integrated state matrix
 inputFileVariational2	filename	approximate position and integrated state matrix
 ephemerides	ephemerides	
 parametrizationGravity	parametrizationGravity	gravity field parametrization
 parametrizationAcceleration1	parametrizationAcceleration	orbit1 force parameters
 parametrizationAcceleration2	parametrizationAcceleration	orbit2 force parameters
 parametrizationSst	parametrizationSatelliteTracking	satellite tracking parameter
 integrationDegree	uint	integration of forces by polynomial approximation of degree n
 interpolationDegree	uint	orbit interpolation by polynomial approximation of degree n
 covarianceSst	sequence	
 sigma	double	apriori factor of covariance function
 inputFileSigmasPerArc	filename	apriori different accuracies for each arc (multiplied with sigma)
 inputFileSigmasPerEpoch	filename	apriori different accuracies for each epoch
 inputFileCovarianceFunction	filename	approximate covariances in time
 inputFileCovarianceMatrixArc	filename	Must be given per sst arc with correct dimensions.
 inputFileSigmasCovarianceMatrixArc	filename	Vector with one sigma for each \$;inputfileCovarianceMatrixArc\$; [seconds] sampling of the covariance function
 sampling	double	
 covariancePod1	sequence	apriori factor of covariance function
 sigma	double	
 inputFileSigmasPerArc	filename	apriori different accuracies for each arc (multiplied with sigma)
 inputFileSigmasPerEpoch	filename	apriori different accuracies for each epoch
 inputFileCovarianceFunction	filename	approximate covariances in time
 inputFileCovariancePodEpoch	filename	3x3 epoch covariances
 sampling	double	[seconds] sampling of the covariance function
 covariancePod2	sequence	apriori factor of covariance function
 sigma	double	
 inputFileSigmasPerArc	filename	apriori different accuracies for each arc (multiplied with sigma)

---

 <code>inputfileSigmasPerEpoch</code>	<code>filename</code>	apriori different accuraries for each epoch
 <code>inputfileCovarianceFunction</code>	<code>filename</code>	approximate covariances in time
 <code>inputfileCovariancePodEpoch</code>	<code>filename</code>	3x3 epoch covariances
 <code>sampling</code>	<code>double</code>	[seconds] sampling of the covariance function
 <code>estimateShortTimeVariations</code>	<code>sequence</code>	co-estimate short time gravity field variations
 <code>estimateSigma</code>	<code>boolean</code>	estimate standard deviation via VCE
 <code>autoregressiveModelSequence</code>	<code>autoregressiveModelSequence</code>	AR model sequence for constraining short time gravity variations
 <code>parameterSelection</code>	<code>parameterSelector</code>	parameters describing the short time gravity field
 <code>downweightPod</code>	<code>double</code>	downweight factor for POD
 <code>inputfileArcList</code>	<code>filename</code>	list to correspond points of time to arc numbers
 <code>iterationCount</code>	<code>uint</code>	(maximum) number of iterations for the estimation of calibration parameter and error PSD
 <code>variableNameIterations</code>	<code>string</code>	All output fileNames in pre-processing iteration are expanded with this variable prior to writing to disk
 <code>defaultBlockSize</code>	<code>uint</code>	block size of static normal equation blocks

---

This program is [parallelized \(1.5\)](#).

### 4.13.5 PreprocessingVariationalEquation

This program integrates an orbit dynamically using the given forces and setup the state transition matrix for each time step. These are the prerequisite for a least squares adjustment (e.g. gravity field determination) using the variational equation approach. The variational equations are computed arc wise as defined by **inputfileOrbit**. This means for each arc new initial state parameters are setup.

In a first step the **forces** acting on the satellite are evaluated at the apriori positions given by **inputfileOrbit**. Non-conservative forces like solar radiation pressure needs the orientation of the satellite (**inputfileStarCamera**) and additional a satellite macro model (**satelliteModel**) with the surface properties. Furthermore **inputfileAccelerometer** observations are also considered.

In a second step the accelerations are integrated twice to an dynamic orbit unsing a moving polynomial with the degree **integrationDegree**. The orbit is corrected to be self-consistent. This means the forces should be evaluated at the new integrated positions instead of the apriori ones. This correction is computed in a linear approximation using the gradient of the forces with respect to the positions (**gradientfield**). As this term is small generally only the largest force components has to be considered. A low degree spherical harmonic expansion of the static gravity field (about up to degree 5) is sufficient in almost all cases. In this step also the state transition matrix (the partial derivatrices of the current state, position and velocity) with respect to the initial state is computed. The integrated orbit together with the state transitions are stored in **outputfileVariational**, the integrated orbit only in **outputfileOrbit**.

To improve the numerical stability a reference ellipse can be reduced beforehand using Enke's method (**useEnke**). Mathematically the result is the same, but as the large central term is removed before and restored afterwards more digits are available for the computation.

The integrated orbit should be fitted to observations afterwards by the programs **PreprocessingVariationalEquationOrbitFit** and/or **PreprocessingVariationalEquationSstFit**. They apply a least squares adjustment by estimating some satellite parameters (e.g. an accelerometer bias). If the fitted orbit is to far away from the original **inputfileOrbit** the linearization may not be accurate enough. In this case **PreprocessingVariationalEquation** should be run again with the fitted orbit as **inputfileOrbit** and introducing the **estimatedParameters** as additional forces.

Name	Type	Annotation
<b>outputfileVariational</b>	filename	approximate position and integrated state matrix
<b>outputfileOrbit</b>	filename	integrated orbit
<b>inputfileSatelliteModel</b>	filename	satellite macro model
<b>inputfileOrbit</b>	filename	approximate position, used to evaluate the force
<b>inputfileStarCamera</b>	filename	rotation from body frame to CRF
<b>inputfileAccelerometer</b>	filename	non-gravitational forces in satellite reference frame
<b>forces</b>	<b>forces</b>	
<b>estimatedParameters</b>	sequence	satellite parameters e.g. from orbit fit
<b>parametrizationAcceleration</b>	<b>parametrizationAcceleration</b>	orbit force parameters
<b>inputfileParameter</b>	filename	estimated orbit force parameters
<b>earthRotation</b>	<b>earthRotation</b>	
<b>ephemerides</b>	<b>ephemerides</b>	
<b>gradientfield</b>	<b>gravityfield</b>	low order field to estimate the change of the gravity by position adjustement
<b>integrationDegree</b>	uint	integration of forces by polynomial approximation of degree n
<b>useEnke</b>	sequence	integrate differential forces to an elliptical reference trajectory



GM

double

geocentric gravitational constant used  
for elliptical reference orbit

---

This program is [parallelized \(1.5\)](#).

### 4.13.6 PreprocessingVariationalEquationOrbitFit

This program fits an **inputfileVariational** to an observed **inputfileOrbit** by estimating parameters in a least squares adjustment. Additional to the initial satellite state for each arc, these parameters can be **parametrizationGravity**, satellite **parametrizationAcceleration** and stochastic pulses (velocity jumps) at given times, **stochasticPulse**. The estimated parameters can be stored with **outputfileSolution** and an extra file with the parameter names is created. The fitted orbit is written as new reference in **outputfileVariational** and additionally in **outputfileOrbit**.

The observed orbit positions (**inputfileOrbit**) together with the epoch wise covariance matrix (**inputfileCovariancePodEpoch**) must be splitted in the same arcs as the variational equations but not necessarily uniform distributed (use irregularData in **InstrumentSynchronize**). An iterative down-weighting of outliers is performed by M-Huber method.

The observation equations (parameter sensitivity matrix) are computed by integration of the variational equations (**inputfileVariational**) using a polynomial with **integrationDegree** and interpolated to the observation epochs using a polynomial with **interpolationDegree**.

All parameters used here must be reestimated in the full least squares adjustment for the gravity field determination to get a solution which is not biased towards the reference field. The solution of additional estimations are relative (deltas) as the parameters are already used as Taylor point in the reference orbit.

See also **PreprocessingVariationalEquation**.

Name	Type	Annotation
<b>outputfileVariational</b>	filename	approximate position and integrated state matrix
<b>outputfileOrbit</b>	filename	integrated orbit
<b>outputfileSolution</b>	filename	estimated calibration and state parameters
<b>inputfileVariational</b>	filename	approximate position and integrated state matrix
<b>inputfileOrbit</b>	filename	kinematic positions of satellite as observations
<b>inputfileCovariancePodEpoch</b>	filename	3x3 epoch wise covariances
<b>ephemerides</b>	ephemerides	may be needed by parametrizationAcceleration
<b>parametrizationGravity</b>	parametrizationGravity	gravity field parametrization
<b>parametrizationAcceleration</b>	parametrizationAcceleration	orbit force parameters
<b>stochasticPulse</b>	timeSeries	
<b>integrationDegree</b>	uint	integration of forces by polynomial approximation of degree n
<b>interpolationDegree</b>	uint	orbit interpolation by polynomial approximation of degree n
<b>iterationCount</b>	uint	for the estimation of calibration parameter and error PSD

This program is parallelized (1.5).

### 4.13.7 PreprocessingVariationalEquationSstFit

This program fits two **inputfileVariational1/2** to satellite-to-satellite-tracking (SST) and orbit observations in a GRACE like configuration. It works similar to **PreprocessingVariationalEquationOrbitFit**, see there for details.

As the relative weighting of the observation types is important complex description of the covariances can be set with **covarianceSst**, **covariancePod1**, **covariancePod2**.

Name	Type	Annotation
outputfileVariational1	filename	approximate position and integrated state matrix
outputfileVariational2	filename	approximate position and integrated state matrix
outputfileOrbit1	filename	integrated orbit
outputfileOrbit2	filename	integrated orbit
outputfileSolution1	filename	estimated calibration and state parameters
outputfileSolution2	filename	estimated calibration and state parameters
rightHandSide	sequence	input for observation vectors
inputfileSatelliteTracking	filename	ranging observations and corrections
inputfileOrbit1	filename	kinematic positions of satellite A as observations
inputfileOrbit2	filename	kinematic positions of satellite B as observations
sstType	choice	
range		
rangeRate		
none		
inputfileVariational1	filename	approximate position and integrated state matrix
inputfileVariational2	filename	approximate position and integrated state matrix
ephemerides	ephemerides	gravity field parametrization
parametrizationGravity	parametrizationGravity	orbit1 force parameters
parametrizationAcceleration1	parametrizationAcceleration	orbit2 force parameters
parametrizationAcceleration2	parametrizationAcceleration	satellite tracking parameter
parametrizationSst	parametrizationSatelliteTracking	integration of forces by polynomial approximation of degree n
integrationDegree	uint	orbit interpolation by polynomial approximation of degree n
interpolationDegree	uint	covariance matrix of satellite to satellite tracking observations
covarianceSst	covarianceSst	covariance matrix of kinematic orbits (satellite 1)
covariancePod1	covariancePod	covariance matrix of kinematic orbits (satellite 2)
covariancePod2	covariancePod	for the estimation of calibration parameter and error PSD
iterationCount	uint	

This program is [parallelized \(1.5\)](#).

## 4.14 Programs: Simulation

### 4.14.1 NoiseAccelerometer

This program adds noise and biases to simulated  accelerometer data generated by  **SimulateAccelerometer**. See  **noiseGenerator** for details on noise generation.

Name	Type	Annotation
 <b>outputfileAccelerometer</b>	filename	
 <b>inputfileAccelerometer</b>	filename	
 <b>biasAlong</b>	double	[m/s**2]
 <b>biasCross</b>	double	[m/s**2]
 <b>biasRadial</b>	double	[m/s**2]
 <b>noiseAlong</b>	<b>noiseGenerator</b>	[m/s**2]
 <b>noiseCross</b>	<b>noiseGenerator</b>	[m/s**2]
 <b>noiseRadial</b>	<b>noiseGenerator</b>	[m/s**2]

This program is parallelized (1.5).

### 4.14.2 NoiseGriddedData

This program adds noise to `gridded data data`. See `noiseGenerator` for details on noise generation.

Name	Type	Annotation
<code>outputfileGriddedData</code>	filename	
<code>inputfileGriddedData</code>	filename	
<code>noise</code>	<code>noiseGenerator</code>	
<code>startDataFields</code>	uint	start
<code>countDataFields</code>	uint	number of data fields (default: all after start)

### 4.14.3 NoiseInstrument

This program adds noise to [instrument data](#). See [noiseGenerator](#) for details on noise generation.

Name	Type	Annotation
 <code>outputfileInstrument</code>	filename	
 <code>inputfileInstrument</code>	filename	
 <code>noise</code>	<a href="#">noiseGenerator</a>	
 <code>startDataFields</code>	uint	start
 <code>countDataFields</code>	uint	number of data fields (default: all after start)

This program is parallelized (1.5).

#### 4.14.4 NoiseNormalsSolution

The inverse of the normal matrix of **inputfileNormalEquation** represents the covariance matrix of the estimated parameters. This program generates a noise vector with

$$\Sigma(\mathbf{e}) = \mathbf{N}^{-1}, \quad (4.68)$$

if generated input noise is standard white noise.

The noise vector is computed with

$$\mathbf{e} = \mathbf{W}^{-T} \mathbf{z}, \quad (4.69)$$

where **z** is the generated **noise** and **W** is the cholesky upper triangle matrix of the normal matrix  $\mathbf{N} = \mathbf{W}^T \mathbf{W}$ .

Name	Type	Annotation
outputfileNoise	filename	generated noise as matrix: parameterCount x sampleCount
inputfileNormalEquation	filename	
noise	noiseGenerator	
sampleCount	uint	number of samples to be generated
useEigenvalueDecomposition	boolean	use eigenvalue decomposition

This program is [parallelized \(1.5\)](#).

#### 4.14.5 NoiseOrbit

This program adds noise to simulated `satellite`'s positions and velocities generated by `SimulateOrbit` (along, cross, radial). See `noiseGenerator` for details on noise options.

Name	Type	Annotation
<code>outfileOrbit</code>	filename	
<code>infileOrbit</code>	filename	
<code>noisePosition</code>	<code>noiseGenerator</code>	along, cross, radial [m]
<code>noiseVelocity</code>	<code>noiseGenerator</code>	along, cross, radial [m/s]

This program is parallelized (1.5).

#### 4.14.6 NoiseSatelliteTracking

This program adds noise to simulated satellite tracking data generated by [SimulateSatelliteTracking](#). See [noiseGenerator](#) for details on noise generation.

Name	Type	Annotation
<a href="#">outputfileSatelliteTracking</a>	filename	
<a href="#">inputfileSatelliteTracking</a>	filename	
<a href="#">noiseRange</a>	<a href="#">noiseGenerator</a>	[m]
<a href="#">noiseRangeRate</a>	<a href="#">noiseGenerator</a>	[m/s]
<a href="#">noiseRangeAcceleration</a>	<a href="#">noiseGenerator</a>	[m/s <sup>2</sup> ]

This program is [parallelized \(1.5\)](#).

#### 4.14.7 NoiseStarCamera

This program adds noise to rotation observations. The noise is computed via a pseudo random sequence. See [noiseGenerator](#) for details on noise options.

Name	Type	Annotation
 <code>outputfileStarCamera</code>	filename	
 <code>inputfileStarCamera</code>	filename	
 <code>noiseRoll</code>	<code>noiseGenerator</code>	[rad]
 <code>noisePitch</code>	<code>noiseGenerator</code>	[rad]
 <code>noiseYaw</code>	<code>noiseGenerator</code>	[rad]

This program is parallelized (1.5).

#### 4.14.8 NoiseTimeSeries

This program generates **outputfileNoise** with the requested characteristics. See **noiseGenerator** for details on noise options.

Name	Type	Annotation
outputfileNoise	filename	
outputfileCovarianceFunction	filename	
noise	noiseGenerator	
timeSeries	timeSeries	
columns	uint	number of noise series (columns)

#### 4.14.9 SimulateAccelerometer

This program simulate `Accelerometer data`. The orientation of the accelerometer is given by `inputfileStarCamera` otherwise the celestial reference frame (CRF) is used. For computation of non-conservative forces a `satelliteModel` is needed.

Name	Type	Annotation
<code>outputfileAccelerometer</code>	filename	
<code>inputfileSatelliteModel</code>	filename	satellite macro model
<code>inputfileOrbit</code>	filename	
<code>inputfileStarCamera</code>	filename	
<code>earthRotation</code>	<code>earthRotation</code>	
<code>ephemerides</code>	<code>ephemerides</code>	
<code>forces</code>	<code>forces</code>	

This program is parallelized (1.5).

#### 4.14.10 SimulateAccelerometerCoMOffset

This program generates an [accelerometer file](#) containing perturbing accelerations due to a given center of mass (CoM) offset. This includes centrifugal effects, Euler forces and the effect of gravity gradients.

Name	Type	Annotation
outputfileAccelerometer	filename	effect of offset
inputfileOrbit	filename	
inputfileStarCamera	filename	
applyAngularRate	boolean	compute effect of centrifugal forces
applyAngularAccelerations	boolean	compute effect of Euler forces
gradientfield	gravityfield	low order field to estimate the change of the gravity by position adjustement
earthRotation	earthRotation	
interpolationDegree	uint	derivation of quaternions by polynomial interpolation of degree n
CoMOffsetX	double	offset [m]
CoMOffsetY	double	offset [m]
CoMOffsetZ	double	offset [m]

This program is [parallelized \(1.5\)](#).

#### 4.14.11 SimulateGradiometer

This program simulates error free `gradiometer` data along a satellite's orbit. The orientation of the full tensor gradiometer is given by `inputfileStarCamera` otherwise the celestial reference frame (CRF) is used. The gravity gradients are given by `gravityfield` and `tides`.

Name	Type	Annotation
outfileGradiometer	filename	
infileOrbit	filename	
infileStarCamera	filename	
earthRotation	earthRotation	
ephemerides	ephemerides	
gravityfield	gravityfield	
tides	tides	

This program is parallelized (1.5).

### 4.14.12 SimulateKeplerOrbit

This program simulates a Keplerian **orbit** at a given **timeSeries** starting from the given **integrationConstants**.

Name	Type	Annotation
outfileOrbit	filename	
timeSeries	timeSeries	
GM	double	Geocentric gravitational constant
integrationConstants	choice	
kepler	sequence	
majorAxis	double	[m]
eccentricity	double	[‐]
inclination	angle	[degree]
ascendingNode	angle	[degree]
argumentOfPerigee	angle	[degree]
meanAnomaly	angle	[degree]
time	time	integration constants are valid at this epoch
positionAndVelocity	sequence	
position0x	double	[m] in CRF
position0y	double	[m] in CRF
position0z	double	[m] in CRF
velocity0x	double	[m/s]
velocity0y	double	[m/s]
velocity0z	double	[m/s]
time	time	integration constants are valid at this epoch

### 4.14.13 SimulateOrbit

This program integrates an **orbit** from a given force function (dynamic orbit). The force functions are given by **forces**. For computation of non-conservative forces a **satelliteModel** is needed. The integration method must be selected with **propagator**. Because the orbit data are calculated in the celestial reference frame (CRF) you need **earthRotation** to transform the force function from the terrestrial reference frame (TRF). The integration start and end time, as well as the sampling, are derived from the **timeSeries** option. It is possible to integrate the arc in **reverse**, where the initial conditions are assumed to be met at the end time of the **timeSeries**.

Name	Type	Annotation
outputfileOrbit	filename	orbit file to be written.
inputfileSatelliteModel	filename	satellite macro model
timeSeries	timeSeries	time points for simulated orbit epochs.
integrationConstants	choice	
kepler	sequence	
majorAxis	double	[m]
eccentricity	double	[ ]
inclination	angle	[degree]
ascendingNode	angle	[degree]
argumentOfPerigee	angle	[degree]
meanAnomaly	angle	[degree]
GM	double	Geocentric gravitational constant
positionAndVelocity	sequence	
position0x	double	[m] in CRF
position0y	double	[m] in CRF
position0z	double	[m] in CRF
velocity0x	double	[m/s]
velocity0y	double	[m/s]
velocity0z	double	[m/s]
file	sequence	
inputfileOrbit	filename	only epoch at timeStart is used
margin	double	[seconds] used when finding initial epoch in orbitFile
propagator	orbitPropagator	orbit propagation method.
earthRotation	earthRotation	
ephemerides	ephemerides	
forces	forces	considered in orbit propagation.
reverse	boolean	start integration at last epoch in timeSeries, going backward in time.

#### 4.14.14 SimulateSatelliteTracking

This program simulates **tracking data** (range, range-rate, range-accelerations) between 2 satellites. The range is given by

$$\rho(t) = \|\mathbf{r}_B(t) - \mathbf{r}_A(t)\| = \mathbf{e}_{AB}(t) \cdot \mathbf{r}_{AB}(t), \quad (4.70)$$

with  $\mathbf{r}_{AB} = \mathbf{r}_B - \mathbf{r}_A$  and the unit vector in line of sight (LOS) direction

$$\mathbf{e}_{AB} = \frac{\mathbf{r}_{AB}}{\|\mathbf{r}_{AB}\|} = \frac{\mathbf{r}_{AB}}{\rho}. \quad (4.71)$$

Range-rates  $\dot{\rho}$  and range accelerations  $\ddot{\rho}$  are obtained by differentiation

$$\dot{\rho} = \mathbf{e}_{AB} \cdot \dot{\mathbf{r}}_{AB} + \dot{\mathbf{e}}_{AB} \cdot \mathbf{r}_{AB} = \mathbf{e}_{AB} \cdot \dot{\mathbf{r}}_{AB}, \quad (4.72)$$

$$\ddot{\rho} = \mathbf{e}_{AB} \cdot \ddot{\mathbf{r}}_{AB} + \dot{\mathbf{e}}_{AB} \cdot \dot{\mathbf{r}}_{AB} = \mathbf{e}_{AB} \cdot \ddot{\mathbf{r}}_{AB} + \frac{1}{\rho} (\dot{\mathbf{r}}_{AB}^2 - \dot{\rho}^2). \quad (4.73)$$

with the derivative of the unit vector

$$\dot{\mathbf{e}}_{AB} = \frac{d}{dt} \left( \frac{\mathbf{r}_{AB}}{\rho} \right) = \frac{\dot{\mathbf{r}}_{AB}}{\rho} - \frac{\dot{\rho} \cdot \mathbf{r}_{AB}}{\rho^2} = \frac{1}{\rho} (\dot{\mathbf{r}}_{AB} - \dot{\rho} \cdot \mathbf{e}_{AB}). \quad (4.74)$$

The **inputfileOrbits** must contain positions, velocities, and acceleration (see **OrbitAddVelocityAndAcceleration**).

Name	Type	Annotation
outputfileSatelliteTracking	filename	
inputfileOrbit1	filename	
inputfileOrbit2	filename	

This program is parallelized (1.5).

#### 4.14.15 SimulateStarCamera

This program simulates [star camera](#) measurements at each satellite's position. The satellite's orientation follows a local orbit frame with the x-axis in along track (along velocity), y-axis is cross track (normal to position and velocity vector) and z-axis pointing nadir (negative position vector). As for non circular orbit the position and velocity are not exact normal, the default is the x-axis to be exact along velocity and the z-axis forms a right hand system (not exact nadir) or with [nadirPointing](#) the z-axis is exact nadir and x-axis approximates along. The resulting rotation matrices rotate from satellite frame to inertial frame.

Name	Type	Annotation
<code>outputfileStarCamera</code>	filename	rotation from satellite to inertial frame (x: along, y: cross, z: nadir)
<code>inputfileOrbit</code>	filename	position and velocity defines the orientation of the satellite at each epoch
<code>nadirPointing</code>	boolean	false: exact along and nearly nadir, true: nearly along and exact nadir

This program is [parallelized \(1.5\)](#).

#### 4.14.16 SimulateStarCameraGnss

This program simulates **star camera** measurements at each satellite position of **inputfileOrbit**. The resulting rotation matrices rotate from body frame to inertial frame. The body frame refers to the IGS-specific (not the manufacturer-specific) body frame, as described by Montenbruck et al. (2015). The **inputfileOrbit** must contain velocities (use **OrbitAddVelocityAndAcceleration** if needed).

Information about the attitude mode(s) used by the GNSS satellite may be provided via **inputfileAttitudeInfo**. This file can be created with **GnssAttitudeInfoCreate**. It contains one or more time-dependent entries, each defining the default attitude mode, the attitude modes used around orbit noon and midnight, and some parameters required by the various modes. If no **inputfileAttitudeInfo** is selected, the program defaults to a nominal yaw-steering attitude model. A sufficiently high **modelingResolution** ensures that the attitude behavior is modeled properly at all times.

The attitude behavior is defined by the respective mode. Here is a list of the supported modes with a brief explanation and references:

- **nominalYawSteering**: Yaw to keep solar panels aligned to Sun (e.g. most GNSS satellites outside eclipse) [1]
- **orbitNormal**: Keep fixed yaw angle, for example point X-axis in flight direction (e.g. BDS-2G, BDS-3G, QZS-2G) [1]
- **catchUpYawSteering**: Yaw at maximum yaw rate to catch up to nominal yaw angle (e.g. GPS-\* (noon), GPS-IIR (midnight)) [2, 3]
- **shadowMaxYawSteeringAndRecovery**: Yaw at maximum yaw rate from shadow start to end, recover after shadow (e.g. GPS-IIA (midnight)) [2]
- **shadowMaxYawSteeringAndStop**: Yaw at maximum yaw rate from shadow start until nominal yaw angle at shadow end is reached, then stop (e.g. GLO-M (midnight)) [4]
- **shadowConstantYawSteering**: Yaw at constant yaw rate from shadow start to end (e.g. GPS-IIF (midnight)) [3]
- **centeredMaxYawSteering**: Yaw at maximum yaw rate centered around noon/midnight (e.g. QZS-2I, GLO-M (noon)) [4, 8]
- **smoothedYawSteering1**: Yaw based on an auxiliary Sun vector for a smooth yaw maneuver (e.g. GAL-1) [5]
- **smoothedYawSteering2**: Yaw based on a modified yaw-steering law for a smooth yaw maneuver (e.g. GAL-2, BDS-3M, BDS-3I) [5, 6]
- **betaDependentOrbitNormal**: Switch to orbit normal mode if below beta angle threshold (e.g. BDS-2M, BDS-2I, QZS-1) [7, 8]

See **GnssAttitudeInfoCreate** for more details on which satellite uses which attitude modes and the required parameters for each mode.

References for the attitude modes:

1. Montenbruck et al. (2015)
2. Kouba (2009)
3. Kuang et al. (2017)

4. Dilssner et al. (2011)
5. <https://www.gsc-europa.eu/support-to-developers/galileo-satellite-metadata#3>
6. Wang et al. (2018)
7. Li et al. (2018)
8. <https://qzss.go.jp/en/technical/qzssinfo/index.html>

Name	Type	Annotation
outputfileStarCamera	filename	rotation from body frame to CRF
inputfileOrbit	filename	attitude is modeled based on this orbit
inputfileAttitudeInfo	filename	attitude modes used by the satellite and respective parameters
interpolationDegree	uint	polynomial degree for orbit interpolation
modelingResolution	double	[s] resolution for attitude model evaluation
ephemerides	ephemerides	
eclipse	eclipse	model to determine if satellite is in Earth's shadow

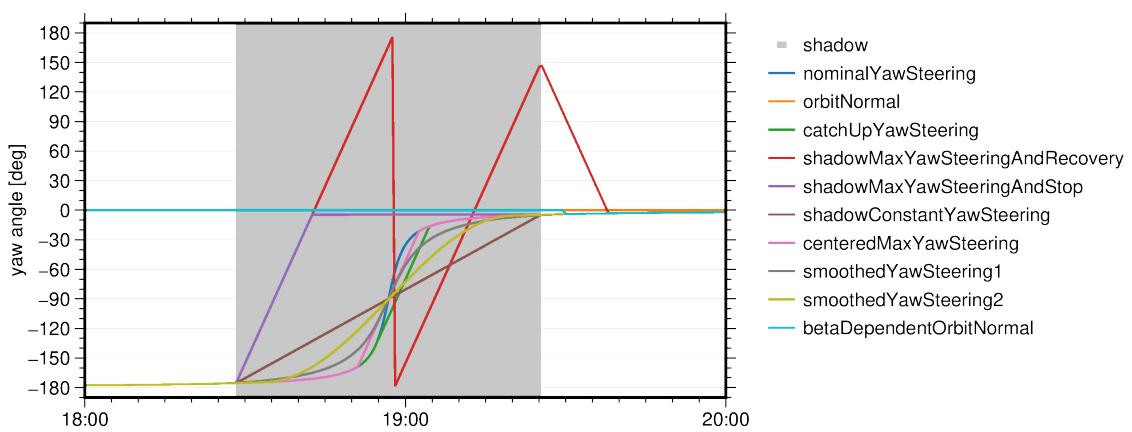


Figure 4.26: Overview of attitude modes used by GNSS satellites

#### 4.14.17 SimulateStarCameraGrace

Simulates [star camera data](#) of the two GRACE satellites.

- x: the antenna center pointing to the other satellite.
- y: normal to line of sight and the radial direction.
- z: forms a right handed system.

Name	Type	Annotation
outputfileStarCamera1	filename	
outputfileStarCamera2	filename	
inputfileOrbit1	filename	position define the orientation of the satellite at each epoch
inputfileOrbit2	filename	position define the orientation of the satellite at each epoch
antennaCenters	choice	KBR antenna phase center
value	sequence	
center1X	double	x-coordinate of antenna position in SRF [m] for GRACEA
center1Y	double	y-coordinate of antenna position in SRF [m] for GRACEA
center1Z	double	z-coordinate of antenna position in SRF [m] for GRACEA
center2X	double	x-coordinate of antenna position in SRF [m] for GRACEB
center2Y	double	y-coordinate of antenna position in SRF [m] for GRACEB
center2Z	double	z-coordinate of antenna position in SRF [m] for GRACEB
file	sequence	
inputAntennaCenters	filename	

#### 4.14.18 SimulateStarCameraSentinel1

This program simulates `star camera` measurements at each satellite's position for the Sentinel 1A satellite. The `inputfileOrbit` must contain positions and velocities (see [OrbitAddVelocityAndAcceleration](#)). The resulting rotation matrices rotate from satellite frame to inertial frame.

Name	Type	Annotation
<code>outputfileStarCamera</code>	filename	
<code>inputfileOrbit</code>	filename	position and velocity defines the orientation of the satellite at each epoch

This program is parallelized (1.5).

## 4.15 Programs: System

### 4.15.1 FileConvert

Converts GROOPS file between different file formats (ASCII, XML, binary), see [file formats \(1.7\)](#) for details. It prints also some information about the content. Therefore it can be used to get an idea about the content of binary files.

Name	Type	Annotation
▶ outputfile	filename	GROOPS formats: .xml, .txt, .dat
◀ inputfile	filename	GROOPS formats: .xml, .txt, .dat

### 4.15.2 FileCreateDirectories

Creates the directory and parent directories as needed.

Name	Type	Annotation
directory	filename	

### 4.15.3 FileMove

Move/rename file or directory. If the **outfile** is an existing directory the **infile** is moved into it.

Name	Type	Annotation
outfile	filename	target name or directory for the move/rename
infile	filename	

#### 4.15.4 FileRemove

Remove files or directories. Deletes also the content recursively if one of  files is a directory.

Name	Type	Annotation
 files	filename	

#### 4.15.5 FileTextCreate

Create text **» outputFile** containing **» lines**. This program can be a powerful tool, if the **» line** is repeated with a **» loop** together with the [text parser \(1.2.2\)](#).

Name	Type	Annotation
<b>» outputFile</b>	filename	
<b>» line</b>	string	

#### 4.15.6 GroupPrograms

Runs ▶ **programs** in a group, which can be used to structure a config file. If ▶ **catchErrors** is enabled and an error occurs, the remaining ▶ **programs** are skipped and execution continues with ▶ **errorPrograms**, in case any are defined. Otherwise an exception is thrown.

The ▶ **silently** option disables the screen output of the ▶ **programs**. With ▶ **outputfileLog** a log file is written for this group additional to a global log file. This might be helpful within ▶ **LoopPrograms** with parallel iterations.

Name	Type	Annotation
▶ <b>outputfileLog</b>	filename	additional log file
▶ <b>silently</b>	boolean	without showing the output.
▶ <b>program</b>	programType	
▶ <b>catchErrors</b>	sequence	
└▶ <b>errorProgram</b>	programType	executed if an error occurred

This program is parallelized (1.5).

### 4.15.7 IfPrograms

Runs a list of **programs** if a **condition** is met. Otherwise **elsePrograms** are executed.

Name	Type	Annotation
condition	condition	
program	programType	executed if condition evaluates to true
elseProgram	programType	executed if condition evaluates to false

This program is [parallelized \(1.5\)](#).

### 4.15.8 LoopPrograms

This program runs a list of programs in a **loop**.

If **continueAfterError=yes** and an error occurs, the remaining programs in the current iteration are skipped and the loop continues with the next iteration. Otherwise an exception is thrown.

If this program is executed on multiple processing nodes, the iterations can be computed in parallel, see [parallelization \(1.5\)](#). The first process serves as load balancer and the other processes are assigned to iterations according to **processCountPerIteration**. For example, running a loop containing three iterations on 13 processes with **processCountPerIteration=4**, runs the three iterations in parallel, with each iteration being assigned four processes. With **parallelLog=yes** all processes write output to screen and the log file. As the output can be quite confusing in this case, running **GroupPrograms** with an extra **outputfileLog** for each iteration (use the loop variables for the name of the log files) might be helpful.

Name	Type	Annotation
loop	loop	subprograms are called for every iteration
continueAfterError	boolean	continue with next iteration after error, otherwise throw exception
processCountPerIteration	uint	0: use all processes for each iteration
parallelLog	boolean	write to screen/log file from all processing nodes in parallelized loops
program	programType	

This program is [parallelized \(1.5\)](#).

### 4.15.9 RunCommand

Execute system **commands**. If **executeParallel** is set and multiple **commands** are given they are executed in parallel at distributed nodes, otherwise they are executed consecutively at master node only.

Name	Type	Annotation
command	filename	
silently	boolean	without showing the output.
continueAfterError	boolean	continue with next command after error, otherwise throw exception
executeParallel	boolean	execute several commands in parallel

This program is [parallelized \(1.5\)](#).

## 4.16 Programs: Conversion

### 4.16.1 BerneseKinematic2Orbit

Read kinematic orbits in Bernese format.

Name	Type	Annotation
outputfileOrbit	filename	
outputfileCovariance	filename	
earthRotation	earthRotation	from TRF to CRF
infile	filename	

## 4.16.2 Champ2AccStar

This program reads in CHAMP accelerometer and star camera data given in the special CHAMP format. In case of CHAMP accelerometer and star camera data is both stored in one file. A description of the format can be found under: [http://op.gfz-potsdam.de/champ/docs\\_CHAMP/CH-GFZ-FD-001.pdf](http://op.gfz-potsdam.de/champ/docs_CHAMP/CH-GFZ-FD-001.pdf).

Name	Type	Annotation
▶ outputfileAccelerometer	filename	
▶ outputfileAngularAcceleration	filename	
▶ outputfileStarCamera	filename	
✖ inputFile	filename	

### 4.16.3 Champ2Orbit

This program reads in CHAMP precise science orbits in the special CHORB format. A description of the format can be found under: [http://op.gfz-potsdam.de/champ/docs\\_CHAMP/CH-GFZ-FD-002.pdf](http://op.gfz-potsdam.de/champ/docs_CHAMP/CH-GFZ-FD-002.pdf)

Name	Type	Annotation
outputfileOrbit	filename	
earthRotation	earthRotation	
timeSeries	timeSeries	
inputOrbit	sequence	
inputfile	filename	orbits in SP3 format

#### 4.16.4 Cosmic2OrbitStar

This program reads in cosmic orbit and star camera data given in the CHAMP format. In case of cosmic orbit and star camera data is stored in one file. A description of the format can be found under: [http://op.gfz-potsdam.de/champ/docs\\_CHAMP/CH-GFZ-FD-001.pdf](http://op.gfz-potsdam.de/champ/docs_CHAMP/CH-GFZ-FD-001.pdf)

Name	Type	Annotation
outputfileOrbit	filename	
outputfileStarCamera	filename	
inputfile	filename	

#### 4.16.5 DoodsonAdmittance2SupplementaryFiles

The publication of an ocean tide model includes not only the atlas in the form of spherical harmonics coefficients, but also the matrix of Doodson multipliers (**▶ outputFileDoodsonMatrix**) and the **▶ outputFileAdmittanceMatrix**.

The **▶ outputFileMajorTideList** contains the **▶ fileNames** for each constituent. The required information is taken from the **▶ inputFileAdmittance**.

See also **▶ DoodsonHarmonics2PotentialCoefficients**.

Name	Type	Annotation
<b>▶ outputFileMajorTideList</b>	filename	
<b>▶ fileNames</b>	filename	template for fileList, variables: doodson, name, cossin
<b>▶ outputFileDoodsonMatrix</b>	filename	
<b>▶ outputFileAdmittanceMatrix</b>	filename	
<b>▶ inputFileAdmittance</b>	filename	interpolation of minor constituents

#### 4.16.6 DoodsonHarmonics2IersPotential

Convert doodson harmonics to IERS conventions according to FES2004. cf. <ftp://tai.bipm.org/iers/conv2010/chapter6/tidemodels/fes2004.dat>.

Name	Type	Annotation
❶ <code>outputfile</code>	filename	according to IERS2010, chapter 6.3.2, footnote 7
❶ <code>inputfileDoodsonHarmonics</code>	filename	
❶ <code>header</code>	string	info for output header
❶ <code>factor</code>	double	
❶ <code>minDegree</code>	uint	
❶ <code>maxDegree</code>	uint	

#### 4.16.7 DoodsonHarmonics2IersWaterHeight

Convert doodson harmonics to IERS conventions according to FES2004. cf. <ftp://tai.bipm.org/iers/conv2010/chapter6/tidemodels/fes2004.dat>.

Name	Type	Annotation
❶ <code>outputfile</code>	filename	according to IERS2010, chapter 6.3.2, footnote 7
❶ <code>inputfileDoodsonHarmonics</code>	filename	
❶ <code>inputfileTideGeneratingPotential</code>	filename	to compute Xi phase correction
❷ <code>header</code>	string	info for output header
❶ <code>kernel</code>	kernel	data type of output values
❷ <code>factor</code>	double	e.g. from [m] to [cm]
❷ <code>minDegree</code>	uint	
❷ <code>maxDegree</code>	uint	

#### 4.16.8 GnssAntex2AntennaDefinition

Converts metadata and antenna definitions from the [IGS ANTEX format](#), to [►transmitterInfo](#), [►antennaDefinition](#), [►svnBlockTable](#), and [►transmitterList](#) files for the respective GNSS and for the list of ground station antennas.

Name	Type	Annotation
►outputfileAntennaDefinitionStation	filename	antenna center variations
►outputfileAntennaDefinitionTransmitter	filename	antenna center variations
►outputfileTransmitterInfo	filename	PRN is appended to file name
►outputfileSvnBlockTableGps	filename	SVN to satellite block mapping
►outputfileSvnBlockTableGlonass	filename	SVN to satellite block mapping
►outputfileSvnBlockTableGalileo	filename	SVN to satellite block mapping
►outputfileSvnBlockTableBeiDou	filename	SVN to satellite block mapping
►outputfileSvnBlockTableQzss	filename	SVN to satellite block mapping
►outputfileTransmitterListGps	filename	list of PRNs
►outputfileTransmitterListGlonass	filename	list of PRNs
►outputfileTransmitterListGalileo	filename	list of PRNs
►outputfileTransmitterListBeiDou	filename	list of PRNs
►outputfileTransmitterListQzss	filename	list of PRNs
★inputfileAntex	filename	
►timeStart	time	ignore older antenna definitions
►createZeroModel	boolean	create empty antenna patterns

#### 4.16.9 GnssAttitude2Orbex

Convert attitude of GNSS satellites to [ORBEX file format](#) (quaternions).

If **earthRotation** is provided, the output file contains quaternions for rotation from TRF to satellite body frame (IGS/ORBEX convention), otherwise the rotation is from CRF to satellite body frame.

See also [GnssOrbex2StarCamera](#), [SimulateStarCameraGnss](#).

Name	Type	Annotation
outputfileOrbex	filename	ORBEX file
inputfileTransmitterList	filename	ASCII list with transmitter PRNs
inputfileAttitude	filename	instrument file containing attitude
variablePrn	string	loop variable for PRNs from transmitter list
timeSeries	timeSeries	resample to these epochs (otherwise input file epochs are used)
earthRotation	earthRotation	rotate data into Earth-fixed frame
interpolationDegree	uint	for attitude and Earth rotation interpolation
description	string	description of file contents
createdBy	string	name of agency
inputData	string	description of input data (see ORBEX description)
contact	string	email address
referenceFrame	string	reference frame used in file
comment	string	

### 4.16.10 GnssClock2ClockRinex

Converts GNSS clocks from GROOPS format to **IGS clock RINEX format**. Clocks can be provided via **satelliteData** and/or **stationData**. Observed signal types are inferred from **inputfileSignalBias**. Satellites/stations used as clock references can be provided via **referenceClock**.

See IGS clock RINEX format description for further details on header information.

Name	Type	Annotation
outfileClockRinex	filename	
satelliteData	sequence	one element per satellite
inputfileClock	filename	clock instrument file
inputfileSignalBias	filename	signal bias file
identifier	string	PRN (e.g. G23)
stationData	sequence	one element per station
inputfileClock	filename	clock instrument file
inputfilePosition	filename	station position file
inputfileStationInfo	filename	station info file
identifier	string	station name (e.g. wtzz)
comment	string	comment in header
program	string	name of program (for first line)
institution	string	name of agency (for first line)
analysisCenter	string	name of analysis center
differentialCodeBias	string	program and source for applied differential code bias
phaseCenterVariations	string	program and source for applied phase center variations
referenceClock	string	identifier of reference satellite/station
referenceFrame	string	terrestrial reference frame for the stations

#### 4.16.11 GnssClockRinex2InstrumentClock

This program converts clocks from the [IGS clock RINEX format](#), which contains the clocks of all satellites and stations in a single file, into an [Instrument file \(MISCVALUE\)](#) for each [Identifier](#) (satellite and/or station).

Name	Type	Annotation
outfileInstrument	filename	identifier is appended to each file
infileClockRinex	filename	
identifier	string	satellite or station identifier, e.g. G23 or alic
intervals	timeSeries	
minEpochsPerInterval	uint	minimum number of epochs in an interval

### 4.16.12 GnssEop2IgsErp

Write GNSS Earth orientation parameters to [IGS ERP file format](#).

Requires polar motion, polar motion rate, dUT1 and LOD parameters in the solution vector **inputfileSolution** and their sigmas in **inputfileSigmax**. Solution usually comes out of [GnssProcessing](#).

Name	Type	Annotation
<b>outputfileIgsErp</b>	filename	IGS ERP file
<b>epoch</b>	sequence	e.g. daily solution
<b>inputfileSolution</b>	filename	parameter vector
<b>inputfileSigmax</b>	filename	standard deviations of the parameters (sqrt of the diagonal of the inverse normal equation)
<b>inputfileParameterNames</b>	filename	parameter names
<b>inputfileTransmitterList</b>	filename	transmitter PRNs used in solution (used for transmitter count)
<b>inputfileStationList</b>	filename	stations used in solution (used for station count)
<b>time</b>	time	reference time for epoch
<b>comment</b>	string	

### 4.16.13 GnssGriddedDataTimeSeries2Ionex

Converts TEC maps from GROOPS  gridded data time series format to IGS [IONEX file format](#).

Currently only supports 2D TEC maps.

See also  [GnssIonex2GriddedDataTimeSeries](#),  [IonosphereMap](#).

Name	Type	Annotation
 <code>outputfileIonex</code>	filename	
 <code>inputfileGriddedDataTimeSeries</code>	filename	must contain regular grid
 <code>value</code>	expression	expression (e.g. data column)
 <code>timeSeries</code>	<code>timeSeries</code>	(empty = use input file time series)
 <code>program</code>	string	name of program (for first line)
 <code>institution</code>	string	name of agency (for first line)
 <code>description</code>	string	description in header
 <code>comment</code>	string	comment in header
 <code>mappingFunction</code>	string	see IONEX documentation
 <code>elevationCutoff</code>	double	see IONEX documentation (0 if unknown)
 <code>observablesUsed</code>	string	see IONEX documentation
 <code>exponent</code>	int	factor $10^{\text{exponent}}$ is applied to all values

#### 4.16.14 GnssIonex2GriddedDataTimeSeries

Converts TEC maps from IGS [IONEX](#) file format to GROOPS [gridded data time series](#) format.

Currently only supports 2D TEC maps.

See also [GnssGriddedDataTimeSeries2Ionex](#), [IonosphereMap](#).

Name	Type	Annotation
FLAG outputfileGriddedDataTimeSeries	filename	
FLAG inputFileIonex	filename	

### 4.16.15 GnssNormals2Sinex

Write GNSS data/metadata and **normal equations** to **SINEX** format.

Normal equations usually come from **GnssProcessing** (e.g. from GNSS satellite orbit determination and station network analysis (3.2)). Metadata input files include **stationInfo/transmitterInfo**, **antennaDefinition**, and **stationList/transmitterList**, see **GnssAntex2AntennaDefinition**.

See also **Sinex2Normals** and **NormalsSphericalHarmonics2Sinex**.

Name	Type	Annotation
▶ outputfileSinexNormals	filename	full SINEX file including normal equations
▶ outputfileSinexCoordinates	filename	SINEX file without normal equations (station coordinates file)
▶ inputFileNormals	filename	normal equation matrix
▶ inputFileSolution	filename	parameter vector
▶ inputFileSigmaX	filename	standard deviations of the parameters (sqrt of the diagonal of the inverse normal equation)
▶ inputFileApriori	filename	apriori parameter vector
▶ inputFileAprioriSigma	filename	constraint sigmas for apriori parameter vector
▶ inputFileAprioriMatrix	filename	normal equation matrix of applied constraints
▶ transmitterConstellation	sequence	transmitter constellation metadata
▶▶ inputFileTransmitterList	filename	transmitter PRNs used in solution
▶▶ inputFileTransmitterInfo	filename	transmitter info file template
▶▶ inputFileAntennaDefinition	filename	transmitter phase centers and variations (ANTEX)
▶▶ variablePrn	string	loop variable for PRNs from transmitter list
▶ stations	sequence	stations contained in normal equations
▶▶ inputFileStationList	filename	station info file template
▶▶ inputFileStationInfo	filename	station phase centers and variations (ANTEX)
▶▶ inputFileAntennaDefinition	filename	loop variable for station names from station list
▶▶ variableStationName	string	start time for which solution has observations
▶▶ observationTimeStart	time	end time for which solution has observations
▶▶ observationTimeEnd	time	reference time for parameters
▶ time	time	[seconds] observation sampling
▶ sampling	double	e.g. IGS14_WWWW (WWW = ANTEX release GPS week)
▶▶ antennaCalibrationModel	string	
▶▶ sinexHeader	sequence	identify the agency providing the data
▶▶▶ agencyCode	string	start time of the data
▶▶▶ timeStart	time	end time of the data
▶▶▶ timeEnd	time	technique used to generate the SINEX solution
▶▶▶ observationCode	string	0: tight constraint, 1: significant constraint, 2: unconstrained
▶▶▶ constraintCode	string	solution types contained in the SINEX solution (S O E T C A)
▶▶▶ solutionContent	string	organizations gathering/alerting the file contents
▶▶▶▶ description	string	Address of the relevant contact. e-mail
▶▶▶▶ contact	string	Description of the file contents
▶▶▶▶ output	string	Brief description of the input used to generate this solution
▶▶▶▶ input	string	Software used to generate the file
▶▶▶▶ software	string	Computer hardware on which above software was run
▶▶▶▶ hardware	string	comments in the comment block from a file (truncated at 80 characters)
▶▶▶▶▶ inputFileComment	filename	
▶▶▶▶▶ comment	string	comments in the comment block

### 4.16.16 GnssOrbex2StarCamera

Converts GNSS satellite attitude from [ORBEX file format](#) (quaternions) to [Instrument file \(STARCAM-ERA\)](#). The resulting star camera files contain the rotation from satellite body frame to TRF, or to CRF in case [earthRotation](#) is provided.

See also [GnssAttitude2Orbex](#).

