Australian Government
Geoscience Australia

GINAN

Using the toolkit

Ginan documentation

March, 2022

Document approvals

Role	Name	Signature	Date
Prepared:	John Donovan	–	18-3-2022
Prepared:	Ken Harima	–	18-3-2022
Approved:	Simon McClusky	–	18-3-2022

Document history

Version	Dated	Author	Notes
Beta	18-3-2022	John Donovan, Ken Harima	Ginan Beta release
–	–	–	–

Contents

1 Introduction

1.1 The Positioning Australia Program

The Australian Government is making a significant investment in the Positioning Australia program through Geoscience Australia. The program contains three major projects:

- The commercial procurement and operation of a Satellite Based Augmentation System (SBAS) called SouthPAN which will enhance positioning across the region through the provision of extra GNSS signals and data delivered from a geostationary satellite.
- The enhancement of the National Positioning Infrastructure Capability (NPIC) which will see upgrades to and an expansion of the Global Navigation Satellite System (GNSS) Continuously Operating Reference Station (CORS) network across the South Pacific and Antarctica.
- Ginan is an open source Precise Point Positioning (PPP) toolkit. It can produce PPP position correction products and, operating in another mode, use GNSS observations and those correction products to determine positions with an accuracy in the centimetre range.

The program is summarised in Figure 1.1 below.

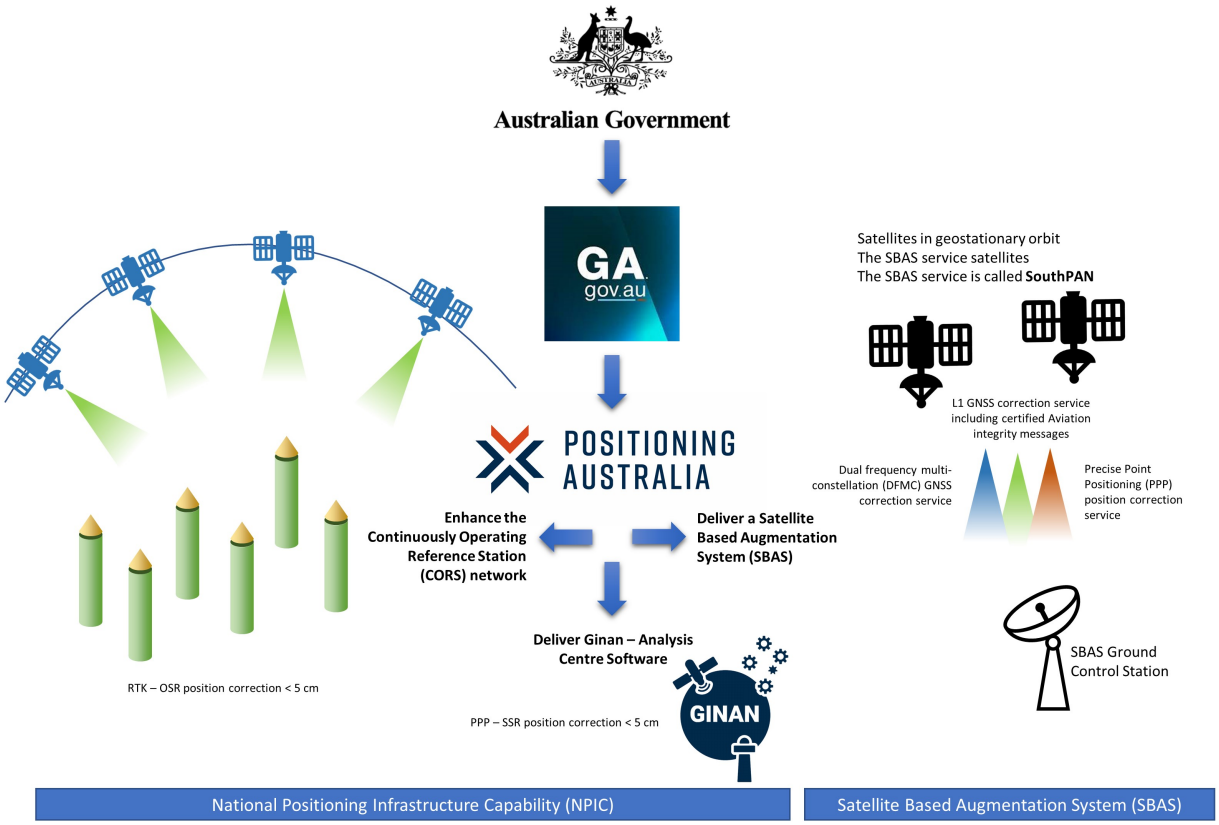

Figure 1.1: The three main projects in the Positioning Australia program.

1.2 Ginan - Analysis Centre Software

Ginan, is being rolled out in a phased approach and will offer products in four distinct categories:

- The software itself. Ginan is open-source software that GA has hosted on a GitHub repository.
- Standard precise point positioning (PPP) product files. An operational version of Ginan, maintained by Geoscience Australia (GA), will produce on a 24 X 7 basis, a range of standard PPP product files including, for example, a precise orbits and clocks file in SP3 format.
- Precise point positioning correction messages. An operational version of Ginan, maintained by GA, will stream over the internet on a 24 X 7 basis, a range of PPP correction messages in the RTCM3 message format.
- New PPP products and applications yet to be defined. The Ginan toolkit gives GA the ability to offer new PPP products, yet to be defined, but which, in collaboration with users, may spawn new applications and commercial opportunities.

Ginan is summarised in Figure 1.2 below.

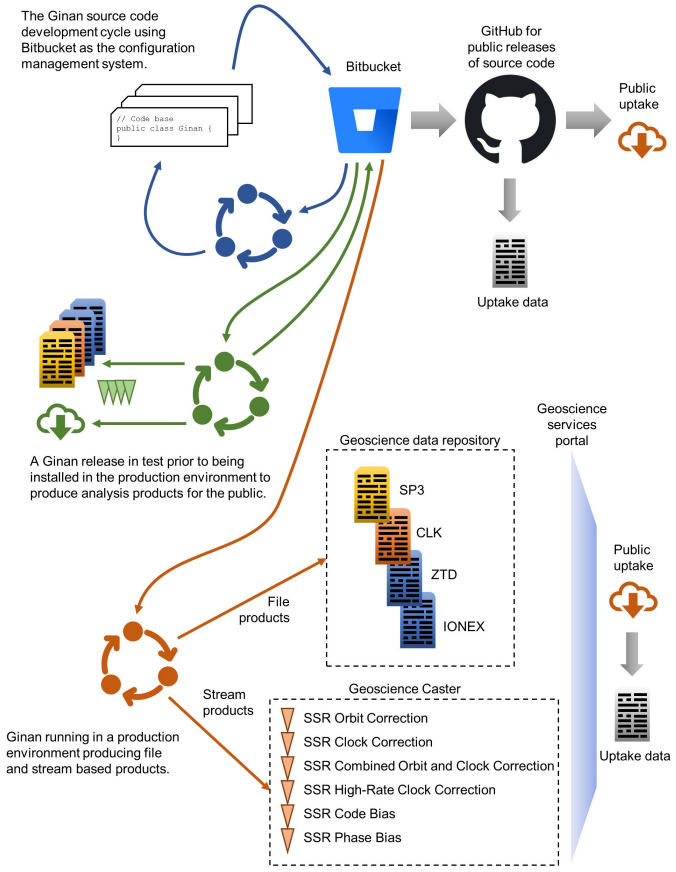

Figure 1.2: The Ginan product offering.

1.3 This document - POD and PEA

This document forms part of the Ginan documentation suite. The Ginan software toolkit consist of two main components. The *precise orbit determination* (POD) component estimates precise satellite position and orbital parameters, while the *parameter estimation algorithm* (PEA) monitors systematic biases associated with GNSS signals. The Ginan software uses *precise point positioning*(PPP) techniques to process GNSS signals. PPP was originally developed as a GNSS based positioning method for calculating location of autonomous receivers with high levels of accuracy and precision. PPP aims to calculate the end user position by rigorously modelling and/or estimating error sources in GNSS measurements. The systematic of errors in GNSS signals can be summarised as:

- Satellite state estimation errors: position, clock offset, hardware biases, antenna effects
- Receiver state estimation errors: position, clock offset, hardware biases, antenna effects
- Atmospheric effects: ionospheric propagation delay, tropospheric propagation delay
- Other (model-able) environmental effects: Relativistic corrections, phase windup

The various components of the Ginan software toolkit are designed to model or estimate these errors as parameters. Figure 1.3 below illustrates the way the Ginan components interact to estimate these parameters.

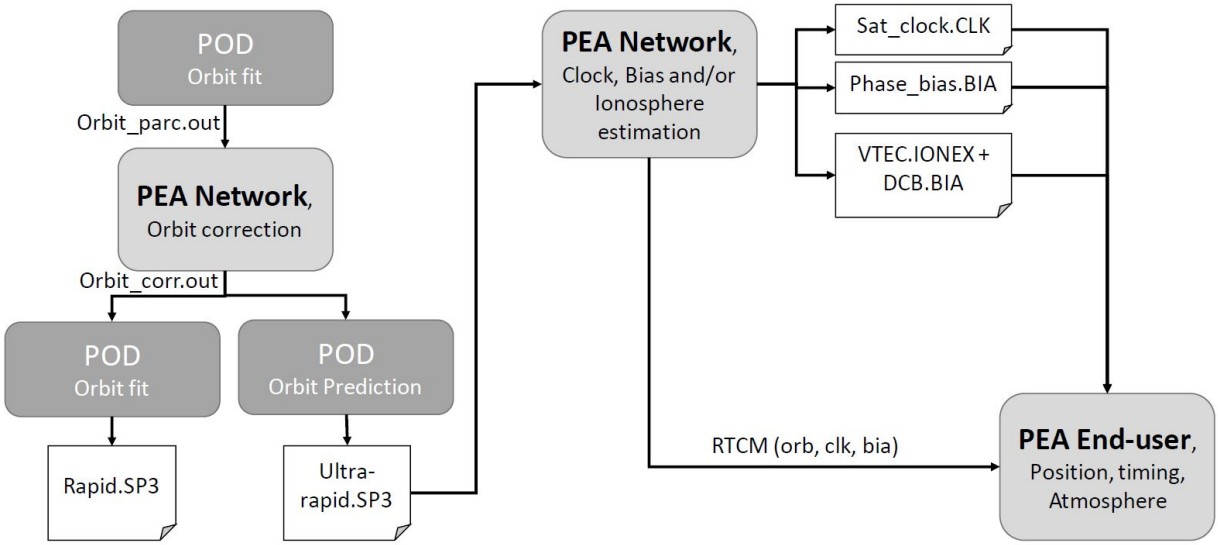

Figure 1.3: Ginan software components

In the example illustrated by Figure 1.3:

1. The POD in orbit fitting mode is used to calculate an a-priori position and the linearization partials of orbit parameters
2. The PEA, in network mode, estimate orbital parameters from orbit partials

3. The POD, use the orbital parameters to estimate and predict precise satellite positions
4. The PEA, in network mode, is used to estimate wide-area parameters: satellite clock offsets, satellite hardware bias and atmospheric delays
5. The PEA, in end-user mode, is used to calculate local parameters like receiver position, receiver clock offset and local atmospheric delays

Other parameters, such as antenna, phase windup and relativistic effects are calculated from predefined models.

