Parallel.GAMIT

Release 1.0.0

Demián D. Gómez

Nov 08, 2024

CONTENTS:

CHAPTER

ONE

INTRODUCTION

Parallel.GAMIT is a Python software solution for parallel GPS processing of large regional or global networks. It also incorporates a metadata and RINEX data management tool that guarantees a consistent archive. It relies on Postgres SQL (https://www.postgresql.org/) to store station metadata and the GPSPACE Precise-Point-Positioning (PPP) software (not included in this repository, available here: https://github.com/CGS-GIS/GPSPACE) to obtain reliable daily a-priori coordinates for GAMIT.

The software is divided into two modules: Parallel.GAMIT (PG) and Parallel.PPP (PP). PG requires all GAMIT-GLOBK (http://www-gpsg.mit.edu/~simon/gtgk/) dependencies installed in the processing nodes. PP requires GPSPACE PPP and several other dependencies detailed later in this document. Although PP was designed to use GPSPACE PPP, it can be easily changed to use any other open source PPP software such as RTKLIB (http://www.rtklib.com/), although this has not been tested.

PG uses dispy (https://github.com/pgiri/dispy) to create Python pickles that are sent to local or remote nodes for execution. PG has the ability to split a network of GPS stations into subnetworks for processing in GAMIT (when the network is larger than 50 stations, depending on PG's configuration). The parallel execution is performed per day-subnetwork. In other words, a GAMIT pickle is built for each subnetwork-day being processed and sent to the available nodes. At the end of each PG run, the subnetworks are combined with GLOBK and inserted as records in the Postgres database for later use. Some routines (such as the SINEX parser) are modified versions of the code from @softwarespartan (https://github.com/softwarespartan).

PP is a Python wrapper for the PGSPACE PPP which uses the same Postgres SQL database to store the daily PPP solutions and medatadata of all station-days in the GPS archive. Some of the abilities of PP are:

- Scan a directory structure containing RINEX files and add them to the Postgres database (DB).
- Manage station metadata in GAMIT's station info format with consistency check of the records.
- Add new RINEX data to the database by geolocation, i.e. the data is incorporated not by station name but by running PPP and finding the corresponding station in the DB. This avoids problems with duplicate station codes and misidentified RINEX files.
- Handle ocean loading coefficients to correct the PPP coordinates and produce consistent time series before running GAMIT. This allows to find problems in the metadata BEFORE executing a long GAMIT run.
- Plot PPP time series using Bevis and Brown's (2014) extended trajectory model.
- Merge stations with different names that in reality are the same station (but renamed or moved a couple of meters), if desired.
- Merge, delete and add metadata directly from GAMIT station info files or using UNAVCO's GSAC (https://www.unavco.org/software/data-management/gsac/user-info/user-info.html).
- Parse all ZTD results and store them in the database.
- Stack the GAMIT solution to produce regional or global reference frames following Bevis and Brown's (2014).

- Both PP and PG tolerate station name duplicates by using a three-letter network code. Although this is not supported by GAMIT, PG converts duplicate station codes (stored in different networks) to unique IDs that are used during processing, which are later converted back to the original names after the GLOBK combination of the subnetworks.
- Because all the information is stored in a relational database, PP and PG can handle very large datasets very easily (it has been tested with ~ 5,600,000 station-days but Postgres can easily handle more than 10 million records in a regular computer). Also, the relational database guarantees then consistency of the data and does not allow accidental duplicates in metadata.

PG and PP require the following dependencies:

- Python version > 3
- GAMIT-GLOBK: although PP does not use GAMIT to process data, it relies on grdtab, otl.grid and sh_rx2apr to obtain the ocean loading coefficients and station coordinates (when PPP fails to process a station-day). Bare in mind that sh_rx2apr needs the following dependencies to run in a computer without GAMIT installed: svdiff, svpos, tform, sh_rx2apr, doy
- gfzrnx: RINEX quality check and conversion tool which supports RINEX 3.
- pygressql: Python interface to connect to Postgres
- tqdm: a Python progress bar to show the processing progress
- rnx2crx: RINEX to CRINEX
- crx2rnx: CRINEX to RINEX
- crz2rnx: this is a script modified by me which is based on the the scripts found in http://terras.gsi.go.jp/ja/crx2rnx.html with a few minor tweaks to handle the most common problems found in CRINEZ files.
- rnx2crz: the regular C-shell script
- compress/gzip
- dispy to schedule parallel jobs
- matplotlib
- numpy
- scandir
- Neicio: the USGS NEIC Python interface and its dependencies found in https://github.com/usgs/neicio