Name	Type	Annotation
<code>outputfileStarCamera</code>	filename	rotation from body frame to TRF/CRF, identifier is appended to each file
<code>inputfileOrbex</code>	filename	
<code>identifier</code>	string	(empty = all) satellite identifier, e.g. G23 or E05
<code>earthRotation</code>	<a href="#">earthRotation</a>	rotation from TRF to CRF

#### 4.16.17 GnssReceiver2RinexObservation

Converts a **inputfileGnssReceiver** into a **RINEX** observation file. The **inputfileStationInfo** contains the antenna and receiver information for the RINEX header. The **useType** and **ignoreType** can be used to filter the observation types that will be exported.

Name	Type	Annotation
outfileRinexObservation	filename	RINEX observation file
infileGnssReceiver	filename	GNSS instrument file
infileStationInfo	filename	antenna and receiver info
comment	string	write comments at begin of header
observer	string	header information
agency	string	header information
useType	gnssType	only use observations that match any of these patterns
ignoreType	gnssType	ignore observations that match any of these patterns

### 4.16.18 GnssRinexNavigation2OrbitClock

Evaluates orbit and clock parameters from RINEX (version 2, 3, and 4) navigation file **inputfileRinex** at epochs given by **timeSeries** and writes them to **outputfileOrbit** and **outputfileClock**, respectively.

Orbits are rotated from TRF (as broadcasted) to CRF via **earthRotation**, but system-specific TRFs (WGS84, PZ-90, etc.) are not aligned to a common TRF.

Furthermore, option is available to remove any satellite ephemeris data that has their satellite flag set to unhealthy.

See also **OrbitAddVelocityAndAcceleration**.

Name	Type	Annotation
outputfileOrbit	filename	PRN is appended to file name
outputfileClock	filename	PRN is appended to file name
inputfileRinex	filename	RINEX navigation file
timeSeries	timeSeries	orbit and clock evaluation epochs
earthRotation	earthRotation	for rotation from TRF to CRF
useType	gnssType	(e.g. ***G12) only use satellites with PRN that match any of these patterns
ignoreType	gnssType	(e.g. ***R**) ignore satellites PRN that match any of these patterns
removeUnhealthySatellites	boolean	Remove satellite ephemeris that have their sat flags set to unhealthy

### 4.16.19 GnssSignalBias2SinexBias

Convert [GNSS signal biases](#) from GROOPS format to [IGS SINEX Bias format](#). Biases can be provided via [transmitterBiases](#) and/or [receiverBiases](#). Phase biases without attribute (e.g. L1\*) are automatically expanded so each code bias has a corresponding phase bias (Example: C1C, C1W, L1\* are converted to C1C, C1W, L1C, L1W).

Time-variable biases (e.g. GPS L5 satellite phase bias) can be provided via [timeVariableBias](#). Their time span will be based on the provided epochs ( $t \pm \Delta t/2$ ). The slope of the bias can be optionally provided in the second data column.

If GLONASS receiver biases depend on frequency number, those must be defined in [inputfileTransmitterInfo](#) to get the correct PRN/SVN assignment to the biases.

See IGS SINEX Bias format description for further details on header information.

See also [GnssSinexBias2SignalBias](#) and [GnssBiasClockAlignment](#).

Name	Type	Annotation
<a href="#">outputfileSinexBias</a>	filename	
<a href="#">inputfileTransmitterInfo</a>	filename	one file per satellite
<a href="#">transmitterBiases</a>	sequence	one element per satellite
<a href="#">inputfileSignalBias</a>	filename	signal bias file
<a href="#">timeVariableBias</a>	sequence	one entry per time variable bias type
<a href="#">inputfileSignalBias</a>	filename	columns: mjd, bias [m], (biasSlope [m/s])
<a href="#">type</a>	<a href="#">gnssType</a>	bias type
<a href="#">identifier</a>	string	PRN or station name (e.g. G23 or wtzz)
<a href="#">receiverBiases</a>	sequence	one element per station
<a href="#">inputfileSignalBias</a>	filename	signal bias file
<a href="#">timeVariableBias</a>	sequence	one entry per time variable bias type
<a href="#">inputfileSignalBias</a>	filename	columns: mjd, bias [m], (biasSlope [m/s])
<a href="#">type</a>	<a href="#">gnssType</a>	bias type
<a href="#">identifier</a>	string	PRN or station name (e.g. G23 or wtzz)
<a href="#">agencyCode</a>	string	identify the agency providing the data
<a href="#">fileAgencyCode</a>	string	identify the agency creating the file
<a href="#">timeStart</a>	time	start time of the data
<a href="#">timeEnd</a>	time	end time of the data
<a href="#">biasMode</a>	choice	absolute or relative bias estimates
<a href="#">absolute</a>		
<a href="#">relative</a>		
<a href="#">observationSampling</a>	uint	[seconds]
<a href="#">intervalLength</a>	uint	[seconds] interval for bias parameter representation
<a href="#">determinationMethod</a>	string	determination method used to generate the bias results (see SINEX Bias format description)
<a href="#">receiverClockReferenceGnss</a>	string	(G, R, E, C) reference GNSS used for receiver clock estimation
<a href="#">satelliteClockReferenceObservables</a>	string	one per system, reference code observable on first and second frequency (RINEX3 format)
<a href="#">description</a>	string	organization gathering/altering the file contents
<a href="#">contact</a>	string	contact name and/or email address
<a href="#">input</a>	string	brief description of the input used to generate this solution
<a href="#">output</a>	string	description of the file contents
<a href="#">software</a>	string	software used to generate the file
<a href="#">hardware</a>	string	computer hardware on which above software was run
<a href="#">comment</a>	string	comments in the comment block

### 4.16.20 GnssSinexBias2SignalBias

Converts GNSS signal biases from [IGS SINEX Bias format](#) to [GnssSignalBias format](#).

Only satellite observable-specific signal biases (OSB) are supported at the moment. If multiple entries exist for the same bias, the weighted average (based on time span) of all entries is used. Time-variable biases are not supported at the moment.

See also [GnssSignalBias2SinexBias](#).

Name	Type	Annotation
❶ <code>outputfileSignalBias</code>	filename	identifier is appended to file name
❷ <code>inputfileSinexBias</code>	filename	
❸ <code>inputfileGlonassSignalDefinition</code>	filename	GLONASS frequency number mapping
❹ <code>identifier</code>	string	(empty = all) satellite PRN, e.g. G23 or E05

#### 4.16.21 GnssStationLog2Platform

Converts IGS station log format to ▶ **outputfileStationPlatform**.

If ▶ **inputfileAntennaDefinition** is provided, station log data is cross-checked with the given antenna definitions. Cross-checking station log data with a **SINEX file** is possible with ▶ **CheckStationsPlatformsWithSinex**.

Name	Type	Annotation
▶ <b>outputfileStationPlatform</b>	filename	
▶ <b>inputfileStationLog</b>	filename	
▶ <b>inputfileAntennaDefinition</b>	filename	used to check antennas

### 4.16.22 GnssStationLog2StationInfo

Converts IGS station log format to ▶ **outputfileStationInfo**.

If ▶ **inputfileAntennaDefinition** is provided, station log data is cross-checked with the given antenna definitions. Cross-checking station log data with a [SINEX file](#) is also possible by providing ▶ **inputfileSinex**. Any failed checks result in warnings in the output log.

Name	Type	Annotation
▶ <b>outputfileStationInfo</b>	filename	
▶ <b>inputfileStationLog</b>	filename	
▶ <b>inputfileAntennaDefinition</b>	filename	used to check antennas
▶ <b>inputfileSinex</b>	filename	used to cross-check station log with SINEX file

### 4.16.23 GnssTroposphere2TropoSinex

Convert GNSS troposphere data from GROOPS format to [IGS SINEX TRO](#) format.

Specification of the station list is done via **inputfileStationList**. **inputfileTroposphereData** needs the troposphere data provided from **GnssProcessing**. Additional following station metadata are required: **inputfileStationInfo** using file type **Station info**, **inputfileAntennaDefinition** using file type **Antenna definition** and **inputfileGridPos** which uses the stations positions provided from **GnssProcessing**. For considering the geoid height use **inputfileGeoidHeight**. The geoid height is provided by **Gravityfield2GriddedData**.

Name	Type	Annotation
outputfileTropoSinex	filename	
stations	sequence	
inputfileStationList	filename	ASCII file with station names
inputfileTroposphereData	filename	Troposphere data estimates template (columns: mjd, trodry, trowet, tgndry, tgnwet, tgedry, tgewet)
inputfileTroposphereSigmas	filename	Troposphere data sigmas template (columns: mjd, sigma_trowet, sigma_tgnwet, sigma_tgewet)
inputfileStationInfo	filename	station info file template
inputfileGeoidHeight	filename	File including geoid height
inputfileGridPos	filename	File including stations positions
inputfileAntennaDefinition	filename	station phase centers and variations (ANTEX)
variableStationName	filename	Loop variable for station names from station list
observationTimeStart	time	Start time for which solution has observations
observationTimeEnd	time	End time for which solution has observations
dataSamplingInterval	double	[sec] GNSS data sampling rate
tropoSamplingInterval	double	[sec] Tropospheric parameter sampling interval
tropoModelingMethod	string	Tropospheric estimation method: Filter, Smoother, Least Squares, Piece-Wise Linear Interpolation
aPrioriTropoModel	string	A priori tropospheric model used
tropoMappingFunction	string	Name of mapping function used for hydrostatic and wet delay
gradientMappingFunction	string	Name of mapping function used for gradients
metDataSource	string	source of surface meteorological observations used (see format desc.)
observationWeighting	string	observation weighting model applied
elevationCutoff	double	[deg]
gnssSystems	string	G=GPS, R=GLONASS, E=Galileo, C=BeiDou
timeSystem	string	G (GPS) or UTC
oceanTideModel	string	Ocean tide loading model applied
atmosphericTideModel	string	Atmospheric tide loading model applied
geoidModel	string	Geoid model name for undulation values
systemCode	string	Terrestrial reference system code
remark	string	Remark used to identify the origin of the coordinates (AC acronym)
antennaCalibrationModel	string	e.g. IGS14_WWWW (WWW = ANTEX release GPS week)
sinexTroHeader	sequence	
agencyCode	string	Identify the agency providing the data
timeStart	time	Start time of the data
timeEnd	time	End time of the data
observationCode	string	Technique used to generate the SINEX solution
solutionContents	string	Marker name if single station, MIX if multiple stations
description	string	Organizations gathering/alerting the file contents

---

 output	string	Description of the file contents
 contact	string	Address of the relevant contact e-mail
 software	string	Software used to generate the file
 hardware	string	Computer hardware on which above software was run
 input	string	Brief description of the input used to generate this solution
 versionNumber	string	Unique identifier of the product, same as in file name, e.g. 000
 inputFileComment	filename	comments in the comment block from a file (truncated at 80 characters per line)
 comment	string	comments in the comment block

---

#### 4.16.24 GoceXml2Gradiometer

Read ESA XML GOCE Data. The **»outputfileGradiometer** is written as **»instrument file (GRADIOMETER)**.

Name	Type	Annotation
<b>»outputfileGradiometer</b>	filename	
<b>»infile</b>	filename	

### 4.16.25 GoceXml2Orbit

Read ESA XML GOCE Data.

Name	Type	Annotation
outputfileOrbit	filename	
earthRotation	earthRotation	rotation from TRF to CRF
inputfile	filename	

#### 4.16.26 GoceXml2StarCamera

Read ESA XML GOCE Data.

Name	Type	Annotation
 <code>outputfileStarCamera</code>	filename	
 <code>inputfile</code>	filename	

### 4.16.27 GoceXmlEggNom1b

Read ESA XML GOCE Data.

Name	Type	Annotation
▶ outputfileGradiometer	filename	
▶ outputfileStarCamera	filename	
▶ outputfileAngularRate	filename	
▶ outputfileAngularAcc	filename	
▶ <b>inputfile</b>	filename	

#### 4.16.28 Grace2PotentialCoefficients

This program converts potential coefficients from the GRACE SDS format into [potential coefficients file](#). The program supports file formats for RL04 to RL06.

Within the program, the variables `epochStart`, `epochEnd` and `epochMid` are populated with the corresponding time-stamps in the file. These can be used in to [outputfilePotentialCoefficients](#) to auto-generate the file name.

Name	Type	Annotation
<a href="#">outputfilePotentialCoefficients</a>	filename	variables: epochStart, epochEnd, epochMid
<a href="#">inputfile</a>	filename	

### 4.16.29 GraceAccelerometer2L1bAscii

Convert GROOPS accelerometer files to the GRACE SDS L1B ASCII format.

Name	Type	Annotation
outputfileAscii	filename	ASCII outputfile
inputfileAccelerometer	filename	GROOPS accelerometer file
satelliteIdentifier	string	satellite identifier (A or B for GRACE, C or D for GRACE-FO)
globalAttributes	string	additional attributes as 'key: value' pairs

#### 4.16.30 GraceAod2DoodsonHarmonics

This program converts the atmospheric and ocean tidal products (AOD1B) from the GRACE SDS format into **outputfileDoodsonHarmonics**.

Name	Type	Annotation
outputfileDoodsonHarmonics	filename	
inputfileTideGeneratingPotential	filename	to compute Xi phase correction
inputfile	filename	

### 4.16.31 GraceAod2TimeSplines

This program converts the atmospheric and ocean de-aliasing product (AOD1B) from the GRACE SDS format into [time spline files](#). Multiple [inputfiles](#) must be given in the correct time order. A linear method is assumed for the interpolation between the given points in time.

The GRACE SDS format is described in "AOD1B Product Description Document" given at <http://podaac.jpl.nasa.gov/grace/documentation.html>.

Name	Type	Annotation
outputfileDealiasing	filename	
outputfileAtmosphere	filename	
outputfileOcean	filename	
outputfileBottomPressure	filename	
outputfileMisc	filename	
inputfile	filename	

### 4.16.32 GraceCoefficients2BlockMeanTimeSplines

This program converts potential coefficients from the GRACE SDS RL06 format into  
**►outputfileTimeSplines**.

The **►outputfileTimeSeries** contains the mid points of non-empty intervals and  
**►outputfileTimeIntervals** contains the monthly interval boundaries from first to last solution.

The output will always be monthly block means. If the SDS solutions do vary or overlap, the nearest solution in terms of reference epoch is used.

Name	Type	Annotation
<b>►</b> <b>outputfileTimeSplines</b>	filename	
<b>►</b> <b>outputfileTimeSplinesCovariance</b>	filename	only the variances are saved
<b>►</b> <b>outputfileTimeSeries</b>	filename	mid points of non-empty intervals
<b>►</b> <b>outputfileTimeIntervals</b>	filename	monthly interval boundaries from first to last solution
<b>►*</b> <b>infile</b>	filename	

### 4.16.33 GraceL1a2Accelerometer

This program converts Level-1A accelerometer data (ACC1A) to the GROOPS instrument file format. The GRACE Level-1A format is described in `GRACEiolib.h` given at [http://podaac-tools.jpl.nasa.gov/drive/files/allData/grace/sw/GraceReadSW\\_L1\\_2010-03-31.tar.gz](http://podaac-tools.jpl.nasa.gov/drive/files/allData/grace/sw/GraceReadSW_L1_2010-03-31.tar.gz). The output is one arc of satellite data which can include data gaps. To split the arc in multiple gap free arcs use  **InstrumentSynchronize**.

Name	Type	Annotation
 <code>outputfileAccelerometer</code>	filename	ACCELEROMETER in SRF
 <code>outputfileAngularAccelerometer</code>	filename	ACCELEROMETER in SRF
 <code>inputfile</code>	filename	ACC1A

#### 4.16.34 GraceL1a2SatelliteTracking

This program converts Level-1A satellite tracking data (KBR1A) to the GROOPS instrument file format. The GRACE Level-1A format is described in `GRACEiolib.h` given at [http://podaac-tools.jpl.nasa.gov/drive/files/allData/grace/sw/GraceReadSW\\_L1\\_2010-03-31.tar.gz](http://podaac-tools.jpl.nasa.gov/drive/files/allData/grace/sw/GraceReadSW_L1_2010-03-31.tar.gz). The output is one arc of satellite data which can include data gaps. To split the arc in multiple gap free arcs use  **InstrumentSynchronize**.

Name	Type	Annotation
 <code>outputfileSatelliteTracking</code>	filename	MISCVALUES(ant_id, K_phase, Ka_phase, K_SNR, Ka_SNR)
 <code>infile</code>	filename	KBR1A

### 4.16.35 GraceL1a2StarCamera

This program converts orientation data measured by the star cameras from the GRACE Level-1A format (SCA1A) to the GROOPS instrument file format. For further information see [GraceL1a2Accelerometer](#).

Name	Type	Annotation
☞ <code>outputfileStarCamera1</code>	filename	STARCAMERA1A, head 1
☞ <code>outputfileStarCamera2</code>	filename	STARCAMERA1A, head 2
☞ <code>inputfile</code>	filename	SCA1A, !GRACE-FO is not working!

#### 4.16.36 GraceL1a2Temperature

This program converts Level-1A temperature measurements (HRT1B or HRT1A) to the GROOPS instrument file format. The GRACE Level-1A format is described in GRACE given at [http://podaac-tools.jpl.nasa.gov/drive/files/allData/grace/sw/GraceReadSW\\_L1\\_2010-03-31.tar.gz](http://podaac-tools.jpl.nasa.gov/drive/files/allData/grace/sw/GraceReadSW_L1_2010-03-31.tar.gz). Multiple **inputfiles** must be given in the correct time order. The output is one arc of satellite data which can include data gaps. To split the arc in multiple gap free arcs use **InstrumentSynchronize**.

Name	Type	Annotation
outfileTemperature	filename	MISCVALUES
infile	filename	HRT1B or HRT1A

### 4.16.37 GraceL1b2AccHousekeeping

This program converts ACC housekeeping data (AHK1B or AHK1A) from the GRACE SDS format into [instrument file \(ACCHOUSEKEEPING\)](#). For further information see [GraceL1b2Accelerometer](#).

Name	Type	Annotation
☞ <code>outputfileAccHousekeeping</code>	filename	ACCHOUSEKEEPING
☞ <code>inputfile</code>	filename	AHK1B or AHK1A

#### 4.16.38 GraceL1b2Accelerometer

This program converts accelerometer data (ACC1B or ACT1B) from the GRACE SDS format into [instrument file \(ACCELEROMETER\)](#).

Multiple **inputfiles** must be given in the correct time order. The output is one arc of satellite data which can include data gaps. To split the arc in multiple gap free arcs use [InstrumentSynchronize](#).

The GRACE SDS format is described in "GRACE Level 1B Data Product User Handbook JPL D-22027" given at [https://podaac-tools.jpl.nasa.gov/drive/files/allData/grace/docs/Handbook\\_1B\\_v1.3.pdf](https://podaac-tools.jpl.nasa.gov/drive/files/allData/grace/docs/Handbook_1B_v1.3.pdf).

Name	Type	Annotation
outputfileAccelerometer	filename	ACCELEROMETER
outputfileAngularAccelerometer	filename	ACCELEROMETER
outputfileFlags	filename	MISCVVALUES(qualflg, acl_res.x, acl_res.y, acl_res.z)
infile	filename	ACC1B or ACT1B

### 4.16.39 GraceL1b2ClockOffset

This program converts clock data (CLK1B or LLK1B) from the GRACE SDS format into [Instrument file \(MISCVALUE\)](#). For further information see [GraceL1b2Accelerometer](#).

Name	Type	Annotation
outputfileClock	filename	MISCVALUE
inputfile	filename	CLK1B or LLK1B

#### 4.16.40 GraceL1b2GnssReceiver

This program converts GPS receiver data (phase and pseudo range) data from the GRACE SDS format (GPS1B or GPS1A) into [Instrument file \(GNSSRECEIVER\)](#). For further information see [GraceL1b2Accelerometer](#).

Name	Type	Annotation
outputfileGnssReceiver	filename	GNSSRECEIVER
inputfile	filename	GPS1B or GPS1A

#### 4.16.41 GraceL1b2Magnetometer

This program converts magnetometer data (MAG1B or MAG1A) from the GRACE SDS format into [instrument file \(MAGNETOMETER\)](#). For further information see [GraceL1b2Accelerometer](#).

Name	Type	Annotation
☛ <code>outputfileMagnetometer</code>	filename	MAGNETOMETER
☛ <code>inputfile</code>	filename	MAG1B or MAG1A

#### 4.16.42 GraceL1b2Mass

This program converts mass data (MAS1B or MAS1A) from the GRACE SDS format into [Instrument file \(MASS\)](#). For further information see [GraceL1b2Accelerometer](#).

Name	Type	Annotation
 <code>outputfileMass</code>	filename	MASS
 <code>infile</code>	filename	MAS1B or MAS1A

#### 4.16.43 GraceL1b2Orbit

This program converts the reduced dynamical orbit from the GRACE/GRACE-FO SDS format (GNV1B, GNI1B) into [Instrument file \(ORBIT\)](#).

When GNV1B is used, the orbit can be rotated from the terrestrial reference frame (TRF) transformed into the celestial reference frame (CRF) by specifying [earthRotation](#).

For further information see [GraceL1b2Accelerometer](#).

Name	Type	Annotation
<a href="#">outputfileOrbit</a>	filename	
<a href="#">earthRotation</a>	<a href="#">earthRotation</a>	to rotate GNV1B into CRF
<a href="#">inputfile</a>	filename	GNV1B/GNI1B

#### 4.16.44 GraceL1b2SatelliteTracking

This program converts low-low satellite data measured by the K-band ranging system from the GRACE SDS format (KBR1B or LRI1B) into **Instrument file (SATELLITETRACKING)**. The **inputfiles** contain also corrections to antenna offsets and the so called light time correction. The corrections can be stored in additional files in the same format as the observations. If a phase break is found an artificial gap is created. For further information see **GraceL1b2Accelerometer**.

Name	Type	Annotation
▶ outfileSatelliteTracking	filename	SATELLITETRACKING
▶ outfileAntCentr	filename	SATELLITETRACKING
▶ outfileLighttime	filename	SATELLITETRACKING
▶ outfileSNR	filename	MISCVALUES(K_A_SNR, Ka_A_SNR, K_B_SNR, Ka_B_SNR, qualflg)
▶ outfileIonoCorr	filename	MISCVALUE
...* infile	filename	KBR1B or LRI1B

#### 4.16.45 GraceL1b2StarCamera

This program converts orientation data measured by a star camera (SRF to CRF) from the GRACE SDS format (SCA1B) into [instrument file \(STARCAMERA\)](#). For further information see [GraceL1b2Accelerometer](#).

Name	Type	Annotation
outputfileStarCamera	filename	
outputfileStarCameraFlags	filename	MISCVALUES(sca_id, qual_rss, qualflg)
inputfile	filename	SCA1B

#### 4.16.46 GraceL1b2StarCameraCovariance

This program computes star camera covariance matrices ([instrument file](#), COVARIANCE3D) for a GRACE satellite under consideration of the active camera heads and an a priori variance factor.

Name	Type	Annotation
outfileStarCameraCovariance	filename	
infileStarCameraFlags	filename	
infileSequenceOfEventsQSA	filename	
sigma0	double	[seconds of arc]

This program is parallelized (1.5).

#### 4.16.47 GraceL1b2SteeringMirror

This program converts GRACE-FO Steering Mirror output (LSM1B) to an [Instrument file \(STARCAM-ERA\)](#).

Name	Type	Annotation
▶ <code>outputfileStarCamera</code>	filename	
✖ <code>inputfile</code>	filename	LSM1B

#### 4.16.48 GraceL1b2Thruster

This program converts thruster data (THR1B or THR1A) from the GRACE SDS format into [Instrument file \(THRUSTER\)](#). For further information see [GraceL1b2Accelerometer](#).

Name	Type	Annotation
☞ outputfileThruster	filename	THRUSTER
☞ inputfile	filename	THR1B or THR1A

#### 4.16.49 GraceL1b2TimeOffset

This program converts time data (TIM1A or TIM1B) from the GRACE SDS format into [Instrument file \(MISCVALUE\)](#). For further information see [GraceL1b2Accelerometer](#).

Name	Type	Annotation
outfileTime	filename	MISCVALUE
fractionalScale	double	1e-6 for GRACE, 1e-9 for GRACE-FO
infile	filename	TIM1A or TIM1B

#### 4.16.50 GraceL1b2Uso

This program converts clock data (USO1B) from the GRACE SDS format into [Instrument file \(MISCVALUES\)](#). For further information see [GraceL1b2Accelerometer](#).

Name	Type	Annotation
outputfileUso	filename	MISCVALUES(uso_freq, K_freq, Ka_freq)
inputfile	filename	USO1B

### 4.16.51 GraceL1b2Vector

This program reads vector orientation data (positions of instruments in the satellite frame) from the GRACE SDS format (VGB1B, VGN1B, VGO1B, VKB1B, or VCM1B). The **outputfileVector** is a  $(3n \times 1)$  matrix containing  $(x, y, z)$  for each record. The GRACE SDS format is described in "GRACE Level 1B Data Product User Handbook JPL D-22027" given at <http://podaac.jpl.nasa.gov/grace/documentation.html>.

Name	Type	Annotation
outputfileVector	filename	
inputfile	filename	VGB1B, VGN1B, VGO1B, VKB1B, or VCM1B

### 4.16.52 GraceSequenceOfEvents

This program converts the GRACE SOE (sequence of events) file/format into **Instrument file (MISCVUES)**. The GRACE SOE format is described in "GRACE Level 1B Data Product User Handbook JPL D-22027" and "TN-03\_SOE\_format.txt" given at <http://podaac.jpl.nasa.gov/grace/documentation.html>. The output is one arc of satellite data which can include data gaps.

Name	Type	Annotation
outputfileGraceA	filename	
outputfileGraceB	filename	
inputfile	filename	SoE file
events	choice	
ACCT	sequence	DSHL HeaterDisconnect
mode	choice	
Heater		DSHL HeaterDisconnect
SetPoint		temperature set point
AOCS	sequence	coarse pointing mode or attitude hold mode
mode	choice	
CPM		coarse pointing mode
AHM		attitude hold mode
SM		science mode
ACCR		ACCR
CMCAL	sequence	CoM calibration maneuver
sampling	double	[seconds] create events between start and end of maneuver
KBRCAL	sequence	KBR calibration maneuver
sampling	double	[seconds] create events between start and end of maneuver
VCM		CoM coordinates in SRF (m)
VKB		KBR phase center coordinates in SRF (m)
ICUVP		ICUVP
IPU		IPU
IPUR		IPUR
KAMI		KAMI: time tag offset to Ka-phase meas.
KMI		K_MI: time tag offset to K-phase meas.
KTOFF		KTOFF: time tag offset to KBR meas.
MANV		MANV
MTE1		MTE1
MTE2		MTE2
OCC		OCC
QSA		SCA to SRF frame rotation
QKS		SCA to KBR frame rotation

### 4.16.53 GrailCdr2Orbit

This program converts the orbit from the GRAIL SDS format into [Instrument file \(ORBIT\)](#).

Name	Type	Annotation
outputfileOrbit	filename	
inputfile	filename	

#### 4.16.54 GrailCdr2SatelliteTracking

This program converts low-low satellite data measured by the K-band ranging system from the GRAIL format into [Instrument file \(SATELLITETRACKING\)](#). The [inputfiles](#) contain also corrections for antenna offsets and the so called light time correction. The corrections can be stored in additional files in the same format as the observations. If a phase break is found an artificial gap is created.

Name	Type	Annotation
▶ outfileSatelliteTracking	filename	
▶ outfileAntCentr	filename	
▶ outfileLighttime	filename	
▶ outfileTemperature	filename	
▶ approximateTimeBias	double [seconds]	
✳️ inputFile	filename	

#### 4.16.55 GrailCdr2StarCamera

This program converts orientation data measured by a star camera (SRF to CRF) from the GRAIL SDS format into [Instrument file \(STARCAMERA\)](#). For further information see [GraceL1b2Accelerometer](#).

Name	Type	Annotation
▶ outputfileStarCamera	filename	
✖ inputfile	filename	

#### 4.16.56 GriddedData2NetCdf

This program converts a **inputfileGriddedData** to a COARDS compliant NetCDF file. The output data can be defined with **dataVariable**. You should add at least the attributes `units`, `long_name`, and maybe `_FillValue` to the variables. For the **dataVariable:value** the standard `dataVariables (1.2.3)` are available to select the data columns of **inputfileGriddedData**.

See also **NetCdfInfo**, **GriddedDataTimeSeries2NetCdf**, **NetCdf2GriddedData**.

Name	Type	Annotation
outputfileNetCdf	filename	file name of NetCDF output
inputfileGriddedData	filename	input grid sequence
dataVariable	sequence	metadata for data variables
name	string	netCDF variable name
value	expression	expression (variables 'height', 'data', 'longitude', 'latitude' and, 'area' are taken from the gridded data
dataType	choice	
double		
float		
int		
attribute	choice	netCDF attributes
text	sequence	
name	string	
value	string	
value	sequence	
name	string	
value	double	
dataType	choice	
double		
float		
int		
globalAttribute	choice	additional meta data
text	sequence	
name	string	
value	string	
value	sequence	
name	string	
value	double	
dataType	choice	
double		
float		
int		

### 4.16.57 GriddedDataTimeSeries2NetCdf

Read a **inputfileGriddedDataTimeSeries** and converts it to a COARDS compliant NetCDF file.

The output data can be defined with **dataVariable**. You should add at least the attributes `units`, `long_name`, and maybe `_FillValue` to the variables. The **dataVariable:inputColumn** selects the data from the input file.

If **timeSeries** is not set the temporal nodal points from the inputfile are used.

See also **NetCdfInfo**, **GriddedData2NetCdf**, **NetCdf2GriddedDataTimeSeries**.

Name	Type	Annotation
outputfileNetCdf	filename	file name of NetCDF output
inputfileGriddedDataTimeSeries	filename	
timeSeries	timeSeries	otherwise times from inputfile are used
dataVariable	sequence	metadata for data variables
name	string	netCDF variable name
inputColumn	uint	input data column
dataType	choice	
double		
float		
int		
attribute	choice	netCDF attributes
text	sequence	
name	string	
value	string	
value	sequence	
name	string	
value	double	
dataType	choice	
double		
float		
int		
globalAttribute	choice	additional meta data
text	sequence	
name	string	
value	string	
value	sequence	
name	string	
value	double	
dataType	choice	
double		
float		
int		

#### 4.16.58 GroopsAscii2Orbit

Read Orbits given in groops kinematic orbit ASCII format with covariance information.

See also [➡ Orbit2GroopsAscii](#).

Name	Type	Annotation
➡ outputfileOrbit	filename	
➡ outputfileCovariance	filename	
➡ earthRotation	earthRotation	
➡ inputfile	filename	

### 4.16.59 Hw2TideGeneratingPotential

Write `tide generating potential` from Hartmann and Wenzel 1995 file, <https://doi.org/10.1029/95GL03324>.

Name	Type	Annotation
outfileTideGeneratingPotential	filename	
inputfile	filename	
headerLines	uint	skip number of header lines
referenceTime	time	reference time

#### 4.16.60 Icgem2PotentialCoefficients

Read spherical harmonics in ICGEM format (<http://icgem.gfz-potsdam.de/>).

Name	Type	Annotation
❶ <code>outputfileStaticCoefficients</code>	filename	static potential coefficients in GROOPS gfc format. Available variables (icgem2.0): epochStart, epochEnd, epochMid; (icgem1.0) epochReference
❷ <code>outputfileTrendCoefficients</code>	filename	trend potential coefficients in GROOPS gfc format. Available variables (icgem2.0): epochStart, epochEnd, epochMid; (icgem1.0) epochReference
❸ <code>outputfileOscillationCosine</code>	filename	oscillation cosine coefficients in GROOPS gfc format. Available variables (icgem2.0): epochStart, epochEnd, epochMid, oscillationPeriod; (icgem1.0) epochReference, oscillationPeriod
❹ <code>outputfileOscillationSine</code>	filename	oscillation sine coefficients in GROOPS gfc format. Available variables (icgem2.0): epochStart, epochEnd, epochMid, oscillationPeriod; (icgem1.0) epochReference, oscillationPeriod
❺ <code>outputfileIntervals</code>	filename	two column ASCII file with all intervals found (only sensible for icgem2.0). The base name will be extended with .static, .trend, .annualCos, and .annualSin.
❻ <code>inputfileIcgem</code>	filename	ICGEM GFC file
❼ <code>useFormalErrors</code>	boolean	use formal errors if both formal and calibrated errors are given

### 4.16.61 Iers2OceanPoleTide

Read ocean pole tide model according to IERS conventions and convert into [OceanPoleTide](#) file.

Name	Type	Annotation
outputfileOceanPole	filename	
inputfile	filename	
inputfileLoadingLoveNumber	filename	
maxDegree	uint	
GM	double	Geocentric gravitational constant
R	double	Reference radius
Omega	double	[rad/s] earth rotation
rho	double	[kg/m**3] density of sea water
G	double	[m**3/(kg*s**2)] gravitational constant
g	double	[m/s**2] gravity

#### 4.16.62 IersC04IAU2000EarthOrientationParameter

Read a IERS Earth orientation data C04 (IAU2000A) file and write it as  **outputfileEOP**.

Name	Type	Annotation
 <b>outputfileEOP</b>	filename	
 <b>inputfile</b>	filename	
 <b>timeStart</b>	time	
 <b>timeEnd</b>	time	

### 4.16.63 IersHighFrequentEop2DoodsonEop

Read Diurnal and Subdiurnal Earth Orientation variations according to updated IERS 2010 conventions and write them as **outputfileDoodsonEOP**.

Name	Type	Annotation
outputfileDoodsonEOP	filename	
inputfile	filename	

#### 4.16.64 IersPotential2DoodsonHarmonics

Read ocean tide file in IERS format.

Name	Type	Annotation
outputfileDoodsonHarmonics	filename	
inputfile	filename	
headerLines	uint	skip number of header lines
minDegree	uint	
maxDegree	uint	
GM	double	Geocentric gravitational constant
R	double	reference radius

#### 4.16.65 IersRapidIAU2000EarthOrientationParameter

Read a IERS Earth orientation rapid data and prediction file (IAU2000) and write it as **»outputfileEOP**.

Name	Type	Annotation
»outputfileEOP	filename	
»inputfile	filename	
»timeStart	time	
»timeEnd	time	

#### 4.16.66 IersWaterHeight2DoodsonHarmonics

Read ocean tide file in IERS format.

Name	Type	Annotation
outputfileDoodsonHarmonics	filename	
inputfile	filename	
headerLines	uint	skip number of header lines
inputfileTideGeneratingPotential	filename	to compute Xi phase correction
kernel	kernel	data type of input values
factor	double	to convert in SI units
minDegree	uint	
maxDegree	uint	
GM	double	Geocentric gravitational constant
R	double	reference radius

#### 4.16.67 Igs2EarthOrientationParameter

Read Rapid Earth Orientation Parameter from IGS daily file and write it as  **outputfileEOP**.

Name	Type	Annotation
 <b>outputfileEOP</b>	filename	
 <b>inputfile</b>	filename	
 <b>timeStart</b>	time	
 <b>timeEnd</b>	time	

#### 4.16.68 Jason2Starcamera

This program reads in Jason star camera data given in a special format. Files available at: [cddis.gsfc.nasa.gov/pub/doris/ancillary/quaternions/ja2/](http://cddis.gsfc.nasa.gov/pub/doris/ancillary/quaternions/ja2/). A description of the format can be found under: [ftp://ftp.ids-doris.org/pub/ids/ancillary/quaternions/jason1\\_2\\_quaternion\\_solar\\_panel.pdf](http://ftp.ids-doris.org/pub/ids/ancillary/quaternions/jason1_2_quaternion_solar_panel.pdf)

Name	Type	Annotation
outfileStarCamera	filename	
jasonNumber	uint	Jason number (different file format), 1 for Sentinel
infile	filename	

### 4.16.69 JplAscii2Ephemerides

Read JPL DExxx (ASCII) ephemerides.

Name	Type	Annotation
outputfileEphemerides	filename	
inputfileHeader	filename	
inputfileData	filename	

#### 4.16.70 Metop2Starcamera

This program reads in star camera data from MetOp satellites given in the special CHAMP format. A description of the format can be found under: [http://op.gfz-potsdam.de/champ/docs\\_CHAMP/CH-GFZ-FD-001.pdf](http://op.gfz-potsdam.de/champ/docs_CHAMP/CH-GFZ-FD-001.pdf)

Name	Type	Annotation
outputfileStarCamera	filename	
inputfile	filename	

### 4.16.71 NetCdf2GriddedData

This program converts a COARDS compliant NetCDF file into an **outfileGriddedData**. If no specific input **variableNameData** are selected all suitable data are used.

If the NETCDF file contains a time axis (**variableNameData**) an specific epoch can be selected with **time**. The nearest epoch in file is used.

See also **NetCdfInfo**, **GriddedData2NetCdf**, **NetCdf2GriddedDataTimeSeries**.

Name	Type	Annotation
<b>outfileGriddedData</b>	filename	
<b>infileNetCdf</b>	filename	
<b>variableNameLongitude</b>	string	name of NetCDF variable
<b>variableNameLatitude</b>	string	name of NetCDF variable
<b>variableNameTime</b>	string	if with time axis: name of NetCDF variable
<b>variableNameData</b>	string	data variables, otherwise all suitable data are used
<b>time</b>	time	if with time axis: nearest epoch is used
<b>R</b>	double	reference radius for ellipsoidal coordinates
<b>inverseFlattening</b>	double	reference flattening for ellipsoidal coordinates

#### 4.16.72 NetCdf2GriddedDataTimeSeries

This program converts a COARDS compliant NetCDF file into **outputfileGriddedDataTimeSeries**. If no specific input **variableNameData** are selected all suitable data are used.

See also **NetCdfInfo**, **NetCdf2GriddedData**, **GriddedDataTimeSeries2NetCdf**.

Name	Type	Annotation
outputfileGriddedDataTimeSeries	filename	
inputfileNetCdf	filename	
variableNameLongitude	string	name of NetCDF variable
variableNameLatitude	string	name of NetCDF variable
variableNameTime	string	name of NetCDF variable)
variableNameData	string	data variables, otherwise all suitable data are used
R	double	reference radius for ellipsoidal coordinates
inverseFlattening	double	reference flattening for ellipsoidal coordinates

### 4.16.73 NetCdfInfo

Print content information of a NetCDF file like dimensions, variables and attributes.

See also [NetCdf2GriddedData](#), [NetCdf2GriddedDataTimeSeries](#), [GriddedData2NetCdf](#), [GriddedDataTimeSeries2NetCdf](#).

Name	Type	Annotation
 inputfileNetCdf	filename	

#### 4.16.74 NormalsSphericalHarmonics2Sinex

Write potential coefficients and [normal equations](#) to SINEX format.

See also [Sinex2Normals](#) and [GnssNormals2Sinex](#).

Name	Type	Annotation
outputfileSinex	filename	solutions in SINEX format
inputfileNormals	filename	normal equation matrix
inputfileSolution	filename	parameter vector
inputfileSigmax	filename	standard deviations of the parameters (sqrt of the diagonal of the inverse normal equation)
inputfileApriori	filename	apriori parameter vector
inputfileAprioriMatrix	filename	normal equation matrix of applied constraints
time	time	reference time for parameters
sinexHeader	sequence	
agencyCode	string	identify the agency providing the data
timeStart	time	start time of the data
timeEnd	time	end time of the data
observationCode	string	technique used to generate the SINEX solution
constraintCode	string	0: tight constraint, 1: significant constraint, 2: unconstrained
solutionContent	string	solution types contained in the SINEX solution (S O E T C A)
description	string	organizations gathering/alerting the file contents
contact	string	Address of the relevant contact. e-mail
output	string	Description of the file contents
input	string	Brief description of the input used to generate this solution
software	string	Software used to generate the file
hardware	string	Computer hardware on which above software was run
inputfileComment	filename	comments in the comment block from a file (truncated at 80 characters)
comment	string	comments in the comment block

### 4.16.75 Orbit2GroopsAscii

Convert groops orbits and corresponding covariance information to ASCII format. The format is used to publish TUG orbits. It contains a two line header with a short description of the orbit defined in **firstLine**. The orbit is rotated to the Earth fixed frame (TRF) with **earthRotation** and given as one line per epoch. The epoch lines contained time [MJD GPS time], position x, y and z [m], and the epoch covariance xx, yy, zz, xy, xz and yz [ $m^2$ ].

See also **GroopsAscii2Orbit**.

Name	Type	Annotation
outputfile	filename	
inputfileOrbit	filename	
inputfileCovariance	filename	
earthRotation	<a href="#">earthRotation</a>	
firstLine	string	Text for first line

### 4.16.76 Orbit2Sp3Format

Writes orbits to [SP3 format](#).

SP3 orbits are usually given in the terrestrial reference frame (TRF), so providing `earthRotation` automatically rotates the orbits from the celestial reference frame (CRF) to the TRF. Since SP3 orbits often use the center of Earth as a reference, a correction from center of mass to center of Earth can be applied to the orbits by providing `gravityfield` (e.g. ocean tides).

See also [Sp3Format2Orbit](#).

Name	Type	Annotation
<code>outputfile</code>	filename	
<code>satellite</code>	sequence	
<code>inputfileOrbit</code>	filename	
<code>inputfileClock</code>	filename	
<code>inputfileCovariance</code>	filename	
<code>identifier</code>	string	3 characters (e.g. GNSS PRN: G01)
<code>orbitAccuracy</code>	double	[m] used for accuracy codes in header (0 = unknown)
<code>earthRotation</code>	<code>earthRotation</code>	rotate data into Earth-fixed frame
<code>gravityfield</code>	<code>gravityfield</code>	degree 1 fluid mantle for CM2CE correction (SP3 orbits should be in center of Earth)
<code>comment</code>	string	comment lines (77 char max)
<code>firstLine</code>	string	Text for first line e.g: u+U IGb14 KIN ITSG
<code>writeVelocity</code>	boolean	write velocity in addition to position
<code>useSp3kFormat</code>	boolean	use the extended sp3k format

### 4.16.77 PotentialCoefficients2Icgem

Write spherical harmonics in ICGEM format. GROOPS uses this format as default but this program enables the possibility to include comments and set the modelname.

Name	Type	Annotation
outputfile	filename	
inputfilePotentialCoefficients	filename	
inputfileTrend	filename	
oscillation	sequence	
inputfileCosPotentialCoefficients	filename	
inputfileSinPotentialCoefficients	filename	
period	string	period of oscillation [year]
comment	string	comment in header
inputfileComment	filename	file containing comments for header
modelname	string	name of the model
tideSystem	choice	tide system of model
zero_tide		
tide_free		
minDegree	uint	
maxDegree	uint	
GM	double	Geocentric gravitational constant
R	double	reference radius
time	time	reference time

#### 4.16.78 PsmslOceanBottomPressure2TimeSeries

This programs reads ocean bottom pressure time series from the Permanent Service for Mean Sea Level (PSMSL).

In addition to the OBP measurements, the recorder position can be written to a [grid file](#).

Name	Type	Annotation
outputfileTimeSeries	filename	
outputfilePosition	filename	recorder position as gridded data
inputfile	filename	
isDaily	boolean	
ignoreBadData	boolean	
R	double	
inverseFlattening	double	
timeSeries	timeSeries	

### 4.16.79 RinexObservation2GnssReceiver

Converts [RINEX](#) (version 2, 3, and 4) and [Compact RINEX](#) observation files to [GnssReceiver Instrument file](#).

In case of [RINEX v2.x](#) observation files containing GLONASS satellites, a mapping from PRN to frequency number must be provided via [inputfileMatrixPrn2FrequencyNumber](#) in the form of a [matrix file](#) with columns: GLONASS PRN, mjdStart, mjdEnd, frequencyNumber. Source for mapping: <http://semisys.gfz-potsdam.de/semisys/api/?symname=2002&format=json&satellite=GLO>. RINEX v3+ observation files already contain this information.

[useType](#) and [ignoreType](#) can be used to filter the observation types that will be exported.

If [inputfileStationInfo](#) is set, RINEX antenna and receiver info will be cross-checked with the provided file and warnings are raised in case of differences.

A list of semi-codeless GPS receivers (observing C2D instead of C2W) can be provided via [inputfileSemiCodelessReceivers](#) in ASCII format with one receiver name per line. Observation types will be automatically corrected for these receivers.

Some LEO satellites use special RINEX observation types, either from the unofficial RINEX v2.20 or custom ones. These can be provided via [inputfileSpecialObservationTypes](#) in ASCII format. The file must contain a table with two columns, the first being the special type, and the second being the equivalent RINEX v3 type.

Name	Type	Annotation
<a href="#">outputfileGnssReceiver</a>	filename	
<a href="#">inputfileRinexObservation</a>	filename	RINEX or Compact RINEX observation files
<a href="#">inputfileMatrixPrn2FrequencyNumber</a>	filename	(required for RINEX v2 files containing GLONASS observations) GROOPS matrix with columns: GLONASS PRN, SVN, mjdStart, mjdEnd, frequencyNumber
<a href="#">inputfileStationInfo</a>	filename	used to determine semi-codeless receivers and to cross-check antenna and receiver info
<a href="#">inputfileSemiCodelessReceivers</a>	filename	ASCII list with one receiver name per line
<a href="#">inputfileSpecialObservationTypes</a>	filename	ASCII table mapping special observation types to RINEX 3 types, e.g.: LA L1C
<a href="#">useType</a>	<a href="#">gnssType</a>	only use observations that match any of these patterns
<a href="#">ignoreType</a>	<a href="#">gnssType</a>	ignore observations that match any of these patterns

#### 4.16.80 Sacc2Orbit

This program reads in SACC orbit data.

Name	Type	Annotation
 outputfileOrbit	filename	
 inputfile	filename	

### 4.16.81 Sentinel2StarCamera

This program reads in Sentinel-1/2/3 star camera data given in the special format.

Name	Type	Annotation
⭐ outputfileStarCamera	filename	
⭐ inputfile	filename	

#### 4.16.82 SentinelXml2Orbit

Read Sentinel orbits from XML format.

Name	Type	Annotation
 <code>outputfileOrbit</code>	filename	
 <code>earthRotation</code>	<a href="#">earthRotation</a>	
 <code>inputfile</code>	filename	

### 4.16.83 Sinex2Normals

Convert normal equations from **SINEX** format to **normal equations**.

See also **GnssNormals2Sinex** and **NormalsSphericalHarmonics2Sinex**.

Name	Type	Annotation
outputfileNormals	filename	N, n: unconstrained normal equations
outputfileNormalsConstraint	filename	N0, n0: normal equations of applied constraints
outputfileSolution	filename	x: parameter vector
outputfileSolutionApriori	filename	x0: a priori parameter vector
inputFileSinex	filename	

#### 4.16.84 Sinex2StationDiscontinuities

Convert station discontinuities from [SINEX format](#) (e.g. ITRF20) to [outfileInstrument](#) (MISC-VALUE). A value of 1 means position discontinuity, a value of 2 means velocity discontinuity. Start and end epochs with value 0 are added in addition to the discontinuities from SINEX to define continuity interval borders.

See also [Sinex2StationPosition](#) and [Sinex2StationPostSeismicDeformation](#).

Name	Type	Annotation
<a href="#">outfileInstrument</a>	filename	loop variable is replaced with station name (e.g. wtzz)
<a href="#">infileDiscontinuities</a>	filename	SINEX (e.g. ITRF20) station discontinuities
<a href="#">variableLoopStation</a>	string	variable name for station loop
<a href="#">stationName</a>	string	only export these stations

### 4.16.85 Sinex2StationPositions

Extracts station positions from **inputfileSinexSolution** (SINEX format description) and writes an **outputfileInstrument** of type VECTOR3D for each station. Positions will be computed at **timeSeries** based on position and velocity of each provided interval in the SINEX file. With **inputfileSinexDiscontinuities** the bounds of these time spans are adjusted to the exact epochs of discontinuities. The **inputfileSinexPostSeismicDeformations** adds the ITRF post-seismic deformation model to the affected stations. The **inputfileSinexFrequencies** adds annual and semi-annual frequencies.

If **extrapolateBackward** or **extrapolateForward** are provided, positions will also be computed for epochs before the first interval/after the last interval, based on the position and velocity of the first/last interval. Position extrapolation will stop at the first discontinuity before the first interval/after the last interval.

Stations can be limited via **stationName**, otherwise all stations in **inputfileSinexSolution** will be used.