2 Using the POD Module

POD uses configuration files in YAML to control its processing. After installing the software and its dependencies (see the README.md in the GitHub repository) and compiling building the POD application, POD processing can be started by typing the command.

```
./pod -y <path_to_config_file>
```

Details on the configuration parameters included in YAML files can be found in chapter 4. Configuration files corresponding to the examples in this section can be found in the *ginan/examples* directory.

The POD module has two main modes of operation, the orbit fitting mode and the orbit integration/prediction mode. In orbit fitting mode, precise orbit parameters are calculated from, potentially inaccurate, satellite position pseudo-observations. In orbit integration mode, precise satellite positions are estimated/predicted from precise orbit parameters.

2.1 Using the POD for orbit fitting

The orbit fitting mode can be selected by setting the *pod_mode_fit* to true and *ic_input_format: sp3* to true. In this mode, the POD will take satellite position pseudo-measurements from a SP3 formatted file and estimate the orbit state of each satellite contained in the SP3 file. The SP3 file containing a priory satellite position needs to be specified as the *pseudobs_orbit_filename* parameter. The orbit state in POD is represented by a set of parameters consisting of

- Satellite position (in ITRF or ICRF) at the first epoch in the SP3 file
- Satellite velocity (in ITRF or ICRF) at the first epoch in the SP3 file
- Up to 9 parameters describing the Solar Radiation Pressure over the fitting time

These initial conditions, and the models described in Ginan Science Manual will allow for the precise determination of satellite positions over the fitting arc (set by the *orbit_arc_determination* parameter).

The main outputs from this mode of operation are the a-posteriori satellite position in SP3 format, and the orbit partials of satellite positions with respect to the initial conditions. The output SP3 file which can be found on *output_directory/gagWWWWD.sp3* where *WWWW* is the GPS week and *D* is the GPS day of the first epoch on the SP3 files. The orbit partials are written in Ginan's proprietary Initial Conditions File (ICF) format, and can be found in *output_directory/gagWWWWD_orbit_partials.out*. Configuration files, *ex21_pod_fit_gps.yaml* and *ex22_pod_fit_gnss.yaml*, for this mode of operation are included in the Ginan *examples* folder.

2.2 Using the POD for orbit integration/prediction

The orbit fitting mode can be selected by setting the *pod_mode_ic_int* to true and *ic_input_format_icf* to true. In this mode, the POD will take the initial conditions contained in the ICF formatted files and propagates the satellite positions forward over the time period specified by the sum of *orbit_arc_determination* and *orbit_arc_prediction* parameters. It also propagates the satellite position backwards by a number of hours specified by the *orbit_arc_backwards* parameter. The ICF file containing the satellites initial condition and radiation pressure parameters needs to be specified as the *ic_input_format_ic_filename* parameter.

It is to note that the orbit fitting mode will also use the orbit integration operation after estimating the initial conditions from pseudo-observations. The mode *pod_mode_fit* will only integrated for a number of hours specified by *orbit_arc_determination*, but will also do the backwards arc specified by *orbit_arc_backwards* and a prediction arc specified by *orbit_arc_prediction*. Selecting the *pod_mode_predict* will propagate the initial conditions a number of hours specified by the sum of *orbit_arc_determination* and *orbit_arc_prediction*. The integrated/predicted satellite position will be output to a SP3 formatted file located in *output_directory/gagWWWWD.sp3*.
The example configuration file to perform orbit integration/prediction from SP3 files is *ex23_pod_prd_gps.yaml*. The example configuration file to perform orbit integration/prediction from ICF files is *ex24_pod_ic_gps.yaml*. Both are located in the Ginan *examples* folder.

3 POD Examples

3.1 Processing Example 1

In this example the pod will perform a dynamic orbit determination for the GPS constellation over a 24 hour arc. The full gravitational force models are applied, with a cannonball model SRP model.

To run the POD, change to the Ginan examples directory, then ...

```
../bin/pod -y ex21_pod_fit_gps.yaml
```

This should output the following to stdout (first satellite (G01) only shown)

```
  PRN: G01, SVN:  63, BLK TYP: GPS-IIF, BLKID:   8, TX PWR: 240, MASS:
    1633.000
 IC:    57842.000000000000        0.0000000000000000
   13449938.290000001       -15187647.773999998        16895359.906999998
      -59.097715290263295        2181.6586220953614        2021.8495117835701
 Orbit Determination
 day of year        2017          89 G01  beta   58.757853105247371
 ECOM1 SRP MODEL IS ACTIVATED
Orbit residuals: ICRF
 RMS-XYZ ICRF FIT G01        0.0050        0.0035        0.0031
External Orbit comparison
Orbit comparison: ICRF
 RMS-RTN ICRF CMP G01        0.0037        0.0035        0.0046
 RMS-XYZ ICRF CMP G01        0.0050        0.0035        0.0031
Orbit comparison: ITRF
 RMS-XYZ ITRF CMP G01        0.0043        0.0044        0.0031

The results above show that our orbits arcs, over 24 hours, are currently
   within 0.5 cm of the final combined IGS orbit.
```

The processing also produces the following output files in the yaml specified output directory (ex21 in this case)

```
POD.status        processing report file
gagWWWWD.erp      Earth Rotation Parameters file
gagWWWWD.obx      ORBEX file resulting from the integration
gagWWWWD.sp3      sp3 file from all integration steps (backwards, fitting, predicti
gagWWWWD_orbits_partials.out     output IC file for pea
gagWWWWD_igsWWWWD_orbdiff_rtn.out      differences in solutions in orbital frame co
gagWWWWD_igsWWWWD_orbitstat_(R|T|N).out      statistical differences in solutions i
```

3.2 Processing Example 2 - ECOM2 SRP

In this example we will change the SRP model to use the ECOM2 model.

this yaml file has ECOM2 selected (see the SRP model selection). It runs over all the gnss constellations, and will take about 10 minutes to run.

```
bin/pod -y ex22_pod_fit_gnss.yaml
```

This should output the following to stdout (G02 output selected)

```
  PRN: G02, SVN:  61, BLK TYP: GPS-IIR-B, BLKID:   6, TX PWR:  60, MASS:
    1080.000
 IC:    58682.000000000000        0.0000000000000000
   20561552.770000000       -13147270.275999999       -9692254.6860000007
      -649.18898038938642        1190.7695703040808       -2831.8342814706266
 Orbit Determination
 day of year        2019         199 G02  beta  -22.344868611303600
 ECOM2 SRP MODEL IS ACTIVATED
Orbit residuals: ICRF
 RMS-XYZ ICRF FIT G02        0.0055        0.0035        0.0035
External Orbit comparison
Orbit comparison: ICRF
 RMS-RTN ICRF CMP G02        0.0043        0.0029        0.0052
 RMS-XYZ ICRF CMP G02        0.0055        0.0035        0.0035
Orbit comparison: ITRF
 RMS-XYZ ITRF CMP G02        0.0042        0.0050        0.0035
```

3.3 Example 3 - (examples/ex23_pod_prd_gps.yaml)

GPS IGS SP3 file orbit fitting, orbit prediction and comparison to next IGS SP3 file

3.4 Example 4 - (examples/ex24_pod_ic_gps.yaml):

Integration of POD initial conditions file generated by the PEA.

3.5 Example 5 - (examples/ex25_pod_fit_gps.yaml):

ECOM1+ECOM2 hybrid SRP model

3.6 Example 6 - (examples/ex26_pod_fit_meo.yaml):

week long integration of a middle earth orbit satellite (L51)

4 YAML Configuration for POD

The YAML configuration file for POD allows you to specify how and what data the POD will process and what results and statistics to report at the end. In order to use the yaml configuration file you will need to specify this at the command line, with the *-y <yaml_filename>* otherwise it will default to the (no longer supported) traditional *POD.in, EQM.in and VEQ.in* option files.

```
pod -y example_configuration.yaml
```

Listing 4.1: calling the yaml configuration file

4.1 POD processing options (pod_options)

These options will control how the pod will process the input files, with four different options available. Only one of the options listed below can be set to true, the remainder must be set to false.

Option	Values	Comments
pod_mode_fit	true or false	Orbit Determination (pseudo-observations; orbit fitting)
pod_mode_predict	true or false	Orbit Determination and Prediction
pod_mode_eqm_int	true or false	Orbit Integration (Equation of Motion only)
pod_mode_ic_int	true or false	Orbit Integration and Partials (Equation of Motion and Variational Equations) initial condition integration

Table 4.1: POD YAML: processing options

```
pod_options:
# Example YAML showing different processing options
#-------------------------------------------------
pod_mode_fit:     true
pod_mode_predict: false
pod_mode_eqm_int: false
pod_mode_ic_int:  false
```

Listing 4.2: pod_options yaml configuration example

1. **pod_mode_fit** - this is used to fit an existing sp3 file (this is sometimes referred to as pseudo observations) with the parameters that are set later on. See pod example 1
2. **pod_mode_predict** - determine an orbit from observations and then predict the orbits path
3. **pod_mode_eqm_int** - determine the equations of motion only
4. **pod_mode_ic_int** -set up the initial conditions

4.2 Time scale(time_scale)

Option	Values	Comments
TT_time	true or false	Terrestrial (TT)
UTC_time	true or false	Universal (UTC)
GPS_time	true or false	Satellite (GPS)
TAI_time	true or false	Atomic (TAI)

Table 4.2: POD YAML: Time scale options

```
    time_scale:
        TT_time:  false
        UTC_time: false
        GPS_time: true
        TAI_time: false
```

Listing 4.3: time_scale yaml configuration example

4.3 Initial Conditions (IC)

4.3.1 IC input format (ic_input_filename)

one is true, the other is false. If icf selected the ic_filename value specifies the path to the file.

```
    ic_input_format:
        sp3:  true   # Input a-priori orbit in sp3 format
        icf:  false  # Input a-priori orbit in POD Initial Conditions File (
    ICF) format
        ic_filename: some_file
```

Listing 4.4: ic_input_format yaml configuration example

Option	Values	Comments
sp3	true or false	sp3 format file
icf	true or false	initial conditions file

Table 4.3: POD YAML: Initial Conditions input format options

4.3.2 IC input reference system (ic_input_refsys)

reference system for the initial conditions one is true, the other is false.

Option	Values	Comments
itrf	true or false	terestrial
icrf	true or false	celestial
kepler	true or false	polar form of celestial

Table 4.4: POD YAML: Initial Conditions reference system

```
1    ic_input_refsys:
2         itrf: true   # Initial Conditions Reference Frame: ITRF, ICRF
3         icrf: false  # Initial Conditions Reference Frame: ITRF, ICRF
4         kepler: false
```

Listing 4.5: ic_input_refsys yaml configuration example

4.4 Using Pseudo observations

These options are used to control how pseudo observations are used by the POD.