CHAPTER

TWO

PARALLEL.GAMIT

2.1 Parallel.GAMIT

2.1.1 com package

Submodules

com.AlterETM module

class com.AlterETM.**SmartFormatter**(*prog*, *indent_increment=2*, *max_help_position=24*, *width=None*)

Bases: HelpFormatter

com.AlterETM.**apply_change**(*cnn*, *station*, *tpar*, *soln*)

com.AlterETM.**insert_modify_param**(*parser*, *cnn*, *stnlist*, *args*)

com.AlterETM.**main**()

com.AlterETM.**print_params**(*cnn*, *stnlist*)

AlterETM.py - CLI interface

Program to alter the default ETM parameters for each station. The command can be executed on several stations at the same time. It is also possible to alter parameters for PPP and GAMIT simultaneously.

```
AlterETM.py [-h] [-fun function [argument ...]] [-soln {ppp,gamit} [{ppp,gamit} ...]]
            [-print]
            all|net.stnm [all|net.stnm ...]
```

AlterETM.py positional arguments

- ***all|net.stnm*** - List of networks/stations to process given in [net].[stnm] format or just [stnm] (separated by spaces; if [stnm] is not unique in the database, all stations with that name will be processed). Use keyword `'all'` to process all stations in the database. If [net].all is given, all stations from network [net] will be processed. Three letter ISO 3166 international standard codes can be provided (always in upper case) to select all stations within a country. If a station name is given using a * in front (e.g. *igs.pwro or *pwro) then the station will be removed from the list. If *net.all or ISO country code was used (e.g. *igs.all or *ARG), then remove the stations within this group. Wildcards are accepted using the regex postgres convention. Use [] to provide character ranges (e.g. ars.at1[3-5] or ars.[a-b]x01). Char %% matches any string (e.g. ars.at%%). Char | represents the OR operator that can be used to select one string or another (e.g. ars.at1[1|2] to choose at11 and at12). To specify a wildcard using a single character, use _ (equivalent to ? in POSIX regular expressions). Alternatively, a file with the station list can be provided (using all the same conventions described above). When using a file, * can be replaced with - for clarity in removing stations from .all lists (default: None)

AlterETM.py options

- ***-h***, ***--help*** - show this help message and exit
- ***-fun*** FUNCTION, ***--function_type*** FUNCTION - R|Specifies the type of function to work with. Can be polynomial (p), jump (j), periodic (q) or bulk earthquake jump removal (t). Each one accepts a list of arguments. p {terms} where terms equals the number of polynomial terms in the ETM, i.e. terms = 2 is constant velocity and terms = 3 is velocity + acceleration, etc. j {action} {type} {date} {relax} where action can be + or -. A + indicates that a jump should be added while a - means that an existing jump should be removed; type = 0 is a mechanic jump and 1 is a geophysical jump; date is the date of the event in all the accepted formats (yyyy/mm/dd yyyy_doy gpswk-wkday fyear); and relax is a list of relaxation times for the logarithmic decays (only used when type = 1, they are ignored when type = 0). q {periods} where periods is a list expressed in days (1 yr = 365.25). t {max_magnitude} {stack_name} removes any earthquake Mw <= max_magnitude from the specified stations' trajectory models; if GAMIT solutions are invoked, provide the stack_name to obtain the ETMs of the stations. m {stack_name} [start_date] [end_date|days] removes mechanical jumps between given dates from the specified stations' trajectory models; if no dates are provided, remove all mechanical jumps. If only first date is provided, remove starting at that date until today. Can also specify {start_date} {days} to add to {start_date}. Provide the stack_name to obtain the ETMs of the stations. If PPP solutions only (-soln ppp), stack_name is ignored. If both solutions are indicated (or -soln is not specified) then stack_name must be provided. (default: [])
- ***-soln*** SOLUTION_TYPE, ***--solution_type*** SOLUTION_TYPE - Specifies the type of solution that this command will affect. If left empty, the ETMs for both PPP and GAMIT will be affected. Otherwise, specify gamit to insert or remove the function on GAMIT ETMs only or ppp to insert or remove the function on PPP ETMs only. (default: ['ppp', 'gamit'])
- ***-print***, ***--print_params*** - Print the parameters present in the database for the selected stations.

com.ApplyCountryCode module

Project: Parallel.GAMIT Date: 11/20/2023 Author: Demian D. Gomez

This script assigns country codes to the stations table

com.ApplyCountryCode.**main**()

com.ArchiveService module

Project: Parallel.Archive Date: 3/19/17 11:41 AM Author: Demian D. Gomez

ArchiveService

Main script that scans the repository for new rinex files. It PPPs the rinex files and searches the database for stations

(within configured distance in stations table)

with the same station 4 letter code. If the station exists in the db, it moves the file to the archive

and adds the new file to the "rinex" table.

if the station doesn't exist, then it incorporates the station
with a special NetworkCode (???) and leaves the

file in the repo until you assign the correct NetworkCode and
add the station information.