Name	Type	Annotation
<b>outputfileInstrument</b>	filename	loop variable is replaced with station name (e.g. wtzz)
<b>variableLoopStation</b>	string	variable name for station loop
<b>inputfileSinexSolution</b>	filename	SINEX file
<b>inputfileSinexDiscontinuities</b>	filename	SINEX file
<b>inputfileSinexPostSeismicDeformations</b>	filename	SINEX file
<b>inputfileSinexFrequencies</b>	filename	SINEX file (XYZ or ENU)
<b>timeSeries</b>	timeSeries	compute positions for these epochs based on velocity
<b>extrapolateForward</b>	boolean	also compute positions for epochs after last interval defined in SINEX file
<b>extrapolateBackward</b>	boolean	also compute positions for epochs before first interval defined in SINEX file
<b>stationName</b>	string	convert only these stations

#### 4.16.86 SinexMetadata2GlonassFrequencyNumber

Create  **outputfileMatrixPrn2FrequencyNumber** matrix from [IGS SINEX metadata format](#) with the columns: GLONASS PRN, SVN, mjdStart, mjdEnd, frequencyNumber.

See also  [RinexObservation2GnssReceiver](#).

Name	Type	Annotation
 <b>outputfileMatrixPrn2FrequencyNumber</b>	filename	GROOPS matrix with columns: GLONASS PRN, SVN, mjd-Start, mjdEnd, frequencyNumber
 <b>inputfileSinexMetadata</b>	filename	IGS SINEX metadata file

#### 4.16.87 SinexMetadata2SatelliteModel

Create **»outputfileSatelliteModel** from [IGS SINEX](#) metadata format.

If **»infileSatelliteModel** is provided it is used as a basis and values are updated from the metadata file.

See also **»SatelliteModelCreate**.

Name	Type	Annotation
»outputfileSatelliteModel	filename	
»infileSinexMetadata	filename	IGS SINEX metadata file
»infileSatelliteModel	filename	base satellite model
»svn	string	e.g. G040, R736, E204, C211

### 4.16.88 Sp3Format2Orbit

Read orbits from SP3 format and write an [instrument file \(ORBIT\)](#). The additional [outputfileClock](#) is an [instrument file \(MISCVALUE\)](#) and [outputfileCovariance](#) is an [instrument file \(COVARIANCE3D\)](#).

With [satelliteIdentifier](#) a single satellite can be selected if the [inputfiles](#) contain more than one satellites. If [satelliteIdentifier](#) is empty the first satellite is taken. All satellites can be selected with [satelliteIdentifier=<all>](#). In this case the identifier is appended to each output file.

If [earthRotation](#) is provided the data are transformed from terrestrial (TRF) to celestial reference frame (CRF). Since SP3 orbits often use the center of Earth as a reference, a correction from center of Earth to center of mass can be applied to the orbits by providing [gravityfield](#) (e.g. ocean tides).

See also [Orbit2Sp3Format](#).

Name	Type	Annotation
<a href="#">outputfileOrbit</a>	filename	
<a href="#">outputfileClock</a>	filename	
<a href="#">outputfileCovariance</a>	filename	3x3 epoch covariance
<a href="#">satelliteIdentifier</a>	string	e.g. L09 for GRACE A, empty: take first satellite, \$;all\$;: identifier is appended to each file
<a href="#">earthRotation</a>	<a href="#">earthRotation</a>	rotation from TRF to CRF
<a href="#">gravityfield</a>	<a href="#">gravityfield</a>	degree 1 fluid mantle for CM2CE correction (SP3 orbits should be in center of Earth)
<a href="#">infile</a>	filename	orbits in SP3 format

### 4.16.89 Swarm2Starcamera

This program reads SWARM star camera data given in the cdf format and before converted to an ascii file using the program `cdfexport` provided by the Goddard Space Flight Center (<http://cdf.gsfc.nasa.gov/>).

Name	Type	Annotation
outputfileStarCamera	filename	
earthRotation	<a href="#">earthRotation</a>	
inputfile	filename	

#### 4.16.90 TerraSarTandem2Orbit

This program reads in TerraSar-X or Tandem-X orbits in the special CHORB format and takes the appropriate time frame as stated in the document header. A description of the format can be found under: [http://op.gfz-potsdam.de/champ/docs\\_CHAMP/CH-GFZ-FD-002.pdf](http://op.gfz-potsdam.de/champ/docs_CHAMP/CH-GFZ-FD-002.pdf)

Name	Type	Annotation
outputfileOrbit	filename	
earthRotation	<a href="#">earthRotation</a>	
inputfile	filename	orbits in CHORB format

#### 4.16.91 TerraSarTandem2StarCamera

This program reads in TerraSar-X or Tandem-X star camera data given in the special format.

Name	Type	Annotation
 outputfileStarCamera	filename	
 inputfile	filename	

#### 4.16.92 Tle2Orbit

This program computes the **outfileOrbit** from two-line elements (TLE/3LE) as can be found at e.g. <http://celesttrak.org/NORAD/elements/>. The first satellite in the input file that matches the wildcard of **satelliteName** is used.

The program uses the Simplified General Perturbation (SGP) model. More information can be found in the Revisiting Spacetrack Report 3 by Vallado et al. 2006.

Name	Type	Annotation
outfileOrbit	filename	
infileTLE	filename	two line elements (TLE/3LE)
satelliteName	string	first name of wildcard match is used
timeSeries	timeSeries	output orbit at these times
earthRotation	earthRotation	rotation to CRF

### 4.16.93 ViennaMappingFunctionGrid2File

This program converts the gridded time series of the Vienna Mapping Functions (VMF) into the  GROOPS file format.

Gridded VMF data is available at: [https://vmf.geo.tuwien.ac.at/trop\\_products/GRID/](https://vmf.geo.tuwien.ac.at/trop_products/GRID/)

Name	Type	Annotation
outputfileVmfCoefficients	filename	
inputfile	filename	files must be given for each point in time
timeSeries	timeSeries	times of input files
deltaLambda	angle	[deg] sampling in longitude
deltaPhi	angle	[deg] sampling in latitude
isCellRegistered	boolean	grid points represent cells (VMF3), not grid corners (VMF1)

#### 4.16.94 ViennaMappingFunctionStation2File

Converts Vienna Mapping Functions (VMF) station time series into **GROOPS** file format.

Station-wise VMF data for GNSS is available at: [https://vmf.geo.tuwien.ac.at/trop\\_products/GNSS/](https://vmf.geo.tuwien.ac.at/trop_products/GNSS/)

Name	Type	Annotation
▶ outfileVmfCoefficients	filename	
▶* infileStationInfo	filename	
▶* infileStation	filename	
▶* infileVmf	filename	

## 4.17 Programs: Deprecated

### 4.17.1 GnssPrn2SvnBlockVariables

DEPRECATED. This program no longer works!

Setup up a **loop:platformEquipment** instead with

- **inputfilePlatform**: the old **inputfileTransmitterInfo**
- **equipmentType = gnssAntenna**
- **variableLoopName = block**
- **variableLoopSerial = svn**
- **variableLoopTimeStart = svnTimeStart**
- **variableLoopTimeEnd = svnTimeEnd**
- **condition:expression**
  - **expression = (svnTimeStart <= time) && (time < svnTimeEnd)**

Attribute this loop to programs, which uses the variables.

### 4.17.2 GridRectangular2NetCdf

DEPRECATED. Please use [GriddedData2NetCdf](#) or [GriddedDataTimeSeries2NetCdf](#) instead.

Name	Type	Annotation
<input type="checkbox"/> <code>outputfileNetCdf</code>	filename	file name of NetCDF output
<input type="checkbox"/> <code>inputfileGridRectangular</code>	filename	input grid sequence
<input type="checkbox"/> <code>times</code>	<code>timeSeries</code>	values for time axis (COARDS specification)
<input type="checkbox"/> <code>dataVariable</code>	sequence	metadata for data variables
<input type="checkbox"/> <code>selectDataField</code>	uint	input data column
<input type="checkbox"/> <code>name</code>	string	netCDF variable name
<input type="checkbox"/> <code>dataType</code>	choice	
<input type="checkbox"/> <code>double</code>		
<input type="checkbox"/> <code>float</code>		
<input type="checkbox"/> <code>int</code>		
<input type="checkbox"/> <code>attribute</code>	choice	netCDF attributes
<input type="checkbox"/> <code>text</code>	sequence	
<input type="checkbox"/> <code>name</code>	string	
<input type="checkbox"/> <code>value</code>	string	
<input type="checkbox"/> <code>value</code>	sequence	
<input type="checkbox"/> <code>name</code>	string	
<input type="checkbox"/> <code>value</code>	double	
<input type="checkbox"/> <code>dataType</code>	choice	
<input type="checkbox"/> <code>double</code>		
<input type="checkbox"/> <code>float</code>		
<input type="checkbox"/> <code>int</code>		
<input type="checkbox"/> <code>globalAttribute</code>	choice	additional meta data
<input type="checkbox"/> <code>text</code>	sequence	
<input type="checkbox"/> <code>name</code>	string	
<input type="checkbox"/> <code>value</code>	string	
<input type="checkbox"/> <code>value</code>	sequence	
<input type="checkbox"/> <code>name</code>	string	
<input type="checkbox"/> <code>value</code>	double	
<input type="checkbox"/> <code>dataType</code>	choice	
<input type="checkbox"/> <code>double</code>		
<input type="checkbox"/> <code>float</code>		
<input type="checkbox"/> <code>int</code>		

### 4.17.3 NetCdf2GridRectangular

DEPRECATED. Please use [NetCdf2GriddedData](#) or [NetCdf2GriddedDataTimeSeries](#) instead.

Name	Type	Annotation
outputfileGridRectangular	filename	One grid for each epoch in the NetCDF file is written. Use loopTimeVariable as template.
loopTimeVariable	string	
inputfileNetCdf	filename	
variableNameLongitude	string	name of NetCDF variable
variableNameLatitude	string	name of NetCDF variable
variableNameTime	string	name of NetCDF variable (leave blank for static grids)
variableNameData	string	name of NetCDF variable
R	double	reference radius for ellipsoidal coordinates
inverseFlattening	double	reference flattening for ellipsoidal coordinates

#### 4.17.4 Sinex2StationPosition

DEPRECATED. Please use [Sinex2StationPositions](#) instead.

Name	Type	Annotation
☛ <code>outputfileInstrument</code>	filename	loop variable is replaced with station name (e.g. wtzz)
☛ <code>inputfileSinex</code>	filename	SINEX file (.snx or .ssc)
☛ <code>inputfileDiscontinuities</code>	filename	discontinuities file per station; loop variable is replaced with station name (e.g. wtzz)
☛ <code>variableLoopStation</code>	string	variable name for station loop
☛ <code>stationName</code>	string	convert only these stations
☛ <code>timeSeries</code>	timeSeries	compute positions for these epochs based on velocity
☛ <code>extrapolateForward</code>	boolean	also compute positions for epochs after last interval defined in SINEX file
☛ <code>extrapolateBackward</code>	boolean	also compute positions for epochs before first interval defined in SINEX file

#### 4.17.5 Sinex2StationPostSeismicDeformation

DEPRECATED. Please use [Sinex2StationPositions](#) instead.

Name	Type	Annotation
outputfileInstrument	filename	deformation time series
inputfileSinex	filename	ITRF post-seismic deformation SINEX file
timeSeries	timeSeries	compute deformation for these epochs
stationName	string	
localLevelFrame	boolean	output in North, East, Up local-level frame

# Chapter 5

## Classes

This chapter details classes included in GROOPS, describing what they are and what they do. For usage examples see the cookbook in Chapter 3.

### 5.1 AutoregressiveModelSequence

Represents a sequence of multivariate autoregressive (AR) models with increasing order  $p$ . The AR models should be stored as [Matrix file](#) in the [GROOPS definition of AR models \(2.3\)](#). The required AR models can be computed with [CovarianceMatrix2AutoregressiveModel](#), and passed to this class through [inputfileAutoregressiveModel](#) in increasing order.

The main purpose of AutoregressiveModelSequence is to use AR models of the form

$$\mathbf{y}_e(t_i) = \sum_{k=1}^p \Phi_k^{(p)} \mathbf{y}_e(t_{i-k}) + \mathbf{w}(t_i), \quad \mathbf{w}(t_i) \sim \mathcal{N}(0, \Sigma_w^{(p)}), \quad (5.1)$$

to create pseudo-observation equations

$$0 = \bar{\Phi} \Delta \mathbf{y} + \bar{\mathbf{w}}, \quad \bar{\mathbf{w}} \sim \mathcal{N}(0, \bar{\Sigma}_{\bar{\mathbf{w}}}), \quad (5.2)$$

with

$$\bar{\Phi} = \begin{bmatrix} \mathbf{I} & & & & \\ -\Phi_1^{(1)} & \mathbf{I} & & & \\ -\Phi_2^{(2)} & -\Phi_1^{(2)} & \mathbf{I} & & \\ -\Phi_3^{(3)} & -\Phi_2^{(3)} & -\Phi_1^{(3)} & \mathbf{I} & \\ & -\Phi_3^{(3)} & -\Phi_2^{(3)} & -\Phi_1^{(3)} & \mathbf{I} \\ & & \ddots & \ddots & \ddots & \ddots \end{bmatrix}, \quad \bar{\Sigma}_{\bar{\mathbf{w}}} = \bar{\Sigma}_{\mathbf{w}} = \begin{bmatrix} \Sigma_w^{(0)} & & & & \\ \Sigma_w^{(1)} & \Sigma_w^{(0)} & & & \\ \Sigma_w^{(2)} & & \Sigma_w^{(1)} & & \\ & & & \Sigma_w^{(2)} & \\ & & & & \Sigma_w^{(3)} \\ & & & & & \ddots \end{bmatrix}. \quad (5.3)$$

used to constrain high-frequency temporal gravity field variations (see [KalmanSmoothenLeastSquares](#), [NormalsBuildShortTimeStaticLongTime](#), [PreprocessingSst](#)).

The corresponding normal equation coefficient matrix is given by

$$\bar{\Phi}^T \bar{\Sigma}_{\bar{\mathbf{w}}}^{-1} \bar{\Phi} \quad (5.4)$$

and if all AR models are estimated from the same sample its inverse is a block-Toeplitz covariance matrix

$$(\Sigma_{y_m})_{ij} = \begin{cases} \Sigma(|j - i|) & \text{for } i \leq j \\ \Sigma(|j - i|)^T & \text{otherwise} \end{cases}, \quad (5.5)$$

which can be computed using  [AutoregressiveModel2CovarianceMatrix](#).

A detailed description with applications can be found in: Kvas, A., Mayer-Gürr, T. GRACE gravity field recovery with background model uncertainties. J Geod 93, 2543–2552 (2019). <https://doi.org/10.1007/s00190-019-01314-1>

Name	Type	Annotation
 autoregressiveModelSequenceType	sequence	
 inputFileAutoregressiveModel	filename	matrix file containing an AR model
 sigma0	double	a-priori sigma for white noise covariance

## 5.2 Border

With this class you can select one or more regions on the surface of the Earth. In every instance of Border you can choose whether the specific region is excluded from the overall result with the switch **exclude**. To determine whether a specific point will be used furthermore the following algorithm will be applied: In a first step all points are selected if first border excludes points otherwise all points excluded. When every point will be tested for each instance of border from top to bottom. If the point is not in the selected region nothing happens. Otherwise it will included or excluded depending on the switch **exclude**.

First Example: The border excludes all continental areas. The result are points on the oceans only.

Second Example: First border describes the continent north america. The next borders excludes the great lakes and the last border describes Washington island. In this configuration points are selected if they are inside north america but not in the area of the great lakes. But if the point is on Washington island it will be included again.

### 5.2.1 Rectangle

The region is restricted along lines of geographical coordinates. **minPhi** and **maxPhi** describe the lower and the upper bound of the region. **minLambda** and **maxLambda** define the left and right bound.

Name	Type	Annotation
<b>minLambda</b>	angle	
<b>maxLambda</b>	angle	
<b>minPhi</b>	angle	
<b>maxPhi</b>	angle	
<b>exclude</b>	boolean	dismiss points inside

### 5.2.2 Cap

The region is defined by a spherical cap with the center given in geographical coordinates longitude (**lambdaCenter**) and latitude (**phiCenter**). The radius of the cap is given as aperture angle **psi**.

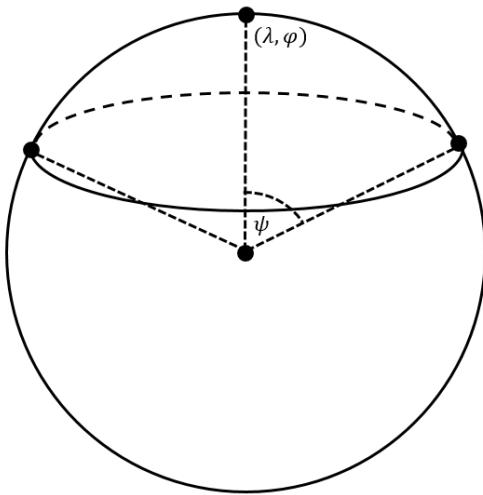


Figure 5.1: spherical cap

Name	Type	Annotation
lambdaCenter	angle	longitude of the center of the cap
phiCenter	angle	latitude of the center of the cap
psi	angle	aperture angle (radius)
exclude	boolean	dismiss points inside

### 5.2.3 Polygon

The region is defined by **inputfilePolygon** containing one or more polygons given in longitude and latitude. An additional **buffer** around the polygon can be defined. Use a negative value to shrink the polygon area.

Name	Type	Annotation
inputfilePolygon	filename	
buffer	double	buffer around polygon [km], \$0: inside
exclude	boolean	dismiss points inside

## 5.3 Condition

Test for conditions. See [Loop and conditions \(1.3\)](#) for usage.

### 5.3.1 FileExist

Check for a file or directory existing. Supports wildcards \* for any number of characters and ? for exactly one character.

Name	Type	Annotation
file	filename	supports wildcards: * and ?

### 5.3.2 Command

Execute command and check success.

Name	Type	Annotation
command	filename	
silently	boolean	without showing the output.

### 5.3.3 Expression

Evaluate expression.

Name	Type	Annotation
expression	expression	

### 5.3.4 Matrix

Evaluate elements of a `matrix` based on an expression. If `all=yes`, all elements of the matrix must evaluate to true for the condition to be fulfilled, otherwise any element evaluating to true is sufficient.

Name	Type	Annotation
matrix	matrixGenerator	expression is evaluated for each element of resulting matrix
expression	expression	(variable: data) evaluated for each element
all	boolean	all (=yes)/any (=no) elements must evaluate to true

### 5.3.5 MatrixEmpty

Evaluate if `matrix` (or `instrument`) file is empty/has zero size.

Name	Type	Annotation
inputfileMatrix	filename	

### 5.3.6 StringContainsPattern

Determines if there is a match between a pattern or a regular expression and some subsequence in a string.

Name	Type	Annotation
▶ string	filename	should contain a {variable}
▶ pattern	filename	
▶ isRegularExpression	boolean	pattern is a regular expression
▶ caseSensitive	boolean	treat lower and upper case as distinct

### 5.3.7 StringMatchPattern

Determines if a pattern or a regular expression matches the entire string.

Name	Type	Annotation
▶ string	filename	should contain a {variable}
▶ pattern	filename	
▶ isRegularExpression	boolean	pattern is a regular expression
▶ caseSensitive	boolean	treat lower and upper case as distinct

### 5.3.8 And

All conditions must be met (with short-circuit evaluation).

Name	Type	Annotation
▶ condition	condition	

### 5.3.9 Or

One of the conditions must be met (with short-circuit evaluation).

Name	Type	Annotation
▶ condition	condition	

### 5.3.10 Not

The result of the condition is inverted.

Name	Type	Annotation
▶ condition	condition	

## 5.4 CovariancePod

Provides arc wise covariance matrices for precise orbit data. Temporal correlations are modeled in the orbit system (along, cross, radial). The **inputfileCovarianceFunction** provides temporal covariance functions for each axis. From the diagonal matrix for each time step

$$\text{Cov}_{3 \times 3}(t) = \text{diag}(\text{cov}_x(t), \text{cov}_y(t), \text{cov}_z(t)) \quad (5.6)$$

the Toeplitz covariance matrix for an arc is constructed

$$\mathbf{C} = \begin{pmatrix} \text{Cov}(t_0) & \text{Cov}(t_1) & \dots \\ \text{Cov}(t_1) & \text{Cov}(t_0) & \text{Cov}(t_1) & \dots \\ \dots & \text{Cov}(t_1) & \text{Cov}(t_0) & \text{Cov}(t_1) & \dots \\ & \dots & \ddots & \ddots & \ddots & \dots \end{pmatrix} \quad (5.7)$$

The epoch wise  $3 \times 3$  covariance matrices given by **inputfileCovariancePodEpoch** are eigen value decomposed

$$\mathbf{C}_{3 \times 3}(t_i) = \mathbf{Q} \boldsymbol{\Lambda} \mathbf{Q}^T, \quad (5.8)$$

where  $\mathbf{Q}$  is an orthgonal matrix and  $\boldsymbol{\Lambda}$  diagonal. This used to split the covariances matrices

$$\mathbf{C}_{3 \times 3}(t_i) = \mathbf{D}(t_i) \mathbf{D}(t_i)^T = (\mathbf{Q} \boldsymbol{\Lambda}^{1/2} \mathbf{Q}^T) (\mathbf{Q} \boldsymbol{\Lambda}^{1/2} \mathbf{Q}^T)^T, \quad (5.9)$$

and to compose a block diagonal matrix for an arc

$$\mathbf{D} = \text{diag}(\mathbf{D}(t_1), \mathbf{D}(t_2), \dots, \mathbf{D}(t_n)). \quad (5.10)$$

The complete covariance matrix of an arc is given by

$$\mathbf{C}_{arc} = \sigma_0^2 \sigma_{arc}^2 \mathbf{D} \mathbf{C} \mathbf{D}^T + \text{diag}(\sigma_1^2 \mathbf{I}_{3 \times 3}, \sigma_2^2 \mathbf{I}_{3 \times 3}, \dots, \sigma_n^2 \mathbf{I}_{3 \times 3}) \quad (5.11)$$

where **sigma**  $\sigma_0$  is an overall factor and the arc specific factors  $\sigma_{arc}$  can be provided with **inputfileSigmasPerArc**. The last matrix can be used to downweight outliers in single epochs and will be added if **inputfileSigmasPerEpoch** is provided.

Name	Type	Annotation
<b>covariancePodType</b>	sequence	
<b>sigma</b>	double	general variance factor
<b>inputfileSigmasPerArc</b>	filename	different accuracies for each arc (multiplied with sigma)
<b>inputfileSigmasPerEpoch</b>	filename	different accuracies for each epoch (added)
<b>inputfileCovarianceFunction</b>	filename	covariances in time for along, cross, and radial direction
<b>inputfileCovariancePodEpoch</b>	filename	$3 \times 3$ epoch wise covariances

## 5.5 CovarianceSst

Provides arc wise covariance matrices for satellite-to-satellite observations (SST). The **inputfileCovarianceFunction** provides a temporal covariance function. From it the Toeplitz covariance matrix is constructed

$$\mathbf{C} = \begin{pmatrix} \text{cov}(t_0) & \text{cov}(t_1) & \cdots & & \\ \text{cov}(t_1) & \text{cov}(t_0) & \text{cov}(t_1) & \cdots & \\ \cdots & \text{cov}(t_1) & \text{cov}(t_0) & \text{cov}(t_1) & \cdots \\ & \cdots & \ddots & \ddots & \ddots & \cdots \end{pmatrix} \quad (5.12)$$

The complete covariance matrix of an arc is given by

$$\mathbf{C}_{arc} = \sigma_0^2 \sigma_{arc}^2 \mathbf{C} + \sigma_{S,arc}^2 \mathbf{S}_{arc} + \text{diag}(\sigma_1^2, \sigma_2^2, \dots, \sigma_n^2) \quad (5.13)$$

where **sigma**  $\sigma_0$  is an overall factor and the arc specific factors  $\sigma_{arc}$  can be provided with **inputfileSigmasPerArc**. The second term describes general covariance matrices for each arc **inputfileCovarianceMatrixArc** together with the factors  $\sigma_{S,arc}$  from **sigmasCovarianceMatrixArc**. The last matrix can be used to downweight outliers in single epochs and will be added if **inputfileSigmasPerEpoch** is provided.

Name	Type	Annotation
covarianceSstType	sequence	
sigma	double	general variance factor
inputfileSigmasPerArc	filename	different accuracies for each arc (multiplied with sigma)
inputfileSigmasPerEpoch	filename	different accuracies for each epoch (added)
inputfileCovarianceFunction	filename	covariance function in time
inputfileCovarianceMatrixArc	filename	one matrix file per arc. Use {arcNo} as template
sigmasCovarianceMatrixArc	filename	vector with one sigma for each covarianceMatrixArc

## 5.6 DigitalFilter

Digital filter implementation for the filtering of equally spaced time series. This class implements the filter equations as

$$\sum_{l=0}^Q a_l y_{n-l} = \sum_{k=-p_0}^{P-p_0-1} b_k x_{n-k}, \quad a_0 = 1, \quad (5.14)$$

where  $Q$  is the autoregressive (AR) order and  $P$  is the moving average (MA) order. Note that the MA part can also be non-causal. The characteristics of a filter cascade can be computed by the programs [DigitalFilter2FrequencyResponse](#) and [DigitalFilter2ImpulseResponse](#). To apply a filter cascade to a time series (or an instrument file) use [InstrumentFilter](#). Each filter can be applied in forward and backward direction by setting [backwardDirection](#). If the same filter is applied in both directions, the combined filter has zero phase and the squared magnitude response. Setting [inFrequencyDomain](#) to true applies the transfer function of the filter to the DFT of the input and synthesizes the result, i.e.:

$$y_n = \mathcal{F}^{-1}\{H \cdot \mathcal{F}\{x_n\}\}. \quad (5.15)$$

This is equivalent to setting [padType](#) to [periodic](#).

To reduce warmup effects, the input time series can be padded by choosing a [padType](#):

- [none](#): no padding is applied
- [zero](#): zeros are appended at the beginning and end of the input time series
- [constant](#): the beginning of the input time series is padded with the first value, the end is padded with the last value
- [periodic](#): periodic continuation of the input time series (i.e. the beginning is padded with the last epochs and the end is padded with the first epochs)
- [symmetric](#): beginning and end are reflected around the first and last epoch respectively

### 5.6.1 MovingAverage

Moving average (boxcar) filter. For odd lengths, this filter is symmetric and has therefore no phase shift. For even lengths, a phase shift of half a cycle is introduced.

$$y_n = \sum_{k=-\lfloor \frac{P}{2} \rfloor}^{\lfloor \frac{P}{2} \rfloor} \frac{1}{P} x_{n-k}$$

Name	Type	Annotation
<a href="#">length</a>	uint	number of epochs in averaging operator
<a href="#">inFrequencyDomain</a>	boolean	apply filter in frequency domain
<a href="#">padType</a>	choice	
<a href="#">none</a>		no padding is applied
<a href="#">zero</a>		zero padding
<a href="#">constant</a>		pad using first and last value
<a href="#">periodic</a>		periodic continuation of matrix
<a href="#">symmetric</a>		symmetric continuation around the matrix edges

## 5.6.2 Median

Moving median filter of length  $n$ . The filter output at epoch  $k$  is the median of the set start at  $k - n/2$  to  $k + n/2$ . The filter length  $n$  should be uneven to avoid a phase shift.

Name	Type	Annotation
length	uint	length of the moving window [epochs]
padType	choice	
none		no padding is applied
zero		zero padding
constant		pad using first and last value
periodic		periodic continuation of matrix
symmetric		symmetric continuation around the matrix edges

## 5.6.3 Derivative

Symmetric MA filter for numerical differentiation using polynomial approximation. The input time series is approximated by a moving polynomial of degree **polynomialDegree**, by solving

$$\begin{bmatrix} x(t_k + \tau_0) \\ \vdots \\ x(t_k + \tau_M) \end{bmatrix} = \begin{bmatrix} 1 & \tau_0 & \tau_0^2 & \cdots & \tau_0^M \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \tau_M & \tau_M^2 & \cdots & \tau_M^M \end{bmatrix} \begin{bmatrix} a_0 \\ \vdots \\ a_M \end{bmatrix} \quad \text{with } \tau_j = (j - M/2) \cdot \Delta t, \quad (5.16)$$

for each time step  $t_k$  ( $\Delta t$  is the **sampling** of the time series). The filter coefficients for the  $k$ -th derivative are obtained by taking the appropriate row of the inverse coefficient matrix  $\mathbf{W}$ :

$$b_n = \prod_{i=0}^{k-1} (k-i) \mathbf{w}_{2,:}. \quad (5.17)$$

The **polynomialDegree** should be even if no phase shift should be introduced.

Name	Type	Annotation
polynomialDegree	uint	degree of approximation polynomial
derivative	uint	take kth derivative
sampling	double	assumed time step between points
padType	choice	
none		no padding is applied
zero		zero padding
constant		pad using first and last value
periodic		periodic continuation of matrix
symmetric		symmetric continuation around the matrix edges

## 5.6.4 Integral

Numerical integration using polynomial approximation. The input time series is approximated by a moving polynomial of degree **polynomialDegree** by solving

$$\begin{bmatrix} x(t_k + \tau_0) \\ \vdots \\ x(t_k + \tau_M) \end{bmatrix} = \begin{bmatrix} 1 & \tau_0 & \tau_0^2 & \cdots & \tau_0^M \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \tau_M & \tau_M^2 & \cdots & \tau_M^M \end{bmatrix} \begin{bmatrix} a_0 \\ \vdots \\ a_M \end{bmatrix} \quad \text{with } \tau_j = (j - M/2) \cdot \Delta t, \quad (5.18)$$

for each time step  $t_k$  ( $\Delta t$  is the **sampling** of the time series). The numerical integral for each time step  $t_k$  is approximated by the center interval of the estimated polynomial.

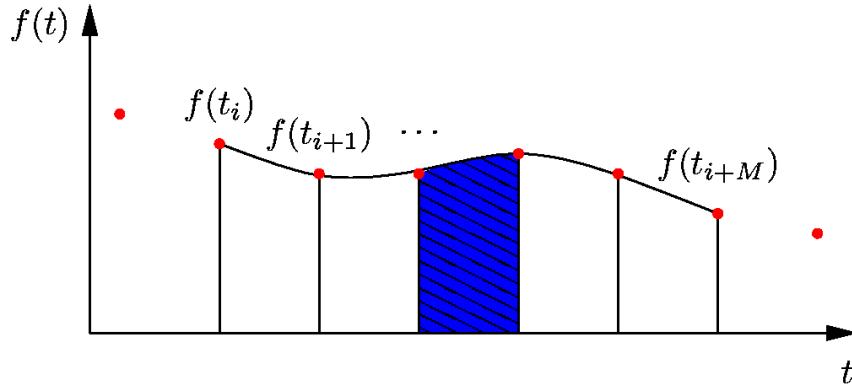


Figure 5.2: Numerical integration by polynomial approximation.

**polynomialDegree** should be even to avoid a phase shift.

Name	Type	Annotation
polynomialDegree	uint	degree of approximation polynomial
sampling	double	assumed time step between points
padType	choice	
none		no padding is applied
zero		zero padding
constant		pad using first and last value
periodic		periodic continuation of matrix
symmetric		symmetric continuation around the matrix edges

### 5.6.5 Correlation

Correlation ( $\rho$ ) of **corr** is introduced into the time series:

$$y_n = \rho \cdot y_{n-1} + \sqrt{1 - \rho^2} x_n. \quad (5.19)$$

Name	Type	Annotation
correlation	double	correlation
backwardDirection	boolean	apply filter in backward direction
inFrequencyDomain	boolean	apply filter in frequency domain
padType	choice	
none		no padding is applied
zero		zero padding
constant		pad using first and last value
periodic		periodic continuation of matrix
symmetric		symmetric continuation around the matrix edges

### 5.6.6 GraceLowpass

Low pass and differentiation filter as used for GRACE KBR and ACC data in the Level1A processing.

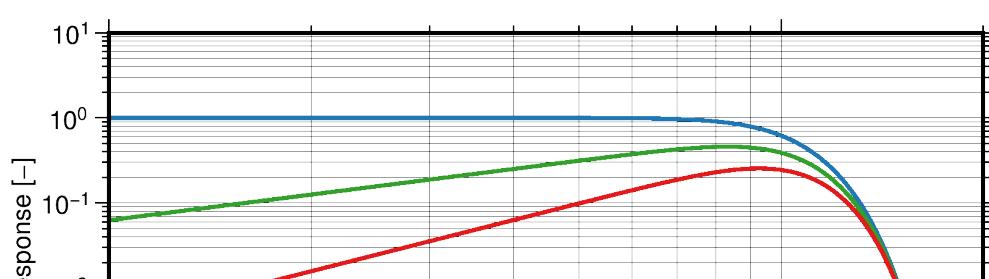
Name	Type	Annotation
rawDataRate	double	sampling frequency in Hz (fs).
convolutionNumber	uint	number of self convolutions of the filter kernel
fitInterval	double	length of the filter kernel [seconds]
lowPassBandwidth	double	target low pass bandwidth
normFrequency	double	norm filter at this frequency [Hz] (default: GRACE dominant (J2) signal frequency)
reduceQuadraticFit	boolean	remove-\$filter-\$restore quadratic fit
derivative	choice	
derivative1st	range	rate
derivative2nd	range	acceleration
inFrequencyDomain	boolean	apply filter in frequency domain
padType	choice	
none		no padding is applied
zero		zero padding
constant		pad using first and last value
periodic		periodic continuation of matrix
symmetric		symmetric continuation around the matrix edges

### 5.6.7 Butterworth

Digital implementation of the Butterworth filter. The design of the filter is done by modifying the analog (continuous time) transfer function, which is then transformed into the digital domain by using the bilinear transform. The filter coefficients are then determined by a least squares adjustment in time domain.

The **filterType** can be **lowpass**, **highpass**, where one cutoff frequency has to be specified, and **bandpass** and **bandstop** where two cutoff frequencies have to be specified. Cutoff frequencies must be given as normalized frequency  $w_n = f / f_{nyq}$ . For a cutoff frequency of 30 mHz for a time series sampled with 5 seconds gives a normalized frequency of  $0.03/0.1 = 0.3$ .

Name	Type	Annotation
order	uint	filter order
type	choice	filter type
lowpass	sequence	
Wn	double	normalized cutoff frequency ( $f_c / f_{nyq}$ )
highpass	sequence	
Wn	double	normalized cutoff frequency ( $f_c / f_{nyq}$ )
bandpass	sequence	
Wn1	double	lower normalized cutoff frequency ( $f_c / f_{nyq}$ )
Wn2	double	upper normalized cutoff frequency ( $f_c / f_{nyq}$ )
bandstop	sequence	
Wn1	double	lower normalized cutoff frequency ( $f_c / f_{nyq}$ )
Wn2	double	upper normalized cutoff frequency ( $f_c / f_{nyq}$ )
backwardDirection	boolean	apply filter in backward direction
inFrequencyDomain	boolean	apply filter in frequency domain
padType	choice	
none		no padding is applied
zero		zero padding
constant		pad using first and last value
periodic		periodic continuation of matrix
symmetric		symmetric continuation around the matrix edges



### 5.6.8 File

Read filter coefficients of (5.14) from a coefficient file. One column might define the index  $n$  of the coefficients  $a_n$  and  $b_n$  in the other columns.

Name	Type	Annotation
inputfileMatrix	filename	matrix with filter coefficients
index	expression	index of coefficients (input columns are named data0, data1, ...)
bn	expression	MA coefficients (moving average) (input columns are named data0, data1, ...)
an	expression	AR coefficients (autoregressive) (input columns are named data0, data1, ...)
backwardDirection	boolean	apply filter in backward direction
inFrequencyDomain	boolean	apply filter in frequency domain
padType	choice	
none		no padding is applied
zero		zero padding
constant		pad using first and last value
periodic		periodic continuation of matrix
symmetric		symmetric continuation around the matrix edges

### 5.6.9 Wavelet

Filter representation of a wavelet.

Name	Type	Annotation
inputfileWavelet	filename	wavelet coefficients
type	choice	filter type
lowpass		
highpass		
level	uint	compute filter for specific decomposition level
backwardDirection	boolean	apply filter in backward direction
inFrequencyDomain	boolean	apply filter in frequency domain
padType	choice	
none		no padding is applied
zero		zero padding
constant		pad using first and last value
periodic		periodic continuation of matrix
symmetric		symmetric continuation around the matrix edges

### 5.6.10 Notch

Implemented after Christian Siemes' dissertation, page 106.

Name	Type	Annotation
notchFrequency	double	normalized notch frequency $w_n = (f_n/f_{nyq})$
bandWidth	double	bandwidth at -3db. Quality factor of filter $Q = w_n/bw$
backwardDirection	boolean	apply filter in backward direction
inFrequencyDomain	boolean	apply filter in frequency domain
padType	choice	
none		no padding is applied

---

	zero	zero padding
	constant	pad using first and last value
	periodic	periodic continuation of matrix
	symmetric	symmetric continuation around the matrix edges

---

### 5.6.11 Decorrelation

Moving average decorrelation filter based on eigendecomposition of a Toeplitz covariance matrix.

---

Name	Type	Annotation
	inputfileCovarianceFunction	filename covariance function of time series
	inFrequencyDomain	boolean apply filter in frequency domain
	padType	choice
	none	no padding is applied
	zero	zero padding
	constant	pad using first and last value
	periodic	periodic continuation of matrix
	symmetric	symmetric continuation around the matrix edges

---

### 5.6.12 TimeLag

Lag operator in digital filter representation.

---

Name	Type	Annotation
	lag	int lag epochs: 1 (lag); -1 (lead)
	inFrequencyDomain	boolean apply filter in frequency domain
	padType	choice
	none	no padding is applied
	zero	zero padding
	constant	pad using first and last value
	periodic	periodic continuation of matrix
	symmetric	symmetric continuation around the matrix edges

---

### 5.6.13 ReduceFilterOutput

Removes the filtered signal from the input, i.e. the input is passed through a **digitalFilter** with a frequency response of  $1 - H(f)$ .

---

Name	Type	Annotation
	<b>digitalFilter</b>	remove filter output from input signal

---

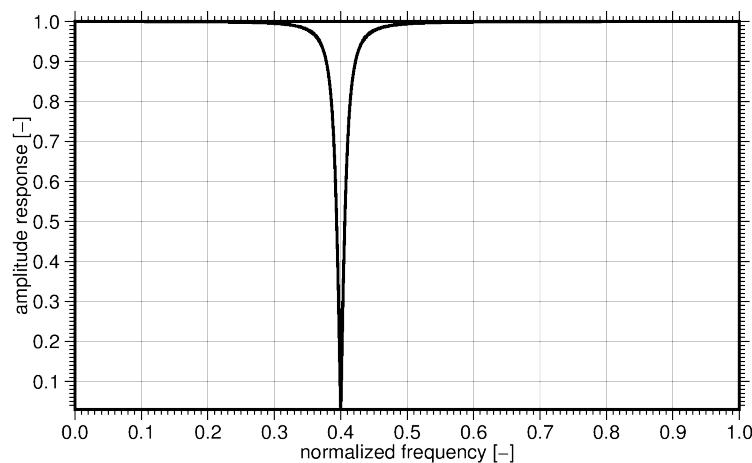


Figure 5.4: Amplitude response of a notch filter of order three with default settings.

## 5.7 Doodson

This is a string which describes a tidal frequency either coded as Doodson number or using Darwin's name, e.g. 255.555 or M2.

The following names are defined:

- 055.565: om1
- 055.575: om2
- 056.554: sa
- 056.555: sa
- 057.555: ssa
- 058.554: sta
- 063.655: msm
- 065.455: mm
- 073.555: msf
- 075.555: mf
- 083.655: mstm
- 085.455: mtm
- 093.555: msq
- 093.555: msqm
- 125.755: 2q1
- 127.555: sig1
- 127.555: sigma1
- 135.655: q1
- 137.455: ro1
- 137.455: rho1
- 145.555: o1
- 147.555: tau1
- 155.655: m1
- 157.455: chi1
- 162.556: pi1
- 163.555: p1
- 164.555: s1
- 165.555: k1
- 166.554: psi1

- 167.555: fi1
- 167.555: phi1
- 173.655: the1
- 173.655: theta1
- 175.455: j1
- 183.555: so1
- 185.555: oo1
- 195.455: v1
- 225.855: 3n2
- 227.655: eps2
- 235.755: 2n2
- 237.555: mu2
- 237.555: mi2
- 245.655: n2
- 247.455: nu2
- 247.455: ni2
- 253.755: gam2
- 254.556: alf2
- 255.555: m2
- 256.554: bet2
- 257.555: dlt2
- 263.655: la2
- 263.655: lmb2
- 263.655: lambda2
- 265.455: l2
- 271.557: 2t2
- 272.556: t2
- 273.555: s2
- 274.554: r2
- 275.555: k2
- 283.655: ksi2
- 285.455: eta2
- 355.555: m3

- 381.555: t3
- 382.555: s3
- 383.555: r3
- 435.755: n4
- 445.655: mn4
- 455.555: m4
- 473.555: ms4
- 491.555: s4
- 655.555: m6
- 855.555: m8

## 5.8 EarthRotation

This class realize the transformation between a terrestrial reference frame (TRF) and a celestial reference frame (CRF).

### 5.8.1 File

This class realize the transformation by interpolation from file. This file can be created with [EarthOrientationParameterTimeSeries](#).

Name	Type	Annotation
inputfileEOP	filename	
interpolationDegree	uint	for polynomial interpolation

### 5.8.2 Iers2010

This class realize the transformation according to the IERS2010 conventions given by the *International Earth Rotation and Reference Systems Service* (IERS). A file with the earth orientation parameter is needed ([inputfileEOP](#)).

Name	Type	Annotation
inputfileEOP	filename	
truncatedNutation	boolean	use truncated nutation model (IAU2006B)

### 5.8.3 Iers2010b

This class realize the transformation according to the IERS2010 conventions given by the *International Earth Rotation and Reference Systems Service* (IERS). A file with the earth orientation parameter is needed ([inputfileEOP](#)). Includes additional high-frequency EOP models ([inputfileDoodsonEOP](#)).

Name	Type	Annotation
inputfileEOP	filename	
inputfileDoodsonEOP	filename	

### 5.8.4 Iers2003

This class realize the transformation according to IERS2003 conventions given by the *International Earth Rotation and Reference Systems Service* (IERS). A file with the earth orientation parameter is needed ([inputfileEOP](#)).

The following subroutines are used:

- BPN2000.f,
- ERA2000.f,
- pmsdnut.f,
- POM2000.f,

- SP2000.f,
- T2C2000.f,
- XYS2000A.f

from <ftp://maia.usno.navy.mil/conv2000/chapter5/> and

- orthoeop.f

from <ftp://maia.usno.navy.mil/conv2000/chapter8/>

Name	Type	Annotation
inputfileEOP	filename	

### 5.8.5 Iers1996

Very old.

Name	Type	Annotation
inputfileEOP	filename	
inputfileNutation	filename	

### 5.8.6 Gmst

The transformation is realized as rotation about the z-axis. The angle ist given by the Greenwich Mean Siderial Time (GMST).

```
Double Tu0 = (timeUTC.mjdInt()-51544.5)/36525.0;

Double GMST0 = (6.0/24 + 41.0/(24*60) + 50.54841/(24*60*60))
    + (8640184.812866/(24*60*60))*Tu0
    + (0.093104/(24*60*60))*Tu0*Tu0
    + (-6.2e-6/(24*60*60))*Tu0*Tu0*Tu0;
Double r = 1.002737909350795 + 5.9006e-11*Tu0 - 5.9e-15*Tu0*Tu0;
GMST = fmod(2*PI*(GMST0 + r * timeUTC.mjdMod()), 2*PI);
```

### 5.8.7 Earth Rotation Angle (ERA)

The transformation is realized as rotation about the z-axis. The angle ist given by the Earth Rotation Angle (ERA).

### 5.8.8 Z-Axis

The transformation is realized as rotation about the z-axis. You must specify the angle (**initialAngle**) at **time0** and the angular velocity (**angularVelocity**).

Name	Type	Annotation
initialAngle	double	Angle at time0 [rad]
angularVelocity	double	[rad/s]
time0	time	

### 5.8.9 StarCamera

This class reads quaternions from an instrument file and interpolates to the given time stamp.

Name	Type	Annotation
inputfileStarCamera	filename	
interpolationDegree	uint	degree of interpolation polynomial

### 5.8.10 MoonRotation

This class realizes the transformation between the moon-fixed system (Principal Axis System (PA) or Mean Earth System (ME)) and the ICRS according to the JPL ephemeris file.

Name	Type	Annotation
inputfileEphemerides	filename	librations
moonfixedSystem	choice	
PA		Principal Axis System
ME		Mean Earth System

## 5.9 Eclipse

Shadowing of satellites by moon and Earth provided as factor between [0, 1] with 0: full shadow and 1: full sun light.

### 5.9.1 Conical

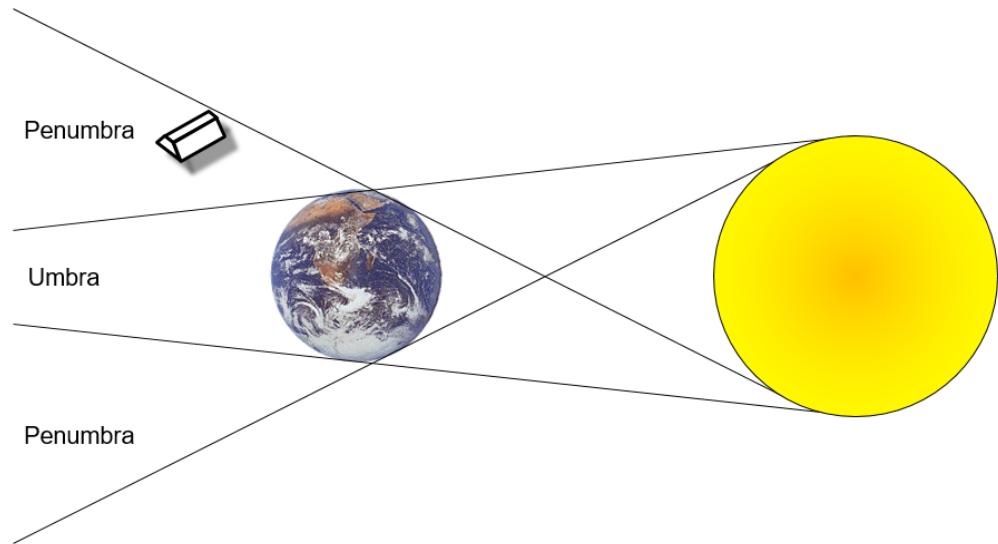


Figure 5.5: Modelling umbra and penumbra.

### 5.9.2 SOLAARS

Earth's penumbra modeling with Solar radiation pressure with Oblateness and Lower Atmospheric Absorption, Refraction, and Scattering (SOLAARS). See Robertson, Robbie. (2015), Highly Physical Solar Radiation Pressure Modeling During Penumbra Transitions (pp. 67-75).

## 5.10 Ephemerides

Ephemerides of Sun, Moon and planets. The coordinate system is defined as center of  origin.

## 5.11 JPL

Using DExxx ephemerides from NASA Jet Propulsion Laboratory (JPL).

Name	Type	Annotation
 inputfileEphemerides	filename	
 origin	planet	center of coordinate system

## 5.12 Forces

This class provides the forces acting on a satellite. This encompasses **gravityfield**, **tides** and **miscAccelerations**.

Name	Type	Annotation
forcesType	sequence	
gravityfield	gravityfield	
tides	tides	
miscAccelerations	miscAccelerations	

## 5.13 GnssAntennaDefintionList

Provides a list of GnssAntennaDefinitions as used in [GnssAntennaDefinitionCreate](#).

### 5.13.1 New

Creates a new antenna.

Name	Type	Annotation
name	string	
serial	string	
radome	string	
comment	string	
pattern	sequence	
type	gnssType	pattern matching of observation types
offsetX	double	[m] antenna center offset
offsetY	double	[m] antenna center offset
offsetZ	double	[m] antenna center offset
deltaAzimuth	angle	[degree] step size
deltaZenith	angle	[degree] step size
maxZenith	angle	[degree]
values	expression	[m] expression (zenith, azimuth: variables)

### 5.13.2 FromFile

Select all or the first antenna from an [antenna definition file](#) which matches the wildcards.