Option	Values	Comments
pseudobs_orbit_filename	filename	*path to the observations file*
pseudobs_interp_step	int	Interval (sec) of the interpolated orbit
pseudobs_interp_points	int	Number of data points used in Lagrange interpolation (at least 2 but recommended 6 to 12)
IC_time	datetime	YYYY MM DD hh mm ss.ss

Table 4.5: POD YAML: Using pseudo observations

```
1     pseudobs_orbit_filename: igs19424.sp3 # Pseudo observations orbit
     filename
2     pseudobs_interp_step:    900          # Interval (sec) of the
     interpolated orbit
3     pseudobs_interp_points:  12           # Number of data points used in
     Lagrange interpolation
4     IC_time: 2017 03 10 03 0 0.0
```

Listing 4.6: pseudo observation model yaml configuration example

4.5 Orbit arc length

```
1 # Orbit arc length (in hours)
2     orbit_arc_determination: 24  # Orbit Estimation arc
3     orbit_arc_prediction:    12  # Orbit Prediction arc
4     orbit_arc_backwards:     2   # Orbit Propagation backwards arc
```

Listing 4.7: orbit arc length yaml configuration example

4.6 External Orbit Comparison

In this section only one of the following options listed below can be set to true, the remainder must be set to false.

```
1
2 # External Orbit Comparison
3     ext_orbit_enabled: true
4     ext_orbit_type_sp3:       false        # Orbit data in sp3 format
```

Option	Values	Comments
orbit_arc_determination	int	*number of hours to integrate*
orbit_arc_prediction	int	*number of hours to predict at end of orbit arc*
orbit_arc_backwards	int	*number of hours to check before start of orbit arc*

Table 4.6: POD YAML: Orbit arc options

```
                                        # (including position and velocity
    vectors)
    ext_orbit_type_interp:    true      # Interpolated orbit based on
   Lagrange
                                        # interpolation of sp3 file
    ext_orbit_type_kepler:    false     # Keplerian orbit
    ext_orbit_type_lagrange:  false     # 3-day Lagrange interpolation
    ext_orbit_type_position_sp3:    false  # Position and SP3 file
    ext_orbit_filename:       igs19424.sp3 # External (comparison) orbit
   filename
    ext_orbit_interp_step:    900       # Interval (sec) of the
   interpolated/Kepler orbit
    ext_orbit_interp_points:  12        # Number of data points used
                                        # in Lagrange interpolation
```

Listing 4.8: orbit arc length yaml configuration example

4.6.1 External orbit reference frame (ext_orbit_frame)

```
   ext_orbit_frame:
        itrf: true            # External orbit reference frame - ITRF
        icrf: false           # External orbit reference frame - ICRF
        kepler: false
```

Listing 4.9: external orbit reference frame yaml configuration example

Option	Values	Comments
ext_orbit_enabled	true or false	
ext_orbit_type_sp3	true or false	
ext_orbit_type_interp	true or false	
ext_orbit_type_kepler	true or false	
ext_orbit_type_lagrange	true or false	
ext_orbit_type_position_sp3	true or false	
ext_orbit_filename	filename	*path to the orbit file*
ext_orbit_interp_step	int	Interval (sec) of the interpolatedKepler orbit
ext_orbit_interp_points	int	Number of data points used in Lagrange interpolation (at least 2 but recommended 6 to 12)

Table 4.7: POD YAML: External orbit options

Option	Values	Comments
itrf	true or false	terrestrial
icrf	true or false	celestial
kepler	true or false	kepler orbital elements

Table 4.8: POD YAML: External orbit reference system

4.7 Earth Orientation Parameters

In this section only one of the following options listed below can be set to true, the remainder must be set to false.

4.7.1 EOP type

Option	Values	Comments
EOP_soln_c04	true or false	C04 is the IERS solution
EOP_soln_rapid	true or false	Rapid is the rapidprediction center solution
EOP_soln_igs	true or false	igs is the ultra-rapid solution using partials. To use this you need both the rapid file and partials file.
EOP_soln_c04_file	filename	
EOP_soln_rapid_file	filename	
ERP_soln_igs_file	filename	
EOP_soln_interp_points	int	

Table 4.9: POD YAML: Earth Orientation Parameter solution options

```
1    EOP_soln_c04:   true         # IERS C04 solution : EOP_sol = 1
2    EOP_soln_rapid: false        # IERS rapid service/prediction center (RS/
    PC) Daily : EOP_sol = 2
3    EOP_soln_igs:   false        # IGS ultra-rapid ERP + IERS RS/PC Daily (dX
    ,dY) : EOP_sol = 3. Need both rapid_file AND igs_file
4    EOP_soln_c04_file: eopc04_14_IAU2000.62-now
5    EOP_soln_rapid_file: finals2000A.daily
6    ERP_soln_igs_file: igu18543_12.erp
7    EOP_soln_interp_points: 4    # EOP solution interpolation points
```

Listing 4.10: eop estimation options

4.7.2 IAU Precession-Nutation model

Option	Values	Comments
eop_soln_interp_points	int	number of data points to be used in an eop interpolation (at least 2!)
iau_model_2000	true or false	
iau_model_2006	true or false	

Table 4.10: POD YAML: EOP model options

```
    # IAU Precession-Nutation model:
    iau_model_2000: true         # IAU2000A: iau_pn_model = 2000
    iau_model_2006: false        # IAU2006/2000A: iau_pn_model = 2006

```

Listing 4.11: Precession model

4.8 Input files

Option	Values	Comments
gravity_model_file	filename	
DE_fname_header	filename	Emphemeris header file
DE_fanme_data	filename	Emphemeris data file
ocean_tides_model_file	filename	
leapsec_filename	filename	leapseconds to be added
satsinex_filename	filename	sinex file with satellite meta-data

Table 4.11: POD YAML: Input files

```
# Gravity model file
gravity_model_file: goco05s.gfc
```

```
# goco05s.gfc, eigen-6s2.gfc, ITSG-Grace2014k.gfc

# Planetary/Lunar ephemeris - JPL DE Ephemeris
DE_fname_header: header.430_229
DE_fname_data:   ascp1950.430

# Ocean tide model file
ocean_tides_model_file: fes2004_Cnm-Snm.dat
# FES2004 ocean tide model

# Leap second filename
leapsec_filename: leap.second

# Satellite metadata SINEX
satsinex_filename: igs_metadata_2063.snx
```

Listing 4.12: yaml example for general input files

4.9 Output options

Option	Values	Comments
sp3_velocity	true or false	if you wish to write out the velocities for comparison
partials_velocity	true or false	if you wish to write velocity vector partials to the output file

Table 4.12: POD YAML: Output options

```
# Write to sp3 orbit format: Option for write Satellite Velocity vector
sp3_velocity: false        #  Write Velocity vector to sp3 orbit

#--------------------------------------------------------------------
# Write partials of the velocity vector w.r.t. parameters into the
   orbits_partials output file:
partials_velocity: false   # Write out velocity vector partials wrt orbital
   state vector elements
```

Listing 4.13: yaml example for output file optionss

4.10 Variational Equation Options

Option	Values	Comments
veq_integration	true or false	pod mode overides it anyway. Ignore.
ITRF	true or false	reference_frame
ICRF	true or false	one must be true
kepler	true or false	

Table 4.13: POD YAML: VEQ ref system

```
# Variational Equations
veq_integration: false

#-----------------------------------------------------------------------
# Reference System for Variational Equations'  - Partials & Parameter
    Estimation
veq_refsys:
itrs: true      # ITRS: Terrestrial Reference System
icrs: false     # ICRS: Celestial Reference System
kepler: false
```

Listing 4.14: yaml example for variational equation options

4.11 General Options

Option	Values	Comments
estimator_iterations	int	integrate this number of times, using the generated initial conditions from the previous run as a start point

Table 4.14: POD YAML: general options

```
    # Parameter Estimation
    estimator_iterations: 2
```

Listing 4.15: yaml example for output file options

Option	Values	Comments
no_model	true or false	point source only
cannon_ball_model	true or false	see **??**
simple_boxwing_model	true or false	as stated
full_boxwing_model	true or false	as stated

Table 4.15: POD YAML: Apriori SRP model

4.12 Apriori solar radiation models

```
srp_apriori_model:
    no_model:   false
    cannon_ball_model:   true
    simple_boxwing_model:   false
    full_boxwing_model:     false
```

Listing 4.16: yaml example for apriori srp model options

4.12.1 Estimated Solar radiation models

Option	Values	Comments
ECOM1	true or false	ECOM1 params only
ECOM2	true or false	ECOM2 params only
hybrid	true or false	any mix of ECOM1 and ECOM2
SBOXW	true or false	Simple box wing model
EMPirical models	true or false	Empirical is independent of the other four

Table 4.16: POD YAML: Estimated SRP models

```
srp_apriori_model:
    no_model:               false
    cannon_ball_model:      true
    simple_boxwing_model:   false
```

```
    full_boxwing_model:     false
```

Listing 4.17: yaml example for apriori srp model options

4.12.2 gravity_model

Type of gravity model to apply, only one option can be true.

Option	Values	Comments
central_force	true or false	
static_gravity_model	true or false	
time_variable_model	true or false	
iers_geopotential_model	true or false	

Table 4.17: POD YAML: Gravity Models

```
gravity_model:
    central_force:           false   # Central force gravity field
      : gravity_model = 0
    static_gravity_model:    false   # Static global gravity field model
      : gravity_model = 1
    time_variable_model:     true    # Time-variable global gravity field
   model : gravity_model = 2
    iers_geopotential_model: false   # IERS conventional geopotential model
      : gravity_model = 3
```

Listing 4.18: yaml example for gravitational force model options

4.12.3 stochastic pulse (pulse)

Do not mix pulses in R/T/N (terrestrial) with pulses in (X/X/Z)

```
pulse:
enabled:      false
epoch_number:     1    # number of epochs to apply pulses
offset:       43200    # since the start of day
interval:     43200    # repeat every N seconds
directions:
    x_direction:   true
    y_direction:   true
    z_direction:   true
    r_direction:   false
    t_direction:   false
    n_direction:   false
reference_frame:
    icrf:          true
    orbital:       false
```

Listing 4.19: yaml example for gravitational force model options

Option	Values	Comments
enabled	true or false	then if true:
epoch_number	int	number of epochs to apply pulses each day
offset	int	seconds until the first pulse of the day
interval	int	seconds between each pulse (after the first)
directions		
x_direction	true or false	
y_direction	true or false	
z_direction	true or false	
r_direction	true or false	
t_direction	true or false	
n_direction	true or false	

Table 4.18: POD YAML:stochastic pulse options

4.13 Inclusions/Exclusions

The pod can be configured to only do certain constellations, and to include or exclude selected PRNs. For constellations just have true or false after each.
Inclusions/exclusions use the same format, and limit to (or exclude) the stated PRNs in the given list

```
# optional limiting to constellations
#-------------------------------------------------------------------
   constellations:
      GPS:      false
      GALILEO:  false
      GLONASS:  true
      BEIDOU:   false
      QZSS:     false
```

Listing 4.20: yaml example for constellation limiting

```
# optional limiting to PRNs
#-------------------------------------------------------------------
   prn_inclusions (or prn_exlusions):
      - G01
      - E01
      - G23
      - R22
```

Listing 4.21: yaml example for prn inclusions/exclusions

4.14 EQM/VEQ options

You can set different parameters for EQM and VEQ, but the sections labels are the same.

4.14.1 Integration Step

Option	Values	Comments
RK4_integrator_method	true or false	Do not use RK4 for veq as it is not implemented
RKN7_integrator_method	true or false	only one can be true
RK8_integrator_method	true or false	
integrator_step	int	step size in seconds

Table 4.19: POD YAML: Integration Step models

```
# Numerical integration method
# Runge-Kutta-Nystrom 7th order RKN7(6): RKN7, Runge-Kutta 4th order: RK4,
   Runge-Kutta 8th order RK8(7)13: RK8
integration_options:
    RK4_integrator_method: false
    RKN7_integrator_method: true
    RK8_integrator_method: false
    integrator_step: 900         # Integrator stepsize in seconds
```

Listing 4.22: yaml example for integration options

4.14.2 Gravity Field

Option	Values	Comments
enabled	true or false	and if true:
gravity_degree_max		maximum model terms in spherical harmonic expansion
timevar_degree_max		maximum time variable model terms in spherical harmonic expansion

Table 4.20: POD YAML: Gravity Models

```
# Gravitational Forces
gravity_field:
    enabled: true
    gravity_degree_max:      15      # Gravity model maximum degree/order (d/
   o)
    timevar_degree_max:      15      # Time-variable coefficients maximum d/o
```

Listing 4.23: yaml example for gravitational force model options

4.14.3 planetary_perturbations:

Option	Values	Comments
enabled	true or false	Uses the emphemeris

Table 4.21: POD YAML: planetary perturbations

```
# Planetary Gravitational Forces
    planetary_perturbations:
        enabled: true
```

Listing 4.24: yaml example for planetary pertubations

4.14.4 tidal_effects:

Option	Values	Comments
solid_tides_nonfreq	True or False	frequency independent Solid Earth Tides
solid_tides_freq	True or False	frequency dependent Solid Earth Tides
ocean_tides	True or False	uses the ocean tides file
solid_earth_pole_tides	True or False	tide induced earth spin rotation not about the centre of the ellipsoid
ocean_pole_tide	True or False	ocean response to the above
ocean_tides_degree_max	True or False	maximum model term in spherical harmonic expansion

Table 4.22: POD YAML: tidal effects

```
1    tidal_effects:
2    enabled: true
3    solid_tides_nonfreq:    true   # Solid Earth Tides frequency-independent
     terms
4    solid_tides_freq:       true   # Solid Earth Tides frequency-dependent
     terms
5    ocean_tides:            true   # Ocean Tides
6    solid_earth_pole_tides: true   # Solid Earth Pole Tide
7    ocean_pole_tide:        true   # Ocean Pole Tide
8    ocean_tides_degree_max: 15     # Ocean Tides model maximum degree/order
```

Listing 4.25: yaml example for tidal effects

4.15 relativistic_effects:

Option	Values	Comments
enabled	true or false	Lens Thinning, SchwarzChild and deSitter effects, there are no means to separate these effects currently. The Lens Thirring effect is calculated but subsequently ignored in the POD.

Table 4.23: POD YAML:relativistic_effects

4.16 non_gravitational_effects:

4.16.1 Models to be applied:

Option	Values	Comments
solar_radiation	true or false	radiation push from the sun
earth_radiation	true or false	radiation push from the earth
antenna_thrust	true or false	reverse thrust from antenna radiation

Table 4.24: POD YAML: non gravitational effects

```
# Non-gravitational Effects
non_gravitational_effects:
    enabled: true
    solar_radiation: true
    earth_radiation: true
    antenna_thrust:  true
```

Listing 4.26: yaml example for non gravitational effects

4.16.2 Empirical parameters

Option	Values	Comments
ecom_d_bias	true or false	
ecom_y_bias	true or false	
ecom_b_bias	true or false	
ecom_d_cpr	true or false	(only ECOM1hybrid)
ecom_y_cpr	true or false	(only ECOM1hybrid)
ecom_b_cpr	true or false	
ecom_d_2_cpr	true or false	(only ECOM2hybrid)
ecom_d_4_cpr	true or false	(only ECOM2hybrid)
emp_r_bias	true or false	
emp_t_bias	true or false	
emp_n_bias	true or false	
emp_r_cpr	true or false	
emp_t_cpr	true or false	
emp_n_cpr	true or false	
cpr_count	int	empirical cpr count

Table 4.25: POD YAML: Configuration options for solar radiation pressure models

```
# Non-gravitational Effects
srp_parameters:
    ECOM_D_bias:  true
    ECOM_Y_bias:  true
    ECOM_B_bias:  true
```

```
    EMP_R_bias:   true
    EMP_T_bias:   true
    EMP_N_bias:   true
    ECOM_D_cpr:   true
    ECOM_Y_cpr:   true
    ECOM_B_cpr:   true
    ECOM_D_2_cpr: false
    ECOM_D_4_cpr: false
    EMP_R_cpr:    true
    EMP_T_cpr:    true
    EMP_N_cpr:    true
    cpr_count:    1
```

Listing 4.27: yaml example for srp parameters

NB EQM and VEQ srp parameters MUST be identical. May move into pod_options in future. overrides are not implemented yet. Ignore for now. We imagine overrides at the system, block (sat type) and individual satellite level

4.17 overides*

***This section has not yet been implemented in the POD, and is a placeholder for future versions.**

In this section put any system, block or PRN overrides that are different to the ones chosen before

```
overrides:
    system:
    GPS:
    srp_apriori_model:
    no_model:   false
    cannon_ball_model:   true
    simple_boxwing_model:   false
    full_boxwing_model:     false
    GAL:
    srp_apriori_model:
    no_model:   false
    cannon_ball_model:   false
    simple_boxwing_model:   false
    full_boxwing_model:     true
    GLO:
    srp_apriori_model:
    no_model:   false
    cannon_ball_model:   false
    simple_boxwing_model:   true
    full_boxwing_model:     false
    BDS:
    srp_apriori_model:
    no_model:   false
    cannon_ball_model:   false
    simple_boxwing_model:   true
    full_boxwing_model:     false
block:
    GPS-IIF:
        srp_apriori_model:
        no_model:   false
        cannon_ball_model:   false
```

```
        simple_boxwing_model:   true
        full_boxwing_model:     false
    # GPS BLK IIF use ECOM2 parameters
    srp_parameters:
        ECOM_D_bias:   true
        ECOM_Y_bias:   true
        ECOM_B_bias:   true
        ECOM_D_2_cpr: false
        ECOM_D_4_cpr: false
        ECOM_B_cpr:   true
prn:
    G01:
        srp_apriori_model:
            no_model:   false
            cannon_ball_model:   false
            simple_boxwing_model:   false
            full_boxwing_model:     true
```

Listing 4.28: yaml example for override

5 Overview of the PEA

The software execution of the Parameter Estimation Algorithm (PEA), written in C++, will be largely sequential - using threads sparingly to limit the overhead of collision avoidance. Where possible tasks will be completed in parallel using parallelisation libraries to take advantage of all cpu cores in multi-processor systems while still retaining a linear flow through the execution.

Sections of the software that create and modify global objects, such as while reading ephemeris data, will be executed on a single core only. This will ensure that collisions are avoided and the debugging of these functions is deterministic.

For sections of the software that have clear delineation between objects, such as per-receiver calculations, these may be completed in parallel, provided they do not attempt to modify or create objects with more global scope. When globally accessible objects need to be created for individual receivers, they should be pre-initialised before the entry to parallel execution section.

5.1 Data Input and Synchronisation

Before the processing of data from an epoch is initiated, all other relevant data is accumulated. As this code affects global objects that have effects in multiple places, this code is run in a single thread until data processing is ready to begin.

5.1.1 Config

Configurations are defined in YAML files. At the beginning of each epoch the timestamp of the configuration file is read, and if there has been a modification, new parameters in the configuration will be loaded into memory.

5.1.2 Product Input

Various external products may be required for operation of the software, as defined in the configuration file. At the beginning of each epoch, if any product input files have been added to the config, or if the inputs are detected to have been modified, they will be re-read into memory.

5.1.3 Metadata Input

Metadata such as GNSS ephemerides are available from external sources to augment the capability of the software. This data is ingested at the beginning of each epoch before processing begins.

5.1.4 Observation Data Input

Observation data forms the basis for operation of the software. Observations from various sources are synchronised and collated at the beginning of each epoch before processing begins. The software uses class inheritance and polymorphism such that all data type inputs are retrieved using a single common instruction, with backend functions performing any retrieval and parsing required.

Observation data is synchronised by timestamp - when the main function requests data of a specific timestamp all data until that point is parsed (but may be discarded), before the observation data corresponding to that timestamp being used in processing.

5.1.5 Initialisation of Objects

During the following stages of processing many receiver-specific objects may be created within global objects. To prevent thread collision in the global objects, the receiver-specific objects are created here sequentially.

5.2 Preprocessor

The preprocessor is run on input data to detect the anomalies and other metrics that are available before complete processing of the data is performed. This enables low-latency reporting of issues, and prepares data for more efficient processing in the later stages of operation.

- Cycle Slip Detection
- Low Latency Anomaly Detection
- Missing Data
- Output / Reporting

5.3 Precise Point Positioning

The largest component of the software, the PPP module ingests all of the data available, and applies scientific models to estimate and predict current and future parameters of interest.

Version 1 of the GINAN toolkit satisfies many of the requirements for GNSS modelling, but has been achieved by incrementally adding features as they became available and as scientific models have been developed. Many of the components make assumptions about the outputs of previous computations performed in the software, and require care before adding or making changes to the code, or even setting configuration options.

It is intended that the software will be reorganised with the benefit of hindsight, to remove interactions between modules and explicitly execute each processing step in a manner similar to an algebraic formulation used by experts in the field.

It is tempting for researchers to apply heuristics or corrections that may have been historically used to assist in computation, but these must be limited to effects that can be modelled and applied through the Kalman filter, in order to maintain the efficiency and robustness that it provides.

As the models required for ‘user’, ‘network’, and even ‘ionosphere’ modes are equivalent, the only distinction between the modes is the extent of modelling to be applied, which can be reduced to a simple configuration change. As such the parallel streams within the software will be eliminated and reduced to a single unified model, with example configurations for common use-cases.

5.3.1 Force and Dynamic Models

At the beginning of processing of an epoch, parameters with time-dependent models are updated to reflect the time increment since the previous epoch. Simple models will be well defined when initialised, but more complex models will require updating at every epoch.

Ultimate positioning performance largely depends on accurate dynamic models, with development of these models improving predictive capability, and reducing uncertainty and adjustments at every point in time.

- Gravity
- Solar Radiation Pressure
- Other

5.3.2 Orbit and State Prediction

Before the available observations for an epoch are utilised, a prediction is made of the parameters of interest by utilising the previous estimates and applying dynamic models through the Kalman filter's state transition.

5.3.3 Phenomena Modelling + Estimation

In order to accurately estimate and predict parameters of interest, all phenomena that affect GNSS/SLR observations must be isolated and modelled, as being components comprising the available measurements.

Where the values of parameters are well known they may be used directly to extract other parameters of interest from the data - such as using published corrections to precisely determine a user's position.

When data is unavailable, or when it is desired to compute these products for subsequent publication and use, estimates of the values are derived from the available data.

It is the sophistication of the models available and applied that determines the ultimate performance of the software.