Name	Type	Annotation
inputfileAntennaDefinition	filename	
name	string	
serial	string	
radome	string	
onlyFirstMatch	boolean	otherwise all machting antennas included

### 5.13.3 FromStationInfo

Select all antennas from an [antenna definition file](#) which are used by a station within a defined time interval. With [specializeAntenna](#) an individual antenna is created for each different serial number using the general type specific values from file.

Name	Type	Annotation
inputfileStationInfo	filename	
inputfileAntennaDefinition	filename	
timeStart	time	only antennas used in this time interval
timeEnd	time	only antennas used in this time interval
specializeAntenna	boolean	e.g. separate different serial numbers from stationInfo

### 5.13.4 Resample

The azimuth and elevation dependent antenna center variations (patterns) of all **► antennas** are resampled to a new resolution.

Name	Type	Annotation
► antenna	gnssAntennaDefintionList	
► deltaAzimuth	angle	[degree] step size, empty: no change
► deltaZenith	angle	[degree] step size, empty: no change
► maxZenith	angle	[degree], empty: no change

### 5.13.5 Transform

This class can be used to separate general antenna patterns for different **► gnssTypes**. If the **► antennas** contain only one pattern for all GPS observations on the L1 frequency (**\*1\*G\*\***), the **► patternTypes=C1\*G\*\*** and **L1\*G\*\*** create two patterns with the **\*1\*G\*\*** pattern as template. The first matching pattern in the **► antenna** is used as template. Also new **► additionalPattern** can be added (e.g. for **\*5\*G\*\***). With **► addExistingPatterns** all already existing patterns that don't match completely to any of the above are added.

Name	Type	Annotation
► antenna	gnssAntennaDefintionList	
► patternTypes	gnssType	gnssType for each pattern (first match is used)
► additionalPattern	sequence	additional new patterns
└► type	gnssType	pattern matching of observation types
└► offsetX	double	[m] antenna center offset
└► offsetY	double	[m] antenna center offset
└► offsetZ	double	[m] antenna center offset
└► deltaAzimuth	angle	[degree] step size
└► deltaZenith	angle	[degree] step size
└► maxZenith	angle	[degree]
└► values	expression	[m] expression (zenith, azimuth: variables)
► addExistingPatterns	boolean	add existing patterns that don't match completely any of the above

### 5.13.6 Rename

Replaces parts of the descrption of **► antennas**. The star "\*" left this part untouched.

Name	Type	Annotation
► antenna	gnssAntennaDefintionList	
► name	string	*: left this part untouched
► serial	string	*: left this part untouched
► radome	string	*: left this part untouched
► comment	string	*: left this part untouched

## 5.14 GnssParametrization

This class defines the models and parameters of the linearized observation equations for all phase and code measurements (see [GnssProcessing](#))

$$\mathbf{l} - \mathbf{f}(\mathbf{x}_0) = \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}_0} \Delta \mathbf{x} + \epsilon, \quad (5.20)$$

where the left side is the observation vector minus the effects computed from the a priori models. After each least squares adjustment (see [GnssProcessing:processingStep:estimate](#)) the a priori parameters are updated

$$\mathbf{x}_0 := \mathbf{x}_0 + \Delta \hat{\mathbf{x}}. \quad (5.21)$$

The vector  $\mathbf{x}_0$  can be written with [GnssProcessing:processingStep:writeAprioriSolution](#). Any [outputfiles](#) defined in the parametrizations are written with [GnssProcessing:processingStep:writeResults](#).

Each parametrization (and possible constraint equations) has a [name](#) which enables activating/deactivating the estimation of subsets of  $\Delta \mathbf{x}$  with [GnssProcessing:processingStep:selectParametrizations](#). The a priori model  $\mathbf{f}(\mathbf{x}_0)$  is unaffected and is always reduced.

The model for the different observation types can be described as

$$\begin{aligned} f[\tau\nu a]_r^s(\mathbf{x}) &= \text{geometry}(\mathbf{r}_r^s) + \text{clock}^s(t) + \text{clock}_r(t) \\ &+ \text{iono}([tn], t, \mathbf{r}_r^s) + \text{tropo}(t, \mathbf{r}_r^s) \\ &+ \text{ant}[\tau\nu a]^s + \text{ant}[\tau\nu a]_r \\ &+ \text{bias}[\tau\nu a]^s + \text{bias}[\tau\nu a]_r + \lambda[Ln]N[Lna]_r^s + \text{other}(\dots) + \epsilon[\tau\nu a]_r^s \end{aligned} \quad (5.22)$$

The notation  $[\tau\nu a]_r^s$  describes the attribution to a signal type  $\tau$  (i.e., C or L), frequency  $\nu$ , signal attribute  $a$  (e.g., C, W, Q, X), transmitting satellite  $s$ , and observing receiver  $r$ . It follows the [RINEX 3 definition](#), see [GnssType](#) (5.18).

See also [GnssProcessing](#).

### 5.14.1 IonosphereSTEC

The influence of the ionosphere is modelled by a STEC parameter (slant total electron content) in terms of  $[TECU]$  between each transmitter and receiver at each epoch. These parameters are pre-eliminated from the observation equations before accumulating the normal equations. This is similar to using the ionosphere-free linear combination as observations but only one STEC parameter is needed for an arbitrary number of observation types.

The influence to the code and phase observation is modelled as

$$\begin{aligned} f_C(STEC) &= \frac{40.3}{f^2} STEC + \frac{7525\mathbf{b}^T\mathbf{k}}{f^3} STEC + \frac{r}{f^4} STEC^2 \\ f_L(STEC) &= -\frac{40.3}{f^2} STEC - \frac{7525\mathbf{b}^T\mathbf{k}}{2f^3} STEC - \frac{r}{3f^4} STEC^2 + \text{bending}(E)STEC^2 \end{aligned} \quad (5.23)$$

The second order term depends on the [magnetosphere](#)  $\mathbf{b}$  and the direction of the signal  $\mathbf{k}$ .

If further information about the ionosphere is available (in the form of a prior model or as additional parametrizations such as [parametrization:ionosphereMap](#) or [parametrization:ionosphereVTEC](#)) the STEC parameters describe local and short-term scintillations. The STEC parameters are estimated as additions to the model and it is advised to constrain them towards zero with a standard deviation of [sigmaSTEC](#).

Name	Type	Annotation
▶ name	string	used for parameter selection
▶ apply2ndOrderCorrection	boolean	apply ionospheric correction
▶ apply3rdOrderCorrection	boolean	apply ionospheric correction
▶ applyBendingCorrection	boolean	apply ionospheric correction
▶ magnetosphere	magnetosphere	
▶ nameConstraint	string	used for parameter selection
▶ sigmaSTEC	double	(0 = unconstrained) sigma [TECU] for STEC constraint

### 5.14.2 IonosphereVTEC

The influence of the ionosphere is modelled by a VTEC parameter (vertical total electron content) in terms of [TECU] for every selected receiver each epoch. The slant TEC is computed using the elevation  $E$  dependent Modified Single-Layer Model (MSLM) mapping function

$$STEC = \frac{VTEC}{\cos z'} \quad \text{with} \quad \sin z' = \left( \frac{R}{R+H} \right) \sin(\alpha(\pi/2 - E)) \quad (5.24)$$

inserted into eq. (5.23).

The result is written as a `times series file` at epochs with observations depending on **GnssProcessing:processingStep:selectEpochs**.

This class provides a simplified model of the ionosphere for single receivers and enables the separation of the TEC and signal biases, meaning **parametrization:tecBiases** becomes estimable. Local and short-term scintillations should be considered by adding loosely constrained **parametrization:ionosphereSTEC**.

The `parameter names` are `<station>:VTEC:<time>`.

Name	Type	Annotation
▶ name	string	
▶ selectReceivers	platformSelector	
▶ outputfileVTEC	filename	variable {station} available
▶ mapR	double	constant of MSLM mapping function
▶ mapH	double	constant of MSLM mapping function
▶ mapAlpha	double	constant of MSLM mapping function

### 5.14.3 IonosphereMap

Apriori VTEC maps can be removed from the observations with **inputfileGriddedDataTimeSeries** (e.g. from **GnssIonex2GriddedDataTimeSeries**).

The ionosphere is parametrized in terms of [TECU] in a single layer sphere with **radiusIonosphericLayer** as a **temporally** changing (e.g. hourly linear splines) spherical harmonics expansion

$$VTEC(\lambda, \theta, t) = \sum_{n=0}^{n_{max}} \sum_{m=0}^n c_{nm}(t) C_{nm}(\lambda, \theta) + s_{nm}(t) S_{nm}(\lambda, \theta) \quad (5.25)$$

up to **maxDegree=15** in a solar-geomagnetic frame defined by **magnetosphere**. The VTEC values are mapped to STEC values in the observation equations via eq. (5.24).

The estimated VTEC inclusive the apriori **inputfileGriddedDataTimeSeries** can be written to **outputfileGriddedDataTimeSeries** evaluated at **outputGrid** and **outputTimeSeries**.

Local and short-term scintillations should be considered by adding constrained **parametrization:ionosphereSTEC**. To account for signal biases add **parametrization:tecBiases**.

The **parameter names** are

- VTEC:sphericalHarmonics.c\_<degree>\_<order>:<temporal>:<interval>,
- VTEC:sphericalHarmonics.s\_<degree>\_<order>:<temporal>:<interval>.

Name	Type	Annotation
► name	string	
► selectReceivers	platformSelector	
► outputFileGriddedDataTimeSeries	filename	single layer VTEC [TECU]
► outputGrid	grid	
► outputTimeSeries	timeSeries	
► inputFileGriddedDataTimeSeries	filename	single layer VTEC [TECU]
► maxDegree	uint	spherical harmonics parametrization
► temporal	parametrizationTemporal	temporal evolution of VTEC values
► radiusIonosphericLayer	double	[m] radius of ionospheric single layer
► mapR	double	[m] constant of MSLM mapping function
► mapH	double	[m] constant of MSLM mapping function
► mapAlpha	double	constant of MSLM mapping function
► magnetosphere	magnetosphere	

#### 5.14.4 Clocks

Clock errors are estimated epoch-wise for each **selectTransmitter/Receiver**. No clock errors are estimated if no valid observations are available (e.g. data gaps in the observations).

These parameters are linearly dependent and would lead to a rank deficiency in the normal equation matrix. To circumvent this issue, the estimation requires an additional zero-mean constraint added in each epoch. This is realized with an additional observation equation

$$0 = \frac{1}{n_i + n_k} \left( \sum_i \Delta t_{s_i} + \sum_k \Delta t_{r_k} \right) \quad (5.26)$$

summed over all **selectTransmitters/ReceiversZeroMean** with a standard deviation of **sigmaZeroMeanConstraint**.

The **parameter names** are `<station or prn>:clock::<time>`.

Name	Type	Annotation
► name	string	used for parameter selection
► selectTransmitters	platformSelector	
► selectReceivers	platformSelector	
► outputFileClockTransmitter	filename	variable {prn} available
► outputFileClockReceiver	filename	variable {station} available
► nameConstraint	string	used for parameter selection
► selectTransmittersZeroMean	platformSelector	
► selectReceiversZeroMean	platformSelector	
► sigmaZeroMeanConstraint	double	(0 = unconstrained) sigma [m] for zero-mean constraint over all selected clocks

### 5.14.5 ClocksModel

This parametrization is an alternative to **parametrization:clocks**. Clock errors are estimated epoch-wise for each **selectTransmitter/Receiver** and, opposed to **parametrization:clocks**, are also estimated for epochs that have no valid observations available (e.g. data gaps).

The clock error of the an epoch can be predicted by the clock error of the preceding epoch and an unknown clock drift

$$\Delta t_{i+1} = \Delta t_i + t_{drift}dt + \epsilon_i. \quad (5.27)$$

This equation is applied as an additional constraint equation in each epoch

$$0 = \Delta t_{i+1} - \Delta t_i - t_{drift}dt + \epsilon_i. \quad (5.28)$$

The variance  $\sigma^2(\epsilon)$  is estimated iteratively by variance component estimation (VCE). Clock jumps are treated as outliers and are automatically downweighted as described in **GnssProcessing:processingStep:estimate**.

The absolute initial clock error and clock drift cannot be determined if all receiver and transmitter clocks are estimated together due to their linear dependency. This linear dependency would lead to a rank deficiency in the normal equation matrix in the same manner as described in **parametrization:clocks**. To circumvent the rank deficiency additional zero-mean constraints are required for the first and last epoch. The realization of the constraint is done as an additional observation equation in the form

$$0 = \frac{1}{n_i + n_k} \left( \sum_i \Delta t^{s_i} + \sum_k \Delta t_{r_k} \right) \quad (5.29)$$

summed over all **selectTransmitters/ReceiversZeroMean** with a standard deviation of **sigmaZeroMeanConstraint**.

The **parameter names** are `<station or prn>:clock:<time>` and `<station or prn>:clockDrift:<time>`

Name	Type	Annotation
<b>name</b>	string	used for parameter selection
<b>selectTransmitters</b>	<b>platformSelector</b>	
<b>selectReceivers</b>	<b>platformSelector</b>	
<b>outputfileClockTransmitter</b>	filename	variable {prn} available
<b>outputfileClockReceiver</b>	filename	variable {station} available
<b>huber</b>	double	clock jumps \$huber*sigma0 are downweighted
<b>huberPower</b>	double	clock jumps \$huber*sigma0 are downweighted clock jumps \$huber*sigma0 are downweighted clock jumps \$huber*sigma0 are downweighted
<b>nameConstraint</b>	string	used for parameter selection
<b>selectTransmittersZeroMean</b>	<b>platformSelector</b>	use these transmitters for zero-mean constraint
<b>selectReceiversZeroMean</b>	<b>platformSelector</b>	use these receivers for zero-mean constraint
<b>sigmaZeroMeanConstraint</b>	double	(0 = unconstrained) sigma [m] for zero-mean constraint over all selected clocks

### 5.14.6 SignalBiases

Each code and phase observation (e.g C1C or L2W) contains a bias at transmitter/receiver level

$$[\tau\nu a]^s_r(t) = \dots + \text{bias}[\tau\nu a]^s + \text{bias}[\tau\nu a]_r + \dots \quad (5.30)$$

This class provides the apriori model  $\mathbf{f}(\mathbf{x}_0)$  of eq. (5.20) only.

The **inputfileSignalBiasTransmitter/Receiver** are read for each receiver and transmitter. The file name is interpreted as a template with the variables `{prn}` and `{station}` being replaced by the name. (Infos regarding the variables `{prn}` and `{station}` can be found in **gnssTransmitterGeneratorType** and **gnssReceiverGeneratorType** respectively). The files can be converted with **GnssSinexBias2SignalBias**.

The estimation of the biases is complex due to different linear dependencies, which result in rank deficiencies in the system of normal equations. For simplification the parametrization for  $\Delta\mathbf{x}$  has been split into: **parametrization:codeBiases**, **parametrization:tecBiases**, and **parametrization:ambiguities** (including phase biases). The file handling on the other hand still remains within this class. Any prior values for the receiver/transmitter biases are read with the respective **inputFileSignalBias**. All biases for a receiver/transmitter are accumulated and written to the respective **outputFileSignalBias**.

Name	Type	Annotation
<b>name</b>	string	used for parameter selection
<b>selectTransmitters</b>	platformSelector	
<b>selectReceivers</b>	platformSelector	
<b>outputfileSignalBiasTransmitter</b>	filename	variable <code>{prn}</code> available
<b>outputfileSignalBiasReceiver</b>	filename	variable <code>{station}</code> available
<b>inputfileSignalBiasTransmitter</b>	filename	variable <code>{prn}</code> available
<b>inputfileSignalBiasReceiver</b>	filename	variable <code>{station}</code> available

### 5.14.7 Ambiguities

Sets up an ambiguity parameter for each track and phase observation type.

$$[L\nu a]_r^s(t) = \dots + \text{bias}[L\nu a]^s + \text{bias}[L\nu a]_r + \lambda[L\nu]N[L\nu a]_r^s \quad (5.31)$$

As the phase observations contain a float bias at transmitter/receiver level, not all ambiguities are resolvable to integer values. The number of resolvable ambiguities can be increased with known phase biases read from file via **parametrization:signalBiases**. In this case, **estimateTransmitter/ReceiverPhaseBiasTransmitter** should not be used for the corresponding transmitters and receivers.

In case of GLONASS, the phase biases at receiver level differ between different frequency channels (frequency division multiple access, FDMA) and for each channel an extra float phase bias is estimated. With **linearGlonassBias** a linear relationship between bias and frequency channel is assumed, which reduces the number of float bias parameters and increases the number of resolvable integer ambiguities.

The integer ambiguities can be resolved and fixed in **GnssProcessing:processingStep:resolveAmbiguities**. Resolved integer ambiguities are not estimated as unknown parameters in **gnssProcessingStepType:estimate** anymore and are removed from the system of normal equations.

The estimated phase biases can be written to files in **parametrization:signalBiases**.

The **parameter names** are

- `<station>:phaseBias(<gnssType>)::,`
- `<prn>:phaseBias(<gnssType>)::,`
- `<station>.<prn>:ambiguity<index>of<count>(<GnssTypes>)::<track interval>.`

Name	Type	Annotation
▶ name	string	used for parameter selection
▶ estimateTransmitterPhaseBias	platformSelector	
▶ estimateReceiverPhaseBias	platformSelector	
▶ linearGlonassBias	boolean	bias depends linear on frequency channel number

### 5.14.8 CodeBiases

Each code observation (e.g C1C or C2W) contains a bias at transmitter/receiver level

$$[C\nu a]_r^s(t) = \dots + \text{bias}[C\nu a]^s + \text{bias}[C\nu a]_r + \dots \quad (5.32)$$

The code biases cannot be estimated together with clock errors and ionospheric delays in an absolute sense as rank deficiencies will occur in the system of normal equations. Therefore, the biases are not initialized and set up as parameters directly but only estimable linear combinations are parametrized.

The basic idea is to set up simplified normal equations with the biases, clock and STEC parameters of one single receiver or transmitter, eliminate clock and STEC parameters and perform an eigen value decomposition of the normal equation matrix

$$\mathbf{N} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^T. \quad (5.33)$$

Instead of estimating the original bias parameter  $\mathbf{x}$  a transformed set  $\bar{\mathbf{x}}$  is introduced:

$$\bar{\mathbf{x}} = \mathbf{Q}^T \mathbf{x}. \quad (5.34)$$

The new parameters corresponding to eigen values  $\lambda > 0$  are estimable, the others are left out (set to zero). The behavior can be controlled by explicitly setting up to two bias types with ▶ **typesClockDatum** for each transmitter to zero. These then define the ionosphere-free clock datum of the transmitter. The missing linear combinations, which depend on the STEC parameters, can be added with ▶ **parametrization:tecBiases**.

Additional rank deficiencies may also occur when biases of transmitters and receivers are estimated together. The minimum norm nullspace (also via eigen value decomposition) is formulated as zero constraint equations and added with a standard deviation of ▶ **sigmaZeroMeanConstraint**.

In case of GLONASS the code biases at receiver level can differ between different frequency channels (frequency division multiple access, FDMA) and for each channel an extra code bias is estimated. With ▶ **linearGlonassBias** a linear relationship between bias and frequency channel is assumed, which reduces the number of bias parameters.

The estimated biases can be written to files in ▶ **parametrization:signalBiases**.

The ▶ **parameter names** are `<station or prn>:codeBias0<index><combi of gnssTypes>:::`

Name	Type	Annotation
▶ name	string	used for parameter selection
▶ selectTransmitters	platformSelector	
▶ selectReceivers	platformSelector	
▶ linearGlonassBias	boolean	bias depends linear on frequency channel number
▶ typesClockDatum	gnssType	first two matching types define the ionosphere free transmitter clock (e.g. C1WG, C2WG)
▶ nameConstraint	string	used for parameter selection
▶ sigmaZeroMeanConstraint	double	(0 = unconstrained) sigma [m] for null space constraint

### 5.14.9 TecBiases

Each code observation (e.g C1C or C2W) contains a bias at transmitter/receiver level

$$[C\nu a]_r^s(t) = \dots + \text{bias}[C\nu a]^s + \text{bias}[C\nu a]_r + \dots \quad (5.35)$$

This parametrization represents the linear combination of signal biases which completely depend on the STEC parameters. Ignoring these bias combinations would result in a biased STEC estimation (all other parameters are nearly unaffected). To determine this part of the signal biases the **parametrization:ionosphereSTEC** should be constrained. Furthermore, additional information about the ionosphere is required from **parametrization:ionosphereVTEC** or **parametrization:ionosphereMap**.

Rank deficiencies due to the signal bias parameters may occur if biases of transmitters and receivers are estimated together. The minimum norm nullspace is formulated as zero constraint equations and added with a standard deviation of **sigmaZeroMeanConstraint**.

The accumulated estimated result can be written to files in **parametrization:signalBiases**.

The **parameter names** are `<station or prn>:tecBias0<index><combi of gnssTypes>::`

Name	Type	Annotation
name	string	used for parameter selection
selectTransmitters	platformSelector	
selectReceivers	platformSelector	
linearGlonassBias	boolean	phase or code biases depend linear on frequency channel number
nameConstraint	string	used for parameter selection
sigmaZeroMeanConstraint	double	(0 = unconstrained) sigma [m] for null space constraint

### 5.14.10 TemporalBias

This parametrization resolves the issue of some phase observations suffering from time-variable biases. Such a phenomenon has been found to affect GPS block IIF satellites on the L5 phase measurements (see Montenbruck et al. 2011, DOI: [10.1007/s10291-011-0232-x](https://doi.org/10.1007/s10291-011-0232-x)).

For these time-variable biases an appropriate temporal representation has to be defined in **parametrizationTemporal**. For example, time-variable biases for GPS block IIF L5 phase observations (**type=L5\*G**) can be represented by a cubic spline with a nodal distance of one hour.

The result is written as a **times series file** at the processing sampling or the sampling set by **GnssProcessing:processingStep:selectEpochs**.

This parametrization should be set up in addition to the constant **parametrization:signalBiases**. Depending on the temporal representation a temporal zero-mean constraint is needed to separate this parametrization from the constant component. The constraint equations are added with a standard deviation of **sigmaZeroMeanConstraint**.

The **parameter names** are `<prn>:signalBias.<gnssType>:<temporal>:<interval>`.

Name	Type	Annotation
name	string	used for parameter selection
selectTransmitters	platformSelector	
outputfileBiasTimeSeries	filename	variable {prn} available

▶ <code>inputfileBiasTimeSeries</code>	<code>filename</code>	variable {prn} available
▶ <code>type</code>	<code>gnssType</code>	
▶ <code>parametrizationTemporal</code>	<code>parametrizationTemporal</code>	
▶ <code>nameConstraint</code>	<code>string</code>	used for parameter selection (0 = unconstrained) sigma [m] for temporal
▶ <code>sigmaZeroMeanConstraint</code>	<code>double</code>	zero-mean constraint

### 5.14.11 StaticPositions

Estimates a static position for all ▶ `selectReceivers` in the terrestrial frame.

No-net constraints can be applied for a subset of stations, ▶ `selectNoNetReceivers`, with a standard deviation of ▶ `noNetTranslationSigma` and ▶ `noNetRotationSigma` and ▶ `noNetScaleSigma`. If the template ▶ `inputfileNoNetPositions` is provided the constraints are applied relatively to these positions. Only stations with an existing position file are considered. Without ▶ `inputfileNoNetPositions` the constraints are applied towards the apriori values from ▶ `GnssProcessing:receiver`. As a single corrupted station position can disturb the no-net conditions, the rotation/translation parameters are estimated in a **robust least squares adjustment (2.1)** beforehand. The computed weight matrix is used to downweight corrupted stations in the constraint equations.

In case you want to align to an ITRF/IGS reference frame, precise coordinates can be generated with ▶ `Sinex2StationPositions`.

The ▶ `parameter names` are

- `<station>.position.x::,`
- `<station>.position.y::,`
- `<station>.position.z::.`

Name	Type	Annotation
▶ <code>name</code>	<code>string</code>	used for parameter selection
▶ <code>selectReceivers</code>	<code>platformSelector</code>	
▶ <code>outputfileGriddedPosition</code>	<code>filename</code>	delta north east up for all stations
▶ <code>outputfilePosition</code>	<code>filename</code>	variable {station} available, full estimated coordinates (in TRF)
▶ <code>nameConstraint</code>	<code>string</code>	used for parameter selection
▶ <code>selectNoNetReceivers</code>	<code>platformSelector</code>	
▶ <code>inputfileNoNetPositions</code>	<code>filename</code>	variable {station} available, precise coordinates used for no-net constraints (in TRF)
▶ <code>noNetTranslationSigma</code>	<code>double</code>	(0 = unconstrained) sigma [m] for no-net translation constraint on station coordinates
▶ <code>noNetRotationSigma</code>	<code>double</code>	(0 = unconstrained) sigma [m] at Earth's surface for no-net rotation constraint on station coordinates
▶ <code>noNetScaleSigma</code>	<code>double</code>	(0 = unconstrained) sigma [m] for no-net scale constraint on station coordinates
▶ <code>huber</code>	<code>double</code>	stations \$i\$ huber*sigma0 are downweighted in no-net constraint
▶ <code>huberPower</code>	<code>double</code>	stations \$i\$ huber: sigma=(e/huber)^ huberPower*sigma0

### 5.14.12 KinematicPositions

Estimates the epoch-wise **▶outputfilePositions** in an Earth-fixed frame (or in case of LEO satellites in an inertial frame).

The  $3 \times 3$  epoch wise **▶outputfileCovarianceEpoch** are computed within **▶GnssProcessing:processingStep:computeCovarianceMatrix**

The **◀parameter names** are

- <station>:position.x::<time>,
- <station>:position.y::<time>,
- <station>:position.z::<time>.

Name	Type	Annotation
▶ name	string	used for parameter selection
▶ selectReceivers	platformSelector	
▶ outputfilePositions	filename	variable {station} available, estimated kinematic positions/orbit
▶ outputfileCovarianceEpoch	filename	variable {station} available, 3x3 epoch covariances

### 5.14.13 LeoDynamicOrbits

The estimation of (reduced) dynamic orbits is formulated as variational equations. It is based on **▶inputfileVariational** calculated with **▶PreprocessingVariationalEquation**. Necessary integrations are performed by integrating a moving interpolation polynomial of degree **▶integrationDegree**. The **▶parametrizationAcceleration** must include at least those parameters that were estimated in **▶PreprocessingVariationalEquationOrbitFit**. Additional **▶stochasticPulse** parameters can be set up to reduce orbit mismodeling. If not enough epochs with observations are available (**▶minEstimableEpochsRatio**) the LEO satellite is disabled.

The parameters and **◀parameter names** are divided into global

- <station>:<parametrizationAcceleration>::\*:,
- <station>:stochasticPulse.x::<time>,
- <station>:stochasticPulse.y::<time>,
- <station>:stochasticPulse.z::<time>,

and arc related parameters

- <station>:arc<no>. <parametrizationAcceleration>::\*:,
- <station>:arc<no>.position0.x:::,
- <station>:arc<no>.position0.y:::,
- <station>:arc<no>.position0.z:::,
- <station>:arc<no>.velocity0.x:::,
- <station>:arc<no>.velocity0.y:::,
- <station>:arc<no>.velocity0.z:::,

Name	Type	Annotation
▶ name	string	used for parameter selection
...* selectReceivers	platformSelector	
▶ outputfileOrbit	filename	variable {station} available
▶ outputfileParameters	filename	variable {station} available
▶ inputfileVariational	filename	variable {station} available
▶ stochasticPulse	timeSeries	[mu/s] parametrization of stochastic pulses
... parametrizationAcceleration	parametrizationAcceleration	orbit force parameters
▶ ephemerides	ephemerides	
▶ minEstimableEpochsRatio	double	drop satellites with lower ratio of estimable epochs to total epochs
▶ integrationDegree	uint	integration of forces by polynomial approximation of degree n
▶ interpolationDegree	uint	for orbit interpolation and velocity calculation

### 5.14.14 TransmitterDynamicOrbits

Same as ▶ [leoDynamicOrbits](#) but for transmitting GNSS satellites. For more details see [orbit integration \(3.2.2\)](#).

Name	Type	Annotation
▶ name	string	used for parameter selection
...* selectTransmitters	platformSelector	
▶ outputfileOrbit	filename	variable {prn} available
▶ outputfileParameters	filename	variable {prn} available
▶ inputfileVariational	filename	variable {prn} available
▶ stochasticPulse	timeSeries	[mu/s] parametrization of stochastic pulses
... parametrizationAcceleration	parametrizationAcceleration	orbit force parameters
▶ ephemerides	ephemerides	
▶ minEstimableEpochsRatio	double	drop satellites with lower ratio of estimable epochs to total epochs
▶ integrationDegree	uint	integration of forces by polynomial approximation of degree n
▶ interpolationDegree	uint	for orbit interpolation and velocity calculation

### 5.14.15 Troposphere

A priori tropospheric correction is handled by a ▶ [troposphere](#) model (e.g. Vienna Mapping Functions 3). Additional parameters in [m] for zenith wet delay and gradients can be set up via ▶ [troposphereWetEstimation](#) (usually 2-hourly linear splines) and ▶ [troposphereGradientEstimation](#) (usually a daily trend). These parameters can be soft-constrained using ▶ [parametrization:constraints](#) to avoid an unsolvable system of normal equations in case of data gaps.

The ▶ [parameter names](#) are

- <station>:troposphereWet:<temporal>:<interval>,

- <station>:troposphereGradient.x:<temporal>:<interval>,
- <station>:troposphereGradient.y:<temporal>:<interval>.

Name	Type	Annotation
▶ name	string	used for parameter selection
… selectReceivers	platformSelector	
▶ outputFileTroposphere	filename	columns: MJD, ZHD, ZWD, dry north gradient, wet north gradient, dry east gradient, wet east gradient, ...
▶ troposphere	troposphere	a priori troposphere model
… troposphereWetEstimation	parametrizationTemporal	[m] parametrization of zenith wet delays
▶ troposphereGradientEstimation	parametrizationTemporal	[degree] parametrization of north and east gradients

### 5.14.16 EarthRotation

Earth rotation parameters (ERPs) can be estimated by defining ▶ **estimatePole** ( $x_p, y_p [mas]$ ) and ▶ **estimateUT1** ( $dUT1 [ms], LOD$ ).

Estimating length of day (LOD) with the sign according to IGS conventions requires a negative value in ▶ **parametrizationTemporal:trend:timeStep**.

Constraints on the defined parameters can be added via ▶ **parametrization:constraints**. An example would be to set up ▶ **estimateUT1:constant** so the  $dUT1$  parameter is included in the normal equation system . Since  $dUT1$  cannot be determined by GNSS, a hard constraint to its a priori value can then be added.

The ▶ **parameter names** are

- earth:polarMotion.xp:<temporal>:<interval>,
- earth:polarMotion.yp:<temporal>:<interval>,
- earth:UT1:<temporal>:<interval>,
- earth:nutation.X:<temporal>:<interval>,
- earth:nutation.>:<temporal>:<interval>.

Name	Type	Annotation
▶ name	string	used for parameter selection
▶ outputFileEOP	filename	EOP time series (mjd, xp, yp, sp, dUT1, LOD, X, Y, S)
… estimatePole	parametrizationTemporal	xp, yp [mas]
… estimateUT1	parametrizationTemporal	rotation angle [ms]
… estimateNutation	parametrizationTemporal	dX, dY [mas]

### 5.14.17 ReceiverAntennas

This class is for parametrization the antenna for their antenna center offsets (ACO) and antenna center variations (ACV) by ▶ **antennaCenterVariations**. The receivers to be estimated can be selected by ▶ **selectReceivers**.

The amount of patterns to be estimated is configurable with a list of **patternTypes**. For each added **patternTypes** a set of parameters will be evaluated. The observations will be assigned to the first **patternTypes** that matches their own. E.g. having the patterns: \*\*\*G and L1\* would lead to all GPS observations be assigned to the observation equations of the first pattern. The patterntype L1\* would then consist of all other GNSS L1 phase observations. **addNonMatchingTypes** will, if activated, create automatically patterns for **observations** that are not selected within the list **patternTypes**. Furthermore, it is possible to group same antenna build types from different receivers by **groupAntennas**. The grouping by same antenna build ignores antenna serial numbers.

To estimate the antenna variation parameters, a longer period of observations might be necessary for accurate estimations. Hence one should use this parametrization by accumulating normal equations from several epochs. This can be accomplished as the last steps in the **processing steps** by adding **ReceiverAntennas** to current selected parameters with **GnssProcessing:processingStep:selectParametrizations** and write the normal equation matrix with **GnssProcessing:processingStep:writeNormalEquations**. The written normal equations can then be accumulated with **NormalsAccumulate** and solved by **NormalsSolverVCE**. Further, one should apply constraints to the normal equations by **GnssAntennaNormalsConstraint** since the estimation of ACO and ACV can lead to rank deficiencies in the normal equation matrix. Last the solved normal equation can be parsed to a **antenna definition file** with the program **ParameterVector2GnssAntennaDefinition**.

As example referring to the cookbook GNSS satellite orbit determination and station network analysis (3.2), one could add additionally **receiverAntennas** as parametrization. Since the estimations are done on a daily basis for each receiver we add an additional **selectParametrizations** which disables `parameter.receiverAntenna`. After all stations are processed together with all parameters, one adds `parameter.receiverAntenna` with **selectParametrizations** to the current selected parametrizations. The last **processingStep** is **GnssProcessing:processingStep:writeNormalEquations** to write the daily normal equations including the parametrization **receiverAntennas** into files. These normal equation files are then processed with the programs:

- **NormalsAccumulate**: accumulates normal equations.
- **GnssAntennaNormalsConstraint**: apply constraint to the normal equations.
- **NormalsSolverVCE**: solves the normal equations.
- **ParameterVector2GnssAntennaDefinition**: writes the solution into a **antenna definition file**

Note that the apriori value  $x_0$  for this parametrization is always zero and never updated according to eq. (5.21).

The **parameter names** are `<antennaName>:<antennaCenterVariations>.<gnssType>::`

Name	Type	Annotation
<b>name</b>	string	used for parameter selection
<b>selectReceivers</b>	platformSelector	
<b>antennaCenterVariations</b>	parametrizationGnssAntenna	estimate antenna center variations
<b>patternTypes</b>	gnssType	gnssType for each pattern (first match is used)
<b>addNonMatchingTypes</b>	boolean	add patterns for additional observed gnssTypes that don't match any of the above
<b>groupAntennas</b>	boolean	common ACVs for same antenna build types (ignores antenna serial number)

### 5.14.18 TransmitterAntennas

Same as **receiverAntennas** but for transmitting antennas (GNSS satellites).

The **parameter names** are `<antennaName>:<antennaCenterVariations>.<gnssType>::`.

Name	Type	Annotation
name	string	used for parameter selection
selectTransmitters	platformSelector	
antennaCenterVariations	parametrizationGnssAntenna	estimate antenna center variations
patternTypes	gnssType	gnssType for each pattern (first match is used)
addNonMatchingTypes	boolean	add patterns for additional observed gnssTypes that don't match any of the above
groupAntennas	boolean	common ACVs for same antenna build types (ignores antenna serial number)

### 5.14.19 Constraints

Add a pseudo observation equation (constraint) for each selected **parameters**

$$b - x_0 = 1 \cdot dx + \epsilon, \quad (5.36)$$

where  $b$  is the **bias** and  $x_0$  is the a priori value of the parameter if **relativeToApriori** is not set. The standard deviation **sigma** is used to weight the observation equations.

Name	Type	Annotation
name	string	
parameters	parameterSelector	parameter to constrain
sigma	double	sigma of the constraint (same unit as parameter)
bias	double	constrain all selected parameters towards this value
relativeToApriori	boolean	constrain only dx and not full x=dx+x0

### 5.14.20 Group

Groups a set of parameters. This class can be used to structure complex parametrizations and has no further effect itself.

Name	Type	Annotation
parametrization	gnssParametrization	

## 5.15 GnssProcessingStep

Processing step in **GnssProcessing**.

Processing steps enable a dynamic definition of the consecutive steps performed during any kind of GNSS processing. The most common steps are **estimate**, which performs an iterative least squares adjustment, and **writeResults**, which writes all output files defined in **GnssProcessing** and is usually the last step. Some steps such as **selectParametrizations**, **selectEpochs**, **selectNormalsBlockStructure**, and **selectReceivers** affect all subsequent steps. In case these steps are used within a **group** or **forEachReceiverSeparately** step, they only affect the steps within this level.

For usage examples see cookbooks on [GNSS satellite orbit determination and network analysis \(3.2.3\)](#) or [Kinematic orbit determination of LEO satellites \(3.4\)](#).

### 5.15.1 Estimate

Iterative non-linear least squares adjustment. In every iteration it accumulates the system of normal equations, solves the system and updates the estimated parameters. The estimated parameters serve as a priori values in the next iteration and the following processing steps. Iterates until either every single parameter update (converted to an influence in meter) is below a **convergenceThreshold** or **maxIterationCount** is reached.

With **computeResiduals** the observation equations are computed again after each update to compute the observation residuals.

The overall standard deviation of a single observation used for the weighting is composed of several factors

$$\hat{\sigma}_i = \hat{\sigma}_i^{huber} \hat{\sigma}_{[\tau\nu a]}^{recv} \sigma_{[\tau\nu a]}^{recv}(E, A), \quad (5.37)$$

where  $[\tau\nu a]$  is the signal type, the azimuth and elevation dependent  $\sigma_{[\tau\nu a]}^{recv}(E, A)$  is given by **receiver:inputfileAccuracyDefinition** and the other factors are estimated iteratively from the residuals.

With **computeWeights** a standardized variance  $\hat{s}_i^2$  for each residual  $\hat{\epsilon}_i$  is computed

$$\hat{s}_i^2 = \frac{1}{\hat{\sigma}_{[\tau\nu a]}^{recv} \sigma_{[\tau\nu a]}^{recv}(E, A)} \frac{\hat{\epsilon}_i^2}{r_i} \quad \text{with} \quad r_i = \left( \mathbf{A} (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \right)_{ii} \quad (5.38)$$

taking the redundancy  $r_i$  into account. If  $\hat{s}_i$  is above a threshold **huber** the observation gets a higher standard deviation used for weighting according to

$$\hat{\sigma}_i^{huber} = \begin{cases} 1 & s < huber, \\ (\hat{s}_i/huber)^{huberPower} & s \geq huber \end{cases}, \quad (5.39)$$

similar to [robust least squares adjustment \(2.1\)](#).

With **adjustSigma0** individual variance factors can be computed for each station and all phases of a system and each code observation **type** (5.18) (e.g. for each L\*\*G, L\*\*E, C1CG, C2WG, C1CE, ...) separately

$$\hat{\sigma}_{[\tau\nu a]}^{recv} = \sqrt{\frac{\hat{\epsilon}^T \mathbf{P} \hat{\epsilon}}{r}}. \quad (5.40)$$

Name	Type	Annotation
<b>computeResiduals</b>	boolean	
<b>adjustSigma0</b>	boolean	adjust sigma0 by scale factor (per receiver and type)
<b>computeWeights</b>	boolean	downweight outliers
<b>huber</b>	double	residuals $\$;\$$ huber*sigma0 are downweighted
<b>huberPower</b>	double	residuals $\$;\$$ huber: sigma=(e/huber)^huberPower*sigma0
<b>convergenceThreshold</b>	double	[m] stop iteration once full convergence is reached
<b>maxIterationCount</b>	uint	maximum number of iterations

### 5.15.2 ResolveAmbiguities

Performs a least squares adjustment like **processingStep:estimate** but with additional integer phase ambiguity resolution. After this step all resolved ambiguities are removed from the normal equation system. Only ambiguities are resolved with involved **selectTransmitters/Receivers**.

Integer ambiguity resolution is performed based on the least squares ambiguity decorrelation adjustment (LAMBDA) method (Teunissen 1995, DOI [10.1007/BF00863419](https://doi.org/10.1007/BF00863419)), specifically the modified algorithm (MLAMBDA) by Chang et al. (2005, DOI [10.1007/s00190-005-0004-x](https://doi.org/10.1007/s00190-005-0004-x)). First the covariance matrix of the integer ambiguity parameters is computed by eliminating all but those parameters from the full normal equation matrix and inverting it. Then, a Z-transformation is performed as described by Chang et al. (2005) to decorrelate the ambiguity parameters without losing their integer nature.

The search process follows MLAMBDA and uses integer minimization of the weighted sum of squared residuals. It is computationally infeasible to search a hyper-ellipsoid with a dimension of ten thousand or more. Instead, a blocked search algorithm is performed by moving a window with a length of, for example, **searchBlockSize=200** parameters over the decorrelated ambiguities, starting from the most accurate. In each step, the window is moved by half of its length and the overlapping parts are compared to each other. If all fixed ambiguities in the overlap agree, the algorithm continues. Otherwise, both windows are combined and the search is repeated using the combined window, again comparing with the overlapping part of the preceding window. If not all solutions could be checked for a block after **maxSearchSteps**, the selected **incompleteAction** is performed. If the algorithm reaches ambiguities with a standard deviation higher than **sigmaMaxResolve**, ambiguity resolution stops and the remaining ambiguities are left as float values. Otherwise, all ambiguity parameters are fixed to integer values.

In contrast to an integer least squares solution over the full ambiguity vector, it is not guaranteed that the resulting solution is optimal in the sense of minimal variance with given covariance. This trade-off is necessary to cope with large numbers of ambiguities.

Name	Type	Annotation
<b>outputfileAmbiguities</b>	filename	resolved ambiguities
<b>selectTransmitters</b>	<b>platformSelector</b>	only resolve ambiguities with these participating transmitters
<b>selectReceivers</b>	<b>platformSelector</b>	only resolve ambiguities with these participating receivers
<b>sigmaMaxResolve</b>	double	max. allowed std. dev. of ambiguity to resolve [cycles]
<b>searchBlockSize</b>	uint	block size for blocked integer search
<b>maxSearchSteps</b>	uint	max. steps of integer search for each block
<b>incompleteAction</b>	choice	if not all solutions tested after maxSearchSteps
<b>stop</b>		stop searching, ambiguities remain float in this block
<b>resolve</b>		use best integer solution found so far
<b>shrinkBlockSize</b>		try again with half block size
<b>throwException</b>		stop and throw an exception
<b>computeResiduals</b>	boolean	
<b>adjustSigma0</b>	boolean	adjust sigma0 by scale factor (per receiver and type)
<b>computeWeights</b>	boolean	downweight outliers
<b>huber</b>	double	residuals $\$i\$ \text{huber} * \text{sigma0}$ are downweighted
<b>huberPower</b>	double	residuals $\$i\$ \text{huber} : \text{sigma} = (\text{e}/\text{huber})^{\text{huberPower}} * \text{sigma0}$

### 5.15.3 ComputeCovarianceMatrix

Accumulates the normal equations and computes the covariance matrix as inverse of the normal matrix. It is not the full inverse but only the elements which are set in the normal matrix (see **gnssProcessingStep:selectNormalsBlockStructure**) are computed. The matrix is passed to the

► **parametrizations**. Only used in ► **parametrizations:kinematicPositions** to get the epoch wise covariance information at the moment.

#### 5.15.4 WriteResults

In this step all ► **outputfiles** defined in ► **parametrizations** are written. It considers the settings of ► **processingStep:selectParametrizations**, ► **processingStep:selectEpochs**, and ► **processingStep:selectReceivers**.

It is usually the last processing step, but can also be used at other points in the processing in combination with ► **suffix** to write intermediate results, for example before ► **gnssProcessingStep:resolveAmbiguities** to output the float solution.

Name	Type	Annotation
► suffix	string	appended to every output file name (e.g. orbit.G01.suffix.dat)

#### 5.15.5 WriteNormalEquations

Accumulates the normal equations matrix and writes it. If ► **remainingParameters** is set only the selected parameters are written to the normal equations and all other parameters are eliminated beforehand (implicitly solved).

The solution of the normals would result in  $\Delta\mathbf{x}$  (see ► **parametrizations**). To write the appropriate apriori vector  $\mathbf{x}_0$  use ► **processingStep:writeAprioriSolution**.

Name	Type	Annotation
► outputFileNormalEquations	filename	normals
► remainingParameters	parameterSelector	parameter order/selection of output normal equations
► constraintsOnly	boolean	write only normals of constraints without observations
► defaultNormalsBlockSize	uint	block size for distributing the normal equations, 0: one block, empty: original block size

#### 5.15.6 WriteAprioriSolution

Writes the current apriori vector  $\mathbf{x}_0$  (see ► **parametrizations**). If ► **remainingParameters** is set only the selected parameters are written.

Name	Type	Annotation
► outputFileAprioriSolution	filename	a priori parameters
► outputFileParameterNames	filename	parameter names
► remainingParameters	parameterSelector	parameter order/selection of output normal equations

#### 5.15.7 WriteResiduals

Writes the ► **observation residuals** for all ► **selectReceivers**. For each station a file is written. The file name is interpreted as a template with the variable {station} being replaced by the station name.

Name	Type	Annotation
selectReceivers	platformSelector	subset of used stations
outputfileResiduals	filename	variable {station} available

### 5.15.8 WriteUsedStationList

Writes a `list` of receivers (stations) which are used in the last step and selected by `selectReceivers`.

Name	Type	Annotation
selectReceivers	platformSelector	subset of used stations
outputfileUsedStationList	filename	ascii file with names of used stations

### 5.15.9 WriteUsedTransmitterList

Writes a `list` of transmitters which are used in the last step and selected by `selectTransmitters`.

Name	Type	Annotation
selectTransmitters	platformSelector	subset of used transmitters
outputfileUsedTransmitterList	filename	ascii file with PRNs

### 5.15.10 PrintResidualStatistics

Print residual statistics.

```

areq: C1CG**: factor = 0.64, sigma0 = 1.00, count = 2748, outliers = 48 (1.75 \%)
areq: C1WG**: factor = 0.50, sigma0 = 1.00, count = 2748, outliers = 43 (1.56 \%)
areq: C2WG**: factor = 0.50, sigma0 = 1.00, count = 2748, outliers = 59 (2.15 \%)
areq: C5XG**: factor = 0.46, sigma0 = 1.00, count = 1279, outliers = 23 (1.80 \%)
areq: L1CG**: factor = 0.86, sigma0 = 0.96, count = 2748, outliers = 40 (1.46 \%)
areq: L1WG**: factor = 0.86, sigma0 = 1.02, count = 2748, outliers = 40 (1.46 \%)
areq: L2WG**: factor = 0.86, sigma0 = 0.96, count = 2748, outliers = 40 (1.46 \%)
areq: L5XG**: factor = 0.86, sigma0 = 1.30, count = 1279, outliers = 14 (1.09 \%)
areq: C1PR**: factor = 0.48, sigma0 = 1.00, count = 1713, outliers = 53 (3.09 \%)
areq: C2PR**: factor = 0.55, sigma0 = 1.00, count = 1713, outliers = 51 (2.98 \%)
areq: L1PR**: factor = 0.85, sigma0 = 1.09, count = 1713, outliers = 29 (1.69 \%)
areq: L2PR**: factor = 0.85, sigma0 = 0.88, count = 1713, outliers = 29 (1.69 \%)
areq: C1XE**: factor = 0.44, sigma0 = 1.00, count = 1264, outliers = 21 (1.66 \%)
areq: C5XE**: factor = 0.33, sigma0 = 1.00, count = 1264, outliers = 27 (2.14 \%)
areq: C7XE**: factor = 0.28, sigma0 = 1.00, count = 1264, outliers = 41 (3.24 \%)
areq: L1XE**: factor = 0.82, sigma0 = 1.14, count = 1264, outliers = 15 (1.19 \%)
areq: L5XE**: factor = 0.82, sigma0 = 0.84, count = 1264, outliers = 15 (1.19 \%)
areq: L7XE**: factor = 0.82, sigma0 = 0.94, count = 1264, outliers = 15 (1.19 \%)
badg: C1CG**: factor = 1.25, sigma0 = 1.00, count = 2564, outliers = 47 (1.83 \%)
...

```

### 5.15.11 SelectParametrizations

Enable/disable parameter groups and constraint groups for subsequent steps, e.g. `processingStep:estimate` or `processingStep:writeResults`. The `name` and `nameConstraint`

of these groups are defined in **parametrizations**. Prior models or previously estimated parameters used as new apriori  $\mathbf{x}_0$  values are unaffected and they are always reduced from the observations. This means all unselected parameters are kept fixed to their last result.

An example would be to process at a 5-minute sampling using **processingStep:selectEpochs** and then at the end to densify the clock parameters to the full 30-second observation sampling while keeping all other parameters fixed (**disable=\***, **enable=\*.clock\***, **enable=parameter.STEC**).

Name	Type	Annotation
parametrization	choice	
enable	sequence	
name	string	wildcards: * and ?
disable	sequence	
name	string	wildcards: * and ?

### 5.15.12 SelectEpochs

Select epochs for subsequent steps. This step can be used to reduce the processing sampling while keeping the original observation sampling for all preprocessing steps (e.g. outlier and cycle slip detection). Another example is to process at a 5-minute sampling by setting **nthEpoch=10** and then at the end to densify only the clock parameters to the full 30-second observation sampling by setting **nthEpoch=1** while keeping all other parameters fixed with **processingStep:selectParametrizations**.

Name	Type	Annotation
nthEpoch	uint	use only every nth epoch in all subsequent processing steps

### 5.15.13 SelectNormalsBlockStructure

Select block structure of sparse normal equations for subsequent steps.

This step can be used to define the structure of the different parts of the normal equation system, which can have a major impact on computing performance and memory consumption depending on the processing setup.

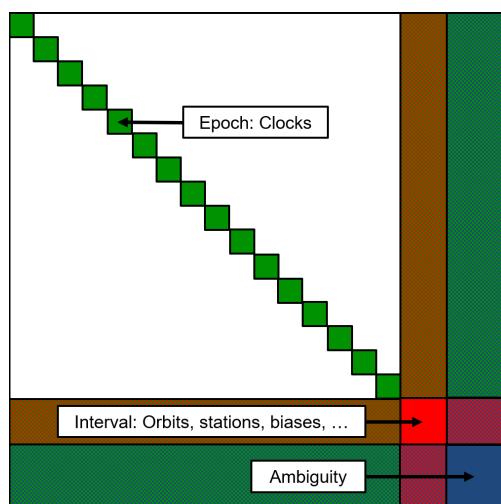


Figure 5.6: Structure of normal equations in GNSS processing

The normal equation system is divided into three parts for epoch, interval, and ambiguity parameters. The epoch part is subdivided further into one subpart per epoch. Each part is divided into blocks and only non-zero blocks are stored in memory to reduce memory consumption and to prevent unnecessary matrix computations. **►defaultBlockSizeEpoch**, **►defaultBlockSizeInterval**, and **►defaultBlockSizeAmbiguity** control the size of the blocks within each part of the normal equations. **►defaultBlockReceiverCount** can be set to group a number of receivers into one block within the epoch and interval parts.

If **►keepEpochNormalsInMemory=no** epoch blocks are eliminated after they are set up to reduce the number of parameters in the normal equation system. **►defaultBlockCountReduction** controls after how many epoch blocks an elimination step is performed. For larger processing setups or high sampling rates epoch block elimination is recommended as the large number of clock parameters require a lot of memory.

Name	Type	Annotation
<b>►defaultBlockSizeEpoch</b>	uint	block size of epoch parameters, 0: one block
<b>►defaultBlockSizeInterval</b>	uint	block size of interval parameters, 0: one block
<b>►defaultBlockSizeAmbiguity</b>	uint	block size of ambiguity parameters, 0: one block
<b>►defaultBlockReceiverCount</b>	uint	number of receivers to group into one block for epoch and interval
<b>►defaultBlockCountReduction</b>	uint	minimum number of blocks for epoch reduction
<b>►keepEpochNormalsInMemory</b>	boolean	speeds up processing but uses much more memory
<b>►accumulateEpochObservations</b>	boolean	set up all observations per epoch and receiver at once

### 5.15.14 SelectReceivers

This step can be used to process only a subset of stations in subsequent processing steps. The most common use is to start the processing with a well-distributed network of core stations as seen in [GNSS satellite orbit determination and network analysis \(3.2.3\)](#). To later process all other stations individually, use the processing step **►processingStep:forEachReceiverSeparately** and select all stations excluding the core stations in that step.

Name	Type	Annotation
<b>►selectReceivers</b>	<a href="#">platformSelector</a>	

### 5.15.15 ForEachReceiverSeparately

Perform these processing steps for each **►selectReceivers** separately. All non-receiver related parameters parameters are disabled in these processing steps (see .

This step can be used for individual precise point positioning (PPP) of all stations. During [GNSS satellite orbit determination and network analysis \(3.2.3\)](#) this step is used after the initial processing of the core network to process all other stations individually. In that case provide the same station list as **►inputfileExcludeStationList** in this step that was used as **►inputfileStationList** in the **►selectReceivers** step where the core network was selected.

Name	Type	Annotation
<b>►selectReceivers</b>	<a href="#">platformSelector</a>	
<b>►variableReceiver</b>	string	variable is set for each receiver
<b>►processingStep</b>	<a href="#">gnssProcessingStep</a>	steps are processed consecutively

### 5.15.16 Group

Perform these processing steps. This step can be used to structure complex processing flows. The  **select..** processing steps defined within a group only affect the steps within this group.

Name	Type	Annotation
 processingStep	gnssProcessingStep	steps are processed consecutively

### 5.15.17 DisableTransmitterShadowEpochs

Disable transmitter epochs during eclipse. With proper attitude modeling (see  **SimulateStarCameraGnss**) this is usually not necessary.

Name	Type	Annotation
 selectTransmitters	platformSelector	
 disableShadowEpochs	boolean	disable epochs if satellite is in Earth's/Moon's shadow
 disablePostShadowRecoveryEpochs	boolean	disable epochs if satellite is in post-shadow recovery maneuver for GPS block IIA
 ephemerides	ephemerides	
 eclipse	eclipse	eclipse model used to determine if a satellite is in Earth's shadow

## 5.16 GnssReceiverGenerator

Definition and basic information of GNSS receivers.

Most of the input files are provided in GROOPS file formats at <https://ftp.tugraz.at/outgoing/ITSG/groops> (marked with \* below). These files are regularly updated.

- **inputfileStationInfo**\*: Antenna and receiver information, antenna reference point offsets, antenna orientations. Created via **GnssStationLog2Platform** or **PlatformCreate**.
- **inputfileAntennaDefinition**\*: Antenna center offsets and variations. Created via **GnssAntex2AntennaDefinition** or **GnssAntennaDefinitionCreate**.
- **inputfileReceiverDefinition**: Observed signal types (optional). Created via **GnssReceiverDefinitionCreate** in case you want to define which signal types a receiver model can observe.
- **inputfileAccuracyDefinition**\*: Elevation and azimuth dependent accuracy. Created via **GnssAntennaDefinitionCreate**.
- **inputfileObservation**: Converted from RINEX observation files via **RinexObservation2GnssReceiver**.

It is possible to limit the observation types to be used in the processing by a list of **useType** and any observation types not defined within the list are ignored and discarded. Similarly observations defined in the list of **ignoreType** are ignored and discarded. The codes used follow the [RINEX 3 definition](#).

Each receiver goes through a **preprocessing** step individually, where observation outliers are removed or downweighted, continuous tracks of phase observations are defined for ambiguity parametrization, cycle slips are detected, and receivers are disabled if they do not fulfill certain requirements. The preprocessing step consists of an initial PPP estimation done by [robust least squares adjustment \(2.1\)](#) and checks whether the position error of the solutions exceeds **codeMaxPositionDiff**. If the error exceeds the threshold the receiver will be discarded. The preprocessing also sets initial clock error values and removes tracks that stay below a certain elevation mask (**elevationTrackMinimum**).

See also **GnssProcessing** and **GnssSimulateReceiver**.

### 5.16.1 StationNetwork

A network of GNSS ground stations is defined via **inputfileStationList**. Each line can contain more than one station. The first station in each line for which **inputfileObservations** exists and contains enough observations is used for the processing. All input files except **inputfileAntennaDefinition**, **inputfileReceiverDefinition**, and **inputfileAccuracyDefinition** are read for each station. The file name is interpreted as a template with the variable {station} being replaced by the station name.

The effects of loading and tidal deformation on station positions can be corrected for via **loadingDisplacement** and **tidalDisplacement**, respectively. Tidal deformations typically include:

- **earthTide**: Earth tidal deformations (IERS conventions)
- **doodsonHarmonicTide**: ocean tidal deformations (e.g. fes2014b\_n720, **minDegree=1**)
- **doodsonHarmonicTide**: atmospheric tidal deformation (e.g. AOD1B RL06, **minDegree=1**)
- **poleTide**: pole tidal deformations (IERS conventions)
- **poleOceanTide**: ocean pole tidal deformations (IERS conventions)

Name	Type	Annotation
inputfileStationList	filename	ascii file with station names
maxStationCount	uint	maximum number of stations to be used
inputfileStationInfo	filename	variable {station} available. station metadata (antennas, receivers, ...)
inputfileAntennaDefinition	filename	antenna center offsets and variations
noAntennaPatternFound	choice	what should happen if no antenna pattern is found for an observation
ignoreObservation		ignore observation if no matching pattern is found
useNearestFrequency		use pattern of nearest frequency if no matching pattern is found
throwException		throw exception if no matching pattern is found
inputfileReceiverDefinition	filename	observed signal types
inputfileAccuracyDefinition	filename	elevation and azimuth dependent accuracy
inputfileStationPosition	filename	variable {station} available.
inputfileClock	filename	variable {station} available
inputfileObservations	filename	variable {station} available
loadingDisplacement	gravityfield	loading deformation
tidalDisplacement	tides	tidal deformation
ephemerides	ephemerides	for tidal deformation
inputfileDeformationLoadLoveNumber	filename	
inputfilePotentialLoadLoveNumber	filename	if full potential is given and not only loading potential
useType	gnssType	only use observations that match any of these patterns
ignoreType	gnssType	ignore observations that match any of these patterns
elevationCutOff	angle	[degree] ignore observations below cutoff
elevationTrackMinimum	angle	[degree] ignore tracks that never exceed minimum elevation
minObsCountPerTrack	uint	tracks with less number of epochs with observations are dropped
minEstimableEpochsRatio	double	[0,1] drop stations with lower ratio of estimable epochs to total epochs
preprocessing	sequence	settings for preprocessing of observations/stations
printStatistics	boolean	print preprocesssing statistics for all receivers
huber	double	residuals $\$; \$_{huber} * \sigma_0$ are downweighted
huberPower	double	residuals $\$; \$_{huber} = \sigma_0 / (\epsilon / huber)^{1/huberPower}$
codeMaxPositionDiff	double	[m] max. allowed position error by PPP code
denoisingLambda	double	only clock error estimation
tecWindowSize	uint	regularization parameter for total variation denoising used in cycle slip detection
tecSigmaFactor	double	(0 = disabled) window size for TEC smoothness evaluation used in cycle slip detection
		factor applied to moving standard deviation used as threshold in TEC smoothness evaluation during cycle slip detection

---

 <code>outputfileTrackBefore</code>	filename	variables {station}, {prn}, {trackTimeStart}, {trackTimeEnd}, {types}, TEC and MW-like combinations in cycles for each track before cycle slip detection
 <code>outputfileTrackAfter</code>	filename	variables {station}, {prn}, {trackTimeStart}, {trackTimeEnd}, {types}, TEC and MW-like combinations in cycles for each track after cycle slip detection

---

### 5.16.2 LowEarthOrbiter

A single low-Earth orbiting (LEO) satellite with an onboard GNSS receiver. An apriori orbit is needed as  `inputfileOrbit`. Attitude data must be provided via  `inputfileStarCamera`. If no attitude data is available from the satellite operator, the star camera data can be simulated by using  `SimulateStarCamera`.

Name	Type	Annotation
 <code>inputfileStationInfo</code>	filename	satellite metadata (antenna, receiver, ...)
 <code>inputfileAntennaDefinition</code>	filename	antenna center offsets and variations
 <code>noAntennaPatternFound</code>	choice	what should happen if no antenna pattern is found for an observation
 <code>ignoreObservation</code>		ignore observation if no matching pattern is found
 <code>useNearestFrequency</code>		use pattern of nearest frequency if no matching pattern is found
 <code>throwException</code>		throw exception if no matching pattern is found
 <code>inputfileReceiverDefinition</code>	filename	observed signal types
 <code>inputfileAccuracyDefinition</code>	filename	elevation and azimuth dependent accuracy
 <code>inputfileObservations</code>	filename	
 <code>inputfileOrbit</code>	filename	approximate positions
 <code>inputfileStarCamera</code>	filename	satellite attitude
 <code>sigmaFactorPhase</code>	expression	PHASE: factor = f(FREQ, ELE, SNR, ROTI, dTEC, IONOINDEX)
 <code>sigmaFactorCode</code>	expression	CODE: factor = f(FREQ, ELE, SNR, ROTI, dTEC, IONOINDEX)
 <code>supportsIntegerAmbiguities</code>	boolean	receiver tracks full cycle integer ambiguities
 <code>wavelengthFactor</code>	double	factor to account for half-wavelength observations (collected by codeless squaring techniques)
 <code>useType</code>	gnssType	only use observations that match any of these patterns
 <code>ignoreType</code>	gnssType	ignore observations that match any of these patterns
 <code>elevationCutOff</code>	angle	[degree] ignore observations below cutoff
 <code>minObsCountPerTrack</code>	uint	tracks with less number of epochs with observations are dropped
 <code>preprocessing</code>	sequence	settings for preprocessing of observations/stations
 <code>printStatistics</code>	boolean	print preprocessing statistics for all receivers
 <code>huber</code>	double	residuals $\$; \$$ huber * sigma0 are downweighted
 <code>huberPower</code>	double	residuals $\$; \$$ huber: sigma = (e/huber)^huberPower * sigma0
 <code>codeMaxPositionDiff</code>	double	[m] max. allowed position error by PPP code only clock error estimation
 <code>denoisingLambda</code>	double	regularization parameter for total variation denoising used in cycle slip detection
 <code>tecWindowSize</code>	uint	(0 = disabled) window size for TEC smoothness evaluation used in cycle slip detection
 <code>tecSigmaFactor</code>	double	factor applied to moving standard deviation used as threshold in TEC smoothness evaluation during cycle slip detection

 <b>outputfileTrackBefore</b>	filename	variables {station}, {prn}, {timeStart}, {timeEnd}, {types}, TEC and MW-like combinations in cycles for each track before cycle slip detection
 <b>outputfileTrackAfter</b>	filename	variables {station}, {prn}, {timeStart}, {timeEnd}, {types}, TEC and MW-like combinations in cycles for each track after cycle slip detection

---

## 5.17 GnssTransmitterGenerator

Definition and basic information of GNSS transmitters.

See also [GnssProcessing](#) and [GnssSimulateReceiver](#).

### 5.17.1 GNSS

A list of satellite PRNs (i.e for GPS: G01, G02, G03, ...) must be provided via [inputfileTransmitterList](#). Satellite system codes follow the [RINEX 3 definition](#), see [GnssType](#) (5.18). All input files except [inputfileAntennaDefinition](#), and [inputfileReceiverDefinition](#) are read for each satellite. The file name is interpreted as a template with the variable {prn} being replaced by the satellite PRN.

Metadata input files (marked with \* below) are provided in GROOPS file formats at <https://ftp.tugraz.at/outgoing/ITSG/groops>. These files are regularly updated.

- [inputfileTransmitterInfo](#)\*: PRN-SVN mapping, antenna offsets and orientations. Created via [GnssAntex2AntennaDefinition](#) or [PlatformCreate](#).
- [inputfileAntennaDefinition](#)\*: Antenna center variations. Created via [GnssAntex2AntennaDefinition](#) or [GnssAntennaDefinitionCreate](#).
- [inputfileReceiverDefinition](#)\*: Transmitted signal types. Created via [GnssReceiverDefinitionCreate](#) in case you want to define which signal types a satellite transmits.
- [inputfileOrbit](#): Converted via [Sp3Format2Orbit](#) or output of [GnssProcessing](#).
- [inputfileAttitude](#): Rotation from body frame to CRF. Created via [SimulateStarCameraGnss](#) or converted via [GnssOrbex2StarCamera](#).
- [inputfileClock](#): Converted via [GnssClockRinex2InstrumentClock](#) or [GnssRinexNavigation2OrbitClock](#) or output of [GnssProcessing](#).

Name	Type	Annotation
<a href="#">inputfileTransmitterList</a>	filename	ascii file with transmitter PRNs, used to loop variable {prn}
<a href="#">inputfileTransmitterInfo</a>	filename	variable {prn} available
<a href="#">inputfileAntennaDefintion</a>	filename	phase centers and variations (ANTEX like)
<a href="#">noAntennaPatternFound</a>	choice	what should happen if no antenna pattern is found for an observation
<a href="#">ignoreObservation</a>		ignore observation if no matching pattern is found
<a href="#">useNearestFrequency</a>		use pattern of nearest frequency if no matching pattern is found
<a href="#">throwException</a>		throw exception if no matching pattern is found
<a href="#">inputfileSignalDefintion</a>	filename	transmitted signal types
<a href="#">inputfileOrbit</a>	filename	variable {prn} available
<a href="#">inputfileAttitude</a>	filename	variable {prn} available
<a href="#">inputfileClock</a>	filename	variable {prn} available
<a href="#">interpolationDegree</a>	uint	for orbit interpolation and velocity calculation

## 5.18 GnssType

A GnssType string consists of six parts (type, frequency, attribute, system, PRN, frequency number) represented by seven characters.

- The first three characters (representing type, frequency, and attribute) correspond to the observation codes of the [RINEX 3 definition](#).
- The satellite system character also follows the RINEX 3 definition:
  - G = GPS
  - R = GLONASS
  - E = Galileo
  - C = BeiDou
  - S = SBAS
  - J = QZSS
  - I = IRNSS
- PRN is a two-digit number identifying a satellite.
- Frequency number is only used for GLONASS, where the range -7 to 14 is represented by letters starting with A.

Each part of a GnssType string can be replaced by a wildcard '\*', enabling the use of these strings as patterns, for example to select a subset of observations (e.g. **C\*\*G\*\*** matches all GPS code/range observations). Trailing wildcards are optional, meaning **L1\*R** is automatically expanded to **L1\*R\*\*\***. For some RINEX 2 types (e.g. Galileo L5) the RINEX 3 attribute is unknown/undefined and can be replaced by ?, for example **L5?E01**.

Examples:

- **C1CG23** = code/range observation, L1 frequency, derived from C/A code, GPS, PRN 23
- **L2PR05B** = phase observation, G2 frequency, derived from P code, GLONASS, PRN 05, frequency number -6
- **\*5\*E\*\*** = all observation types, E5a frequency, all attributes, Galileo, all PRNs

## 5.19 Gravityfield

This class computes functionals of the time depending gravity field, e.g potential, gravity anomalies or gravity gradients.

If several instances of the class are given the results are summed up. Before summation every single result is multipllicated by a **factor**. To subtract a normal field like GRS80 from a potential to get the disturbance potential you must choose one factor by 1 and the other by -1. To get the mean of two fields just set each factor to 0.5.

Some of the instances gives also information about the accuracy. The variance of the result (sum) is computed by means of variance propagation.

### 5.19.1 PotentialCoefficients

Reads coefficients of a spherical harmonics expansion from file. The potential is given by

$$V(\lambda, \vartheta, r) = \frac{GM}{R} \sum_{n=0}^{\infty} \sum_{m=0}^n \left( \frac{R}{r} \right)^{n+1} (c_{nm} C_{nm}(\lambda, \vartheta) + s_{nm} S_{nm}(\lambda, \vartheta)). \quad (5.41)$$

If set the expansion is limited in the range between **minDegree** and **maxDegree** inclusivly. The computed result is multiplied with **factor**. If **setSigmasToZero** is true the variances are set to zero. This option is only important for variance propagation and does not change the result of the gravity field functionals.

Name	Type	Annotation
inputfilePotentialCoefficients	filename	
minDegree	uint	
maxDegree	uint	
factor	double	the result is multiplied by this factor, set -1 to subtract the field
setSigmasToZero	boolean	set variances to zero, should be used by adding back reference fields

### 5.19.2 PotentialCoefficientsInterior

Reads coefficients of a spherical harmonics expansion (for inner space) from file. If set the expansion is limited in the range between **minDegree** and **maxDegree** inclusivly. The computed result is multiplied with **factor**. If **setSigmasToZero** is true the variances are set to zero. This option is only important for error propagation and does not change the result of the gravity field functionals.

Name	Type	Annotation
inputfilePotentialCoefficients	filename	
minDegree	uint	
maxDegree	uint	
factor	double	the result is multiplied by this factor, set -1 to subtract the field
setSigmasToZero	boolean	set variances to zero, should be used by adding back reference fields

### 5.19.3 FromParametrization

Reads a solution vector from file **inputfileSolution** which may be computed by a least squares adjustment (e.g. by **NormalsSolverVCE**). The coefficients of the vector are interpreted from position **indexStart** (counting from zero) with help of **parametrizationGravity**. If the solution file contains solution of several right hand sides you can choose one with number **rightSide** (counting from zero). You can also read a vector from file **inputfileSigmax** containing the accuracies of the coefficients.

The computed result is multiplied with **factor**.

Name	Type	Annotation
parametrization	parametrizationGravity	
inputfileSolution	filename	solution vector
inputfileSigmax	filename	standards deviations or covariance matrix of the solution
indexStart	uint	position in the solution vector
rightSide	uint	if solution contains several right hand sides, select one
factor	double	the result is multiplied by this factor, set -1 to subtract the field

### 5.19.4 TimeSplines

Read a time variable gravity field from file **inputfileTimeSplinesGravityfield** represented by a spherical harmonics expansion in the spatial domain and spline functions in the time domain. If set the expansion is limited in the range between **minDegree** and **maxDegree** inclusivly.

This file can be created for example by **Gravityfield2TimeSplines** or **PotentialCoefficients2BlockMeanTimeSplines**.

The computed result is multiplied with **factor**.

Name	Type	Annotation
inputfileTimeSplinesGravityfield	filename	
inputfileTimeSplinesCovariance	filename	
minDegree	uint	
maxDegree	uint	
factor	double	the result is multiplied by this factor, set -1 to subtract the field

### 5.19.5 Trend

The given **gravityfield** is interpreted as trend function and the result is computed at time  $t$  as follows

$$V(\mathbf{x}, t) = \frac{t - t_0}{\Delta t} V(\mathbf{x}), \quad (5.42)$$

with  $t_0$  is **timeStart** and  $\Delta t$  is **timeStep**.

Name	Type	Annotation
gravityfield	gravityfield	this field is multiplicated by (time-time0)/timeStep
timeStart	time	reference time
timeStep	time	

### 5.19.6 Oscillation

The given **gravityfield** is interpreted as oscillation function and the result is computed at time  $t$  as follows

$$V(\mathbf{x}, t) = \cos(\omega)V_{cos}(\mathbf{x}) + \sin(\omega)V_{sin}(\mathbf{x}), \quad (5.43)$$

with  $\omega = \frac{2\pi}{T}(t - t_0)$ .

Name	Type	Annotation
gravityfieldCos	gravityfield	multiplicated by $\cos(2\pi/T(\text{time}-\text{time0}))$
gravityfieldSin	gravityfield	multiplicated by $\sin(2\pi/T(\text{time}-\text{time0}))$
time0	time	reference time
period	time	[day]

### 5.19.7 InInterval

A **gravityfield** is only evaluated in the interval between **timeStart** inclusively and **timeEnd** exclusively. Outside the interval the result is zero.

This class is useful to get a time series of monthly mean GRACE gravity field solutions. In each month another file of potentialCoefficients is valid. This can easily be created with **loop**.

Name	Type	Annotation
gravityfield	gravityfield	
timeStart	time	first point in time
timeEnd	time	last point in time will be less or equal timeEnd

### 5.19.8 Tides

Treat **tides** as gravitational forces. The tides need a realization of **earthRotation** to transform between the CRF and TRF and to compute rotational deformation from polar motion. It also needs **ephemerides** from Sun, moon, and planets.

Name	Type	Annotation
tides	tides	
earthRotation	earthRotation	
ephemerides	ephemerides	

### 5.19.9 Topography

The gravity is integrated from a topographic mass distribution. For each grid point in **inputfileGridRectangular** a prisma with **density** is assumed. The horizontal extension is computed from the grid spacing and the vertical extension is given by **radialLowerBound** and **radialUpperBound** above ellipsoid. All values are expressions and computed for each point with given data in the grid file. The standard variables for grids are available, see [dataVariables \(1.2.3\)](#).

Example: The grid file contains the orthometric height of the topography in the first column, the geoid height in the second and the mean density of each prism in the third column. In this case the following settings should be used:

- **radialUpperBound** = data0+data1,
- **radialLowerBound** = data1,
- **density** = data2.

As the prim computation is time consuming a maximum distance around the evaluation point can be defined with **distancePrism**. Afterwards a simplified radial line (the prism mass is concentrated to a line in the center) is used up to a distance of **distanceLine**. At last the prim is approximated by a point mass in the center up to a distance **distanceMax** (if set). Prisms nearby the evaluation point can be excluded with **distanceMin**.

Name	Type	Annotation
inputfileGridRectangular	filename	Digital Terrain Model
<b>density</b>	expression	expression [kg/m**3]
<b>radialUpperBound</b>	expression	expression (variables 'height', 'data', 'L', 'B' and, 'area' are taken from the gridded data)
<b>radialLowerBound</b>	expression	expression (variables 'height', 'data', 'L', 'B' and, 'area' are taken from the gridded data)
<b>distanceMin</b>	double	[km] min. influence distance (ignore near zone)
<b>distancePrism</b>	double	[km] max. distance for prism formula
<b>distanceLine</b>	double	[km] max. distance for radial integration
<b>distanceMax</b>	double	[km] max. influence distance (ignore far zone)
<b>factor</b>	double	the result is multiplied by this factor, set -1 to subtract the field

### 5.19.10 EarthquakeOscillation

The given **gravityfield** is interpreted as an oscillation function in the gravitational potential field, caused by large earthquakes. The result is computed at time  $t$  as follows:

$$C_{lm}(\mathbf{t}) = \sum_{n=0}^N C_{nlm} \left(1 - \cos(\omega) \exp\left(\frac{-\omega}{2Q_{nlm}}\right)\right), \quad (5.44)$$

with  $\omega = \frac{2\pi}{T_{nlm}}(t - t_0)$ . In this equation,  $Q_{nlm}$  is the attenuation factor,  $n$  is the overtone factor,  $m$  is degree,  $l$  is order, and  $t$  is time in second.  $T_{nlm}$  and  $Q_{nlm}$  are computed with the elastic Earth model or observed from the long period record of superconducting gravimeter measurements after the earthquakes.

Name	Type	Annotation
inputCoefficientMatrix	filename	oscillation model parameters
time0	time	the time earthquake happened
minDegree	uint	
maxDegree	uint	
GM	double	Geocentric gravitational constant
R	double	reference radius

### 5.19.11 Filter

Convert **gravityfield** to spherical harmonics and **filter** the coefficients.

Name	Type	Annotation

---

➡ gravityfield	gravityfield
➡ filter	sphericalHarmonicsFilter

---

### 5.19.12 Group

Groups a set of ➡ **gravityfield** and has no further effect itself.

---

Name	Type	Annotation
➡ gravityfield	gravityfield	
➡ factor	double	the result is multiplied by this factor, set -1 to subtract the field

---

## 5.20 Grid

This class generates a set of grid points. In a first step, the grid is always generated globally, with **border** a regional subset of points can be extracted from the global grid. The parameters **R** and **inverseFlattening** define the shape of the ellipsoid on which the grid is generated. In case **inverseFlattening** is chosen as zero, a sphere is used. With **height** the distance of the points above the ellipsoid can be defined. In addition to the location of the points, weights are assigned to each of the points. These weights can be regarded as the surface element associated with each grid point.

### 5.20.1 Geograph

The geographical grid is an equal-angular point distribution with points located along meridians and along circles of latitude. **deltaLambda** denotes the angular difference between adjacent points along meridians and **deltaPhi** describes the angular difference between adjacent points along circles of latitude. The point setting results as follows:

$$\lambda_i = \frac{\Delta\lambda}{2} + i \cdot \Delta\lambda \quad \text{with} \quad 0 \leq i < \frac{360^\circ}{\Delta\lambda}, \quad (5.45)$$

$$\varphi_j = -90^\circ + \frac{\Delta\varphi}{2} + j \cdot \Delta\varphi \quad \text{with} \quad 0 \leq j < \frac{180^\circ}{\Delta\varphi}. \quad (5.46)$$

The number of grid points can be determined by

$$I = \frac{360^\circ}{\Delta\lambda} \cdot \frac{180^\circ}{\Delta\varphi}. \quad (5.47)$$

The weights are calculated according to

$$w_i = \int_{\lambda_i - \frac{\Delta\lambda}{2}}^{\lambda_i + \frac{\Delta\lambda}{2}} \int_{\vartheta_i - \frac{\Delta\vartheta}{2}}^{\vartheta_i + \frac{\Delta\vartheta}{2}} = 2 \cdot \Delta\lambda \sin(\Delta\vartheta) \sin(\vartheta_i). \quad (5.48)$$

Name	Type	Annotation
<b>deltaLambda</b>	angle	
<b>deltaPhi</b>	angle	
<b>height</b>	double	ellipsoidal height expression (variables 'height', 'L', 'B')
<b>R</b>	double	major axis of the ellipsoid/sphere
<b>inverseFlattening</b>	double	flattening of the ellipsoid, 0: sphere
<b>border</b>	border	

### 5.20.2 TriangleVertex

The zeroth level of densification coincides with the 12 icosahedron vertices, as displayed in the upper left part of Fig. 5.7. Then, depending on the envisaged densification, each triangle edge is divided into  $n$  parts, illustrated in the upper right part of Fig. 5.7. The new nodes on the edges are then connected by arcs of great circles parallel to the triangle edges. The intersections of each three corresponding parallel lines become nodes of the densified grid as well. As in case of a spherical triangle those three connecting lines do not exactly intersect in one point, the center of the resulting triangle is used as location for the new node (lower left part of Fig. 5.7). The lower right side of Fig. 5.7 finally shows the densified triangle vertex grid for a level of

$n = 3$ . The number of grid points in dependence of the chosen level of densification can be calculated by

$$I = 10 \cdot (n + 1)^2 + 2. \quad (5.49)$$

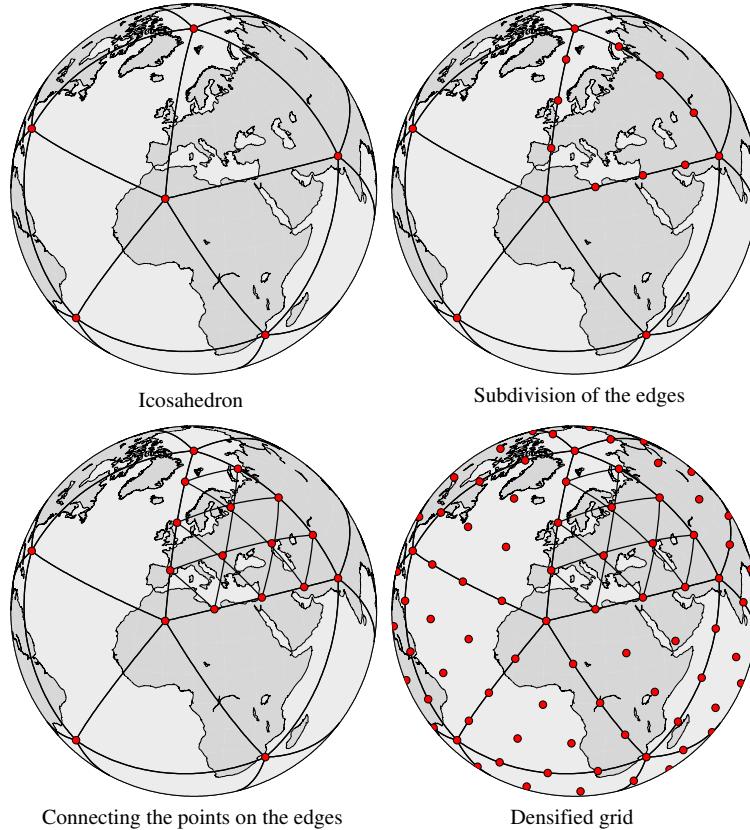


Figure 5.7: TriangleVertex grid.

Name	Type	Annotation
level	uint	division of icosahedron, point count = $10*(n+1)^{**2}+2$
R	double	major axis of the ellipsoid/sphere
inverseFlattening	double	flattening of the ellipsoid, 0: sphere
border	border	

### 5.20.3 TriangleCenter

The points of the zeroth level are located at the centers of the icosahedron triangles. To achieve a finer grid, each of the triangles is divided into four smaller triangles by connecting the midpoints of the triangle edges. The refined grid points are again located at the center of the triangles. Subsequently, the triangles can be further densified up to the desired level of densification  $n$ , which is defined by **level**.

The number of global grid points for a certain level can be determined by

$$I = 20 \cdot 4^n. \quad (5.50)$$

Thus the quantity of grid points depends exponentially on the level  $n$ , as with every additional level the number of grid points quadruples.

Name	Type	Annotation
level	uint	division of icosahedron, point count = $5*4^{**}(n+1)$
R	double	major axis of the ellipsoid/sphere
inverseFlattening	double	flattening of the ellipsoid, 0: sphere
border	border	

#### 5.20.4 Gauss

The grid features equiangular spacing along circles of latitude with  $\blacktriangleright$  **parallelsCount** defining the number  $L$  of the parallels.

$$\Delta\lambda = \frac{\pi}{L} \quad \Rightarrow \quad \lambda_i = \frac{\Delta\lambda}{2} + i \cdot \Delta\lambda \quad \text{with} \quad 0 \leq i < 2L. \quad (5.51)$$

Along the meridians the points are located at  $L$  parallels at the  $L$  zeros  $\vartheta_j$  of the Legendre polynomial of degree  $L$ ,

$$P_L(\cos \vartheta_j) = 0. \quad (5.52)$$

Consequently, the number of grid points sums up to

$$I = 2 \cdot L^2. \quad (5.53)$$

The weights can be calculated according to

$$w_i(L) = \Delta\lambda \frac{2}{(1 - t_i^2)(P'_L(\cos(\vartheta_i)))^2}, \quad (5.54)$$

Name	Type	Annotation
parallelsCount	uint	
R	double	major axis of the ellipsoid/sphere
inverseFlattening	double	flattening of the ellipsoid, 0: sphere
border	border	

#### 5.20.5 Reuter

The Reuter grid features equi-distant spacing along the meridians determined by the control parameter  $\gamma$  according to

$$\Delta\vartheta = \frac{\pi}{\gamma} \quad \Rightarrow \quad \vartheta_j = j\Delta\vartheta, \quad \text{with} \quad 1 \leq j \leq \gamma - 1. \quad (5.55)$$

Thus  $\gamma + 1$  denotes the number of points per meridian, as the two poles are included in the point distribution as well. Along the circles of latitude, the number of grid points decreases with increasing latitude in order to achieve an evenly distributed point pattern. This number is chosen, so that the points along each circle of latitude have the same spherical distance as two adjacent latitudes. The resulting relationship is given by

$$\Delta\vartheta = \arccos(\cos^2 \vartheta_j + \sin^2 \vartheta_j \cos \Delta\lambda_j). \quad (5.56)$$

The left hand side of this equation is the spherical distance between adjacent latitudes, the right hand side stands for the spherical distance between two points with the same polar distance  $\vartheta_j$  and a longitudinal difference of  $\Delta\lambda_j$ . This longitudinal distance can be adjusted depending on  $\vartheta_j$  to fulfill Eq. (5.56). The

resulting formula for  $\Delta\lambda_i$  is

$$\Delta\lambda_j = \arccos \left( \frac{\sin \Delta\vartheta - \cos^2 \vartheta_j}{\sin^2 \vartheta_j} \right). \quad (5.57)$$

The number of points  $\gamma_j$  for each circle of latitude can then be determined by

$$\gamma_j = \left[ \frac{2\pi}{\Delta\lambda_j} \right]. \quad (5.58)$$

Here the Gauss bracket  $[x]$  specifies the largest integer equal to or less than  $x$ . The longitudes are subsequently determined by

$$\lambda_{ij} = \frac{\Delta\lambda_j}{2} + i \cdot (2\pi/\gamma_j), \quad \text{with} \quad 0 \leq i < \gamma_j. \quad (5.59)$$

The number of grid points can be estimated by

$$I = \leq 2 + \frac{4}{\pi} \gamma^2, \quad (5.60)$$

The  $\leq$  results from the fact that the  $\gamma_j$  are restricted to integer values.

Name	Type	Annotation
gamma	uint	number of parallels
height	double	ellipsoidal height
R	double	major axis of the ellipsoid/sphere
inverseFlattening	double	flattening of the ellipsoid, 0: sphere
border	border	

## 5.20.6 Corput

This kind of grid distributes an arbitrarily chosen number of  $I$  points (defined by **globalPointsCount**) following a recursive, quasi random sequence. In longitudinal direction the pattern follows

$$\Delta\lambda = \frac{2\pi}{I} \quad \Rightarrow \quad \frac{\Delta\lambda}{2} + \lambda_i = i \cdot \Delta\lambda \quad \text{with} \quad 1 \leq i \leq I. \quad (5.61)$$

This implies that every grid point features a unique longitude, with equi-angular longitudinal differences.

The polar distance in the form  $t_i = \cos \vartheta_i$  for each point is determined by the following recursive sequence:

- Starting from an interval  $t \in [-1, 1]$ .
- If  $I = 1$ , then the midpoint of the interval is returned as result of the sequence, and the sequence is terminated.
- If the number of points is uneven, the midpoint is included into the list of  $t_i$ .
- Subsequently, the interval is bisected into an upper and lower half, and the sequence is called for both halves.
- $t$  from upper and lower half are alternately sorted into the list of  $t_i$ .
- The polar distances are calculated by

$$\vartheta_i = \arccos t_i. \quad (5.62)$$

Name	Type	Annotation
globalPointsCount	uint	
height	double	ellipsoidal height
R	double	major axis of the ellipsoid/sphere
inverseFlattening	double	flattening of the ellipsoid, 0: sphere
border	border	

### 5.20.7 Driscoll

The Driscoll-Healy grid, has equiangular spacing along the meridians as well as along the circles of latitude. In longitudinal direction (along the parallels), these angular differences for a given **dimension**  $L$  coincide with those described for the corresponding geographical grid and Gauss grid. Along the meridians, the size of the latitudinal differences is half the size compared to the geographical grid. This results in the following point pattern,

$$\begin{aligned}\Delta\lambda &= \frac{\pi}{L} \quad \Rightarrow \quad \lambda_i = \frac{\Delta\lambda}{2} + i \cdot \Delta\lambda \quad \text{with} \quad 0 \leq i < 2L, \\ \Delta\vartheta &= \frac{\pi}{2L} \quad \Rightarrow \quad \vartheta_j = j \cdot \Delta\vartheta \quad \text{with} \quad 1 \leq j \leq 2L.\end{aligned}\tag{5.63}$$

Consequently, the number of grid points is

$$I = 4 \cdot L^2.\tag{5.64}$$

The weights are given by

$$w_i = \Delta\lambda \frac{4}{2L} \sin(\vartheta_i) \sum_{l=0}^{L-1} \frac{\sin[(2l+1)\vartheta_i]}{2l+1}.\tag{5.65}$$

Name	Type	Annotation
dimension	uint	number of parallels = 2*dimension
height	double	ellipsoidal height
R	double	major axis of the ellipsoid/sphere
inverseFlattening	double	flattening of the ellipsoid, 0: sphere
border	border	

### 5.20.8 SinglePoint

Creates one single point.

Name	Type	Annotation
L	angle	longitude
B	angle	latitude
height	double	ellipsoidal height
area	double	associated area element on unit sphere
R	double	major axis of the ellipsoid/sphere
inverseFlattening	double	flattening of the ellipsoid, 0: sphere

### 5.20.9 SinglePointCartesian

Creates one single point.

Name	Type	Annotation
x	double	[m]
y	double	[m]
z	double	[m]
area	double	associated area element on unit sphere

### 5.20.10 File

In this class grid is read from a file, which is given by [inputfileGrid](#). A corresponding file can be generated with [GriddedDataCreate](#) or with [Matrix2GriddedData](#).

Name	Type	Annotation
inputfileGrid	filename	
border	border	

## 5.21 InstrumentType

Defines the type of an [instrument](#) file.

Name	Type	Annotation
instrumentTypeType	choice	instrument type
INSTRUMENTTIME		time without data
MISCVOLUME		single value
MISCVOLUME		multiple values
VECTOR3D		x, y, z
COVARIANCE3D		xx, yy, zz, xy, xz, yz
ORBIT		position [m], velocity [m/s], acceleration [m/s^2] (each x, y, z)
STARCAMERA		quaternions (q0, qx, qy, qz)
ACCELEROMETER		x, y, z [m/s^2]
SATELLITETRACKING		range [m], range rate [m/s], range acceleration [m/s^2]
GRADIOMETER		xx, yy, zz, xy, xz, yz [1/s^2]
GNSSRECEIVER		GNSS phase/code observations [m]
OBSERVATIONSIGMA		accuracy
MASS		
THRUSTER		
MAGNETOMETER		
ACCHOUSEKEEPING		

## 5.22 InterpolatorTimeSeries

This class resamples data of a times series to new points in time.

### 5.22.1 Polynomial

Polynomial prediction using a moving polynomial of **polynomialDegree**. The optimal polynomial is chosen based on the centricity of the data points around the resampling point and the distance to all polynomial data points. All polynomial data points must be within **maxDataPointRange**. Resampling points within **maxExtrapolationDistance** of the polynomial will be extrapolated. The elements **maxDataPointRange** and **maxExtrapolationDistance** are given in the unit of seconds. If negative values are used, the unit is relative to the median input sampling.

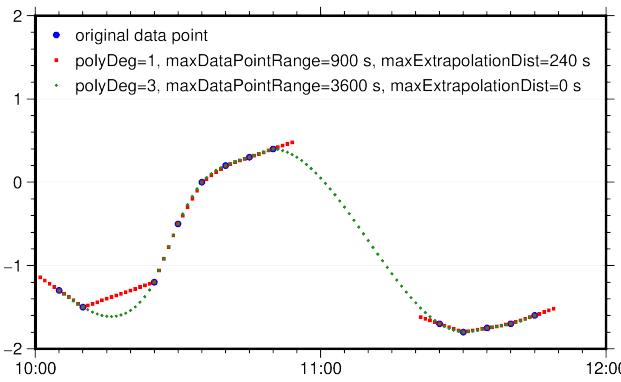


Figure 5.8: Example of polynomial prediction when resampling from 5 to 1 minute sampling

Name	Type	Annotation
polynomialDegree	uint	degree of the moving polynomial
maxDataPointRange	double	[seconds] all degree+1 data points must be within this range for a valid polynomial
maxExtrapolationDistance	double	[seconds] resampling points within this range of the polynomial will be extrapolated

### 5.22.2 Least squares polynomial fit

A polynomial of **polynomialDegree** is estimated using all data points within **maxDataPointDistance** of the resampling point. This polynomial is then used to predict the resampling point. A resampling point will be extrapolated if there are only data points before/after as long as the closest one is within **maxExtrapolationDistance**. The elements **maxDataPointDistance** and **maxExtrapolationDistance** are given in the unit of seconds. If negative values are used, the unit is relative to the median input sampling.

Name	Type	Annotation
polynomialDegree	uint	degree of the estimated polynomial
maxDataPointDistance	double	[seconds] all data points within this distance around the resampling point will be used
maxExtrapolationDistance	double	[seconds] resampling points within this range of the polynomial will be extrapolated

### 5.22.3 Fill gaps with least squares polynomial fit

Name	Type	Annotation
polynomialDegree	uint	degree of the estimated polynomial
maxDataGap	double	[seconds] max data gap to interpolate
maxDataSpan	double	[seconds] time span on each side used for least squares fit
margin	double	[seconds] margin for identical times

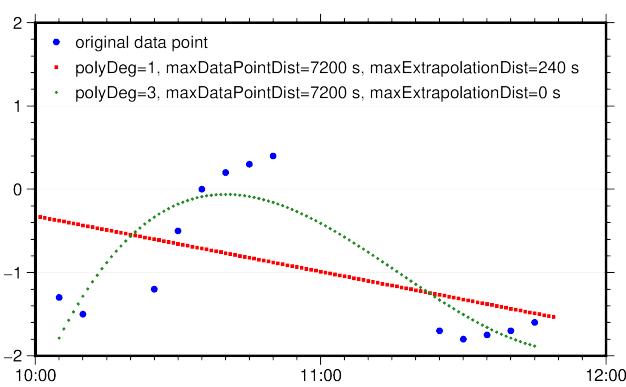


Figure 5.9: Example of least squares polynomial fit when resampling from 5 to 1 minute sampling

## 5.23 Kernel

Kernel defines harmonic isotropic integral kernels  $K$ .

$$T(P) = \frac{1}{4\pi} \int_{\Omega} K(P, Q) \cdot f(Q) d\Omega(Q), \quad (5.66)$$

where  $T$  is the (disturbance)potential and  $f$  is a functional on the spherical surface  $\Omega$ . The Kernel can be expanded into a series of (fully normalized) legendre polynomials

$$K(\cos \psi, r, R) = \sum_n \left( \frac{R}{r} \right)^{n+1} k_n \sqrt{2n+1} \bar{P}_n(\cos \psi). \quad (5.67)$$

On the one hand the kernel defines the type of the functionals  $f$  that are measured or have to be computed, e.g. gravity anomalies given by the Stokes-kernel. On the other hand the kernel functions can be used as basis functions to represent the gravity field, e.g. as spline functions or wavelets.

### 5.23.1 GeoidHeight

The geoid height is defined by Bruns formula

$$N = \frac{1}{\gamma} T \quad (5.68)$$

with  $T$  the disturbance potential and the normal gravity

$$\gamma = \gamma_0 - 0.30877 \cdot 10^{-5} / s^2 (1 - 0.00142 \sin^2(B)) h \quad (5.69)$$

and

$$\gamma_0 = 9.780327 \text{ m/s}^2 (1 + 0.0053024 \sin^2(B) - 0.0000058 \sin^2(2B)) \quad (5.70)$$

where  $h$  is the ellipsoidal height in meter and  $B$  the longitude.

The kernel is given by

$$K(\cos \psi, r, R) = \gamma \frac{R(r^2 - R^2)}{l^3}, \quad (5.71)$$

and the coefficients in (5.67) are

$$k_n = \gamma. \quad (5.72)$$

### 5.23.2 Anomalies

Gravity anomalies in linearized form are defined by

$$\Delta g = -\frac{\partial T}{\partial r} - \frac{2}{r} T. \quad (5.73)$$

The Stokes kernel is given by

$$K(\cos \psi, r, R) = \frac{2R^2}{l} - 3 \frac{Rl}{r^2} - \frac{R^2}{r^2} \cos \psi \left( 5 + 3 \ln \frac{l+r-R \cos \psi}{2r} \right), \quad (5.74)$$

and the coefficients in (5.67) are

$$k_n = \frac{R}{n-1}. \quad (5.75)$$

### 5.23.3 Disturbance

Gravity disturbances in linearized form are defined by

$$\delta g = -\frac{dT}{dr}. \quad (5.76)$$

The Hotine kernel is given by

$$K(\cos \psi, r, R) = \frac{2R^2}{l} - R \ln \frac{l + R - r \cos \psi}{r(1 - \cos \psi)}, \quad (5.77)$$

and the coefficients in (5.67) are

$$k_n = \frac{R}{n+1}. \quad (5.78)$$

### 5.23.4 Potential

The Abel-Poisson kernel is given by

$$K(\cos \psi, r, R) = \frac{R(r^2 - R^2)}{l^3}, \quad (5.79)$$

and the coefficients in (5.67) are

$$k_n = 1. \quad (5.80)$$

### 5.23.5 Density

This kernel defines a point mass or mass on a single layer ( $1/l$ -kernel) taking the effect of the loading into account.

The coefficients of the kernel defined in (5.67) are

$$k_n = 4\pi GR \frac{1 + k'_n}{2n + 1}, \quad (5.81)$$

where  $G$  is the gravitational constant and  $k'_n$  are the load Love numbers.

Name	Type	Annotation
inputfileLoadingLoveNumber	filename	

### 5.23.6 WaterHeight

Height of equivalent water columns taking the effect of the loading into account.

The coefficients of the kernel defined in (5.67) are

$$k_n = 4\pi G\rho R \frac{1 + k'_n}{2n + 1}, \quad (5.82)$$

where  $G$  is the gravitational constant,  $\rho$  is the **density** of water and  $k'_n$  are the load Love numbers.