The software will be developed to allow for all applicable phenomena to be modelled, estimated, such that user's desired constraints may be applied and parameters of interest extracted.

5.3.4 Initialisation of Parameters

Estimation parameters are initialised on the point of first use, automatically by the Kalman filter module. Their initial value may be selected to be user-defined, extracted from a model or input file, or established using a least-squares estimation.

5.3.5 Robust Kalman Filter

It is well known that the Kalman filter is the optimal technique for estimating parameters of interest from sets of noisy data - provided the model is appropriate.

In addition, statistical techniques may be used to detect defects in models or the parameters used to characterise the data, providing opportunities to intervene and make corrections to the model according to the nature of the anomaly.

By incorporating these features into a single generic module, the robustness that was previously available only under certain circumstances may now be automatically applied to all systems to which it is applied. These benefits extend automatically to all related modules (such as RTS), and often perform better than modules designed specifically to address isolated issues.

For further details about the software's robust Kalman filter see the Ginan Science Manual.

5.3.6 RTS Smoothing

The intermediate outputs of a Kalman filter are of use for other algorithms such as RTS smoothers. All intermediate values required for such algorithms are to be recorded in a consistent manner, suitable for later processing.

For further details about the software's RTS smoothing algorithm see the Ginan Science Manual.

5.3.7 Integer Ambiguity Resolution

GNSS phase measurements allow for very precise measurements of biases but require extra processing steps to disambiguate between cycles. Techniques have been demonstrated that perform acceptably under certain conditions and measurement types, but require substantial bookkeeping and may not easily transfer to different measurement applications.

For further details about the software's ambiguity resolution algorithms see the Ginan Science Manual.

5.3.8 Product calculation

In order for estimated and predicted values to be of use to end-users, they must be prepared and distributed in an appropriate format.

Some parameters of interest are not directly estimated by the filter, but may be derived from estimates by secondary operations, which are performed in this section of the code.

In this section, data is written to files or pushed to NTRIP casters and other data sinks.

5.4 Post-processing

5.4.1 Smoothing

The RTS Smoothing algorithm is capable of using intermediate states, covariances, and state transition matrices stored during the Kalman filter stage to calculate reverse smoothed estimates of parameters.

The intermediate data is stored in binary files with messages that contain tail blocks containing the length of the message. This allows for the file to be efficiently traversed in reverse; seeking to the beginning of each message as defined by the tail block.

For further details about the software's RTS Smoothing algorithm see the Ginan Science Manual.

5.4.2 Minimum Constraints

The minimum constraints algorithm is capable of aligning a network of stations to a reference system without introducing any bias to the positions of the stations.

A subset of stations positions are selected and weighted to create pseudo-observations to determine the optimal rigid transformation between the coordinates and the reference frame. The transformation takes the same algebraic form as a Kalman filter stage and is implemented as such in the software.

For further details about the software's minimum constraints algorithm see the Ginan Science Manual.

5.4.3 Unit Testing

The nature of GNSS processing means that well-defined unit tests are difficult to write from first-principles. The software however, is capable of comparing results between runs to determine if the results have changed unexpectedly.

Intermediate variables are tagged throughout the code, and auxiliary files specify which variables should be tested as they are obtained, and the expected values from previous runs.

5.4.4 Logging

Details of processing are logged to trace files according to the processing mode in use.

Per-station files are created with intermediate processing values and information, while a single summary file is generated for the unified filter and combined processing.

Information produced during processing is output to the console, and may also be redirected to other logging sinks such as a database, or json formatted output.

In addition to processing information, inputs may be recorded to file for replaying later.

6 Using PEA in network mode

PEA is designed to estimate the GNSS error parameters that cannot be precisely determined. The PEA will estimate the following parameters:

- Correction to satellite initial conditions estimated by the POD component
- Satellite clock offset and drift
- Satellite hardware bias for two signal carriers
- Satellite differential bias for two signal pseudoranges
- Ionospheric propagation delay
- Tropospheric propagation delay
- Receiver/station position and velocity
- Receiver/station clock offset and drift
- Receiver/station hardware bias for signal carriers
- Receiver/station differential bias for two signal pseudoranges
- Relative carrier phase ambiguities

In order to estimate the full range of parameters, the PEA will need to ingest GNSS observation data from a Global network of sufficient density. Receiver position, clock and Tropospheric delays can also be estimated from a single receiver and thus will be addressed on chapter 7.
As is the case for the POD, the PEA uses YAML formatted configuration files to set the processing options, and is run using the command:

```
./pea --config <path_to_config_file>
```

Details on the configuration parameters included in YAML files can be found in chapter 9.

With one exception (Ionosphere delay modelling using smoothed pseudorange), processing of network data is activated by setting the *processing_options : process_modes : network* to *true*. Configuration files corresponding to the examples in this section can be found in the *ginan/examples* directory. A few of these examples are explained bellow.

6.1 Processing a Global Network to Adjust Satellite Positions

PEA is designed to use integrated/predicted satellite positions for its real-time network processing mode, thus all satellite position corrections can only be made in post process mode. A basic example for PEA network processing is provided in *ex17_pea_pp_netw_gnss_ar.yaml*.

In order to activate estimation of satellite position corrections the entry /textitdefault_filter_parameters : satellites : orb : estimated needs to be set to *true*.

It will also require an ICF file, generated by POD in orbit fitting mode, as input. The path to such file should be given in *input_files : orbfiles* entry.

ANTEX files, with antenna information for both stations and satellites should be provided in *input_files : atxfiles*

SINEX files, with station antennas should be provided in *input_files : snxfiles*

RINEX 3.XX navigation files, with broadcast clocks should be provided in *input_files : navfiles*

BLQ formatted ocean tide loading parameters for each station should be provided in *input_files : blqfiles* if available to correct for OTL, otherwise *processing_options : tide_otl* should be set to *false*.

RINEX 3.XX observation files for the network stations should be provided in *station_data : rnxfiles* (* can be used as a wildcard).
The main output of this processing mode would be an ICF formatted file containing the corrected initial conditions for each of the processed satellites. The file will be created in the same path as the input ICF file with the *_pea* suffix attached. This file can be used by the POD to integrate/predict the satellite positions over the required time window.

Receiver/station positions can also be estimated as part of the network processing. In order to estimate station positions, the parameter *default_filter_parameters : stations : pos : estimated* and *output_files : output_sinex* need to be set to *true* A SINEX formatted file with the estimated station position will be generated in the path specified as *root_output_directory*.

6.2 Post process estimation of Satellite clocks and biases

The example configuration file *ex17_pea_pp_netw_gnss_ar.yaml* corresponds to a post-mission network processing mode for GPS satellite clocks.

The required input files are similar to the satellite position estimation example. The main difference will be that SP3 formatted files can be used as input for satellite position (POD generated ICF files can also be used).

The frequency in which the GNSS error parameters, including satellite and receiver clocks, are estimated should be set by the parameter *processing_options : epoch_interval*.

In order to output the estimated clocks, the parameter *output_files : output_clocks* needs to be set to *true*.

The satellite and receiver will then be output to a RINEX clock formatted file specified in *output_files : clock_filename*.
In order to estimate the satellite and receiver hardware biases, the ambiguities need to be separated from the biases and resolved to integer values.

In order to perform ambiguity resolution, the proper parameters needs to be set on the *ambiguity_resolution_options* fields.

The target constellations need to be selected by setting *GPS_amb_resol* and/or *GAL_amb_resol* to true (only GPS and GAL constellation are supported in the current version).

Both *WL_mode* and *NL_mode* needs to be set to something different than *off*.

It is advised that *round* or *iter_rnd* be used for network processing.

In order to output the estimated biases the parameter *output_files : output_biasSINEX* needs to be to *true* and *ambiguity_resolution_options : bias_output_rate* set to a number (of seconds) different than zero. The satellite biases will then be output to a bias SINEX formatted file specified in *output_files : biasSINEX_filename*.

Receiver biases are also estimated by the process, however they are not reflected on the output files.

6.3 Real-time estimation of Satellite clocks and biases

An example of using PEA for real-time estimation of GPS satellite clocks is provided in *ex17_pea_rt_netw_gnss_ar.yaml*.

The configuration file is similar to that used for post-mission processing. The main difference is that the input data specified in the *station_data* field will correspond to RTCM formatted streams instead of files.

Currently the PEA can only get real-time data by connecting to an NTRIP caster. The host name, user name and password corresponding to the NTRIP should be specified under *station_data : stream_root* using the format *http(s)://user:password@hostname/*.

The mountpoint corresponding to station observables need to be listed under *station_data : obs_streams*.

Ephemeris streams (broadcast ephemeris and SSR corrections) should be listed under *station_data : nav_streams*.

Alternatively the mountpoints can be specified using the full path *http(s)://user:password@hostname/mountpoint*, leaving the *station_data : stream_root* field empty, this allows to use streams from multiple NTRIP casters.

Satellite positions needs to be provided using (predicted) SP3 files or using real-time streams (broadcast + SSR corrections).
In the *processing_options* field, the *ppp_ephemeris* parameter needs to be set to *precise* is using SP3 files and *ssr_apc* or *ssr_com* if using SSR correction streams.

In the same field, the *epoch_interval* is used to set the update interval of the network solutions, and the *wait_next_epoch* and *wait_all_stations* to help synchronise station streams.

The PEA will wait for *wait_next_epoch* seconds from the start of the previous epoch for the first observation to arrive (and skip the current epoch if no observations arrive).

The PEA will wait for *wait_all_stations* seconds from the first observation for data from other stations before processing.

The estimated clock and bias can be output to an NTRIP caster (as well as to local files).

The output NTRIP caster streams need to be specified using the *output_streams* field. The host name, user name and password can be set in the *output_streams : stream_root* parameter. The names for the output streams should be listed under *output_streams : stream_label* Once the label is created, the mountpoint and RTCM messages to encode can be specified in the *output_streams : label* field.

Currently the PEA supports output for GPS and Galileo orbits and clock messages (1060 and 1243), code bias (1059 and 1243) and phase bias (1265 and 1267).

6.4 Post process estimation of atmospheric delays

The PEA is capable of estimation both Tropospheric and Ionospheric delays on GNSS signals. Tropospheric delays can be estimated both in network (*processing_options : process_modes : network* = *true*) and end-user (*processing_options : process_modes : user* = *true*) modes. (End-user mode and will be addressed in the next chapter)

Ionosphere delay estimation and mapping is activated by setting *processing_options : process_modes : ionosphere* to *true*.

Two types of Ionosphere delay estimates are supported by PEA. If all other parameters in the *processing_options : process_modes* field are set to *false* then the Ionospheric delay are estimated based on carrier smoothed pseudoranges. If in addition to *processing_options : process_modes : ionosphere*, *processing_options : process_modes : network* is set to true, the PEA will attempt to calculate Ionospheric delay measurements from ambiguity resolved carrier phase measurements. For this mode to work, the parameters in the *ambiguity_resolution_options* field needs to be set properly. Ionospheric delay estimate are available for GPS signals only.
The Ionosphere slant delay measurements delay measurements can be outputted into STEC files if *output_files : output_ionstec* parameter is set to *true* (the output file name can be set using *ionstec_filename*).

The format of STEC files have one of two forms. If the vector in *ionosphere_filter_parameters : layer_heights* is not empty, each line on the STEC file will contain the slant delay measurements and the piercing points at each layer height:

```
#IONO_MEA, 2102,171000.000, AGGO, G05, -1.6030, 1.9699e-04, 2, 1, 350,
    -33.662, -61.955, 1.443
```

the fields representing, from left to right:

1. ”IONO_MEA” label
2. GPS week
3. GPS TOW in seconds
4. Receiver/station name
5. Satellite ID
6. Slant delay in meters

7. Slant delay variance in meters^2
8. Number of Ionosphere layers N
9. N fields containing:
    (a) Height of layer in Km
    (b) Latitude of piercing point (in degrees)
    (c) Longitude of piercing point (in degrees)
    (d) Slant to vertical mapping function

If the layer heights field is empty the ”Number of Ionosphere layers” field will be 0 and followed by the receiver position in ECEF, and the satellite position in ECEF.
Vertical TEC (VTEC) maps can be estimated from the slant delay measurements and output as IONEX formatted maps (*output_files : output_ionex* = *true*) and its corresponding DCB (*output_files : output_biasSINEX* = *true*).

Ionosphere mapping and output are controlled by parameters in the *ionosphere_filter_parameters* field. Currently only spherical harmonics based mapping is supported by the PEA *model* = *spherical_harmonic*, setting *ionosphere_filter_parameters : model* to *meas_out* will output the ionosphere measurements but will not perform mapping.

If spherical harmonics is selected as mapping method, the ionospheric delays will be mapped into multiple thin layer shells. The height of the shells can be set in the *ionosphere_filter_parameters : layer_heights* vector.

The VTEC at each layer will be fit to spherical harmonic components, with a maximum order and degree of *ionosphere_filter_parameters : func_order*. If *output_files : output_ionex* is set to *true* the Ionosphere map will be outputted in IONEX 1.11 format.

The area of the IONEX map can be set using the *lat_center*, *lon_center*, *lat_width*, *lon_width* parameters. The horizontal resolution of the IONEX map can be set using the *lat_res*, *lon_res* parameters. The temporal resolution of the IONEX file is defined by the *time_res* parameter.

A configuration file, *ex16_pea_pp_ionosphere.yaml* can be used to generate a single layer IONEX map and accompanying biasSINEX file from smoothed pseudorange observations.

7 Using PEA in user mode

When set to end user mode, the PEA component of Ginan will process each station separately. This mode will allow the estimation of parameters available to users with single receivers.

- Receiver position
- Receiver clock offset
- Tropospheric delay at receiver location
- Ionospheric delay at the receiver location (not yet available)
- Carrier phase ambiguities

In order to use the PEA in end-user mode, the *processing_options : process_modes : user* parameter needs to be set to *true*.

The results of PEA run in end user mode are printed in the trace files. Trace file outputs can be activated by setting the *output_files : output_trace* parameter to *true*. The most commonly used outputs from the PEA used in end-user mode are expected to be: the receiver position, receiver velocity, receiver clocks and tropospheric delays.

7.1 Receiver position

Receiver position results are preceded by the ”$POS” label and thus, in Linux, can be extracted using the command:

```
grep "$POS" <path_to_trace_file>
```

the output line for the for receiver position will have 10 comma separated fields with the following format:

```
$POS, 2166, 278015.000, 6, -4052053.0060, 4212836.8682, -2545105.0796,
    0.0245227, 0.0231919, 0.0163678
```

the fields represent, from left to right:

1. ”$POS” label
2. GPS week
3. GPS TOW in seconds
4. Solution type (6 for float PPP, 1 for ambiguity fixed PPP)
5. Receiver ECEF X position in meters
6. Receiver ECEF Y position in meters
7. Receiver ECEF Z position in meters

8. Standard deviation of ECEF X positions in meters
9. Standard deviation of ECEF X positions in meters
10. Standard deviation of ECEF X positions in meters

7.2 Receiver clock

Receiver clock offset results are preceded by the ”$CLK” label and thus, in Linux, can be extracted using the command:

```
grep "$CLK" <path_to_trace_file>
```

the output line for the for receiver position will have 13 comma separated fields with the following format:

```
$CLK, 2166, 278015.000, 6, 14, 3.1902, 0.0000, 1.1924, 0.0000, 0.0860,
    0.0000, 0.0953, 0.0000
```

the fields represent, from left to right:

1. ”$CLK” label
2. GPS week
3. GPS TOW in seconds
4. Solution type (6 for float PPP, 1 for ambiguity fixed PPP)
5. Number of satellites used in the solution
6. Receiver clock offset for with respect to GPS clock, in nanoseconds
7. Receiver clock offset for with respect to GLONASS clock, in nanoseconds
8. Receiver clock offset for with respect to Galileo clock, in nanoseconds
9. Receiver clock offset for with respect to Beidou clock, in nanoseconds
10. Standard deviation of clock offset wrt. GPS, in nanoseconds
11. Standard deviation of clock offset wrt. GLONASS, in nanoseconds
12. Standard deviation of clock offset wrt. Galileo, in nanoseconds
13. Standard deviation of clock offset wrt. Beidou, in nanoseconds

If clock offsets for a particular constellation are not available both the offset and its variance will be set to 0.

7.3 Tropospheric delays

Tropospheric delays at the receiver position are preceded by the ”$TROP” label and thus, in Linux, can be extracted using the command:

```
grep "$TROP" <path_to_trace_file>
```

the tropospheric delay solutions will be represented to either a single line, with the ”$TROP” or three lines, as follows:

```
$TROP, 2166, 278015.000, 6, 14 ,2.294950, 0.0030977
$TROP_N, 2166, 278015.000, 6, 14, -0.174797, 0.0181385
$TROP_E, 2166, 278015.000, 6, 14, -0.223868, 0.0250276
```

each of the troposphere output line will contain 7 comma separated field, of which the first five are:

1. Label, ”$TROP”, ”$TROP_N” or ”$TROP_E”
2. GPS week
3. GPS TOW in seconds
4. Solution type (6 for float PPP, 1 for ambiguity fixed PPP)
5. Number of satellites used in the solution

The line starting with ”$TROP” contain the Zenith Tropospheric Delay (ZTD) and its standards deviation, both in meters, as their last two fields. The line starting with ”$TROP_N” contains the tropospheric delay gradient in north-south direction, and the line starting with ”$TROP_E” contains the tropospheric delay gradient in east-west direction.

Configuration files for specific examples have been added to the *examples* folder in the repository. Examples corresponding to end user processing are explained bellow. In order to

7.4 Dual frequency PPP with floating ambiguities

As the end-user processing mode cannot calculate satellite states, the satellite position and clock offset needs to be provided externally.

The PEA supports SP3 formatted satellite position inputs, specified in *input_files : sp3files*, and RINEX clock files, *input_files : clkiles*, as satellite clock inputs.

ANTEX files, with antenna information for both stations and satellites should be provided in *input_files : atxfiles*

SINEX files, with station antennas should be provided in *input_files : snxfiles*

RINEX 3.XX navigation files, with broadcast clocks should be provided in *input_files : navfiles*

BLQ formatted ocean tide loading parameters for each station should be provided in *input_files : blqfiles* if available to correct for OTL, otherwise *processing_options : tide_otl* should be set to *false*.

RINEX 3.XX observation files for the network stations should be provided in *station_data : rnxfiles*

The configuration files named *examples/ex11_pea_pp_user_gps.yaml* and *examples/ex12_pea_pp_user_gnss.yaml* set the PEA to calculate a post-process end user solution for a static receiver.

The constellations to be used in processing can be specified in the *processing_options : process_sys* field.

The tracking of a moving receiver can be done by setting the *default_filter_parameters : stations : pos : proc_noise* parameter to the maximum expected velocity.

Receiver velocity can also be estimated by setting *default_filter_parameters : stations : pos_rate : estimate* to *true*.

Tropospheric delays are estimated as a combination of hydrostatic and wet components, each component is in turn estimated as the products of the zenith delay and a mapping function. If *default_filter_parameters : stations : trop : estimate* is set to *true*, the PEA estimates the zenith wet delay. If *default_filter_parameters : stations : trop_grad : estimate* is set to *true*, the PEA also estimates azimuthal components of tropospheric mapping functions.

The hydrostatic zenith delays and elevation dependent component of mapping functions are calculated based on pre-defined models. Available models, which can be selected using the *processing_options : troposphere : model* parameter, are the GPT2 and VMF3 models.

If using the GPT2 model the path to the necessary grid file needs to be specified in *processing_options : troposphere : gpt2grid*

If using the VMF3 model, the tropospheric parameters corresponding to the observation times need to be provided in a directory specified by *processing_options : troposphere : vmf3dir*, and the orography file for atmospheric circulation models need to be specified in *processing_options : troposphere : orography*.

7.5 Single frequency PPP

It is possible to perform end user PPP processing using single frequency data (although at reduced accuracy) by providing external Ionospheric delay data.

The configuration files named *examples/ex13_pea_pp_user_gps_sf.yaml* set an example to process single frequency observations.

The PEA currently uses IONEX formatted VTEC maps as Ionosphere delay data. The path to the IONEX file needs to be specified in *input_files : ionfiles*.

In order for the PEA to use the VTEC maps, the *processing_options : ionosphere : corr_mode* parameter should to be set to *total_electron_content*.

If provided separately, files containing the satellite DCB (either RINEX DCB or bias SINEX) should be specified in *input_files :dcbfiles*

7.6 Dual frequency PPP with ambiguity resolution

The PEA (in both network and user processing modes) can be specified to perform ambiguity resolution in an attempt to improve accuracy and convergence times.

In aside from the requirements for floating PPP ambiguities, information on satellite hardware biases needs to be provided order to allow correct ambiguity resolution in end user PEA processing.

For post-process, the PEA use bias SINEX formatted files as input channels for satellite hardware biases. The bias SINEX file can be specified in *input_files :bsxfiles*.

The ambiguity resolution process is controlled by the *ambiguity_resolution_options* field. Currently ambiguity resolution is only supported for GPS and Galileo satellites.

Ambiguity resolution for GPS satellites can be activated by setting the *GPS_amb_resol* parameter to *true*.

Ambiguity resolution for Galileo satellites can be activated by setting the *GAL_amb_resol* parameter to *true*.

In addition the ambiguity resolution algorithm needs to be specified for both the wide-lane ambiguity (*WL_mode*) and narrow-lane ambiguities (*NL_mode*.

For best results, *round* or *iter_rnd* are recommended for Wide-lane ambiguities and *lambda_alt* or *lambda_bie* is recommended for narrow-lane ambiguities.

7.7 Real-time PPP

The PEA can also be used to process GNSS data in real-time. Real-time processing will make use of RTCM formatted streams for receiver observables and satellite data.

Currently the PEA can only get real-time data by connecting to an NTRIP caster.

An example of such a caster, can be accessed by registering at . The host name, user name and password corresponding to the NTRIP can shold be specified under *station_data : stream_root* using the format *http(s)://user:password@hostname/*. The mountpoint corresponding to station observables need to be listed under *station_data : obs_streams*.

Ephemeris streams (broadcast ephemeris and SSR corrections) shold be listed under *station_data : nav_streams*.