Name	Type	Annotation
density	double	[kg/m**3]
inputfileLoadingLoveNumber	filename	

### 5.23.7 BottomPressure

Ocean bottom pressure caused by water and atmosphere masses columns taking the effect of the loading into account.

The coefficients of the kernel defined in (5.67) are

$$k_n = \frac{4\pi G R}{\gamma} \frac{1 + k'_n}{2n + 1}, \quad (5.83)$$

where  $G$  is the gravitational constant,  $\gamma$  is the normal gravity and  $k'_n$  are the load Love numbers.

Name	Type	Annotation
inputfileLoadingLoveNumber	filename	

### 5.23.8 Deformation

Computes the radial deformation caused by loading.

The coefficients of the kernel defined in (5.67) are

$$k_n = \gamma \frac{1 + k'_n}{h'_n}, \quad (5.84)$$

where  $\gamma$  is the normal gravity defined in (5.69),  $h'_n$  and  $k'_n$  are the load Love numbers and the load deformation Love numbers.

Name	Type	Annotation
inputfileDeformationLoadLoveNumber	filename	
inputfilePotentialLoadLoveNumber	filename	if full potential is given and not only loading potential

### 5.23.9 RadialGradient

This kernel defines the second radial derivative of the (disturbance) potential.

$$T_{rr} = \frac{\partial^2 T}{\partial r^2}. \quad (5.85)$$

The coefficients of the kernel defined in (5.67) are

$$k_n = \frac{r^2}{(n + 1)(n + 2)}. \quad (5.86)$$

### 5.23.10 Coefficients

The kernel is defined by the coefficients  $k_n$  given by file.

Name	Type	Annotation
inputfileCoefficients	filename	

### 5.23.11 FilterGauss

Another **kernel** is smoothed by a gauss filter which is defined by

$$F(\cos \psi) = \frac{b \cdot e^{-b(1-\cos \psi)}}{1 - e^{-2b}} \quad (5.87)$$

with  $b = \frac{\ln(2)}{1-\cos(r/R)}$  where  $r$  is the given smoothing **radius** in km and  $R = 6378.1366$  km is the Earth radius. The coefficients  $k_n$  of the **kernel** are multiplicatively by

$$f_n = \frac{1}{2n+1} \int_{-1}^1 F(t) \cdot \bar{P}_n(t) dt. \quad (5.88)$$

Name	Type	Annotation
kernel	kernel	
radius	double	filter radius [km]

### 5.23.12 BlackmanLowpass

Another **kernel** is smoothed by a Blackman low-pass filter. The filter is defined through the beginning and end of the transition from pass-band to stop-band. This transition band is specified by **startDegreeTransition** ( $n_1$ ) and **stopDegreeTransition** ( $n_2$ ).

The coefficients of this kernel are defined as

$$\begin{cases} 1 & \text{for } n < n_1 \\ A_n^2 & \text{for } n_1 \leq n \leq n_2 \\ 0 & \text{for } n > n_2 \end{cases} \quad (5.89)$$

with

$$A_n = 0.42 + 0.5 \cos\left(\pi \frac{n - n_1}{n_2 - n_1}\right) + 0.08 \cos\left(2\pi \frac{n - n_1}{n_2 - n_1}\right). \quad (5.90)$$

Name	Type	Annotation
kernel	kernel	
startDegreeTransition	uint	minimum degree in transition band
stopDegreeTransition	uint	maximum degree in transition band

### 5.23.13 Truncation

Another **kernel** is truncated before **minDegree** and after **maxDegree**. The coefficients of this kernel are defined as

$$k_n = \begin{cases} 1 & \text{for } n_{\minDegree} \leq n \leq n_{\maxDegree} \\ 0 & \text{else.} \end{cases} \quad (5.91)$$

Name	Type	Annotation
kernel	kernel	
minDegree	uint	truncate before minDegree
maxDegree	uint	truncate after maxDegree

### 5.23.14 SelenoidHeight

The selenoid height is defined by Bruns formula

$$N = \frac{1}{\gamma} T \quad (5.92)$$

with  $T$  the disturbance potential and the normal gravity  $\gamma = \frac{GM}{R^2}$  of the moon.

The kernel is given by

$$K(\cos \psi, r, R) = \gamma \frac{R(r^2 - R^2)}{l^3}, \quad (5.93)$$

and the coefficients in (5.67) are

$$k_n = \gamma. \quad (5.94)$$

## 5.24 Loop

Generates a sequence with variables to loop over. The variable names can be set with **variableLoop...** and the current values are assigned to the variables for each loop step.

With **condition** only a subset of loop steps are performed. The **variableLoopIndex** and **variableLoopCount** are not affected by the condition. The result would therefore be the same as using **LoopPrograms** with a nested **IfPrograms**.

See [Loop and conditions \(1.3\)](#) for usage.

### 5.24.1 TimeSeries

Loop over points in time.

Name	Type	Annotation
timeSeries	timeSeries	loop is called for every point in time
variableLoopTime	string	variable with time of each loop
variableLoopIndex	string	variable with index of current iteration (starts with zero)
variableLoopCount	string	variable with total number of iterations
condition	condition	check before each loop step

### 5.24.2 TimeIntervals

Loop over the intervals between points in time.

Name	Type	Annotation
timeIntervals	timeSeries	loop is called for every interval
variableLoopTimeStart	string	variable with starting time of each interval
variableLoopTimeEnd	string	variable with ending time of each interval
variableLoopIndex	string	variable with index of current iteration (starts with zero)
variableLoopCount	string	variable with total number of iterations
condition	condition	check before each loop step

### 5.24.3 ManualList

Loop over list of strings.

Name	Type	Annotation
string	string	explicit list of strings
variableLoopString	string	name of the variable to be replaced
variableLoopIndex	string	variable with index of current iteration (starts with zero)
variableLoopCount	string	variable with total number of iterations
condition	condition	check before each loop step

### 5.24.4 ManualTable

Loop over rows of a table containing strings. The table can be defined **rowWise** or **columnWise**. Each row/column must have the same number of cells.

Name	Type	Annotation
table	choice	define table by rows/columns
rowWise	sequence	define table by rows
row	sequence	define table by rows
cell	string	explicit list of cells in row/column
columnWise	sequence	define table by columns
column	sequence	define table by columns
cell	string	explicit list of cells in row/column
variableLoopString	string	1. variable name for the 1. column, next variable name for the 2. column, ...
variableLoopIndex	string	variable with index of current iteration (starts with zero)
variableLoopCount	string	variable with total number of iterations
condition	condition	check before each loop step

### 5.24.5 FileAscii

Loop over list of strings from files.

Name	Type	Annotation
inputfile	filename	simple ASCII file with strings (separated by whitespace)
sort	boolean	sort entries alphabetically (ascending)
removeDuplicates	boolean	remove duplicate entries (order is preserved)
startIndex	uint	start at element startIndex (counting from 0)
count	uint	use count elements (default: use all)
variableLoopString	string	name of the variable to be replaced
variableLoopIndex	string	variable with index of current iteration (starts with zero)
variableLoopCount	string	variable with total number of iterations
condition	condition	check before each loop step

### 5.24.6 FileAsciiTable

Loop over rows of a table containing strings. Each row must have the same number of columns.

Name	Type	Annotation
inputfile	filename	simple ASCII file with multiple columns (separated by whitespace)
startLine	uint	start at line startLine (counting from 0)
countLines	uint	read count lines (default: all)
variableLoopString	string	1. variable name for the 1. column, next variable name for the 2. column, ...
variableLoopIndex	string	variable with index of current iteration (starts with zero)
variableLoopCount	string	variable with total number of iterations
condition	condition	check before each loop step

### 5.24.7 FileLines

Loop over lines of a text file.

Name	Type	Annotation
inputfile	filename	simple ASCII file with lines
sort	boolean	sort lines alphabetically (ascending)

---

removeDuplicates	boolean	remove duplicate lines (order is preserved)
startIndex	uint	start at element startIndex (counting from 0)
count	uint	use number of lines only (default: use all)
variableLoopLine	string	name of the variable to be replaced
variableLoopIndex	string	variable with index of current iteration (starts with zero)
variableLoopCount	string	variable with total number of iterations
condition	condition	check before each loop step

---

### 5.24.8 Matrix

Loop over rows of a matrix. To define the loop variables the standard data variables of the matrix are available, see [dataVariables \(1.2.3\)](#).

---

Name	Type	Annotation
inputfile	filename	
transpose	boolean	effectively loop over columns
startRow	expression	start at this row (variable: rows)
countRows	expression	use this many rows (variable: rows)
variableLoop	expression	define a variable by name = expression (input columns are named data0, data1, ...)
variableLoopIndex	string	variable with index of current iteration (starts with zero)
variableLoopCount	string	variable with total number of iterations
condition	condition	check before each loop step

---

### 5.24.9 UniformSampling

Loop over sequence of numbers.

---

Name	Type	Annotation
rangeStart	double	start of range
rangeEnd	double	end of range (inclusive)
sampling	double	sampling
variableLoopNumber	string	name of the variable to be replaced
variableLoopIndex	string	variable with index of current iteration (starts with zero)
variableLoopCount	string	variable with total number of iterations
condition	condition	check before each loop step

---

### 5.24.10 DirectoryListing

Loop over files of a directory.

---

Name	Type	Annotation
directory	filename	directory
pattern	string	wildcard pattern
isRegularExpression	boolean	pattern is a regular expression
variableLoopFile	string	name of the variable to be replaced
variableLoopIndex	string	variable with index of current iteration (starts with zero)
variableLoopCount	string	variable with total number of iterations
condition	condition	check before each loop step

---

### 5.24.11 CommandOutput

Loop over lines of command output.

Name	Type	Annotation
command	filename	each output line becomes a loop iteration
silently	boolean	without showing the output.
variableLoopString	string	name of the variable to be replaced
variableLoopIndex	string	variable with index of current iteration (starts with zero)
variableLoopCount	string	variable with total number of iterations
condition	condition	check before each loop step

### 5.24.12 Loop

Loop over nested loops. First **loop** is outermost loop, every subsequent **loop** is one level below the previous **loop**.

Name	Type	Annotation
loop	loop	subloop
variableLoopIndex	string	variable with index of current iteration (starts with zero)
condition	condition	check before each loop step

### 5.24.13 PlatformEquipment

Loop over specific equipment of a [platform file](#).

Name	Type	Annotation
inputfilePlatform	filename	platform info file
equipmentType	choice	equipment type to loop over
all		loop over all types
gnssAntenna		loop over antennas
gnssReceiver		loop over receivers
slrStation		loop over SLR stations
slrRetroReflector		loop over laser retroreflectors
satelliteIdentifier		loop over satellite identifiers
other		loop over other types
variableLoopName	string	variable with name
variableLoopSerial	string	variable with serial
variableLoopInfo	string	variable with radome (antenna) or version (receiver)
variableLoopTimeStart	string	variable with start time
variableLoopTimeEnd	string	variable with end time
variableLoopPositionX	string	variable with position x
variableLoopPositionY	string	variable with position y
variableLoopPositionZ	string	variable with position z
variableLoopIndex	string	variable with index of current iteration (starts with zero)
variableLoopCount	string	variable with total number of iterations
condition	condition	check before each loop step

### 5.24.14 FileGnssStationInfo

DEPRECATDED. Use LoopPlatformEquipment instead.

Name	Type	Annotation
inputfileGnssStationInfo	filename	station/transmitter info file
infoType	choice	info to loop over
antenna		loop over antennas
receiver		loop over receivers
variableLoopName	string	variable with antenna/receiver name
variableLoopSerial	string	variable with antenna/receiver serial
variableLoopInfo	string	variable with radome (antenna) or version (receiver)
variableLoopTimeStart	string	variable with antenna/receiver start time
variableLoopTimeEnd	string	variable with antenna/receiver end time
variableLoopIndex	string	variable with index of current iteration (starts with zero)
variableLoopCount	string	variable with total number of iterations
condition	condition	check before each loop step

## 5.25 Magnetosphere

This class provides functions of the magnetic field of the Earth.

### 5.25.1 IGRF

International Geomagnetic Reference Field.

Name	Type	Annotation
▶ inputFileMagneticNorthPole	filename	time series of north pole

## 5.26 MatrixGenerator

This class provides a matrix used e.g. by **MatrixCalculate**. If multiple matrices are given the resulting matrix is the sum all and the size is exandeded to fit all matrices. Before the computation of each submatrix the variables **rowsBefore** and **columnsBefore** with current size of the overall matrix are set. As all matrices can be manipulated before, complex matrix operations are possible.

### 5.26.1 File

Matrix from **file**.

Name	Type	Annotation
inputFileMatrix	filename	
factor	double	

### 5.26.2 Normals file

Matrix from a **normal equation file**. The symmetric normal matrix, the right hand side vector, the lPI vector, or the observation count ( $1 \times 1$ ) can be selected.

Name	Type	Annotation
inputFileNormalEquation	filename	
type	choice	
normalMatrix		
rightHandSide		
lPI		
observationCount		
factor	double	

### 5.26.3 Expression

Matrix filled by an expression. For each element of the new matrix the variables **row** and **column** are set and the expression **element** is evaluated.

Example: The **element=if (row==column, 1, 0)** generates an identity matrix.

Name	Type	Annotation
rows	expression	(variables: rowsBefore, columnsBefore)
columns	expression	(variables: rowsBefore, columnsBefore)
element	expression	for each element of matrix (variables: row, column, rows, columns, rowsBefore, columnsBefore)

### 5.26.4 Element manipulation

The elements of a matrix are replaced an expression. For each element of the matrix the variables **data**, **row**, **column** are set and the expression **element** is evaluated and replaces the element. Additionally the standard data variables are available (assigned each row), see [dataVariables \(1.2.3\)](#).

Name	Type	Annotation
matrix	matrixGenerator	
element	expression	for each element of matrix (variables: data, row, column, rows, columns, rowsBefore, columnsBefore)

### 5.26.5 ElementWiseOperation

Given two matrices **A** and **B** this class computes  $c_{ij} = f(a_{ij}, b_{ij})$ , where  $f$  is an expression (for example `data0*data1`). For each element of the matrix the variables `data0`, `data1`, `row`, `column` are set and the expression **element** is evaluated.

Name	Type	Annotation
matrix1	matrixGenerator	
matrix2	matrixGenerator	
expression	expression	for each element of matrix (variables: data0, data1, row, column, rows, columns, rowsBefore, columnsBefore)

### 5.26.6 Append

Append matrix to the right (first row) or bottom (first column).

Name	Type	Annotation
matrix	matrixGenerator	
side	choice	
right		
bottom		
diagonal		

### 5.26.7 Shift

Shift start row and start column of a matrix. In other words: zero lines and columns are inserted at the beginning of the matrix.

Name	Type	Annotation
matrix	matrixGenerator	
startRow	expression	start row (variables: rowsBefore, columnsBefore, rows, columns)
startColumn	expression	start column (variables: rowsBefore, columnsBefore, rows, columns)

### 5.26.8 Slice

Slice of a matrix.

Name	Type	Annotation
matrix	matrixGenerator	
startRow	expression	start row of matrix (variables: rowsBefore, columnsBefore, rows, columns)
startColumn	expression	start column of matrix (variables: rowsBefore, columnsBefore, rows, columns)

rows	expression	0: until end (variables: rowsBefore, columnsBefore, rows, columns)
columns	expression	0: until end (variables: rowsBefore, columnsBefore, rows, columns)

### 5.26.9 Reshape

Matrix reshaped columnwise to new row and columns.

Name	Type	Annotation
matrix	matrixGenerator	
rows	expression	0: auto-determine rows, (variables: rowsBefore, columnsBefore)
columns	expression	0: auto-determine columns (variables: rowsBefore, columnsBefore)

### 5.26.10 Reorder

Reorder rows or columns of a matrix by an index vectors. The index vector can be created with  [ParameterSelection2IndexVector](#).

Name	Type	Annotation
matrix	matrixGenerator	
inputfileIndexVectorRow	filename	index in input matrix or -1 for new parameter.
inputfileIndexVectorColumn	filename	index in input matrix or -1 for new parameter.

### 5.26.11 Sort

Sort matrix by  column in ascending order by default or in  descending order.

Name	Type	Annotation
matrix	matrixGenerator	
column	uint	sort by column, top = highest priority
descending	boolean	

### 5.26.12 Transpose

Transposed of a matrix  $\mathbf{A}^T$ .

Name	Type	Annotation
matrix	matrixGenerator	

### 5.26.13 Multiplication

Multiplication of matrices.

Name	Type	Annotation
matrix1	matrixGenerator	
matrix2	matrixGenerator	
factor	double	

### 5.26.14 Inverse

Inverse of a matrix  $\mathbf{A}^{-1}$ .

Name	Type	Annotation
matrix	matrixGenerator	
pseudoInverse	boolean	compute pseudo inverse instead of regular one

### 5.26.15 Cholesky

Upper triangular matrix of the cholesky decomposition of a symmetric matrix  $\mathbf{A} = \mathbf{W}^T \mathbf{W}$ .

Name	Type	Annotation
matrix	matrixGenerator	

### 5.26.16 RankKUpdate

Symmetric matrix from rank k update:  $\mathbf{A}^T \mathbf{A}$ .

Name	Type	Annotation
matrix	matrixGenerator	
factor	double	

### 5.26.17 EigenValues

Computes the eigenvalues of a square matrix and gives a vector of eigenvalues for symmetric matrices or a matrix with 2 columns with real and imaginary parts in general case.

Name	Type	Annotation
matrix	matrixGenerator	
eigenVectors	boolean	return eigen vectors instead of eigen values

### 5.26.18 Diagonal

Extract the diagonal or subdiagonal ( $n \times 1$  vector) of a matrix. The zero  **diagonal** means the main diagonal, a positive value the superdiagonal, and a negative the subdiagonal.

Name	Type	Annotation
matrix	matrixGenerator	
diagonal	int	zero: main diagonal, positive: superdiagonal, negative: subdiagonal

### 5.26.19 FromDiagonal

Generate a matrix from a diagonal vector.

Name	Type	Annotation
matrix	matrixGenerator	(nx1) or (1xn) diagonal vector
diagonal	int	zero: main diagonal, positive: superdiagonal, negative: subdiagonal

### 5.26.20 Set type

Set type (matrix, matrixSymmetricUpper, matrixSymmetricLower, matrixTriangularUpper, matrixTriangularLower) of a matrix. If the type is not matrix, the matrix must be quadratic. Symmetric matrices are filled symmetric and for triangular matrix the other triangle is set to zero.

Name	Type	Annotation
matrix	<a href="#">matrixGenerator</a>	
type	choice	
matrix		
matrixSymmetricUpper		
matrixSymmetricLower		
matrixTriangularUpper		
matrixTriangularLower		

## 5.27 MiscAccelerations

This class gives the non conservative forces acting on satellites.

### 5.27.1 Relativistic effect

The relativistic effect to the acceleration of an artificial Earth satellite according to IERS2010 conventions.

The macro model and the attitude of the satellite is not needed.

Name	Type	Annotation
▶ beta	double	PPN (parameterized post-Newtonian) parameter
▶ gamma	double	PPN (parameterized post-Newtonian) parameter
▶ J	double	Earth's angular momentum per unit mass [m**2/s]
▶ GM	double	Geocentric gravitational constant
▶ factor	double	the result is multiplied by this factor

### 5.27.2 RadiationPressure

This class computes acceleration acting on a satellite caused by Solar and Earth radiation pressure and thermal radiation.

Solar radiation pressure: The solar constant at 1 AU can be set via ▶ **solarFlux**. The ▶ **factorSolarRadation** can be used to scale the computed acceleration of the direct solar radiation.

Earth radiation pressure: Input are a time series of gridded albedo values (unitless) as ▶ **inputfileAlbedoTimeSeries** and a time series of gridded longwave flux ( $\text{W}/\text{m}^2$ ) as ▶ **inputfileLongwaveFluxTimeSeries**. Both files are optional and if not specified, the respective effect on the acceleration is not computed. The ▶ **factorEarthRadation** can be used to scale the computed acceleration of the earth radiation.

The thermal radiation (TRP) of the satellite itself is either computed as direct re-emission or based on the actual temperature of the satellite surfaces, depending on the settings of the ▶ **satellite** macro model. The second one uses a transient temperature model with a temporal differential equation which disallows parallel computing. The ▶ **factorThermalRadiation** can be used to scale the computed acceleration of the TRP.

The algorithms are described in:

Woeske et. al. (2019), GRACE accelerometer calibration by high precision non-gravitational force modeling, Advances in Space Research, <https://doi.org/10.1016/j.asr.2018.10.025>.

Name	Type	Annotation
▶ solarflux	double	solar flux constant in 1 AU [ $\text{W}/\text{m}^2$ ]
▶ eclipse	eclipse	
▶ inputfileAlbedoTimeSeries	filename	GriddedDataTimeSeries of albedo values (unitless)
▶ inputfileLongwaveFluxTimeSeries	filename	GriddedDataTimeSeries of longwave flux values [ $\text{W}/\text{m}^2$ ]
▶ factorSolarRadation	double	Solar radiation pressure is multiplied by this factor
▶ factorEarthRadation	double	Earth radiation pressure is multiplied by this factor
▶ factorThermalRadiation	double	Thermal (re-)radiation is multiplied by this factor

### 5.27.3 AtmosphericDrag

Atmospheric drag model. Algorithm for the atmospheric drag modelling is based on the free molecule flow theory by Sentman 1961. An analytical expression of this treatise is given in Moe and Moe 2005.

Sentman L. (1961), Free molecule flow theory and its application to the determination of aerodynamic forces, Technical report.

Moe K., Moe M. M. (2005), Gas-surface interactions and satellite drag coefficients, Planetary and Space Science 53(8), 793-801, doi:10.1016/j.pss.2005.03.005.

Optional determination steps: Turn temperature on or off. In the first case, the model mentioned above is applied, which estimates variable drag and lift coefficients - in the latter case a constant drag coefficient can be specified.

Turn wind on/off: It enables the usage of the Horizontal Wind Model 2014 to add additional thermospheric winds in the calculation process.

Name	Type	Annotation
thermosphere	thermosphere	
earthRotation	double	[rad/s]
considerTemperature	boolean	compute drag and lift, otherwise simple drag coefficient is used
considerWind	boolean	
factor	double	the result is multiplied by this factor

### 5.27.4 AtmosphericDragFromDensityFile

Atmospheric drag computed from thermospheric density along the orbit (► **inputfileDensity**, MISCVALUE). The ► **thermosphere** is used to compute temperature and wind. For further details see ► **atmosphericDrag**.

Name	Type	Annotation
inputfileDensity	filename	density along orbit, MISCVALUE (kg/m^3)
thermosphere	thermosphere	used to compute temperature and wind
earthRotation	double	[rad/s]
considerTemperature	boolean	compute drag and lift, otherwise simple drag coefficient is used
considerWind	boolean	
factor	double	the result is multiplied by this factor

### 5.27.5 Antenna thrust

The thrust (acceleration) in the opposite direction the antenna is facing which is generated by satellite antenna broadcasts. The thrust is defined in the satellite macro model.

Name	Type	Annotation
factor	double	the result is multiplied by this factor

### 5.27.6 FromParametrization

Reads a solution vector from file ► **inputfileSolution** which may be computed by a least squares adjustment (e.g. by ► **NormalsSolverVCE**). The coefficients of the vector are interpreted from position ► **indexStart**

(counting from zero) with help of **parametrization**. If the solution file contains solution of several right hand sides you can choose one with number **rightSide** (counting from zero).

The computed result is multiplied with **factor**.

Name	Type	Annotation
parametrization	parametrizationAcceleration	
inputfileSolution	filename	solution vector
indexStart	uint	position in the solution vector
rightSide	uint	if solution contains several right hand sides, select one
factor	double	the result is multiplied by this factor, set -1 to subtract the field

## 5.27.7 Group

Groups a set of **miscAccelerations** and has no further effect itself.

Name	Type	Annotation
miscAccelerations	miscAccelerations	
factor	double	the result is multiplied by this factor

## 5.27.8 SolarRadiationPressure

DEPRECATED. Use radiationPressure instead.

Name	Type	Annotation
solarflux	double	solar flux constant in 1 AU [W/m**2]
eclipse	eclipse	
factor	double	the result is multiplied by this factor, set -1 to subtract the field

## 5.27.9 Albedo

DEPRECATED. Use radiationPressure instead.

Name	Type	Annotation
inputfileReflectivity	filename	
inputfileEmissivity	filename	
solarflux	double	solar flux constant in 1 AU [W/m**2]
factor	double	the result is multiplied by this factor, set -1 to subtract the field

## 5.28 NoiseGenerator

This class implements the generation of different types of noise. It provides a generic interface that can be implemented by different types of generators. The characteristics of the generated noise is determined by the generators. See the appropriate documentation for more information.

### 5.28.1 White

The noise is Gaussian with a standard deviation `sigma`. The noise is computed via a pseudo random sequence with a start value given by `initRandom`. The same value always yields the same sequence. Be careful in `parallel (1.5)` mode as all nodes generates the same pseudo random sequence. If this value is set to zero a real random value is used as starting value.

Name	Type	Annotation
<code>sigma</code>	double	standard deviation
<code>initRandom</code>	uint	start value for pseudo random sequence, 0: real random

### 5.28.2 ExpressionPSD

This generator creates noise defined by a one sided PSD. The `psd` is an expression controlled by the variable 'freq'. To determine the frequency `sampling` must be given.

Name	Type	Annotation
<code>noise</code>	<code>noiseGenerator</code>	Basis noise
<code>psd</code>	expression	one sided PSD (variable: freq [Hz]) [unit^2/Hz]
<code>sampling</code>	double	to determine frequency [seconds]

### 5.28.3 Filter

Generated noise `noise` is filtered by a `filter`.

Name	Type	Annotation
<code>filter</code>	<code>digitalFilter</code>	digital filter
<code>noise</code>	<code>noiseGenerator</code>	Basis noise
<code>warmupEpochCount</code>	uint	number of additional epochs at before start and after end
<code>overSamplingFactor</code>	uint	noise with multiple higher sampling -\$; filter -\$; decimate

### 5.28.4 PowerLaw

This generator creates noise that conforms to a power law relationship, where the power of the noise at a frequency is proportional to  $1/f^\alpha$ , with  $\alpha$  typically between -2 and 2.

Name	Type	Annotation
<code>noise</code>	<code>noiseGenerator</code>	Basis noise
<code>alpha</code>	double	Exponent of the power law relationship $1/f^\alpha$

## 5.29 NormalEquation

This class provides a system of normal equations. This total system is the weighted sum of individual normals.

$$\mathbf{N}_{total} = \sum_{k=1} \frac{1}{\sigma_k^2} \mathbf{N}_k \quad \text{and} \quad \mathbf{n}_{total} = \sum_{k=1} \frac{1}{\sigma_k^2} \mathbf{n}_k. \quad (5.95)$$

The normals do not need to have the same dimension. The dimension of the total combined system is chosen to cover all individual systems. For each normal a **startIndex** is required which indicates the position of the first unknown of the individual normal within the combined parameter vector.

The  $\sigma_k$  of the relative weights are defined by **aprioriSigma** in a first step. If an apriori solution **inputfileApproxSolution** is given or the normals are solved iteratively the weights are determined by means of variance component estimation (VCE), see **NormalsSolverVCE**:

$$\sigma_k^2 = \frac{\mathbf{e}_k^T \mathbf{P} \mathbf{e}_k}{n_k - \frac{1}{\sigma_k^2} \text{trace}(\mathbf{N}_k \mathbf{N}_{total}^{-1})}, \quad (5.96)$$

where  $n_k$  is the number of observations. The square sum of the residuals is calculated by

$$\mathbf{e}_k^T \mathbf{P} \mathbf{e}_k = \mathbf{x}^T \mathbf{N}_k \mathbf{x} - 2\mathbf{n}_k^T \mathbf{x} + \mathbf{l}_k^T \mathbf{P}_k \mathbf{l}_k. \quad (5.97)$$

The system of normal equations can be solved with several right hand sides at once. But only one right hand side, which can be selected with the index **rightHandSide** (counting from zero), can be used to compute the variance factors. The combined normal  $\mathbf{N}_{total}$  and the solution  $\mathbf{x}$  are taken from the previous iteration step. In case of **DesignVCE** the algorithm is a little bit different as described below.

### 5.29.1 Design

This class accumulates normal equations from observation equations. The class **observation** computes the linearized and decorrelated equation system for each arc  $i$ :

$$\mathbf{l}_i = \mathbf{A}_i \mathbf{x} + \mathbf{B}_i \mathbf{y}_i + \mathbf{e}_i. \quad (5.98)$$

The arc depending parameters  $\mathbf{y}_i$  are eliminated and the system of normal equations is accumulated according to

$$\mathbf{N} = \sum_{i=1}^m \mathbf{A}_i^T \mathbf{A}_i \quad \text{and} \quad \mathbf{n} = \sum_{i=1}^m \mathbf{A}_i^T \mathbf{l}_i. \quad (5.99)$$

Name	Type	Annotation
<b>observation</b>	<b>observation</b>	
<b>aprioriSigma</b>	double	
<b>startIndex</b>	uint	add this normals at index of total matrix (counting from 0)
<b>inputfileArcList</b>	filename	to accelerate computation

### 5.29.2 DesignVCE

This class accumulates normal equations from observation equations. The class **observation** computes the linearized and decorrelated equation system for each arc  $i$ :

$$\mathbf{l}_i = \mathbf{A}_i \mathbf{x} + \mathbf{B}_i \mathbf{y}_i + \mathbf{e}_i. \quad (5.100)$$

The arc depending parameters  $\mathbf{y}_i$  are eliminated and the system of normal equations is accumulated according to

$$\mathbf{N} = \sum_{i=1} \frac{1}{\sigma_i^2} \mathbf{A}_i^T \mathbf{A}_i \quad \text{and} \quad \mathbf{n} = \sum_{i=1} \frac{1}{\sigma_i^2} \mathbf{A}_i^T \mathbf{l}_i. \quad (5.101)$$

The variance  $\sigma_i^2$  of each individual arc is determined by

$$\sigma_i^2 = \frac{(\mathbf{l}_i - \mathbf{A}_i \mathbf{x})^T (\mathbf{l}_i - \mathbf{A}_i \mathbf{x})}{n_i - \frac{1}{\sigma_i^2} \text{trace}(\mathbf{A}_i^T \mathbf{A}_i \mathbf{N}_{total}^{-1})}, \quad (5.102)$$

where  $n_i$  is the number of observations. If an apriori solution is not given at the first iteration step a zero vector is assumed.

Name	Type	Annotation
observation	observation	
startIndex	uint	add this normals at index of total matrix (counting from 0)
inputfileArcList	filename	to accelerate computation

### 5.29.3 File

Reads a system of normal equations from file **inputfileNormalEquation** as generated by e.g. **NormalsBuild**.

Name	Type	Annotation
inputfileNormalEquation	filename	
aprioriSigma	double	
startIndex	uint	add this normals at index of total matrix (counting from 0)

### 5.29.4 Regularization

Set up a system of normal equations

$$\mathbf{N} = \mathbf{R} \quad \text{and} \quad \mathbf{n} = \mathbf{R}\mathbf{b}, \quad (5.103)$$

where  $\mathbf{R}$  is a diagonal matrix whose elements are given as a vector by **inputfileDiagonalMatrix** and  $\mathbf{b}$  is the right hand side towards which will be regularized. It can be given by **inputfileBiasVector**. The diagonal matrix can be generated with **NormalsRegularizationBorders**, **NormalsRegularizationSphericalHarmonics**, or **MatrixCalculate**. If  $\mathbf{R}$  is not given a unit matrix is assumed. The right hand side  $\mathbf{b}$  may be generated with **Gravityfield2SphericalHarmonicsVector**. If  $\mathbf{b}$  is not given a zero vector is assumed.

Name	Type	Annotation
inputfileDiagonalMatrix	filename	Vector with the diagonal elements of the weight matrix
inputfileBias	filename	Matrix with right hand sides
aprioriSigma	double	
startIndex	uint	regularization of parameters starts at this index (counting from 0)

### 5.29.5 RegularizationGeneralized

Generalized regularization which is represented by the observation equation

$$\mathbf{x}_0 = \mathbf{Ix} + \mathbf{v}, \mathbf{v} \sim \mathcal{N}(0, \sum_k \sigma_k^2 \mathbf{V}_k). \quad (5.104)$$

There are no requirements for partial covariance matrices  $\mathbf{V}_k$  except for them being symmetric. The accumulated covariance matrix  $\sum_k \sigma_k^2 \mathbf{V}_k$  must be positive definite however. The variance components  $\sigma_k^2$  are estimated during the adjustment process and are assumed to be positive. All **inputfilePartialCovarianceMatrix** must be of same size and must match the dimension of **inputfileBiasMatrix** (if provided, otherwise a zero vector of appropriate dimensions is created).

The parameter **aprioriSigma** determines the initial variance factor for the partial covariance matrices. Either one  $\sigma_0$  can be supplied or one for each  $\mathbf{V}_k$ .

The regularization matrix can be applied to a subset of parameters by adjusting **startIndex**.

Name	Type	Annotation
<b>inputfilePartialCovarianceMatrix</b>	filename	symmetric matrix (sum of all matrices must be positive definite)
<b>inputfileBiasMatrix</b>	filename	bias vector (default: zero vector)
<b>aprioriSigma</b>	double	apriori sigmas for initial iteration (default: 1.0)
<b>startIndex</b>	uint	regularization of parameters starts at this index (counting from 0)

## 5.30 Observation

This class set up the oberservation equations in linearized Gauss-Makoff model

$$\mathbf{l} = \mathbf{Ax} + \mathbf{e} \quad \text{and} \quad \mathcal{C}(\mathbf{e}) = \sigma^2 \mathbf{P}^{-1}. \quad (5.105)$$

The observations are divided into short data blocks which can be computed independently and so easily can be parallelized. Usually these data blocks are short arcs of a satellite's orbit. In most cases the unknown parameter vector contains coefficients of a gravity field parametrization given by **parametrizationGravity**. Additional parameters like instrument calibrations parameters are appended at the end of the vector  $\mathbf{x}$ . It is possible to give several observation vectors in one model.

The observations within each arc are decorrelated in the following way: In a first step a Cholesky decomposition of the covariance matrix is performed

$$\mathbf{P}^{-1} = \mathbf{W}^T \mathbf{W}, \quad (5.106)$$

where  $\mathbf{W}$  is an upper regular triangular matrix. In a second step the transformation

$$\bar{\mathbf{A}} = \mathbf{W}^{-T} \mathbf{A} \quad \text{and} \quad \bar{\mathbf{l}} = \mathbf{W}^{-T} \mathbf{l} \quad (5.107)$$

gives an estimation from decorrelated observations with equal variance

$$\bar{\mathbf{l}} = \bar{\mathbf{A}} \mathbf{x} + \bar{\mathbf{e}} \quad \text{and} \quad \mathcal{C}(\bar{\mathbf{e}}) = \sigma^2 \mathbf{I}. \quad (5.108)$$

Usually the arc depending parameters are eliminated in the next step and not mentioned for the parameter names in the following.

### 5.30.1 PodVariational

The observation equations for precise orbit data (POD) are formulated as variational equations. It is based on **inputfileVariational** calculated with **PreprocessingVariationalEquation**. Necessary integrations are performed by integrating a moving interpolation polynomial of degree **integrationDegree**.

The kinematic positions as pseudo observations are taken from **rightHandSide** and should not be given equally spaced in time. The observation equations are interpolated to these times by a moving polynomial of degree **interpolationDegree**.

The accuracy or the full covariance matrix of the precise orbit data is provided in **covariancePod** and can be estimated with **PreprocessingPod**.

**accelerateComputation:** In the event that the sampling of the kinematic orbit is much higher than the sampling of the variational equations (e.g. 1 second vs. 5 seconds) the accumulation of the observation equations can be accelerated by transforming the observation equations

$$\mathbf{l} = \mathbf{J} \mathbf{Ax} + \mathbf{e}, \quad (5.109)$$

where  $\mathbf{J}$  describes the interpolation of the sampling of the variational design matrix  $\mathbf{A}$  to the sampling of the observations  $\mathbf{l}$  with more rows than columns. The QR decomposition

$$\mathbf{J} = (\mathbf{Q}_1 \quad \mathbf{Q}_2) \begin{pmatrix} \mathbf{R} \\ \mathbf{0} \end{pmatrix}. \quad (5.110)$$

can be used to transform the observation equations

$$\begin{pmatrix} \mathbf{Q}_1^T \mathbf{l} \\ \mathbf{Q}_2^T \mathbf{l} \end{pmatrix} = \begin{pmatrix} \mathbf{Q}_1^T \mathbf{R} \\ \mathbf{0} \end{pmatrix} \mathbf{Ax} + \begin{pmatrix} \mathbf{Q}_1^T \mathbf{e} \\ \mathbf{Q}_2^T \mathbf{e} \end{pmatrix}. \quad (5.111)$$

As the zero lines should not be considered the computational time for the accumulation is reduced. This option is not meaningful for evaluating the residuals such in **PreprocessingPod**.

The following parameters with **parameter names** are set up:

- \*:<parametrizationGravity>:\*:\*,
- <satellite>:<parametrizationAcceleration>:\*:\*,
- <satellite>:arc<no>. <parametrizationAcceleration>:\*:\*,
- <satellite>:arc<no>.position0.x::,
- <satellite>:arc<no>.position0.y::,
- <satellite>:arc<no>.position0.z::..
- <satellite>:arc<no>.velocity0.x::,
- <satellite>:arc<no>.velocity0.y::,
- <satellite>:arc<no>.velocity0.z::..

Name	Type	Annotation
rightHandSide	sequence	input for observation vectors
inputfileOrbit	filename	kinematic positions as observations
inputfileVariational	filename	approximate position and integrated state matrix
ephemerides	ephemerides	
parametrizationGravity	parametrizationGravity	gravity field parametrization
parametrizationAcceleration	parametrizationAcceleration	orbit force parameters
integrationDegree	uint	integration of forces by polynomial approximation of degree n
interpolationDegree	uint	orbit interpolation by polynomial approximation of degree n
accelerateComputation	boolean	acceleration of computation by transforming the observations
covariancePod	covariancePod	covariance matrix of kinematic orbits

### 5.30.2 PodIntegral

The observation equations for precise orbit data (POD) of short arcs are given by

$$\mathbf{r}_\epsilon(\tau) = \mathbf{r}_A(1 - \tau) + \mathbf{r}_B\tau - T^2 \int_0^1 K(\tau, \tau') (\mathbf{f}_0(\tau') + \nabla V(\tau')) d\tau' \quad (5.112)$$

with the integral kernel

$$K(\tau, \tau') = \begin{cases} \tau'(1 - \tau) & \text{for } \tau' \leq \tau \\ \tau(1 - \tau') & \text{for } \tau' > \tau \end{cases}, \quad (5.113)$$

and the normalized time

$$\tau = \frac{t - t_A}{T} \quad \text{with} \quad T = t_B - t_A. \quad (5.114)$$

The kinematic positions  $\mathbf{r}_\epsilon(\tau)$  as pseudo observations are taken from **rightHandSide**. From these positions the influence of the reference forces  $\mathbf{f}_0(\tau)$  is subtracted which are computed with the background models in **rightHandSide**. The integral is solved by the integration of a moving interpolation polynomial of degree **integrationDegree**. The boundary values  $\mathbf{r}_A$  and  $\mathbf{r}_B$  (satellite's state vector) are estimated per arc and are usually directly eliminated if **keepSatelliteStates** is not set.

The unknown gravity field  $\nabla V(\mathbf{r}, t)$  parametrized by **parametrizationGravity** is not evaluated at the observed positions but at the orbit given by **inputfileOrbit**. The same is true for the reference forces. The linearized effect of the gravity field change by the position adjustment is taken into account by **gradientfield**. This may be a low order field up to a spherical harmonics degree of  $n = 2$  or  $n = 3$ .

The **inputfileOrbit**, **inputfileStarCamera**, and **inputfileAccelerometer** must be synchronous and must be given with a constant sampling and without any gaps in each short arc (see **InstrumentSynchronize**). The kinematic positions  $\mathbf{r}_\epsilon(\tau)$  should not be given equally spaced in time but must be divided into the same arcs as the other instrument data. The observation equations are interpolated to this time by a polynomial interpolation with degree **interpolationDegree**.

The accuracy or the full covariance matrix of the precise orbit data is provided in **covariancePod** and can be estimated with **PreprocessingPod**.

For **accelerateComputation** see **observation:podVariational**.

The following parameters with **parameter** names are set up:

- \*:<parametrizationGravity>:\*:\*,
- <satellite>:<parametrizationAcceleration>:\*:\*,

and for each arc if **keepSatelliteStates** is set

- <satellite>:arc<no>.position.start.x::,
- <satellite>:arc<no>.position.start.y::,
- <satellite>:arc<no>.position.start.z::,
- <satellite>:arc<no>.position.end.x::,
- <satellite>:arc<no>.position.end.y::,
- <satellite>:arc<no>.position.end.z::.

Name	Type	Annotation
<b>inputfileSatelliteModel</b>	filename	satellite macro model
<b>rightHandSide</b>	<b>podRightSide</b>	input for the reduced observation vector
<b>inputfileOrbit</b>	filename	used to evaluate the observation equations, not used as observations
<b>inputfileStarCamera</b>	filename	
<b>earthRotation</b>	<b>earthRotation</b>	
<b>ephemerides</b>	<b>ephemerides</b>	
<b>gradientfield</b>	<b>gravityfield</b>	low order field to estimate the change of the gravity by position adjustment
<b>parametrizationGravity</b>	<b>parametrizationGravity</b>	gravity field parametrization
<b>parametrizationAcceleration</b>	<b>parametrizationAcceleration</b>	orbit force parameters
<b>keepSatelliteStates</b>	boolean	set boundary values of each arc global integration of forces by polynomial approximation of degree n
<b>integrationDegree</b>	uint	orbit interpolation by polynomial approximation of degree n
<b>interpolationDegree</b>	uint	acceleration of computation by transforming the observations
<b>accelerateComputation</b>	boolean	covariance matrix of kinematic orbits
<b>covariancePod</b>	<b>covariancePod</b>	

### 5.30.3 PodAcceleration

The observation equations for precise orbit data (POD) are given by

$$\ddot{\mathbf{r}}(t) - \mathbf{g}_0(t) = \nabla V(\mathbf{r}, t), \quad (5.115)$$

where the accelerations of the satellite  $\ddot{\mathbf{r}}(t)$  are derived from the kinematic positions in **rightHandSide**. The orbit differentiation is performed by a moving polynomial interpolation or approximation with degree **interpolationDegree** and number of used epochs **numberOfEpochs**. The reference forces  $\mathbf{g}_0(t)$  are computed with the background models in **rightHandSide**.

All instrument data **infileOrbit**, **infileStarCamera**, and **infileAccelerometer** must be synchronous and be given with a constant sampling without any gaps in each short arc (see **InstrumentSynchronize**).

The unknown gravity field  $\nabla V(\mathbf{r}, t)$  parametrized by **parametrizationGravity** is not evaluated at the observed positions but at the orbit given by **infileOrbit**. The same is true for the reference forces. This orbit may be a more accurate dynamical orbit but in most cases the kinematic orbit provides good results.

The accuracy or the full covariance matrix of the precise orbit data is provided in **covariancePod** and can be estimated with **PreprocessingPod**.

The following parameters with **parameter names** are set up:

- \*:<parametrizationGravity>:\*:\*,
- <satellite>:<parametrizationAcceleration>:\*:\*.

Name	Type	Annotation
<b>infileSatelliteModel</b>	filename	satellite macro model
<b>rightHandSide</b>	<b>podRightSide</b>	input for the reduced observation vector
<b>infileOrbit</b>	filename	used to evaluate the observation equations, not used as observations
<b>infileStarCamera</b>	filename	
<b>earthRotation</b>	<b>earthRotation</b>	
<b>ephemerides</b>	<b>ephemerides</b>	
<b>parametrizationGravity</b>	<b>parametrizationGravity</b>	gravity field parametrization
<b>parametrizationAcceleration</b>	<b>parametrizationAcceleration</b>	orbit force parameters
<b>interpolationDegree</b>	uint	orbit differentiation by polynomial approximation of degree n
<b>numberOfEpochs</b>	uint	number of used Epochs for polynomial computation
<b>covariancePod</b>	<b>covariancePod</b>	covariance matrix of kinematic orbits

### 5.30.4 PodEnergy

The observation equations for precise orbit data (POD) are given by

$$\frac{1}{2} \dot{\mathbf{r}}^2 - \dot{\mathbf{r}} \cdot (\boldsymbol{\Omega} \times \mathbf{r}) + \int_{t_0}^t (\dot{\boldsymbol{\Omega}} \times \mathbf{r}) \cdot \dot{\mathbf{r}} dt - \int_{t_0}^t \mathbf{g}_0 \cdot \dot{\mathbf{r}}' dt = V + E. \quad (5.116)$$

where the velocities of the satellite  $\dot{\mathbf{r}}(t)$  are derived from the kinematic positions in **rightHandSide** and the Earth's rotation vector  $\boldsymbol{\Omega}(t)$  is modeled within **earthRotation**. The orbit differentiation is performed

by a polynomial interpolation with degree **► interpolationDegree**. The integrals are solved a polynomial interpolation with degree **► integrationDegree**. The reference forces  $\mathbf{g}_0(t)$  are computed with the background models in **► rightHandSide**.

All instrument data **► inputfileOrbit**, **► inputfileStarCamera**, and **► inputfileAccelerometer** must be synchronous and be given with a constant sampling without any gaps in each short arc (see **► InstrumentSynchronize**).

The unknown gravity potential  $V(\mathbf{r})$  parametrized by **► parametrizationGravity** is not evaluated at the observed positions but at the orbit given by **► inputfileOrbit**. The same is true for the reference forces. This orbit may be a more accurate dynamical orbit but in most cases the kinematic orbit provides good results.

An unknown energy bias  $E$  per arc is parametrized by **► parametrizationBias** and should be a constant in theory but temporal changes might help to absorb other unmodelled effects.

The accuracy or the full covariance matrix of the precise orbit data is provided in **► covariancePod** and can be estimated with **► PreprocessingPod**.

The following parameters with **► parameter names** are set up: \*:<parametrizationGravity>:\*:\*

Name	Type	Annotation
<b>► inputfileSatelliteModel</b>	filename	satellite macro model
<b>► rightHandSide</b>	<b>podRightSide</b>	input for the reduced observation vector
<b>► inputfileOrbit</b>	filename	used to evaluate the observation equations, not used as observations
<b>► inputfileStarCamera</b>	filename	
<b>► earthRotation</b>	<b>earthRotation</b>	
<b>► ephemerides</b>	<b>ephemerides</b>	
<b>► parametrizationGravity</b>	<b>parametrizationGravity</b>	gravity field parametrization (potential)
<b>► parametrizationBias</b>	<b>parametrizationTemporal</b>	unknown total energy per arc
<b>► interpolationDegree</b>	uint	orbit differentiation by polynomial approximation of degree n
<b>► integrationDegree</b>	uint	integration of forces by polynomial approximation of degree n
<b>► covariancePod</b>	<b>covariancePod</b>	covariance matrix of kinematic orbits

### 5.30.5 SstVariational

Like **► observation:podVariational** (see there for details) but with two satellites and additional satellite-to-satellite (SST) observations.

If multiple **► inputfileSatelliteTracking** are given all data are add together. So corrections in extra files like the light time correction can easily be added. Empirical parameters for the SST observations can be setup with **► parametrizationSst**. The accuracy or the full covariance matrix of SST is provided in **► covarianceSst**.

The following parameters with **► parameter names** are set up:

- \*:<parametrizationGravity>:\*:\*,
- <satellite1>:<parametrizationAcceleration>:\*:\*,
- <satellite1>:arc<no>. <parametrizationAcceleration>:\*:\*,
- <satellite1>:arc<no>.position0.x::,

- <satellite1>:arc<no>.position0.y::,
- <satellite1>:arc<no>.position0.z::,
- <satellite1>:arc<no>.velocity0.x::,
- <satellite1>:arc<no>.velocity0.y::,
- <satellite1>:arc<no>.velocity0.z::,
- <satellite2>:<parametrizationAcceleration>::\*:,
- <satellite2>:arc<no>.<parametrizationAcceleration>::\*:,
- <satellite2>:arc<no>.position0.x::,
- <satellite2>:arc<no>.position0.y::,
- <satellite2>:arc<no>.position0.z::,
- <satellite2>:arc<no>.velocity0.x::,
- <satellite2>:arc<no>.velocity0.y::,
- <satellite2>:arc<no>.velocity0.z::,
- <satellite1>.<satellite2>:<parametrizationSatelliteTracking>::\*:.

Name	Type	Annotation
rightHandSide	sequence	input for observation vectors
inputfileSatelliteTracking	filename	ranging observations and corrections
inputfileOrbit1	filename	kinematic positions of satellite A as observations
inputfileOrbit2	filename	kinematic positions of satellite B as observations
sstType	choice	
range		
rangeRate		
none		
inputfileVariational1	filename	approximate position and integrated state matrix
inputfileVariational2	filename	approximate position and integrated state matrix
ephemerides	ephemerides	
parametrizationGravity	parametrizationGravity	gravity field parametrization
parametrizationAcceleration1	parametrizationAcceleration	orbit1 force parameters
parametrizationAcceleration2	parametrizationAcceleration	orbit2 force parameters
parametrizationSst	parametrizationSatelliteTracking	satellite tracking parameter
integrationDegree	uint	integration of forces by polynomial approximation of degree n
interpolationDegree	uint	orbit interpolation by polynomial approximation of degree n
covarianceSst	covarianceSst	covariance matrix of satellite to satellite tracking observations
covariancePod1	covariancePod	covariance matrix of kinematic orbits (satellite 1)
covariancePod2	covariancePod	covariance matrix of kinematic orbits (satellite 2)

### 5.30.6 SstIntegral

Like **observation:podIntegral** (see there for details) but with two satellites and additional satellite-to-satellite (SST) observations.

If multiple **inputfileSatelliteTracking** are given all data are add together. So corrections in extra files like the light time correction can easily be added. Empirical parameters for the SST observations can be setup with **parametrizationSst**. The accuracy or the full covariance matrix of SST is provided in **covarianceSst**.

The following parameters with **parameter names** are set up:

- \*:<parametrizationGravity>:\*:\*,
- <satellite1>:<parametrizationAcceleration>:\*:\*,
- <satellite2>:<parametrizationAcceleration>:\*:\*,
- <satellite1>.<satellite2>:<parametrizationSatelliteTracking>:\*:\*,

and for each arc if **keepSatelliteStates** is set

- <satellite1>:arc<no>.position.start.x::,
- <satellite1>:arc<no>.position.start.y::,
- <satellite1>:arc<no>.position.start.z::,
- <satellite1>:arc<no>.position.end.x::,
- <satellite1>:arc<no>.position.end.y::,
- <satellite1>:arc<no>.position.end.z::,
- <satellite2>:arc<no>.position.start.x::,
- <satellite2>:arc<no>.position.start.y::,
- <satellite2>:arc<no>.position.start.z::,
- <satellite2>:arc<no>.position.end.x::,
- <satellite2>:arc<no>.position.end.y::,
- <satellite2>:arc<no>.position.end.z::,

Name	Type	Annotation
<b>inputfileSatelliteModel1</b>	filename	satellite macro model
<b>inputfileSatelliteModel2</b>	filename	satellite macro model
<b>rightHandSide</b>	<b>sstRightSide</b>	input for the reduced observation vector
<b>sstType</b>	choice	
<b>range</b>		
<b>rangeRate</b>		
<b>rangeAcceleration</b>		
<b>none</b>		
<b>inputfileOrbit1</b>	filename	used to evaluate the observation equations, not used as observations

<code>inputfileOrbit2</code>	filename	used to evaluate the observation equations, not used as observations
<code>inputfileStarCamera1</code>	filename	
<code>inputfileStarCamera2</code>	filename	
<code>earthRotation</code>	<code>earthRotation</code>	
<code>ephemerides</code>	<code>ephemerides</code>	
<code>gradientfield</code>	<code>gravityfield</code>	low order field to estimate the change of the gravity by position adjustement
<code>parametrizationGravity</code>	<code>parametrizationGravity</code>	gravity field parametrization
<code>parametrizationAcceleration1</code>	<code>parametrizationAcceleration</code>	orbit1 force parameters
<code>parametrizationAcceleration2</code>	<code>parametrizationAcceleration</code>	orbit2 force parameters
<code>parametrizationSst</code>	<code>parametrizationSatelliteTracking</code>	satellite tracking parameter
<code>keepSatelliteStates</code>	boolean	set boundary values of each arc global
<code>integrationDegree</code>	uint	integration of forces by polynomial approximation of degree n
<code>interpolationDegree</code>	uint	orbit interpolation by polynomial approximation of degree n
<code>covarianceSst</code>	<code>covarianceSst</code>	covariance matrix of satellite to satellite tracking observations
<code>covariancePod1</code>	<code>covariancePod</code>	covariance matrix of kinematic orbits (satellite 1)
<code>covariancePod2</code>	<code>covariancePod</code>	covariance matrix of kinematic orbits (satellite 2)

### 5.30.7 DualSstVariational

Like `observation:sstVariational` (see there for details) but with two simultaneous satellite-to-satellite (SST) observations.

This class reads two SST observation files (`inputfileSatelliteTracking1` and `inputfileSatelliteTracking2`). Empirical parameters for the SST observations can be setup independently for both SST observation types with `parametrizationSst1` and `parametrizationSst2`.

Both SST observation types are reduced by the same background models and the same impact of accelerometer measurements. The covariance matrix of the reduced observations should not consider the instrument noise only (`covarianceSst1/2`) but must take the cross correlations `covarianceAcc` into account. The covariance matrix of the reduced observations is given by

$$\Sigma \left( \begin{bmatrix} \Delta l_{SST1} \\ \Delta l_{SST2} \end{bmatrix} \right) = \begin{bmatrix} \Sigma_{SST1} + \Sigma_{ACC} & \Sigma_{ACC} \\ \Sigma_{ACC} & \Sigma_{SST2} + \Sigma_{ACC} \end{bmatrix}. \quad (5.117)$$

The following parameters with `parameter names` are set up:

- \*:`parametrizationGravity`:\*:\*>,
- `<satellite1>:<parametrizationAcceleration>:*:*`,
- `<satellite1>:arc<no>. <parametrizationAcceleration>:*:*`,
- `<satellite1>:arc<no>.position0.x::`,
- `<satellite1>:arc<no>.position0.y::`,

- <satellite1>:arc<no>.position0.z::.
- <satellite1>:arc<no>.velocity0.x::,
- <satellite1>:arc<no>.velocity0.y::,
- <satellite1>:arc<no>.velocity0.z::.
- <satellite2>:<parametrizationAcceleration>::\*:,
- <satellite2>:arc<no>.<parametrizationAcceleration>::\*:,
- <satellite2>:arc<no>.position0.x::,
- <satellite2>:arc<no>.position0.y::,
- <satellite2>:arc<no>.position0.z::.
- <satellite2>:arc<no>.velocity0.x::,
- <satellite2>:arc<no>.velocity0.y::,
- <satellite2>:arc<no>.velocity0.z::.
- <satellite1>.<satellite2>:<parametrizationSatelliteTracking1>::\*:.\*
- <satellite1>.<satellite2>:<parametrizationSatelliteTracking2>::\*:.\*

Name	Type	Annotation
rightHandSide	sequence	input for observation vectors
inputfileSatelliteTracking1	filename	ranging observations and corrections
inputfileSatelliteTracking2	filename	ranging observations and corrections
inputfileOrbit1	filename	kinematic positions of satellite A as observations
inputfileOrbit2	filename	kinematic positions of satellite B as observations
sstType	choice	
range		
rangeRate		
none		
inputfileVariational1	filename	approximate position and integrated state matrix
inputfileVariational2	filename	approximate position and integrated state matrix
ephemerides	ephemerides	
parametrizationGravity	parametrizationGravity	gravity field parametrization
parametrizationAcceleration1	parametrizationAcceleration	orbit1 force parameters
parametrizationAcceleration2	parametrizationAcceleration	orbit2 force parameters
parametrizationSst1	parametrizationSatelliteTracking	satellite tracking parameter for first ranging observations
parametrizationSst2	parametrizationSatelliteTracking	satellite tracking parameter for second ranging observations
integrationDegree	uint	integration of forces by polynomial approximation of degree n
interpolationDegree	uint	orbit interpolation by polynomial approximation of degree n

covarianceSst1	covarianceSst	covariance matrix of first satellite to satellite tracking observations
covarianceSst2	covarianceSst	covariance matrix of second satellite to satellite tracking observations
covarianceAcc	covarianceSst	common covariance matrix of reduced satellite to satellite tracking observations
covariancePod1	covariancePod	covariance matrix of kinematic orbits (satellite 1)
covariancePod2	covariancePod	covariance matrix of kinematic orbits (satellite 2)

### 5.30.8 Gradiometer

Observation equations for satellite gravity gradiometry (SGG)

$$\nabla \nabla V(\mathbf{r}) = \begin{pmatrix} \frac{\partial^2 V}{\partial r^2} & \frac{\partial^2 V}{\partial x \partial y} & \frac{\partial^2 V}{\partial x \partial z} \\ \frac{\partial^2 V}{\partial y \partial x} & \frac{\partial^2 V}{\partial y^2} & \frac{\partial^2 V}{\partial y \partial z} \\ \frac{\partial^2 V}{\partial z \partial x} & \frac{\partial^2 V}{\partial z \partial y} & \frac{\partial^2 V}{\partial z^2} \end{pmatrix}. \quad (5.118)$$

From the **inputfileGradiometer** observations precomputed **inputfileReferenceGradiometer** together with other background models are reduced, all given in **rightHandSide**.

All instrument data **inputfileGradiometer**, **inputfileOrbit**, and **inputfileStarCamera** must be synchronous and be divided into each short arcs (see **InstrumentSynchronize**).

Additional to the **parametrizationGravity** an (temporal changing) bias for each gradiometer component and arc can be estimated with **parametrizationBias**.

The accuracy or the full covariance matrix of the gradiometer is provided in **covarianceSgg** and can be estimated with **PreprocessingGradiometer**.

The following parameters with **parameter names** are set up: \*:**<parametrizationGravity>**:\*:\*

Name	Type	Annotation
rightHandSide	sggRightSide	input for the observation vector
inputfileOrbit	filename	
inputfileStarCamera	filename	
earthRotation	earthRotation	
ephemerides	ephemerides	
parametrizationGravity	parametrizationGravity	
parametrizationBias	parametrizationTemporal	per arc
useXX	boolean	
useYY	boolean	
useZZ	boolean	
useXY	boolean	
useXZ	boolean	
useYZ	boolean	
covarianceSgg	sequence	
sigma	double	general variance factor
inputfileSigmasPerArc	filename	different accuracies for each arc (multiplied with sigma)
inputfileCovarianceFunction	filename	covariance function in time

### 5.30.9 Terrestrial

The gravity field is estimated from point wise measurements. The gravity field parametrization is given by **parametrizationGravity**. There is no need to have the data regular distributed or given on a sphere or ellipsoid. The type of the gridded data (e.g gravity anomalies or geoid heights) must be set with **kernel**. A **referencefield** can be reduced beforehand.

The observations at given positions are calculated from **infileGriddedData**. The input columns are enumerated by **data0**, **data1**, ..., see [dataVariables \(1.2.3\)](#).

The observations can be divided into small blocks for parallelization. With **blockingSize** set the maximum count of observations in each block.

The following parameters with **parameter names** are set up: \*:<parametrizationGravity>:\*:\*

Name	Type	Annotation
<b>rightHandSide</b>	sequence	input for observation vectors
<b>infileGriddedData</b>	filename	
<b>observation</b>	expression	[SI units]
<b>sigma</b>	expression	accuracy, 1/sigma used as weighting
<b>referencefield</b>	<b>gravityfield</b>	
<b>kernel</b>	<b>kernel</b>	
<b>parametrizationGravity</b>	<b>parametrizationGravity</b>	
<b>time</b>	time	for reference field and parametrization
<b>blockingSize</b>	uint	segementation of the obervations if designmatrix can't be build at once

### 5.30.10 Deflections

The gravity field parametrized by **parametrizationGravity** is estimated from deflections of the vertical measurements. A **referencefield** can be reduced beforehand.

The observations  $\xi$  in north direction and  $\eta$  in east direction at given positions are calculated from **infileGriddedData**. The input columns are enumerated by **data0**, **data1**, ..., see [dataVariables \(1.2.3\)](#).

The ellipsoid parameters **R** and **inverseFlattening** are used to define the local normal direction.

The observations can be divided into small blocks for parallelization. With **blockingSize** set the maximum count of observations in each block.

The following parameters with **parameter names** are set up: \*:<parametrizationGravity>:\*:\*

Name	Type	Annotation
<b>rightHandSide</b>	sequence	input for observation vectors
<b>infileGriddedData</b>	filename	
<b>observationXi</b>	expression	North-South Deflections of the Vertical [rad]
<b>observationEta</b>	expression	East-West Deflections of the Vertical [rad]
<b>sigmaXi</b>	expression	accuracy, 1/sigma used as weighting
<b>sigmaEta</b>	expression	accuracy, 1/sigma used as weighting
<b>referencefield</b>	<b>gravityfield</b>	
<b>parametrizationGravity</b>	<b>parametrizationGravity</b>	
<b>time</b>	time	for reference field and parametrization
<b>R</b>	double	reference radius for ellipsoid
<b>inverseFlattening</b>	double	reference flattening for ellipsoid, 0: sphere
<b>blockingSize</b>	uint	segementation of the obervations if designmatrix can't be build at once

### 5.30.11 StationLoading

Observation equations for displacements of a list of stations due to the effect of time variable loading masses. The displacement  $\mathbf{u}$  of a station is calculated according to

$$\mathbf{u}(\mathbf{r}) = \frac{1}{\gamma} \sum_{n=0}^{\infty} \left[ \frac{h_n}{1+k_n} V_n(\mathbf{r}) \mathbf{e}_{up} + R \frac{l_n}{1+k_n} \left( \frac{\partial V_n(\mathbf{r})}{\partial \mathbf{e}_{north}} \mathbf{e}_{north} + \frac{\partial V_n(\mathbf{r})}{\partial \mathbf{e}_{east}} \mathbf{e}_{east} \right) \right], \quad (5.119)$$

where  $\gamma$  is the normal gravity, the load Love and Shida numbers  $h_n, l_n$  are given by **inputfileDeformationLoadLoveNumber** and the load Love numbers  $k_n$  are given by **inputfilePotentialLoadLoveNumber**. The  $V_n$  are the spherical harmonics expansion of degree  $n$  of the full time variable gravitational potential (potential of the loading mass + deformation potential) parametrized by **parametrizationGravity**. Additional parameters can be setup to estimate the realization of the reference frame of the station coordinates (**estimateTranslation**, **estimateRotation**, and **estimateScale**).

The observations at stations coordinates are calculated from **inputfileGriddedData**. The input columns are enumerated by `data0`, `data1`, ..., see [dataVariables \(1.2.3\)](#).

The ellipsoid parameters **R** and **inverseFlattening** are used to define the local frame (north, east, up).

The following parameters with **parameter names** are set up:

- \*:<parametrizationGravity>:\*:\*,
- \*:translation.x:\*:\*,
- \*:translation.y:\*:\*,
- \*:translation.z:\*:\*,
- \*:scale:\*:\*,
- \*:rotation.x:\*:\*,
- \*:rotation.y:\*:\*,
- \*:rotation.z:\*:\*.

See also **Gravityfield2DisplacementTimeSeries**.

Reference: Rietbroek (2014): Retrieval of Sea Level and Surface Loading Variations from Geodetic Observations and Model Simulations: an Integrated Approach, Bonn, 2014. - Dissertation, <https://nbn-resolving.org/urn:nbn:de:hbz:5n-35460>

Name	Type	Annotation
rightHandSide	sequence	input for observation vectors
inputfileGriddedData	filename	station positions with displacement data
observationNorth	expression	displacement [m]
observationEast	expression	displacement [m]
observationUp	expression	displacement [m]
sigmaNorth	expression	accuracy, 1/sigma used as weighting
sigmaEast	expression	accuracy, 1/sigma used as weighting
sigmaUp	expression	accuracy, 1/sigma used as weighting
inGlobalFrame	boolean	obs/sigmas given in global x,y,z frame instead of north,east,up

---

 referencefield	<b>gravityfield</b>	
 time	time	for reference field and parametrization
 parametrizationGravity	<b>parametrizationGravity</b>	of loading (+defo) potential
 estimateTranslation	boolean	coordinate center
 estimateScale	boolean	scale factor of position
 estimateRotation	boolean	rotation
 inputfileDeformationLoadLoveNumber	filename	
 inputfilePotentialLoadLoveNumber	filename	if full potential is given and not only loading potential
 R	double	reference radius for ellipsoid
 inverseFlattening	double	reference flattening for ellipsoid, 0: sphere

---

## 5.31 OrbitPropagator

Implements the propagation of a satellite orbit under the influence of **forces** as used in **SimulateOrbit** (dynamic orbits from numerical orbit integration).

### 5.31.1 Euler

This class implements Euler's method to propagate a satellite orbit under the influence of **Forces**. Satellite is assumed to be oriented along-track.

### 5.31.2 RungeKutta4

This class implements the classical Runge-Kutta 4 method of orbit propagation for satellite orbit under the influence of **Forces**. No step-width control or other advanced features are implemented. Satellite is assumed to be oriented along-track. See: Montenbruck, Oliver, and Eberhard Gill. 2000. Satellite Orbits

### 5.31.3 AdamsBashforthMoulton

This class implements the Adams-Moulton class of predictor-corrector orbit propagators for a satellite orbit under the influence of **Forces** using an implicit Adams-Bashforth corrector. The coefficients for the propagator are derived using the equations given in section 4.2.3 of [1]. Satellite is assumed to be oriented along-track. [1] Montenbruck, Oliver, and Eberhard Gill. 2000. Satellite Orbits

Name	Type	Annotation
order	uint	Order of the Adams-Bashforth type propagator.
applyMoultonCorrector	boolean	Corrector step after Adams-Bashforth prediction.
warmup	orbitPropagator	

### 5.31.4 StoermerCowell

This class implements the Stoermer-Cowell class of predictor-corrector orbit propagators for a satellite orbit under the influence of **Forces**. The coefficients for the Stoermer predictor and Cowell corrector are derived using the equations given in section 4.2.6 of [1]. Stoermer-Cowell is a double integration algorithm, yielding positions directly from accelerations. It does not produce velocities. The velocities are derived using Adams-type propagators as suggested in [2]. Satellite is assumed to be oriented along-track. [1] Montenbruck, Oliver, and Eberhard Gill. 2000. Satellite Orbits [2] Berry, Matthew M., and Liam M. Healy. 2004. "Implementation of Gauss-Jackson Integration for Orbit Propagation."

Name	Type	Annotation
order	uint	Order of the Stoermer-Cowell type propagator.
warmup	orbitPropagator	

### 5.31.5 GaussJackson

This class implements the Gauss-Jackson multi-step predictor-corrector method to propagate a satellite orbit under the influence of **Forces**. Satellite is assumed to be oriented along-track. Implementation is based on [1]. [1] Berry, Matthew M., and Liam M. Healy. 2004. "Implementation of Gauss-Jackson Integration for Orbit Propagation."

Name	Type	Annotation
order	uint	of Gauss-Jackson method.
warmup	orbitPropagator	
correctorIterations	uint	Maximum number of iterations to run the corrector step for.
epsilon	double	Convergence criteria for position, velocity, and acceleration tests.

### 5.31.6 Polynomial

This class implements an integration Polynomial method to propagate a satellite orbit under the influence of **Forces**. Satellite is assumed to be oriented along-track. Implementation is based on code by Torsten Mayer-Gürr.

Name	Type	Annotation
degree	uint	polynomial degree to integrate accelerations
shift	int	shift polynomial in future (predicted accelerations)
epsilon	double	[m] max. position change to recompute forces
warmup	orbitPropagator	to compute epochs before start epoch
corrector	boolean	apply corrector iteration if position change is larger than epsilon

### 5.31.7 File

Reads an orbit from file. If the needed epochs are not given an exception is thrown.

Name	Type	Annotation
inputfileOrbit	filename	epoch at timeStart is not used
margin	double	[seconds] to find identical times
recomputeForces	boolean	

## 5.32 ParameterNames

Generate a list of parameter names. All parameters are appended.

### 5.32.1 Name

The parameter is given by explicitly by four parts:

1. object: Object this parameter refers to, e.g. `graceA`, `G023`, `earth`, ...
2. type: Type of this parameter, e.g. `accBias`, `position.x`, ...
3. temporal: Temporal representation of this parameter, e.g. `trend`, `polynomial.degree1`, ...
4. interval: Interval/epoch this parameter represents, e.g. `2017-01-01_00-00-00_2017-01-02_00-00-00`, `2018-01-01_00-00-00`.

Name	Type	Annotation
object	string	object this parameter refers to, e.g. <code>graceA</code> , <code>G023</code> , <code>earth</code>
type	string	type of this parameter, e.g. <code>accBias</code> , <code>position.x</code>
temporal	string	temporal representation of this parameter, e.g. <code>trend</code> , <code>polynomial.degree1</code>
interval	string	interval/epoch this parameter refers to, e.g. <code>2017-01-01_00-00-00_2017-01-02_00-00-00</code> , <code>2008-01-01_00-00-00</code>

### 5.32.2 File

Read parameter names from [file](#).

Name	Type	Annotation
inputfileParameterNames	filename	file with parameter names

### 5.32.3 Gravity

Parameter names of gravity [parametrization](#). An additional [object](#) name can be included in the parameter names.

Name	Type	Annotation
object	string	object these parameters refers to, e.g. <code>earth</code>
parametrization	<a href="#">parametrizationGravity</a>	

### 5.32.4 Acceleration

Parameter names of satellite acceleration [parametrization](#). Arc related parameters are appended if an [inputfileInstrument](#) is provided which defines the arc structure. An additional [object](#) name can be included in the parameter names.

Name	Type	Annotation

---

object	string	object these parameters refers to, e.g. graceA, G023
parameterization	parametrizationAcceleration	
inputfileInstrument	filename	defines the arc structure for arc related parameters

---

### 5.32.5 SatelliteTracking

Parameter names of satellite tracking **parametrization**. An additional **object** name can be included in the parameter names.

---

Name	Type	Annotation
object	string	object these parameters refers to, e.g. grace1.grace2
parameterization	parametrizationSatelliteTracking	

---

### 5.32.6 Temporal

Parameter names from temporal parametrization. It is possible to setup the temporal parameters for each **parameterNameBase**.

---

Name	Type	Annotation
parameterNameBase	parameterNames	
parametrizationTemporal	parametrizationTemporal	

---

### 5.32.7 GnssAntenna

Parameter names of GNSS antenna center variation **parametrization**. An additional **object** name (antenna name) can be included in the parameter names. It is possible to setup the parameters for each **gnssType**.

---

Name	Type	Annotation
object	string	antenna name
parametrization	parametrizationGnssAntenna	
gnssType	gnssType	e.g. C1CG**

---

### 5.32.8 Observation

Parameter names used in **observation equations**.

---

Name	Type	Annotation
observation	observation	

---

### 5.32.9 Rename

Replaces parts of **parameterNames**. The star "\*" left this part untouched.

Name	Type	Annotation
➡ parameterName	parameterNames	
➡ object	string	*: left this part untouched, object
➡ type	string	*: left this part untouched, type
➡ temporal	string	*: left this part untouched, temporal representation
➡ interval	string	*: left this part untouched, interval/epoch

### 5.32.10 Selection

Select a subset of ➡ parameterNames using ➡ parameterSelection.

Name	Type	Annotation
➡ parameterName	parameterNames	
➡ parameterSelection	parameterSelector	parameter order/selection

### 5.32.11 WithoutDuplicates

Removes all duplicate names (keep first) from ➡ parameterName.

Name	Type	Annotation
➡ parameterName	parameterNames	

## 5.33 ParameterSelector

This class provides an index vector from selected parameters, which can be used e.g. to reorder a normal equation matrix. The size of the index vector determines the size of the new matrix. Entries are the indices of the selected parameters in the provided parameter list or NULLINDEX for zero/new parameters.

### 5.33.1 Wildcard

Parameter index vector from name. Name matching supports wildcards \* for any number of characters and ? for exactly one character. Does not add zero/empty parameters if there are no matches.

Name	Type	Annotation
object	string	object this parameter refers to, e.g. graceA, G023, earth (wildcards: * and ?)
type	string	type of this parameter, e.g. accBias, position.x (wildcards: * and ?)
temporal	string	temporal representation of this parameter, e.g. trend, polynomial.degree1 (wildcards: * and ?)
interval	string	interval/epoch this parameter refers to, e.g. 2017-01-01_00-00-00_2017-01-02_00-00-00, 2008-01-01_00-00-00 (wildcards: * and ?)

### 5.33.2 Names

Parameter index vector from list of parameter names.

Name	Type	Annotation
parameterName	parameterNames	

### 5.33.3 Range

Parameter index vector from range.

Name	Type	Annotation
start	expression	start at this index (variables: length)
count	expression	count of parameters, default: all parameters to the end (variables: length)

### 5.33.4 Matrix

Parameter index vector from matrix.

Name	Type	Annotation
inputfileMatrix	filename	index in old parameter list or -1 for new parameter
column	expression	use this column (counting from 0, variables: columns)
startRow	expression	start at this row (counting from 0, variables: rows)
countRows	expression	use these many rows (default: use all, variables: rows)

### 5.33.5 Zeros

Expand parameter index vector by adding zero parameters.

Name	Type	Annotation
count	expression	count of zero parameters (variables: length)

### 5.33.6 Complement

Parameter index vector from a complement of other parameter selector(s).

Name	Type	Annotation
parameterSelection	parameterSelector	parameter order/selection

## 5.34 ParametrizationAcceleration

This class defines parameters of satellite accelerations. It will be used to set up the design matrix in a least squares adjustment. If multiple parametrizations are given the coefficients in the parameter vector are sequentially appended.

### 5.34.1 PerRevolution

Oscillation once, twice, ... per revolution in Satellite Reference Frame (SRF) with the argument of latitude as input angle. If the attitude of the satellite is not provided the Celestial Reference Frame (CRF) is used instead. Parameters are estimated in [ $nm/s^2 = 10^{-9} m/s^2$ ].

The [parameter names](#) are

- \*:perRevolution.cos(<order>\*u).x::<interval>,
- \*:perRevolution.cos(<order>\*u).y::<interval>,
- \*:perRevolution.cos(<order>\*u).z::<interval>,
- \*:perRevolution.sin(<order>\*u).x::<interval>,
- \*:perRevolution.sin(<order>\*u).y::<interval>,
- \*:perRevolution.sin(<order>\*u).z::<interval>.

Name	Type	Annotation
order	uint	once, twice, ...
estimateX	boolean	along
estimateY	boolean	cross
estimateZ	boolean	radial
interval	timeSeries	setup new parameters each interval
perArc	boolean	

### 5.34.2 AccBias

Temporal changing accelerometer bias per axis in [ $m/s^2$ ] in Satellite Reference Frame (SRF). If the attitude of the satellite is not provided the Celestial Reference Frame (CRF) is used instead.

The [parameter names](#) are

- \*:accBias.x::\*:\*,
- \*:accBias.y::\*:\*,
- \*:accBias.z::\*:\*.

Name	Type	Annotation
estimateX	boolean	along
estimateY	boolean	cross
estimateZ	boolean	radial
temporal	parametrizationTemporal	
perArc	boolean	

### 5.34.3 AccelerometerScaleFactors

Accelerometer scale factor per axis.

The  parameter names are

- \*:accScale.x:<temporal>:<interval>,
- \*:accScale.y:<temporal>:<interval>,
- \*:accScale.z:<temporal>:<interval>,
- \*:accScaleCross.xy:<temporal>:<interval>,
- \*:accScaleCross.xz:<temporal>:<interval>,
- \*:accScaleCross.yz:<temporal>:<interval>,
- \*:accScaleRotation.xy:<temporal>:<interval>,
- \*:accScaleRotation.xz:<temporal>:<interval>,
- \*:accScaleRotation.yz:<temporal>:<interval>.

This parametrization needs the attitude of the satellite.

Name	Type	Annotation
 inputFileAccelerometer	filename	
 estimateX	boolean	along
 estimateY	boolean	cross
 estimateZ	boolean	radial
 estimateCrossTalk	boolean	non-orthogonality of axes
 estimateRotation	boolean	misalignment
 temporal	parametrizationTemporal	
 perArc	boolean	

### 5.34.4 GnssSolarRadiation

GNSS solar radiation pressure model. Paramters are estimated in [ $nm/s^2 = 10^{-9} m/s^2$ ].

The  parameter names are

- \*:solarRadiationPressure.ECOM.D0:\*:\*,
- \*:solarRadiationPressure.ECOM.DC2:\*:\*,
- \*:solarRadiationPressure.ECOM.DS2:\*:\*,
- \*:solarRadiationPressure.ECOM.DC4:\*:\*,
- \*:solarRadiationPressure.ECOM.DS4:\*:\*,
- \*:solarRadiationPressure.ECOM.Y0:\*:\*,
- \*:solarRadiationPressure.ECOM.B0:\*:\*,
- \*:solarRadiationPressure.ECOM.BC1:\*:\*,

- \*:**solarRadiationPressure**.ECOM.BS1:\*:\*,
- \*:**solarRadiationPressure**.ECOM.BC3:\*:\*,
- \*:**solarRadiationPressure**.ECOM.BS3:\*:\*.

This parametrization needs the attitude of the satellite.

Name	Type	Annotation
estimateD0	boolean	constant term along D-axis (sat-sun vector)
estimateD2	boolean	2-per-rev terms along D-axis
estimateD4	boolean	4-per-rev terms along D-axis
estimateY0	boolean	constant term along Y-axis (solar panel axis)
estimateB0	boolean	constant term along B-axis (cross product D x Y)
estimateB1	boolean	1-per-rev terms along B-axis
estimateB3	boolean	3-per-rev terms along B-axis
perArc	boolean	
eclipse	eclipse	

### 5.34.5 ThermosphericDensity

Estimate the thermospheric density along the orbit using a satllite macro model. An optional thermospheric model can be used to compute temperature and wind. The temperature is used to estimate variable drag and lift coefficients, otherwise a constant drag coefficient is used. The density is estimated in [ $kg/m^3$ ].

The **parameter names** are \*:**density**:<temporal>:<interval>.

This parametrization needs the macro model and the attitude of the satellite.

Name	Type	Annotation
thermosphere	<b>thermosphere</b>	for wind and temperature
earthRotation	double	[rad/s]
considerTemperature	boolean	compute drag and lift, otherwise simple drag coefficient is used
considerWind	boolean	
temporalDensity	<b>parametrizationTemporal</b>	parameters along orbit
perArc	boolean	

### 5.34.6 ModelScale

Estimate a scale factor for a given model. The **parameter names** are \*:**modelScale**:<temporal>:<interval>.

Name	Type	Annotation
miscAccelerations	<b>miscAccelerations</b>	
temporal	<b>parametrizationTemporal</b>	
perArc	boolean	

## 5.35 ParametrizationGnssAntenna

Parametrization of antenna center variations. It will be used to set up the design matrix in a least squares adjustment. Usually the parametrization is setup separately for different **gnssType**.

If multiple parametrizations are given the parameters are sequentially appended in the design matrix and parameter vector.

### 5.35.1 Center

Antenna center or, if setup for a specific **gnssType**, phase/code center offset (e.g. \*1\*G for GPS L1 phase center offset) in [m].

The **parameter names** are

- \*:antennaCenter.x:\*:\*,
- \*:antennaCenter.y:\*:\*,
- \*:antennaCenter.z:\*:\*.

Name	Type	Annotation
estimateX	boolean	
estimateY	boolean	
estimateZ	boolean	

### 5.35.2 SphericalHarmonics

Parametrization of antenna center variations in [m] in terms of spherical harmonics. As usually only data above the horizon are observed only the even spherical harmonics (degree/order  $m + n$  even), which are symmetric to the equator, are setup.

The total count of parameters is  $((n_{max} + 1)(n_{max} + 2) - n_{min}(n_{min} + 1))/2$  and the **parameter names** are

- \*:antennaCenterVariations.sphericalHarmonics.c\_<degree>\_<order>:\*:\*,
- \*:antennaCenterVariations.sphericalHarmonics.s\_<degree>\_<order>:\*:\*.

Name	Type	Annotation
minDegree	uint	min degree
maxDegree	uint	max degree

### 5.35.3 RadialBasis

Parametrization of antenna center variations with radial basis functions

$$ACV(\mathbf{x}(A, E)) = \sum_i a_i \Phi(\mathbf{x} \cdot \mathbf{x}_i) \quad (5.120)$$

where  $a_i$  in  $[m]$  the coefficients which has to be estimated and  $\Phi$  are the basis functions

$$\Phi(\cos \psi) = \sum_n \sqrt{2n + 1} P_n(\cos \psi). \quad (5.121)$$

The **parameter names** are \*:antennaCenterVariations.radialBasis.<index>.<total count>:\*\*\*.

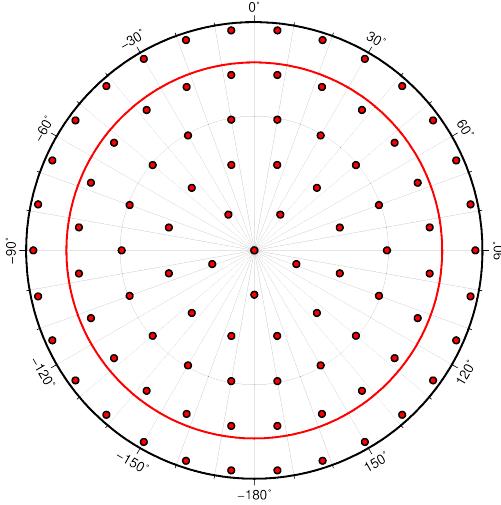


Figure 5.10: Nodal points of the basis functions using a Reuter grid for transmitting satellites (view angle of 18 deg). The red line indicates the view angle of 14 deg of ground stations.

Name	Type	Annotation
grid	grid	nodal points of shannon kernels
minDegree	uint	min degree of shannon kernel
maxDegree	uint	max degree of shannon kernel

## 5.36 ParametrizationGravity

This class gives a parametrization of the time depending gravity field. Together with the class **Observation** it will be used to set up the design matrix in a least squares adjustment. If multiple parametrizations are given the coefficients in the parameter vector are sequentially appended.

### 5.36.1 SphericalHarmonics

The potential  $V$  is parametrized by a expansion of (fully normalized) spherical harmonics

$$V(\lambda, \vartheta, r) = \frac{GM}{R} \sum_{n=0}^{\infty} \sum_{m=0}^n \left( \frac{R}{r} \right)^{n+1} (c_{nm} C_{nm}(\lambda, \vartheta) + s_{nm} S_{nm}(\lambda, \vartheta)). \quad (5.122)$$

You can set the range of degree  $n$  with **minDegree** and **maxDegree**. The sorting sequence of the potential coefficients in the parameter vector can be defined by **numbering**.

The total count of parameters is  $(n_{max} + 1)^2 - n_{min}^2$  and the **parameter names** are

- \*:**sphericalHarmonics.c\_<degree>\_<order>:\*:\***,
- \*:**sphericalHarmonics.s\_<degree>\_<order>:\*:\***.

Name	Type	Annotation
<b>minDegree</b>	uint	
<b>maxDegree</b>	uint	
<b>GM</b>	double	Geocentric gravitational constant
<b>R</b>	double	reference radius
<b>numbering</b>	<b>sphericalHarmonicsNumbering</b>	numbering scheme

### 5.36.2 RadialBasis

The potential  $V$  is represented by a sum of space localizing basis functions

$$V(\mathbf{x}) = \sum_i a_i \Phi(\mathbf{x}, \mathbf{x}_i) \quad (5.123)$$

where  $a_i$  the coefficients which has to be estimated and  $\Phi$  are the basis functions given by isotropic radial **kernel** functions

$$\Phi(\cos \psi, r, R) = \sum_n \left( \frac{R}{r} \right)^{n+1} k_n \sqrt{2n+1} \bar{P}_n(\cos \psi). \quad (5.124)$$

The basis functions are located on a grid  $\mathbf{x}_i$  given by **grid**. This class can also be used to estimate point masses if **kernel** is set to density.

The **parameter names** are \*:**radialBasis.<index>.<total count>:\*:\***.

Name	Type	Annotation
<b>kernel</b>	<b>kernel</b>	shape of the radial basis function
<b>grid</b>	<b>grid</b>	nodal point distribution

### 5.36.3 Temporal

The time variable potential is given by

$$V(\mathbf{x}, t) = \sum_i V_i(\mathbf{x}) \Psi_i(t), \quad (5.125)$$

where  $V_i(\mathbf{x})$  is the spatial parametrization of the gravity field and can be chosen with **parametrizationGravity**. The parametrization in time domain  $\Psi_i(t)$  is selected by **parametrizationTemporal**. The total parameter count is the parameter count of **parametrizationTemporal** times the parameter count of **parametrizationGravity**.

Name	Type	Annotation
parametrizationGravity	parametrizationGravity	
parametrizationTemporal	parametrizationTemporal	

### 5.36.4 LinearTransformation

Parametrization of the gravity field on the basis of a linear transformation of a source parametrization. The linear transformation changes the original solution space represented by **parametrizationGravitySource** from

$$\mathbf{l} = \mathbf{Ax} + \mathbf{e} \quad (5.126)$$

to

$$\mathbf{l} = \mathbf{Af} + \mathbf{e} \quad (5.127)$$

through the linear transformation  $\mathbf{x} = \mathbf{f}$ . It follows that the rows of the matrix  $\mathbf{F}$  in **inputfileTransformationMatrix** coincides with the number of parameters in **parametrizationGravitySource**. The new parameter count is given by the number of columns in  $\mathbf{F}$  and may be smaller, equal or larger than the original parameter count.

The **parameter names** are \*:**transformedParameter.<index>.<total count>:\***:

Name	Type	Annotation
parametrizationGravitySource	parametrizationGravity	
inputfileTransformationMatrix	filename	transformation matrix from target to source parametrization (rows of this matrix must coincide with the parameter count of the source parametrization)

### 5.36.5 EarthquakeOscillation

This class is used to estimate the earthquake oscillation function parameters, i.e.  $C_{nlm}$ ,  $\omega_{nlm}$ , and  $P_{nlm}$ . The results describes the variation in the gravitational potential field caused by large earthquakes.

$$C_{lm}(\mathbf{t}) = \sum_{n=0}^N C_{nlm}(1 - \cos(\omega_{nlm} dt) \exp(P_{nlm} \omega_{nlm} dt)), \quad (5.128)$$

with  $\omega_{nlm} = \frac{2\pi}{T_{nlm}}$  and  $P_{nlm} = \frac{-1}{2Q_{nlm}}$ . In this equation,  $Q_{nlm}$  is the attenuation factor,  $n$  is the overtone factor,  $m$  is degree,  $l$  is order, and  $t$  is time after earthquake in second.

The **parameter names** are

- \*:earthquakeParameter.c\_<degree>\_<order>\_A:\*:\*,
- \*:earthquakeParameter.s\_<degree>\_<order>\_A:\*:\*,
- \*:earthquakeParameter.c\_<degree>\_<order>\_W:\*:\*,
- \*:earthquakeParameter.s\_<degree>\_<order>\_W:\*:\*,
- \*:earthquakeParameter.c\_<degree>\_<order>\_P:\*:\*,
- \*:earthquakeParameter.s\_<degree>\_<order>\_P:\*:\*.

Name	Type	Annotation
inputInitialCoefficient	filename	initial values for oscillation parameters
time0	time	the time earthquake happened
minDegree	uint	
maxDegree	uint	
GM	double	Geocentric gravitational constant
R	double	reference radius
numbering	sphericalHarmonicsNumbering	numbering scheme

## 5.37 ParametrizationSatelliteTracking

This class defines parameters of Satellite-to-Satellite tracking observations. It will be used to set up the design matrix in a least squares adjustment. If multiple parametrizations are given the coefficients in the parameter vector are sequently appended.

### 5.37.1 AntennaCenter

Estimate the KBR antenna phase centre (APC) coordinates in [m] for each spacecraft in satellite reference frame (SRF) as constant per axis. The observation equations are computed by taking the derivative of the antenna offset correction equation w.r.t. the KBR APC coordinates.

The **parameter names** are

- `satellite1.satellite2:sstAntennaCenter1.x:*`:
- `satellite1.satellite2:sstAntennaCenter1.y:*`:
- `satellite1.satellite2:sstAntennaCenter1.z:*`:
- `satellite1.satellite2:sstAntennaCenter2.x:*`:
- `satellite1.satellite2:sstAntennaCenter2.y:*`:
- `satellite1.satellite2:sstAntennaCenter2.z:*`.

Name	Type	Annotation
estimate1X	boolean	along (satellite 1)
estimate1Y	boolean	cross (satellite 1)
estimate1Z	boolean	nadir (satellite 1)
estimate2X	boolean	along (satellite 2)
estimate2Y	boolean	cross (satellite 2)
estimate2Z	boolean	nadir (satellite 2)
interpolationDegree	uint	differentiation by polynomial approximation of degree n

### 5.37.2 Bias

Estimate bias for SST observations in [m] or [m/s]. The temporal variation is defined by **parametrizationTemporal**.

The **parameter names** are `satellite1.satellite2:sstBias:<temporal>:<interval>`.

Name	Type	Annotation
temporal	<a href="#">parametrizationTemporal</a>	
perArc	boolean	

### 5.37.3 Scale

Estimate scale factor for SST observations with respect to reference SST data **inputfileSatelliteTracking**. The temporal variation is defined by **parametrizationTemporal**.

The **parameter names** are `satellite1.satellite2:sstScale:<temporal>:<interval>`.

Name	Type	Annotation
inputfileSatelliteTracking	filename	
temporal	parametrizationTemporal	
perArc	boolean	

### 5.37.4 TimeBias

Estimate time shift in seconds in SST observations, with defined temporal variation by **parametrizationTemporal**. The design matrix is computed by taking the derivative of the ranging data w.r.t. time.

The **parameter names** are `satellite1.satellite2:sstTimeBias:<temporal>:<interval>`.

Name	Type	Annotation
polynomialDegree	uint	polynomial degree
temporal	parametrizationTemporal	
perArc	boolean	

### 5.37.5 ScaleModel

Estimate scale factors for deterministic signal models from satellite tracking instrument file **inputfileSatelliteTracking**, see **EnsembleAveragingScaleModel**. Amplitude variation of model waveforms is defined by **parametrizationTemporal**.

The **parameter names** are `satellite1.satellite2:scaleModel:<temporal>:<interval>`.

Name	Type	Annotation
inputfileSatelliteTracking	filename	
temporal	parametrizationTemporal	
perArc	boolean	

### 5.37.6 SpecialEffect

Estimate deterministic signals in the GRACE K-Band measurements caused by Sun intrusions into the star camera baffles of GRACE-A and eclipse transits of the satellites. These events can be time-indexed beforehand using satellite position and orientation, see **GraceSstSpecialEvents**. The shape of this short-period waveform is nearly constant within one month and can be approximated by a polynomial. The amplitude variation of the waveform can also be taken into account by **parametrizationTemporal**.

The **parameter names** are `satellite1.satellite2:<type>.legendrePolynomial.n<degree>:<temporal>:<interval>`.

Name	Type	Annotation
inputfileEvents	filename	instrument with GRACE events
type	choice	
eclipse1		
eclipse2		
starCameraBox1		
starCameraBox2		
starCameraBox3		
starCameraBox4		

---

└─ starCameraBox5		
└─ starCameraBox6		
└─ marginLeft	double	margin size (on both sides) [seconds]
└─ marginRight	double	margin size (on both sides) [seconds]
└─ minNumberOfEvents	uint	min. number of events to setup parameters
└─ polynomialDegree	uint	polynomial degree
└─ temporal	parametrizationTemporal	

---

## 5.38 ParametrizationTemporal

This class gives a parametrization of time depending parameters (gravity field, positions, ...). It will be used to set up the design matrix in a least squares adjustment. If multiple parametrizations are given the coefficients in the parameter vector are sequentially appended.

Usually time intervals are defined half open meaning the last time belongs not to the interval. This behaviour can be changed for the last interval with **includeLastTime**.

### 5.38.1 Constant

Represents a parameter being constant in time in each **interval**.

The **parameter names** are `*:*:<interval>`.

Name	Type	Annotation
<b>interval</b>	<b>timeSeries</b>	
<b>includeLastTime</b>	boolean	

### 5.38.2 Trend

A time variable function is given by a linear trend

$$f(x, t) = \frac{1}{T}(t - t_0) \cdot f_t(x), \quad (5.129)$$

with  $t_0$  is **timeStart** and  $T$  is **timeStep** in days. A constant term is not included and must be added separately.

The **parameter name** is `*:*:trend.<timeStep(days)>*(t-<timeStart>):*`.

Name	Type	Annotation
<b>timeStart</b>	time	reference time
<b>timeStep</b>	time	

### 5.38.3 Splines

A time variable function is given by

$$f(x, t) = \sum_i f_i(x) \Psi_i(t), \quad (5.130)$$

with the (spatial) coefficients  $f_i(x)$  as parameters and the temporal basis functions  $\Psi_i(t)$ . Basis splines are defined as polynomials of degree  $n$  in intervals between nodal points in time  $t_i$ , for details see [basis splines \(2.2\)](#).

The parameters are ordered timewise. First all parameters of  $f_{i=1}(x)$  then  $f_{i=2}(x)$  and so on. The total parameter count in each **interval** is  $N = N_t + d - 1$ , where  $N_t$  is the count of time points from **timeSeries** in each interval and  $d$  is the **degree**.

The **parameter names** are `*:*:spline.n<degree>:<interval of each spline>`.

Name	Type	Annotation
degree	uint	degree of splines
timeSeries	timeSeries	nodal points in time domain
intervals	timeSeries	
includeLastTime	boolean	

### 5.38.4 Polynomial

A time variable function is represented by Legendre polynomials in each **interval**. The time is normed to  $[-1, 1]$  in each interval.

The total parameter count is  $(N + 1)M$ , where  $N$  is the polynomial degree and  $M$  the number of intervals with the **parameter names** `*:::legendrePolynomial.n<degree>:<interval>`.

Name	Type	Annotation
polynomialDegree	uint	polynomial degree
interval	timeSeries	intervals of polynomials
includeLastTime	boolean	

### 5.38.5 Oscillation

A time variable function is given by a oscillation

$$f(x, t) = f^c(\mathbf{x}) \cos(\omega_i(t)) + f^s(\mathbf{x}) \sin(\omega_i(t)) \quad (5.131)$$

with  $\omega_i = \frac{2\pi}{T_i}(t - t_0)$ ,  $t_0$  is **timeStart** and  $T$  is **timePeriod** in days.

The **parameter names** are `*:::oscillation.cos(2*pi/<period(days)>*(t-<timeStart>)):*` and `*:::oscillation.sin(2*pi/<period(days)>*(t-<timeStart>)):*`.

Name	Type	Annotation
period	time	[day]
time0	time	reference time

### 5.38.6 Fourier

A time variable function is given by a fourier expansion

$$f(x, t) = \sum_{m=1}^M f_m^c(\mathbf{x}) \cos(2\pi m \tau) + f_m^s(\mathbf{x}) \sin(2\pi m \tau) \quad (5.132)$$

with the normalized time

$$\tau = \frac{t - t_A}{t_B - t_A}, \quad (5.133)$$

and  $t_A$  is **timeStart**,  $t_B$  is **timeEnd** in each **interval** and  $M$  is the **fourierDegree**.

The total parameter count is  $2MN$ , where  $N$  is the number of intervals. The parameters are sorted in following order:  $f_1^c, f_1^s, f_2^c, \dots$  with the **parameter names** `*:::fourier.cos(<m>*<x>:<interval>)` and `*:::fourier.sin(<m>*<x>:<interval>)`.

Name	Type	Annotation
fourierDegree	uint	
interval	timeSeries	intervals of fourier series
includeLastTime	boolean	

### 5.38.7 DoodsonHarmonic

The time variable function is given by a fourier expansion

$$f(x, t) = \sum_i f_i^c(x) \cos(\Theta_i(t)) + f_i^s(x) \sin(\Theta_i(t)), \quad (5.134)$$

where  $\Theta_i(t)$  are the arguments of the tide constituents  $i$

$$\Theta_i(t) = \sum_{k=1}^6 n_i^k \beta_k(t), \quad (5.135)$$

where  $\beta_k(t)$  are the Doodson's fundamental arguments  $(\tau, s, h, p, N', p_s)$  and  $n_i^k$  are the Doodson multipliers for the term at frequency  $i$ . The multipliers must be given by **doodson** coded as Doodson number (e.g. 255.555) or as names introduced by Darwin (e.g. M2).