The PEA support MSM4, MSM5, MSM6 and MSM7 messages for observations, and orbit and clock messages, code bias messages and phase bias messages for GPS and Galileo.
Real-time outputs are not yet defined for PEA, the processed receiver solution are printed in real time on the TRACE files.

8 PEA examples

In this section we go through a number of different ways that the pea can be used to process GNSS data.

1. Precise Point Positioning (PPP) processing - In this section we will demonstrate how to processing in PPP mode using the Ionosphere free combination, we will provide an example on how to use IGS products to obtain a float solution, and then an example on how to obtain an ambiguity fixed solution. We will also cover how to process gnss streams in real-time.
2. Obtain an orbit solution from a global tracking network
3. Obtain an orbit and clock solution from a global tracking network
4. How to process a Global solution in real-time
5. How to obtain an ionosphere model

9 PEA Configuration File - YAML

The PEA processing engine uses a single YAML file for configuration of all processing options.

9.1 YAML Syntax

The YAML format allows for heirarchical, self descriptive configurations of parameters, and has a straightforward syntax.

White-space (indentation) is used to specify heirarchies, with each level typically indented with 4 space characters.

Colons (:) are used to separate configuration keys from their values.

Lists may be created by either appending multiple values on a single line, wrapped in square brackets and separated by commas, or, by adding each value on a separate indented line with a dash before the value.

Adding a hash symbol (#) to a line will render the remainder of the line as a comment to be ignored by the parser.

Strings with special characters or spaces should be wrapped in quotation marks.

You will see all of these used in the example configuration files, but the files may be re-ordered, or re-formatted to suit your application.

9.2 Default Values

Many processing options have default values associated with them. To prevent repetition, and to ensure that the values are reported correctly, these values may be viewed in the acsConfig.hpp file within the source code directories.

9.3 Globbing

Files may be specified individually, as lists, or by searching available files using a globbed filename using the star character (*)

9.4 Wildcard Tags

Output filenames can include wildcards wrapped in < > brackets to allow more generic names to be used. While processing, these tags are replaced with details gathered from processing, and allows for automatic generation of, for example, hourly output files.

<CONFIG>

This is replaced with the 'config_description' value entered in the yaml file.

<STATION>

This is replaced with the 4 character station id of each station that generates a trace file.

<LOGTIME>
This is replaced with the (rounded) time of the epochs within the trace file.

If trace file rotation is configured for 1 hour, the `<LOGTIME>` wildcard will be rounded down to the closest hour, and subsequently change value once per hour and generate a separate output file for each hour of processing.

<DDD>, <D>, <WWWW>, <YYYY>, <YY>, <MM>, <DD>, <HH>
These are replaced with the components of time of the start epoch.

9.5 input_files:
This section of the yaml file specifies the lists of files to be used for general metadata inputs, and inputs of external product data.

```
# Example
input_files:

    root_input_directory: /data/acs/pea/proc/exs/products/

    atxfiles:    [ igs14_2045_plus.atx                                ]
    snxfiles:    [ igs19P2062.snx                                     ]
    blqfiles:    [ OLOAD_GO.BLQ                                       ]
    navfiles:    [ brdm1990.19p                                       ]
    orbfiles:    [ orb_partials/gag20624_orbits_partials_new.out      ]
    sp3files:    [ "*.sp3"                                            ]
    clkfiles:    [ jpl20624.clk                                       ]
    erpfiles:    [ igs19P2062.erp                                     ]
    dcbfiles:    [ CAS0MGXRAP_20191990000_01D_01D_DCB.BSX             ]
    bsxfiles:    []
    ionfiles:    []
```

Listing 9.1: A typical input_files section

root_input_directory:
This specifies a root directory to be prepended to all other file paths specified in this section. For file paths that are absolute, (ie. starting with a /), this parameter is not applied.

atxfiles:
A list of ANTEX files to be used in processing. These may supply the antenna parameters to be used by satellites and receivers.

snxfiles:
A list of SINEX files to be used in processing. These may supply the initial positions and other metadata for receivers.

blqfiles:
A list of BLQ files to be used in processing. These may supply the ocean tide loading data.

navfiles:
A list of NAV files to be used in processing. These may supply the basic broadcast ephemerides for satellites.

orbfiles:
A list of ORB files to be used in processing. These may supply the PEA with orbital and ratiation pressure data from the Ginan's POD module, allowing precise orbit data to be passed between the two pieces of software.

sp3files:
A list of SP3 files to be used in processing. These may supply the ephemerides for higher precision processing.

clkfiles:
A list of CLK files to be used in processing. These may supply the clock offsets for satellites and receivers for higher precision processing.

erpfiles:
A list of ERP files to be used in processing. These may supply the earth rotation parameter information.

dcbfiles:
A list of DCB files to be used in processing. These may supply the differential code biases to assist with ambiguity resolution.

bsxfiles:
A list of BSINEX files to be used in processing. These may supply biases to assist with ambiguity resolution.

ionfiles:
A list of ION files to be used in processing. These may supply the ionospheric modelling parameters for single frequency processing.

9.6 station_data:
This section specifies the sources of observation data to be used in positioning.

There are numerous ways that the *pea* can access GNSS observations to process. You can specify individual files to process, set it up so that it will search a particular directory, or you can use a command line flag `--rnx <rnxfilename>` to add an additional file to process. The data should be uncompressed rinex (gunzipped, and not in hatanaka format), or RTCM3 formatted binary data.

It may consist of RINEX files, or RTCM streams or files, which are specified as follows:

9.6.1 Post processing:

```
# post processing example
station_data:
    root_stations_directory: /data/acs/ginan/examples/data
    rnxfiles:
        - "ALIC*.rnx"
        - "BAKO*.rnx"

    #obs_rtcmfiles:
    #   - "*-OBS.rtcm3"

    #nav_rtcmfiles:
    #   - "*-NAV.rtcm3"
```

Listing 9.2: station_data:

root_stations_directory:

This specifies a root directory to be prepended to all other file paths specified in this section. For file paths that are absolute, (ie. starting with a /), this parameter is not applied.

rnxfiles:

This is a list of RINEX files to be used for observation data. The first 4 characters of the filename are used as the receiver ID.

If multiple files are supplied with the same ID, they are all processed in sequence - according to the epoch times specified within the files. In this case, it is advisible to correctly specify the start_epoch for the filter, or the first epoch in the first file will likely be used.

obs_rtcmfiles:

This is a list of RTCM binary files to be used for observation data. The first 4 characters of the filename are used as the receiver ID.

This can be used to read data that has been saved from a stream for later testing.

nav_rtcmfiles:

This is a list of RTCM binary files to be used for navigation and correction data. No receiver is to be associated with these files.

9.6.1.1 Real-time processing:

To process data in real-time you will need to set up the location, username annd password for the caster that you will be obtaining the input data streams from in the configuration file.

The pea supports obtaining streams from casters that use NTRIP 2.0 over http and https.

```
# realtime streaming example
station_data:

    stream_root: "http://<username>:<password>@auscors.ga.gov.au:2101/"

    nav_streams:
        - BCEP00BKG0
        - SSRA00CNE0

    obs_streams:
        - STR100AUS0

    ssr_input_antenna_offset: APC
```

Listing 9.3: station_data:

As shown in listing: 9.3, the caster url, username and password are specified within double quotes with the *stream_root* tag. In this example the streams are being obtained from the auscors caster run by Geoscience Australia. The broadcast information is being obtained from the stream *BCEP00BKG0* being supplied by BKG, and corrections to the

utlra-rapid predicted orbit are being obtained from the stream *SSRA00CNE0*. The real-time data being processed is for the continuous GNSS station located at Mount Stromlo obtained from the stream *STR100AUS0*.

You can test your username and password is working correctly by running the curl command:

```
curl https://ntrip.data.gnss.ga.gov.au/ALIC00AUS0 -H "Ntrip-Version: NTRIP
    /2.0" -i  --output - -u <user>
```

stream_root:
This specifies a root url to be prepended to all other streams specified in this section. If the streams used have individually specified root urls, usernames, or passwords, this should not be used.

obs_streams:
This is a list of RTCM streams to read realtime data from. The first 4 characters of the filename are used as the receiver ID.

In combination with the stream_root parameter, they may require a username, password, port and mountpoint.

The streams in this section are processed for observations from receivers.

nav_streams:
This is a list of RTCM streams to read real-time data from.

In combination with the stream_root parameter, they may require a username, password, port and mountpoint.

The streams in this section are processed separately from observations, and will typically be used for receiving SSR messages or other navigational data from an external service.

ssr_input_antenna_offset:
This setting should match the ephemeris type that is provided in the listed SSR stream, i.e. satellite antenna-phase-centre (APC) or centre-of-mass (COM). This information is listed in the NTRIP Caster's sourcetable - in general, use *APC* for *SSRA** streams, and *COM* for *SSRC** streams.

9.7 output_files:
This section of the yaml file specifies options to enable outputs and specify file locations.

An example of this section follows:

```
output_files:

root_output_directory:          /data/acs/ginan/examples/<CONFIG>/

output_trace:                   true
trace_level:                    3
trace_directory:                ./
trace_filename:                 <CONFIG>-<STATION><LOGTIME>.TRACE

```

```
output_residuals:               false

output_config:                  true

output_summary:                 false
summary_directory:              ./
summary_filename:               <CONFIG>-<YYYY><DDD><HH>.SUM

output_clocks:                  true
clocks_directory:               ./
clocks_filename:                <CONFIG>.clk
```

Listing 9.4: output_files:

root_output_dir:

This specifies a root directory to be prepended to all other file paths specified in this section. For file paths that are absolute, (ie. starting with a /), this parameter is not applied.

[X]_directory:

Directory to output file [X] to, where [X] are the features below. May contain wildcard tags. May be relative to root_output_dir, or absolute. If the directory does not exist, it will be created.

- trace_directory
- summary_directory
- clocks_directory
- ionex_directory
- biasSinex_directory
- sinex_directory
- persistance_directory
- rtcm_directory

[X]_filename:

Filename to use for output of [X]. May contain wildcard tags. File will be created or overwritten if it already exists.

- trace_filename
- summary_filename
- clocks_filename
- ionex_filename
- biasSinex_filename
- sinex_filename
- persistance_filename
- obs_rtcm_filename

- nav_rtcm_filename

trace_level:
Integer from 0-5 to specify verbosity of trace outputs. (5 - print everything)

trace_rotate_period, trace_rotate_period_units:
Granularity of length of time used for <LOGTIME>tags. These parameters may be used such that the filename of an output will change intermittently, and thus distribute the output over multiple files.