The major constituents given by **doodson** can be used to interpolate minor tidal constituents using the file **inputfileAdmittance**. This file can be created with **DoodsonHarmonicsCalculateAdmittance**.

The total parameter count is  $2N$  with  $N$  the number of doodson frequencies. The parameters are sorted in following order:  $f_1^c, f_1^s, f_2^c, \dots$  with the **parameter names \*:\*:doodson.cos(<doodsonName>):\*** and **\*:\*:doodson.sin(<doodsonName>):\***.

Name	Type	Annotation
doodson	doodson	code number (e.g. 255.555) or darwin name (e.g. M2)
inputfileAdmittance	filename	interpolation of minor constituents

## 5.39 Planet

Defines the planet to compute the  **ephemeris**.

Name	Type	Annotation
 planetType	choice	planet
 earth		
 sun		
 moon		
 mercury		
 venus		
 mars		
 jupiter		
 saturn		
 uranus		
 neptune		
 pluto		
 solarBaryCenter		
 earthMoonBaryCenter		

## 5.40 PlatformSelector

Select a list of platforms (stations, satellites, ...).

See also [GnssProcessing](#).

### 5.40.1 All

Select all platforms.

### 5.40.2 Wildcard

Select all receivers/transmitters which match the `name`, `markerName`, and `markerNumber`.

Name	Type	Annotation
▶ name	string	wildcards: * and ?
▶ markerName	string	wildcards: * and ?, from platform
▶ markerNumber	string	wildcards: * and ?, from platform

### 5.40.3 File

Select receivers/transmitters from each row of `inputfileStringTable`. Additional columns in a row represent alternatives if previous names are not available (e.g. without observation file).

Name	Type	Annotation
▶ inputfileStringTable	filename	list of names with alternatives

### 5.40.4 Equipment

Select all platforms which has the specified equipment in the processed time interval.

Name	Type	Annotation
▶ name	string	wildcards: * and ?
▶ serial	string	wildcards: * and ?
▶ equipmentType	choice	equipment type
▶ all		all types
▶ gnssAntenna	sequence	antennas
▶ radome	string	wildcards: * and ?
▶ gnssReceiver	sequence	receivers
▶ version	string	wildcards: * and ?
▶ slrStation		SLR station
▶ slrRetroReflector		laser retroreflector
▶ satelliteIdentifier	sequence	satellite identifier
▶ cospar	string	wildcards: * and ?
▶ norad	string	wildcards: * and ?
▶ sic	string	wildcards: * and ?
▶ sp3	string	wildcards: * and ?
▶ other		other types

### 5.40.5 Exclude

Select all receivers/transmitters except  **selector**.

Name	Type	Annotation
 selector	platformSelector	

## 5.41 PlotAxis

Defines the style of the axes of [PlotGraph](#).

### 5.41.1 Standard

General axis for arbitrary input data.

Name	Type	Annotation
min	double	The minimum value of the axis. If no value is given, the minimum scale value is set automatically.
max	double	The maximum value of the axis. If no value is given, the maximum scale value is set automatically.
majorTickSpacing	double	The boundary annotation.
minorTickSpacing	double	The spacing of the frame tick intervals.
gridLineSpacing	double	The spacing of the grid line intervals
gridLine	plotLine	The style of the grid lines.
unit	string	Naming unit to append to the axis values.
label	string	The description of the axis.
logarithmic	boolean	If set to 'yes', a logarithmic scale is used for the axis.
color	plotColor	Setting the color of the axis bars and labels.
changeDirection	boolean	If set to 'yes', the directions right/up are changed to left/down.

### 5.41.2 Time

The input data are interpreted as MJD (modified Julian date). The unit of the tick spacings should be appended to the number and can be any of

- Y (year, plot with 4 digits)
- y (year, plot with 2 digits)
- O (month, plot using `FORMAT_DATE_MAP`)
- o (month, plot with 2 digits)
- U (ISO week, plot using `FORMAT_DATE_MAP`)
- u (ISO week, plot using 2 digits)
- r (Gregorian week, 7-day stride from start of week `TIME_WEEK_START`)
- K (ISO weekday, plot name of day)
- D (date, plot using `FORMAT_DATE_MAP`)
- d (day, plot day of month 0-31 or year 1-366, via `FORMAT_DATE_MAP`)
- R (day, same as d, aligned with `TIME_WEEK_START`)
- H (hour, plot using `FORMAT_CLOCK_MAP`)
- h (hour, plot with 2 digits)
- M (minute, plot using `FORMAT_CLOCK_MAP`)

- m (minute, plot with 2 digits)
- S (second, plot using `FORMAT_CLOCK_MAP`)
- s (second, plot with 2 digits).

A secondary time axis can be added to specify larger intervals (e.g dates of hourly data).

Examples: Settings for Fig. 5.11: `majorTickSpacing=6H`, secondary: `majorTickSpacing=1D`.

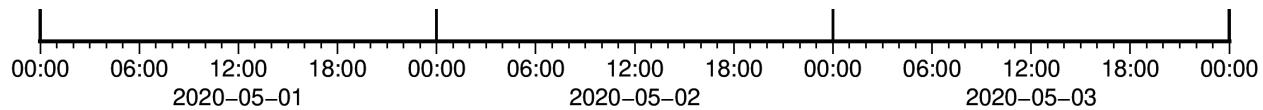


Figure 5.11: Time axis for daily data.

Settings for Fig. 5.12: `majorTickSpacing=2d`, secondary: `majorTickSpacing=10, options=FORMAT_DATE_MAP="o yyyy"`.

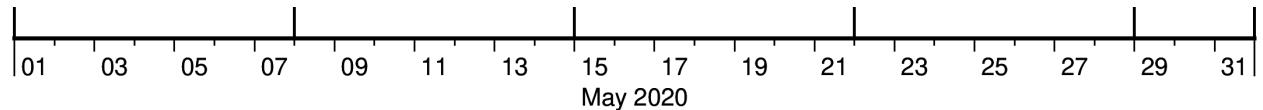


Figure 5.12: Time axis for monthly data.

Settings for Fig. 5.13: `majorTickSpacing=1o, secondary: majorTickSpacing=1Y, options=FORMAT_DATE_MAP="mm"`.

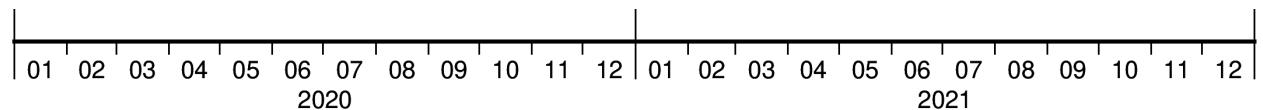


Figure 5.13: Time axis for yearly data.

Name	Type	Annotation
<code>min</code>	time	The minimum value of the time axis. If no value is given, the minimum scale value is set automatically.
<code>max</code>	time	The maximum value of the time axis. If no value is given, the maximum scale value is set automatically.
<code>majorTickSpacing</code>	string	Y: year, o: month
<code>minorTickSpacing</code>	string	D: date, d: day
<code>gridLineSpacing</code>	string	H: clock, h: hour, m: minute, s: second
<code>secondary</code>	sequence	secondary time axis
<code>majorTickSpacing</code>	string	Y: year, o: month
<code>minorTickSpacing</code>	string	D: date, d: day
<code>gridLineSpacing</code>	string	H: clock, h: hour, m: minute, s: second
<code>color</code>	<code>plotColor</code>	color of axis bars and labels
<code>gridLine</code>	<code>plotLine</code>	The style of the grid lines.
<code>changeDirection</code>	boolean	right-\$i\$left / up-\$i\$down
<code>options</code>	string	adjust date format

### 5.41.3 Labeled

Axis with string labels. The coordinate system is based on the label indices (e.g. 0, 1, 2).

Name	Type	Annotation
labels	string	tick labels (ticks are placed at their index. e.g. 0, 1, ..., 5)
min	expression	minimum value of the axis
max	expression	maximum values of the axis
orthogonalLabels	boolean	labels are oriented orthogonal to axis
gridLine	plotLine	The style of the grid lines.
color	plotColor	set the color of the axis and labels
changeDirection	boolean	If set to 'yes', the directions right/up are changed to left/down.

## 5.42 PlotColor

Selects a color. Used in [PlotDegreeAmplitudes](#), [PlotGraph](#), [PlotMap](#), [PlotMatrix](#), [PlotSphericalHarmonicsTriangle](#).

Name	Type	Annotation
plotColorType	choice	color
black		
red		
blue		
green		
orange		
darkred		
yellow		
lightgreen		
gray		
rgb	sequence	
red	uint	0.255
green	uint	0.255
blue	uint	0.255
grayscale	sequence	
value	uint	0.255
namedColor	sequence	
colorName	string	name after GMT definition
cycler	sequence	
index	uint	pick color based on index expression
inputfileColorList	filename	list of colors as defined by GMT

## 5.43 PlotColorbar

A colorbar as used in [PlotMap](#), [PlotMatrix](#), [PlotSphericalHarmonicsTriangle](#).

Name	Type	Annotation
plotColorbarType	sequence	
min	double	
max	double	
annotation	double	boundary annotation
unit	string	appended to axis values
label	string	description of the axis
logarithmic	boolean	use logarithmic scale
triangleLeft	boolean	
triangleRight	boolean	
illuminate	boolean	
vertical	boolean	plot vertical color bar on the right
length	double	length of colorbar in percent
margin	double	between colorbar and figure [cm]
colorTable	string	name of the color bar
reverse	boolean	reverse direction
showColorbar	boolean	

## 5.44 PlotGraphLayer

Defines the content of an xy-plot of **PlotGraph**. Multiple layers are plotted sequentially. With **plotOnSecondAxis** the alternative y-axis on the right hand side can be selected if provided.

### 5.44.1 LinesAndPoints

Draws a **line** and/or points (**symbol**) of xy data. The standard **dataVariables (1.2.3)** are available to select the data columns of **inputfileMatrix**. If no **color** of the **symbol** is given a **colorbar** is required and the color is determined by **valueZ**. Additionally a vertical error bar can be plotted at each data point with size **valueErrorBar**.

See **Gravityfield2AreaMeanTimeSeries** for an example plot.

Name	Type	Annotation
<b>inputfileMatrix</b>	filename	each line contains x,y
<b>valueX</b>	expression	expression for x-values (input columns are named data0, data1, ...)
<b>valueY</b>	expression	expression for y-values (input columns are named data0, data1, ...)
<b>valueZ</b>	expression	expression for the colorbar
<b>valueErrorBar</b>	expression	expression for error bars (input columns are named data0, data1, ...)
<b>description</b>	string	text of the legend
<b>line</b>	<b>plotLine</b>	
<b>symbol</b>	<b>plotSymbol</b>	
<b>plotOnSecondAxis</b>	boolean	draw dataset on a second Y-axis (if available).

### 5.44.2 ErrorEnvelope

Draws a symmetrical envelope around **valueY** as function of **valueX** using deviations **valueErrors**. The standard **dataVariables (1.2.3)** are available to select the data columns of **inputfileMatrix**. The data line itself is not plotted but must be added as extra **layer:linesAndPoints**.

Name	Type	Annotation
<b>inputfileMatrix</b>	filename	each line contains x,y
<b>valueX</b>	expression	expression for x-values (input columns are named data0, data1, ...)
<b>valueY</b>	expression	expression for y-values (input columns are named data0, data1, ...)
<b>valueErrors</b>	expression	expression for error values
<b>description</b>	string	text of the legend
<b>fillColor</b>	<b>plotColor</b>	fill color of the envelope
<b>edgeLine</b>	<b>plotLine</b>	edge line style of the envelope
<b>plotOnSecondAxis</b>	boolean	draw dataset on a second Y-axis (if available).

### 5.44.3 Bars

Creates a bar plot with vertical or **horizontal** bars out of the given x- and y-values. The standard **dataVariables (1.2.3)** are available to select the data columns of **inputfileMatrix**. The bars ranges from **valueBase** (can be also an expression) to the **valueY**. If no **color** is given a **colorbar** is required and the color is determined by **valueZ**.

See **Instrument2Histogram** for an example plot.

Name	Type	Annotation
inputfileMatrix	filename	each line contains x,y
valueX	expression	expression for x-values (input columns are named data0, data1, ...)
valueY	expression	expression for y-values (input columns are named data0, data1, ...)
valueZ	expression	expression for the colorbar
valueBase	expression	base value of bars (default: minimum y-value)
width	expression	width of bars (default: minimum x-gap)
horizontal	boolean	draw horizontal bars instead of vertical
description	string	text of the legend
color	plotColor	
edgeLine	plotLine	line
plotOnSecondAxis	boolean	draw dataset on a second Y-axis (if available).

#### 5.44.4 Gridded

Creates a regular grid of yxz values. The standard [dataVariables \(1.2.3\)](#) are available to select the data columns of **inputfileMatrix**. Empty grid cells are not plotted. Cells with more than one value will be set to the mean value. The grid spacing is determined by the median spacing of the input data or set by **incrementX/Y**.

See [Orbit2ArgumentOfLatitude](#) for an example plot.

Name	Type	Annotation
inputfileMatrix	filename	each line contains x,y,z
valueX	expression	expression for x-values (input columns are named data0, data1, ...)
valueY	expression	expression for y-values (input columns are named data0, data1, ...)
valueZ	expression	expression for the colorbar
incrementX	double	the grid spacing
incrementY	double	the grid spacing
plotOnSecondAxis	boolean	draw dataset on a second Y-axis (if available).

#### 5.44.5 Rectangle

Plots a rectangle to highlight an area.

Name	Type	Annotation
minX	double	empty: left
maxX	double	empty: right
minY	double	empty: bottom
maxY	double	empty: top
description	string	text of the legend
edgeLine	plotLine	
fillColor	plotColor	
plotOnSecondAxis	boolean	draw dataset on a second Y-axis (if available).

#### 5.44.6 Text

Writes a **text** at **originX** and **originY** position in the graph. With **clip** the text is cutted at the boundaries of the plotting area.

Name	Type	Annotation
text	string	
originX	double	
originY	double	
offsetX	double	[cm] x-offset from origin
offsetY	double	[cm] y-offset from origin
alignment	string	L, C, R (left, center, right) and T, M, B (top, middle, bottom)
fontSize	double	[pt]
fontColor	plotColor	
clip	boolean	clip at boundaries
plotOnSecondAxis	boolean	draw dataset on a second Y-axis (if available).

### 5.44.7 DegreeAmplitudes

Plot degree amplitudes of potential coefficients computed by **Gravityfield2DegreeAmplitudes** or **PotentialCoefficients2DegreeAmplitudes**. The standard **dataVariables** (1.2.3) are available to select the data columns of **inputfileMatrix**. It plots a solid line for the **valueSignal** and a dotted line for the **valueError** per default.

Name	Type	Annotation
inputfileMatrix	filename	degree amplitudes
valueDegree	expression	expression for x-values (degrees) (input columns are named data0, data1, ...)
valueSignal	expression	expression for y-values (signal) (input columns are named data0, data1, ...)
valueErrors	expression	expression for y-values (formal errors)
description	string	text of the legend
lineSignal	plotLine	
lineErrors	plotLine	
plotOnSecondAxis	boolean	draw dataset on a second Y-axis (if available).

## 5.45 PlotLegend

Plot a legend of the descriptions provided in `plotGraphLayer` in `PlotGraph`.

Name	Type	Annotation
plotLegendType	sequence	
width	double	legend width [cm]
height	double	legend height [cm] (default: estimated)
positionX	double	legend x-position in normalized (0-1) coordinates.
positionY	double	legend y-position in normalized (0-1) coordinates.
anchorPoint	string	Two character combination of L, C, R (for left, center, or right) and T, M, B for top, middle, or bottom. e.g., TL for top left
columns	uint	number of columns in legend
textColor	plotColor	color of the legend text
fillColor	plotColor	fill color of the legend box
edgeLine	plotLine	style of the legend box edge

## 5.46 PlotLine

Defines the line style to be plotted.

### 5.46.1 Solid

Draws a solid line.

Name	Type	Annotation
width	double	line width [p]
color	plotColor	

### 5.46.2 Dashed

Draws a dashed line.

Name	Type	Annotation
width	double	line width [p]
color	plotColor	

### 5.46.3 Dotted

Draws a dotted line.

Name	Type	Annotation
width	double	line width [p]
color	plotColor	

### 5.46.4 Custom

Draws a custom line. The line `style` code is described in <https://docs.generic-mapping-tools.org/latest/cookbook/features.html#specifying-pen-attributes>.

Name	Type	Annotation
style	string	line style code
width	double	line width [p]
color	plotColor	

## 5.47 PlotMapLayer

Defines the content of a map of **PlotMap**. Multiple layers are plotted sequentially.

### 5.47.1 GriddedData

Creates a regular grid of yxz values. The standard **dataVariables** (1.2.3) are available to select the data column of **inputfileGriddedData**. Empty grid cells are not plotted. Cells with more than one value will be set to the mean value. The grid spacing can be determined automatically for regular rectangular grids otherwise it must be set with **increment**. To get a better display together with some projections the grid should be internally **resampled** to higher resolution. It is assumed that the points of **inputfileGriddedData** represents centers of grid cells. This assumption can be changed with **gridlineRegistered** (e.g if the data starts at the north pole).

Name	Type	Annotation
inputfileGriddedData	filename	
value	expression	expression to compute values (input columns are named data0, data1, ...)
increment	angle	the grid spacing [degrees]
illuminate	boolean	illuminate grid
resample	sequence	
intermediateDpi	double	oversample grid for a smoother visual effect
interpolationMethod	choice	interpolation method for oversampling
bspline		B-Spline interpolation
bicubic		bicubic interpolation
bilinear		bilinear interpolation
nearest		nearest neighbour interpolation
threshold	double	A threshold of 1.0 requires all (4 or 16) nodes involved in interpolation to be non-NaN. 0.5 will interpolate about half way from a non-NaN value; 0.1 will go about 90% of the way.
gridlineRegistered	boolean	treat input as point values instead of cell means

### 5.47.2 Points

Draws points (**symbol**) and/or **lines** between the points. If no **color** of the **symbol** is given a **colorbar** is required and the color is determined by the **value** expression. The standard **dataVariables** (1.2.3) are available to select the data column of **inputfileGriddedData**.

Name	Type	Annotation
inputfileGriddedData	filename	
value	expression	expression to compute color (input columns are named data0, data1, ...)
symbol	plotSymbol	
line	plotLine	style of connecting lines
drawLineAsGreatCircle	boolean	draw connecting lines as great circles (otherwise, a straight line is drawn instead)

### 5.47.3 Arrows

Draws an arrow for each point in **inputfileGriddedData**. The arrows are defined by the expressions **valueNorth/East**. The standard [dataVariables \(1.2.3\)](#) are available to select the correspondent data columns of **inputfileGriddedData**. The **scale** factor converts the input units to cm in the plot. If no **color** is given a **colorbar** is required and the color is determined by the **value** expression. With **scaleArrow** a reference arrow as legend can be plotted inside or outside the map.

Name	Type	Annotation
inputfileGriddedData	filename	grid file with north and east values for arrows
valueNorth	expression	expression to compute north values (input columns are named data0, data1, ...)
valueEast	expression	expression to compute east values (input columns are named data0, data1, ...)
value	expression	expression to compute arrow color (input columns are named data0, data1, ...)
scale	double	[cm per input unit] length scale factor
penSize	double	[pt] width of arrow shaft
headSize	double	[pt] size of arrow head, 0: no head, negative: reverse head
color	plotColor	empty: from value
scaleArrow	sequence	draw an arrow for scale reference
originX	double	[0-1] 0: left, 1: right
originY	double	[0-1] 0: bottom, 1: top
length	double	in same unit as valueNorth and valueEast
unit	string	displayed unit text (e.g. 1 cm)
label	string	description of the arrows

### 5.47.4 Polygon

Draws a **inputfilePolygon**. If **fillColor** is not set and a **value** is given the fill color is taken from a **colorbar**.

Name	Type	Annotation
inputfilePolygon	filename	
line	plotLine	style of border lines
fillColor	plotColor	polygon fill color (no fill color: determine from value if given, else: no fill)
value	double	value to compute fill color from a colorbar (ignored if a fillColor is given)
drawLineAsGreatCircle	boolean	draw connecting lines as great circles (otherwise, a straight line is drawn instead)

### 5.47.5 Coast

Plots coastlines. GMT provides them in different **resolutions**. Features with an area smaller than **minArea** in  $km^2$  will not be plotted.

Name	Type	Annotation
resolution	choice	
crude		
low		

---

	medium	
	high	
	full	
	line	plotLine line style for coastlines
	landColor	plotColor fill land area
	oceanColor	plotColor fill ocean area
	minArea	uint [km^2] features with a smaller area than this are dropped

---

### 5.47.6 Rivers

Plots rivers and lakes. GMT provides different classes (<https://docs.generic-mapping-tools.org/latest/coast.html>).

Name	Type	Annotation
	choice	
	riversCanalsLakes	
	riversCanals	
	permanentRiversLakes	
	permanentRivers	
	intermittentRivers	
	canals	
	singleClass	sequence
	class	uint 0-10. See GMT documentation
	line	plotLine

---

### 5.47.7 PoliticalBoundary

Plots national boundaries. GMT provides them in different resolutions.

Name	Type	Annotation
	choice	
	crude	
	low	
	medium	
	high	
	full	
	line	plotLine

---

### 5.47.8 BlueMarble

An image of the Earth's surface as seen from outer space - the image is known as *blue marble*. The directory of **inputfileChannels** contains several files in different resolutions representing the Earth's surface each month throughout a year.

Name	Type	Annotation
	filename	Blue Marble image file
	brightness	double brightness of bitmap [-1, 1]
	illuminate	sequence add hillshade based on topography
	inputfileTopography	filename GMT grid file containing topography.
	azimuth	angle direction of lighting source [deg]

---

 elevation	angle	direction of lighting source [deg]
 ambient	double	ambient lighting
 diffuse	double	diffuse lighting
 specular	double	specular reflection
 shine	double	surface shine
 amplitude	double	scale gradient by factor

---

### 5.47.9 Text

Writes a  **text** at  **originLongitude** and  **originLatitude** position in the map. With  **clip** the text is cutted at the boundaries of the plotting area.

---

Name	Type	Annotation
 text	string	
 originLongitude	angle	[deg]
 originLatitude	angle	[deg]
 offsetX	double	[cm] x-offset from origin
 offsetY	double	[cm] y-offset from origin
 alignment	string	L, C, R (left, center, right) and T, M, B (top, middle, bottom)
 fontSize	double	
 fontColor	plotColor	
 clip	boolean	clip at boundaries

---

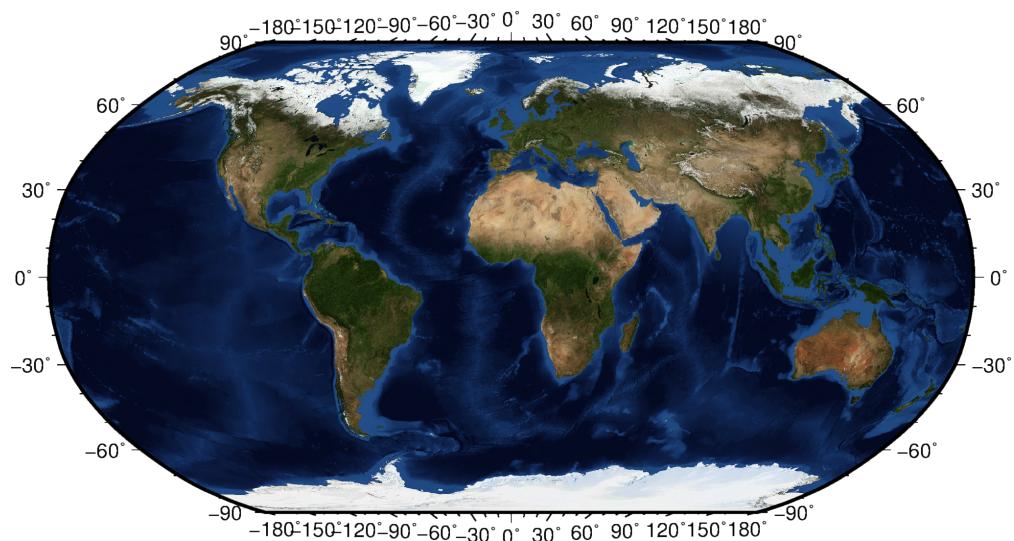


Figure 5.14: The blue marble.

## 5.48 PlotMapProjection

Selects the underlying projection of  **PlotMap**.

### 5.48.1 Robinson

The Robinson projection, presented by Arthur H. Robinson in 1963, is a modified cylindrical projection that is neither conformal nor equal-area. Central meridian and all parallels are straight lines; other meridians are curved. It uses lookup tables rather than analytic expressions to make the world map look right.

Name	Type	Annotation
 centralMeridian	angle	central meridian [degree]

### 5.48.2 Orthographic

The orthographic azimuthal projection is a perspective projection from infinite distance. It is therefore often used to give the appearance of a globe viewed from space.

Name	Type	Annotation
 lambdaCenter	angle	central point [degree]
 phiCenter	angle	central point [degree]

### 5.48.3 Perspective sphere

The orthographic azimuthal projection is a perspective projection from infinite distance. It is therefore often used to give the appearance of a globe viewed from space.

Name	Type	Annotation
 lambdaCenter	angle	longitude of central point in degrees
 phiCenter	angle	latitude of central point in degrees
 altitude	double	[km]
 azimuth	angle	to the east of north of view [degrees]
 tilt	angle	upward tilt of the plane of projection, if negative, then the view is centered on the horizon [degrees]
 viewpointTwist	angle	clockwise twist of the viewpoint [degrees]
 viewpointWidth	angle	width of the viewpoint [degrees]
 viewpointHeight	angle	height of the viewpoint [degrees]

### 5.48.4 Polar

Stereographic projection around given central point.

Name	Type	Annotation
 lambdaCenter	angle	longitude of central point in degrees
 phiCenter	angle	latitude of central point in degrees

### 5.48.5 Skyplot

Skyplot used to plot azimuth/elevation data as generated by [► GnssAntennaDefinition2Skyplot](#) or [► GnssResiduals2Skyplot](#).

### 5.48.6 UTM

A particular subset of the transverse Mercator is the Universal Transverse Mercator (UTM) which was adopted by the US Army for large-scale military maps. Here, the globe is divided into 60 zones between 84°S and 84°N, most of which are 6° wide. Each of these UTM zones have their unique central meridian.

Name	Type	Annotation
zone	string	UTM zone code (e.g. 33N)

### 5.48.7 Lambert

This conic projection was designed by Lambert (1772) and has been used extensively for mapping of regions with predominantly east-west orientation.

Name	Type	Annotation
lambda0	angle	longitude of projection center [deg]
phi0	angle	latitude of projection center [deg]
phi1	angle	latitude of first standard parallel [deg]
phi2	angle	latitude of second standard parallel [deg]

### 5.48.8 Linear

Linear mapping of longitude/latitude to x/y (Plate Caree).

### 5.48.9 Mollweide

This pseudo-cylindrical, equal-area projection was developed by Mollweide in 1805. Parallels are unequally spaced straight lines with the meridians being equally spaced elliptical arcs. The scale is only true along latitudes 40°44' north and south. The projection is used mainly for global maps showing data distributions.

Name	Type	Annotation
centralMeridian	angle	central meridian [degree]

## 5.49 PlotSymbol

Plots a symbol as used e.g. in **plotGraphLayer:linesAndPoints** or **plotMapLayer:points**.

Name	Type	Annotation
plotSymbolType	choice	symbol
circle	sequence	
color	plotColor	empty: determined from value
size	double	size of symbol [point]
blackContour	boolean	
star	sequence	
color	plotColor	empty: determined from value
size	double	size of symbol [point]
blackContour	boolean	
cross	sequence	
color	plotColor	empty: determined from value
size	double	size of symbol [point]
blackContour	boolean	
square	sequence	
color	plotColor	empty: determined from value
size	double	size of symbol [point]
blackContour	boolean	
triangle	sequence	
color	plotColor	empty: determined from value
size	double	size of symbol [point]
blackContour	boolean	
diamond	sequence	
color	plotColor	empty: determined from value
size	double	size of symbol [point]
blackContour	boolean	
dash	sequence	
color	plotColor	empty: determined from value
size	double	size of symbol [point]
blackContour	boolean	

## 5.50 PodRightSide

Observation vector for precise orbit data (POD) of **observation** equations in a least squares adjustment. The observations are reduced by the effect of **inputfileAccelerometer** and **forces** (observed minus computed).

Name	Type	Annotation
podRightSideType	sequence	
inputfileOrbit	filename	kinematic positions of satellite as observations
inputfileAccelerometer	filename	non-gravitational forces in satellite reference frame
forces	forces	

## 5.51 SggRightSide

Observation vector for gradiometer data (satellite gravity gradiometry, SGG) of **observation** equations in a least squares adjustment. The observations are reduced by an **inputfileReferenceGradiometer**, the effect of **referencefield**, and **tides** (observed minus computed).

The reference gradiometer data can be precomputed with **SimulateGradiometer**.

Name	Type	Annotation
sggRightSideType	sequence	
inputfileGradiometer	filename	observed gravity gradients
inputfileReferenceGradiometer	filename	precomputed gradients at orbit positions
referencefield	gravityfield	
tides	tides	

## 5.52 SphericalHarmonicsFilter

Filtering of a spherical harmonics expansion.

### 5.52.1 DDK

Orderwise filtering with the DDK filter by Kusche et al. 2009.

Name	Type	Annotation
level	uint	DDK filter level (1, 2, 3, ..., 8)
inputfileNormalEquation	filename	

### 5.52.2 Gauss

Filtering the spherical harmonics expansion with a Gaussian filter. **radius** gives the filter radius on the Earth surface in km.

Name	Type	Annotation
radius	double	filter radius [km]

### 5.52.3 Matrix

Filtering the spherical harmonics expansion with a matrix filter.

Name	Type	Annotation
inputfileMatrix	filename	
minDegree	uint	of matrix
maxDegree	uint	of matrix
numbering	sphericalHarmonicsNumbering	numbering scheme of the matrix

## 5.53 SphericalHarmonicsNumbering

This class organizes the numbering scheme of spherical harmonics coefficients in a parameter vector (e.g.  **Gravityfield2SphericalHarmonicsVector** and the design matrix of  **parametrizationGravity:sphericalHarmonics**).

### 5.53.1 Degree

Numbering degree by degree:

$c_{20}, c_{21}, s_{21}, c_{22}, s_{22}, c_{30}, c_{31}, s_{31}, c_{32}, s_{32}, \dots$

### 5.53.2 Order

Numbering order by order:

$c_{20}, c_{30}, c_{40}, \dots, c_{21}, s_{21}, c_{31}, s_{31}, \dots, c_{22}, s_{22}$

### 5.53.3 OrderNonAlternating

Numbering order by order with cnm, snm non-alternating:

$c_{20}, c_{30}, c_{40}, \dots, c_{21}, c_{31}, c_{41}, \dots, s_{21}, s_{31}, s_{41},$

### 5.53.4 File

Numbering as specified in the chosen file. The  **inputfile** is a matrix with the first column indicating cnm/snm with 0 or 1. The second and third column specify degree and order.

Name	Type	Annotation
 <b>inputfile</b>	filename	

## 5.54 SstRightSide

Observation vector for GRACE like data (satellite-tracking and precise orbit data (POD)) of  
 ▶ **observation** equations in a least squares adjustment. The observations are reduced by the effect of  
 ▶ **inputfileAccelerometer** and ▶ **forces** (observed minus computed).

Name	Type	Annotation
sstRightSideType	sequence	
inputfileSatelliteTracking	filename	ranging observations and corrections
inputfileOrbit1	filename	kinematic positions of satellite A as observations
inputfileOrbit2	filename	kinematic positions of satellite B as observations
inputfileAccelerometer1	filename	non-gravitational forces in satellite reference frame A
inputfileAccelerometer2	filename	non-gravitational forces in satellite reference frame B
forces	forces	

## 5.55 Thermosphere

This class provides functions for calculating the density, temperature and velocity in the thermosphere. The wind is computed by HWM14 model if **hwm14DataDirectory** is provided. A quiet thermosphere is assumed if **inputfileMagnetic3hAp** is not given.

### 5.55.1 JB2008

Thermosphere parameters from the JB2008 model:

Bowman, B. R., Tobiska, W. K., Marcos, F. A., Huang, C. Y., Lin, C. S., Burke, W. J. (2008). A new empirical thermospheric density model JB2008 using new solar and geomagnetic indices. In AIAA/AAS Astrodynamics Specialist Conference and Exhibit. <https://doi.org/10.2514/6.2008-6438>

Name	Type	Annotation
inputfileSolsmy	filename	solar indices
inputfileDtc	filename	
inputfileMagnetic3hAp	filename	indicies for wind model
hwm14DataDirectory	filename	directory containing dwm07b104i.dat, gd2qd.dat, hwm123114.bin

### 5.55.2 NRLMSIS2

Thermosphere parameters from the NRLMSIS2 model:

Emmert J.D, D.P.Drob, J.M. Picone, et al. (2020), NRLMSIS 2.0: A whole-atmosphere empirical model of temperature and neutral species densities. Earth and Space Science, Volume 8, 3 <https://doi.org/10.1029/2020EA001321>

Name	Type	Annotation
inputfileMsis	filename	input NRLMSIS 2.0
inputfileModelParameters	filename	path to msis20.parm file
inputfileMagnetic3hAp	filename	indicies for wind model
hwm14DataDirectory	filename	directory containing dwm07b104i.dat, gd2qd.dat, hwm123114.bin

## 5.56 Tides

This class computes functionals of the time depending tide potential, e.g potential, acceleration or gravity gradients.

If several instances of the class are given the results are summed up. Before summation every single result is multiplicated by a **factor**. To get the difference between two ocean tide models you must choose one factor by 1 and the other by -1. To get the mean of two models just set each factor to 0.5.

### 5.56.1 AstronomicalTide

This class computes the tide generating potential (TGP) of sun, moon and planets (Mercury, Venus, Mars, Jupiter, Saturn). It takes into account the flattening of the Earth (At the moment only at the acceleration level).

The computed result is multiplied with **factor**.

Name	Type	Annotation
useMoon	boolean	TGP of moon
useSun	boolean	TGP of sun
usePlanets	boolean	TGP of planets
useEarth	boolean	TGP of Earth
c20Earth	double	J2 flattening of the Earth
factor	double	the result is multiplied by this factor, set -1 to subtract the field

### 5.56.2 EarthTide

This class computes the earth tide according to the IERS2003 conventions. The values of solid Earth tide external potential Love numbers and the frequency dependent corrections of these values are given in the file **inputfileEarthTide**. The effect of the permanent tide is removed if **includePermanentTide** is set to false.

The computed result is multiplied with **factor**.

Name	Type	Annotation
inputfileEarthTide	filename	
includePermanentTide	boolean	results in FALSE: zero tide, TRUE: tide free gravity field
factor	double	the result is multiplied by this factor, set -1 to subtract the field

### 5.56.3 PoleTide

The potential coefficients of the solid Earth pole tide according to the IERS2003 conventions are given by

$$\begin{aligned}\Delta c_{21} &= s \cdot (m_1 + o \cdot m_2), \\ \Delta s_{21} &= s \cdot (m_2 - o \cdot m_1),\end{aligned}\tag{5.136}$$

with  $s$  is the **scale**,  $o$  is the **outPhase** and  $(m_1, m_2)$  are the wobble variables in seconds of arc. They are related to the polar motion variables  $(x_p, y_p)$  according to

$$\begin{aligned}m_1 &= (x_p - \bar{x}_p), \\ m_2 &= -(y_p - \bar{y}_p),\end{aligned}\tag{5.137}$$

The mean pole  $(\bar{x}_p, \bar{y}_p)$  is approximated by a polynomial read from **inputfileMeanPole**.

The displacement is calculated with

$$\begin{aligned} S_r &= -v \sin 2\vartheta(m_1 \cos \lambda + m_2 \sin \lambda), \\ S_\vartheta &= -h \cos 2\vartheta(m_1 \cos \lambda + m_2 \sin \lambda), \\ S_\lambda &= h \cos \vartheta(m_1 \sin \lambda - m_2 \cos \lambda), \end{aligned} \quad (5.138)$$

where  $h$  is the **horizontalDisplacement** and  $v$  is the **verticalDisplacement**.

The computed result is multiplied with **factor**.

Name	Type	Annotation
<b>scale</b>	double	
<b>outPhase</b>	double	
<b>inputfileMeanPole</b>	filename	
<b>horizontalDisplacement</b>	double	[m]
<b>verticalDisplacement</b>	double	[m]
<b>factor</b>	double	the result is multiplied by this factor, set -1 to subtract the field

#### 5.56.4 OceanPoleTide

The ocean pole tide is generated by the centrifugal effect of polar motion on the oceans. The potential coefficients of this effect is given by IERS2003 conventions are given by

$$\begin{Bmatrix} \Delta c_{nm} \\ \Delta s_{nm} \end{Bmatrix} = \begin{Bmatrix} c_{nm}^R \\ s_{nm}^R \end{Bmatrix} (m_1 \gamma^R + m_2 \gamma^I) + \begin{Bmatrix} c_{nm}^I \\ s_{nm}^I \end{Bmatrix} (m_2 \gamma^R - m_1 \gamma^I) \quad (5.139)$$

where the coefficients are read from file **inputfileOceanPole**,  $\gamma = \gamma^R + i\gamma^I$  is given by **gammaReal** and **gammaImaginary** and  $(m_1, m_2)$  are the wobble variables in radians. They are related to the polar motion variables  $(x_p, y_p)$  according to

$$\begin{aligned} m_1 &= (x_p - \bar{x}_p), \\ m_2 &= -(y_p - \bar{y}_p), \end{aligned} \quad (5.140)$$

The mean pole  $(\bar{x}_p, \bar{y}_p)$  is approximated by a polynomial read from **inputfileMeanPole**.

The computed result is multiplied with **factor**.

Name	Type	Annotation
<b>inputfileOceanPole</b>	filename	
<b>minDegree</b>	uint	
<b>maxDegree</b>	uint	
<b>gammaReal</b>	double	
<b>gammaImaginary</b>	double	
<b>inputfileMeanPole</b>	filename	
<b>factor</b>	double	the result is multiplied by this factor, set -1 to subtract the field

#### 5.56.5 DoodsonHarmonicTide

The time variable potential of ocean tides is given by a fourier expansion

$$V(\mathbf{x}, t) = \sum_f V_f^c(\mathbf{x}) \cos(\Theta_f(t)) + V_f^s(\mathbf{x}) \sin(\Theta_f(t)), \quad (5.141)$$

where  $V_f^c(\mathbf{x})$  and  $V_f^s(\mathbf{x})$  are spherical harmonics expansions and are read from the file **inputfileDoodsonHarmonic**. If set the expansion is limited in the range between **minDegree** and **maxDegree** inclusivly.  $\Theta_f(t)$  are the arguments of the tide constituents  $f$ :

$$\Theta_f(t) = \sum_{i=1}^6 n_f^i \beta_i(t), \quad (5.142)$$

where  $\beta_i(t)$  are the Doodson's fundamental arguments  $(\tau, s, h, p, N', p_s)$  and  $n_f^i$  are the Doodson multipliers for the term at frequency  $f$ .

The major constituents given by **inputfileDoodsonHarmonic** can be used to interpolate minor tidal constituents using the file **inputfileAdmittance**. This file can be created with **DoodsonHarmonicsCalculateAdmittance**.

After the interpolation step a selection of the computed constituents can be choosen by **selectDoodson**. Only these constituents are considered for the results. If no **selectDoodson** is set all constituents will be used. The constituents can be coded as Doodson number (e.g. 255.555) or as names introduced by Darwin (e.g. M2).

The computed result is multiplied with **factor**.

Name	Type	Annotation
<b>inputfileTides</b>	filename	
<b>inputfileAdmittance</b>	filename	interpolation of minor constituents
<b>selectDoodson</b>	doodson	consider only these constituents, code number (e.g. 255.555) or darwin name (e.g. M2)
<b>minDegree</b>	uint	
<b>maxDegree</b>	uint	
<b>nodeCorr</b>	uint	nodal corrections: 0-no corr, 1-IHO, 2-Schureman
<b>factor</b>	double	the result is multiplied by this factor, set -1 to subtract the field

## 5.56.6 Centrifugal

Computes the centrifugal potential in a rotating system

$$V(\mathbf{r}, t) = \frac{1}{2}(\omega(t) \times \mathbf{r})^2. \quad (5.143)$$

The current rotation vector  $\omega(t)$  is computed from the **earthRotation** provided by the calling program. The computed result is multiplied with **factor**.

Be careful, the centrifugal potential is not harmonic. Convolution with a harmonic kernel (e.g. to compute gravity anomalies) is not meaningful.

Name	Type	Annotation
<b>factor</b>	double	the result is multiplied by this factor, set -1 to subtract the field

## 5.56.7 SolidMoonTide

This class computes the solid moon tide according to the IERS2010 conventions. The values of solid Moon tide external potential Love numbers are given and there are no frequency dependent corrections of these values. The computed result is multiplied with **factor**.

Name	Type	Annotation
▶ k20	double	
▶ k30	double	
▶ factor	double	the result is multiplied by this factor, set -1 to subtract the field

## 5.56.8 Group

Groups a set of ▶ [tides](#) and has no further effect itself.

Name	Type	Annotation
▶ tides	<a href="#">tides</a>	
▶ factor	double	the result is multiplied by this factor

## 5.57 TimeSeries

This class generates a series of points in time. The series is always sorted in ascending order. Depending of the application the series is interpreted as list of points or as intervals between the points.

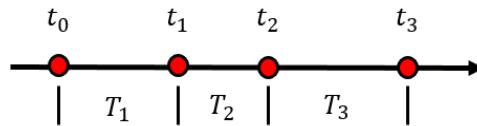


Figure 5.15: List of points  $t_i$  vs. intervals  $T_i$ .

### 5.57.1 UniformSampling

Generates a time series with uniform sampling. The first point in time will be **timeStart**. The last generated point in time will be less or equal **timeEnd**. The time step between generated points in time is given by **sampling**.

Name	Type	Annotation
timeStart	time	first point in time
timeEnd	time	last point in time will be less or equal timeEnd
sampling	time	time step between points in time

### 5.57.2 UniformInterval

Generates a time series with uniform sampling between **timeStart** and **timeEnd**. **intervalCount** gives the count of intervals. This class generates  $\text{count}+1$  points in time inclusive **timeStart** and **timeEnd**.

Name	Type	Annotation
timeStart	time	1st point of the time series
timeEnd	time	last point of the time series
intervalCount	uint	count of intervals, $\text{count}+1$ points in time will generated

### 5.57.3 Irregular

The points of the time series are given explicitly with **time**.

Name	Type	Annotation
time	time	explicit list of points in time

### 5.57.4 Monthly

If **monthMiddle** is set, time points are generated at mid of each month inclusively the **monthStart** in **yearStart** and **monthEnd** in **yearEnd**. Otherwise times are given at the first of each month and a time point after the last month.

Name	Type	Annotation
monthStart	uint	
yearStart	uint	
monthEnd	uint	
yearEnd	uint	
useMonthMiddle	boolean	time points are mid of months, otherwise the 1st of each month + a time point behind the last month

### 5.57.5 Yearly

If **yearMiddle** is set, time points are generated at mid of each year inclusively **yearStart** and **yearEnd**. Otherwise times are given at the first of each year and a time point after the last year.

Name	Type	Annotation
yearStart	uint	
yearEnd	uint	
useYearMiddle	boolean	time points are mid of years, otherwise the 1st of each year + a time point behind the last year

### 5.57.6 EveryMonth

Generates a time series with monthly sampling. The first point in time will be **timeStart** and the following points are generated for each month at the same day and time in month. The last generated point in time will be less or equal **timeEnd**.

Name	Type	Annotation
timeStart	time	first point in time
timeEnd	time	last point in time will be less or equal timeEnd

### 5.57.7 EveryYear

Generates a time series with yearly sampling. The first point in time will be **timeStart** and the following points are generated for each year at the same day and time in year. The last generated point in time will be less or equal **timeEnd**.

Name	Type	Annotation
timeStart	time	first point in time
timeEnd	time	last point in time will be less or equal timeEnd

### 5.57.8 Instrument

Read a time series (epochs) from an **instrument file**. The time series can be restricted to the interval starting from **timeStart** and before **timeEnd**.

Name	Type	Annotation
inputfileInstrument	filename	
timeStart	time	exclude epochs before this epoch
timeEnd	time	only epochs before this time are used

### 5.57.9 InstrumentArcIntervals

Reconstruct a time series from an [Instrument file](#). The time series is the first epoch of each arc plus one time step beyond the last epoch of the last arc (using median sampling).

Name	Type	Annotation
inputfileInstrument	filename	Must be regular. Time series is first epoch of each arc plus one time step extrapolated from last epoch of last arc.

### 5.57.10 Revolution

Reads an [Orbit file](#) and create a time stamp for each ascending equator crossing. The time series can be restricted to the interval starting from [timeStart](#) and before [timeEnd](#).

Name	Type	Annotation
inputfileOrbit	filename	
timeStart	time	exclude epochs before this epoch
timeEnd	time	only epochs before this time are used

### 5.57.11 Exclude

In a first step a [timeSeries](#) is generated. In a second step all times are removed which are in range before or after [excludeMargin](#) seconds of the times given by [timeSeriesExclude](#).

Name	Type	Annotation
timeSeries	timeSeries	time series to be created
timeSeriesExclude	timeSeries	exclude this time points from time series (within margin)
excludeMargin	double	on both sides [seconds]

### 5.57.12 Conditional

Only times for which the [condition](#) is met are included in the time series. The [variableLoopTime](#) is set to every time and the [condition](#) is evaluated.

Name	Type	Annotation
timeSeries	timeSeries	only times for which condition is met will be included
variableLoopTime	string	variable with time of each loop
condition	condition	test for each time

### 5.57.13 Interpolate

Interpolates [nodeInterpolation](#) count points between the given [timeSeries](#) uniformly.

Name	Type	Annotation
timeSeries	timeSeries	time series to be created
nodeInterpolation	uint	interpolates count points in each time interval given by the time series

## 5.58 Troposphere

This class provides functions for calculating and estimating the signal delay in the dry and wet atmosphere.

### 5.58.1 ViennaMapping

Tropospheric delays based on the Vienna Mapping Functions 3 (VMF3) model (Landskron and Boehm 2017, DOI: [10.1007/s00190-017-1066-2](https://doi.org/10.1007/s00190-017-1066-2)).

Hydrostatic and wet mapping function coefficients ( $a_h$ ,  $a_w$ ) and zenith delays (ZHD, ZWD) have to be provided via **inputfileVmfCoefficients**. This file can contain either station-specific data (see **ViennaMappingFunctionStation2File**) or data on a regular global grid (see **ViennaMappingFunctionGrid2File**). In the second case mapping coefficients and zenith delays are interpolated to the requested coordinates. This includes a height correction that requires approximate meteorological data provided via **inputfileGpt**.

Name	Type	Annotation
<b>inputfileVmfCoefficients</b>	filename	ah, aw, zhd, zwd coefficients
<b>inputfileGpt</b>	filename	gridded GPT data
<b>aHeight</b>	double	parameter a (height correction)
<b>bHeight</b>	double	parameter b (height correction)
<b>cHeight</b>	double	parameter c (height correction)

### 5.58.2 GPT

Tropospheric delays based on the Global Pressure and Temperature 3 (GPT3) model (Landskron and Boehm 2017, DOI: [10.1007/s00190-017-1066-2](https://doi.org/10.1007/s00190-017-1066-2)).

It is an empirical model derived from the Vienna Mapping Functions 3 (VMF3, see **viennaMapping**) and thus does not require additional mapping coefficients and zenith delay values.

Name	Type	Annotation
<b>inputfileGpt</b>	filename	gridded GPT data
<b>aHeight</b>	double	parameter a (height correction)
<b>bHeight</b>	double	parameter b (height correction)
<b>cHeight</b>	double	parameter c (height correction)