The <LOGTIME>tag is updated according to the epoch time, not the current clock time.

trace_rotate_period must be a numeric value, and trace_rotate_period_units may be one of seconds (default), minutes, hours, days, weeks, years, (with or without plural s).

output_residuals:
Boolean to print the residuals from kalman filter operation to relevant trace files.

output_config:
Boolean to print a copy of the yaml file to the top of each trace file. This may assist with keeping a record of the parameters used to generate the particular results contained in the file.

output_trace:
Boolean to generate per-station trace files.

output_summary:
Boolean to generate a network summary file.

output_clocks:
Boolean to generate RINEX formatted clock files from processed data.

output_AR_clocks:
Boolean to specify that the ambiguity resolved version of clocks should be output if output_clocks is enabled.

output_ionex:
Boolean to generate an IONEX file from processed ionosphere data.

output_ionstec:
Boolean to generate an IONSTEC file from processed ionosphere data.

output_biasSINEX:
Boolean to generate a biasSINEX from processed network data.

output_sinex:
Boolean to generate a sinex file containing processed solutions, and the metadata used to generate them.

output_persistance:
Boolean to save the network filter state, and navigation and ephemerides structure to disk once per epoch. For realtime processing where ephemerides are sourced from a a stream over several minutes, this may enable quicker start-up if the processor is restarted.

input_persisance:
Boolean to try to load a saved filter and navigation structure from disk.

output_mongo_measurements:
Boolean to output kalman filter measurements and residuals to a mongo database.

output_mongo_states:
Boolean to output the results of kalman filter processing to a mongo database.

output_mongo_logs:
Boolean to output timestamped log data from the console to a mongo database.

output_mongo_metadata:
Boolean to output timestamped metadata from processing to a mongo database. (unimplemented)

delete_mongo_history:
Boolean to delete a previous database using the same <CONFIG> tag before processing, to prevent collisions.

mongo_uri:
The URL to the location of the mongo database server.

9.8 processing_options:
This sections specifies the extent of processing that is performed by the engine.

epoch_interval:
Increment in nominal epoch time for each processing epoch. This parameter may be used to sub-sample datasets by using an epoch_interval that is a multiple of the dataset's internal interval between epochs.

start_epoch
Nominal time of the first epoch to process. Time is formatted as YYYY-MM-DD HH:MM:SS. This parameter may be left undefined to use the first available data point.

end_epoch
Maximum nominal time of the last epoch to process. This parameter may be left undefined.

max_epochs:
Maximum epochs to process before completion. This parameter may be left undefined.

process_modes:
```
process_modes:
    user:                   true
    network:                false
    minimum_constraints:    false
    rts:                    false
    ionosphere:             false
```
Listing 9.5: process_modes:

user:
Boolean to process all stations individually. Typically used for determining position of individual receivers.

network:
Boolean to process all stations in a single filter. May be used for determination of orbits, clocks, etc.

minimum_constraints:
Boolean to apply a rigid transformation to the results of the network filter after completion.

ionosphere:
Boolean to compute an ionosphere model from observations.

unit_tests:
Boolean to run tests to compare intermediate values during processing to stored results.

process_sys:
```
process_sys:
    gps:            true
    glo:            false
    gal:            false
    bds:            false
```

Listing 9.6: process_sys:

Booleans to enable the various GNSS satellite systems.

- gps
- glo
- gal
- bds

elevation_mask:
Minimum elevation required for observations to be used, measured in degrees.

ppp_ephemeris:
Option to specify source of satellite ephemeris used in PPP processing. Sources are:

- broadcast
- precise
- precise_com
- sbas
- ssr_apc
- ssr_com

tide_solid:
Boolean to apply solid tide model to station positions.

tide_otl:
Boolean to apply ocean tide loading model to station positions.

tide_pole:
Boolean to apply pole tide model to station positions.

phase_windup
Boolean to apply phase windup model to satellite phase measurements.

reject_eclipse
Boolean to exclude eclipsed satellites from processing.

raim
Boolean to perform ’Receiver autonomous integrity monitoring’ to detect and exclude observations that result in SPP failures.

cycle_slip: thres_slip:
Threshold to apply to geometry free phase values to determine if an observation should be rejected due to a slip.

max_inno:
Maximum innovation in PPP measurement before both phase and code measurements are excluded.

deweight_factor:
Factor by which measurement variances are increased upon detection of a bad measurement.

max_gdop:
Maximum ’geometric dilution of precision’ allowed for an SPP result to be valid.

antexacs:
Internal processing option. Bad things will likely happen if this is set to false.

sat_pcv:
Boolean to model satellite phase center variations.

pivot_station:
Station specified as origin for receiver clocks. Clocks for this station will be constrained to zero. May be set to <AUTO >or undefined to use first available station.

pivot_satellite
: Unused.

wait_next_epoch:
Expected time interval between successive epochs data arriving. For real-time this should be set equal to epoch_interval.

wait_all_stations:
Window of delay to allow observation data to be received for processing. Processing will begin at the earliest of:

- Observations received for all stations
- wait_all_stations has elapsed since any station has received observations
- wait_all_stations has elapsed since wait_next_epoch expired.

code_priorities:
List of observation codes that may be used in processing, and the order of priority for use. (Currently only a single code is used per frequency)

joseph_stabilisation:
Boolean to apply additional calculations in filter to ensure numerical stability.

9.9 troposphere:

```
    troposphere:
        model:       vmf3    #gpt2
        vmf3dir:     grid5/
        orography:   orography_ell_5x5
        # gpt2grid: EX03/general/gpt_25.grd
```

Listing 9.7: troposphere:

model:
Unused

vmf3dir:
Location of vmf3 files.

orography:
Orography filename for vmf3 troposphere.

gpt2grid:
Name of gpt2 grid file. Will be used as fallback in case of errors with vmf3.

9.10 ionosphere:

corr_mode:
Ionosphere correction/model mode. May be one of:

- broadcast - broadcast model
- sbas - SBAS model
- iono_free_linear_combo - L1/L2 or L1/L5 iono-free LC
- estimate - estimation
- total_electron_content - IONEX TEC model
- qzs - QZSS broadcast model
- lex - QZSS LEX ionospehre
- stec - SLANT TEC model

iflc_freqs:
Frequency pairs to be used in ionosphere-free linear combinations. May be one of:

- any
- l1l2_only
- l1l5_only

9.11 unit_test_options:

output_pass:
Boolean to print pass messages in output file when tests pass. This may produce very long test files for not much benefit if we are just looking for failures.

stop_on_fail:
Boolean to halt processing as soon as an error is located. This may allow testing and reporting to complete far sooner if there is a failure.

stop_on_done:
Boolean to halt further processing if all required tests have been completed.

output_errors:
Boolean to print debug information about the error to the output file.

absorb_errors:
Boolean to replace incorrect values found in processing with the correct test values and continue processing as if the test had passed. This may be useful for preventing a single bad test from causing cascading test failures as the values diverge from the original result.

directory, filename:
File and directory to store and open test files.

9.12 ionosphere_filter_parameters:

model:
Model to use in ionosphere routines. May be one of:

- meas_out
- bspline
- spherical_caps
- spherical_harmonics

model_noise:
Process noise to be applied to ionosphere kalman filter. (deprecated)

lat_center, lon_center:
Longitude and latitude of center of ionosphere map in degrees.

lat_width, lon_width:
Width of ionosphere maps in degrees.

lat_res, lon_res:
Resolution of ionosphere maps in degrees.

time_res:
Resolution of ionosphere maps in time.

func_order:
Order of Legendre function used in spherical caps ionosphere.

layer_heights:
List of heights of modelled ionosphere layers.

9.13 output_options:
This section specifies values to be used in the generation of output files.

```
output_options:

    config_description:             ex11
    analysis_agency:                GAA
    analysis_center:                Geoscience Australia
    analysis_program:               AUSACS
    rinex_comment:                  AUSNETWORK1
```

Listing 9.8: output_options:

config_description:
The value entered here is used to complete the <CONFIG> wildcard. This may enable a single change in the yaml file to make changes to many options, including output folders and filenames.

analysis_agency, analysis_center, analysis_program, rinex_comment:
String to be written within files during various output files' generation.

9.14 user_filter_parameters, network_filter_parameters, ionosphere_filter_parameters:

```
user_filter_parameters:

    max_filter_iterations:      5
    max_prefit_removals:        3

    rts_lag:                    -1      #-ve for full reverse, +ve for
   limited epochs
    rts_directory:              ./
    rts_filename:               PPP-<CONFIG>-<STATION>.rts

    inverter:                   LLT         #LLT LDLT INV
```

Listing 9.9: Kalman Filter Configuration

For details on the configuration of kalman filters refer to **??**, and **??**

9.15 default_filter_parameters:

```
default_filter_parameters:

    stations:

        error_model:        elevation_dependent         #uniform
    elevation_dependent
        code_sigmas:        [0.15]
        phase_sigmas:       [0.0015]

        pos:
            estimated:          true
            sigma:              [0.1]
            proc_noise:         [0.00057] #0.57 mm/sqrt(s), Gipsy default
    value from slow-moving
            proc_noise_dt:      second
```

Listing 9.10: Default_filter_parameters

error_model:
The GNSS observations can be weighted in three different ways in the PEA:

- uniform - all observations are assigned the same variance
- elevation_dependent - an elevation dependent function is used to scale the observations, those at higher elevation are given more weight (a smaller standard deviation) than those observed at lower elevation
- SNR observations (coming soon) - the Carrier to Noise observations supplied by the receiver are used to determine the observation weight. Generally speaking this is very similar to elevation weighting, but is useful when use observations obtained from a non-geodetic grade receiver/antenna.

code_sigmas, phase_sigmas:
Lists of the default sigma values for GNSS measurements, measured in meters. Separate values may be entered for L1, L2 frequencies if desired, or the last value will be used for any undefined values in the list.

pos, clk, amb, trop...:
For details on the configuration of estimated elements refer to **??**

9.16 minimum_constraints:
For details on the configuration of minimum constraints refer to **??**

9.17 ambiguity_resolution_options:
Min_elev_for_AR:
Minimum elevation to perform ambiguity resolution (degrees)

Set_size_for_lambda:
Candidate set size for lambda.

Max_round_iterat:
Maximum number of iterations when performing integer rounding.

GPS_amb_resol, GLO_amb_resol, GAL_amb_resol, BDS_amb_resol, QZS_amb_resol:
Booleans to enable the resolution of ambiguities for system X.

WL_mode, NL_mode:
Mode of ambiguity resolution for widelanes and narrowlanes.

May be one of the following:

- off
- round
- iter_rnd
- bootst
- lambda
- lambda_alt
- lambda_alt2
- lambda_bie

WL_succ_rate_thres, NL_succ_rate_thres:
Threshold for success rate test in LAMBDA. Values between 0 and 1 are valid.

WL_sol_ratio_thres, NL_sol_ratio_thres:
Thresholds for ambiguity validation: Ratio is performance of best solution compared to next best performance set. (greater than 1)

WL_procs_noise_sat, WL_procs_noise_sta:
Process noise applied for stations or satellites in the ambiguity resolution computations.

NL_proc_start:
Time before starting to calculate (and output) NL ambiguities/biases (in seconds)

read_OSB, read_DSB, read_SSR, read_satellite_bias, read_station_bias, read_GLONASS_IFB:
Booleans to enable reading of bias types from file.

write_OSB, write_DSB, write_SSR_bias, write_satellite_bias, write_station_bias:
Booleans to enable writing of bias types to file.

10 Attribution

Ginan - Analysis Centre Software

A project funded as Part of the Positioning Australia program.

Geoscience Australia

https://www.ga.gov.au/scientific-topics/positioning-navigation/positioning-australia

clientservices@ga.gov.au

Cnr Jerrabomberra Ave and Hindmarsh Drive

Symonston ACT 2609

Australia