It is invoked just by calling python ArchiveService.py Requires the config file gnss_data.cfg (in the running folder)

Options: –purge_locks: deletes any locked files from repository and database –no_parallel: runs without parallelizing the execution

com.ArchiveService.**callback_handle**(*job*)

com.ArchiveService.**check_rinex_timespan_int**(*rinex*, *stn*)

com.ArchiveService.**error_handle**(*cnn*, *event*, *crinez*, *folder*, *filename*, *no_db_log=False*)

com.ArchiveService.**insert_data**(*cnn*, *archive*, *rinexinfo*)

com.ArchiveService.**insert_station_w_lock**(*cnn*, *StationCode*, *filename*, *lat*, *lon*, *h*, *x*, *y*, *z*, *otl*)

com.ArchiveService.**main**()

com.ArchiveService.**print_archive_service_summary**()

com.ArchiveService.**process_crinex_file**(*crinez*, *filename*, *data_rejected*, *data_retry*)

com.ArchiveService.**remove_empty_folders**(*folder*)

com.ArchiveService.**verify_rinex_multiday**(*cnn*, *rinexinfo*, *Config*)

com.ArchiveService.**write_error**(*folder*, *filename*, *msg*)

ArchiveService.py - CLI interface

Archive operations Main Program

```
ArchiveService.py [-h] [-purge] [-np]
```

ArchiveService.py options

- ***-h***, ***--help*** - show this help message and exit
- ***-purge***, ***--purge_locks*** -

  from the stations table and purge the contents of the locks table, deleting the associated files from data_in.
- ***-np***, ***--noparallel*** - Execute command without parallelization.

com.CloseStationInfo module

Project: Parallel.GAMIT Date: 08/30/22 12:44 PM Author: Demian D. Gomez

Script to close the StationInfo of a station that has not been collecting data for X time

com.CloseStationInfo.**main**()

CloseStationInfo.py - CLI interface

Close an opened station information record for a station using the last available RINEX file date time

```
CloseStationInfo.py [-h] all|net.stnm [all|net.stnm ...]
```

CloseStationInfo.py positional arguments

- ***all|net.stnm*** -

in [net].[stnm] format or just [stnm] (separated by spaces; if [stnm] is not unique in the database, all stations with that name will be processed). Use keyword `'all'` to process all stations in the database. If [net].all is given, all stations from network [net] will be processed. Alternatively, a file with the station list can be provided.

(default: `None`)

CloseStationInfo.py options

- ***-h***, ***--help*** - show this help message and exit

com.CompareDBs module

com.ConvertDate module

Project: Parallel.GAMIT Date: Jul 20 2023 11:40 AM Author: Demian D. Gomez Script to convert from one date type to others

`com.ConvertDate.`**`main`**`()`

ConvertDate.py - CLI interface

Convert from one date type to others

```
ConvertDate.py [-h] date to convert
```

ConvertDate.py positional arguments

- ***date to convert*** -

  are yyyy/mm/dd yyyy_doy wwww-d format

  (default: `None`)

ConvertDate.py options

- ***-h***, ***--help*** - show this help message and exit

com.ConvertTrimble module

Project: Parallel.GAMIT Date: 10/28/2022 Author: Demian D. Gomez Script to convert T0x files to RINEX

`com.ConvertTrimble.`**`main`**`()`

ConvertTrimble.py - CLI interface

Script to convert T0x files to RINEX

```
ConvertTrimble.py [-h] [-stnm STATION_NAME] [-ant {atx_file} {antenna_name} [SN]
                  [{atx_file} {antenna_name} [SN] ...]]
                  [path to dir] [path to dir]
```

ConvertTrimble.py positional arguments

- ***[path to dir]*** - Path to directory with T0x files (default: `None`)
- ***[path to dir]*** -

(a folder with station name will be created)

(default: None)

ConvertTrimble.py options

- ***-h***, ***--help*** - show this help message and exit
- ***-stnm*** STATION_NAME, ***--station_name*** STATION_NAME - Name of the station to form that RINEX files (default: dftl)
- ***-ant*** {ATX_FILE} {ANTENNA_NAME} [SN], ***--antenna_name*** {ATX_FILE} {ANTENNA_NAME} [SN] -

  with name provided in {antenna_name}. Antenna has to exist in ATX file provided in {atx_file}. Radome will be set to NONE by default. Antenna name can have wildcards but a single match is expected (otherwise an exception will be raised). Optionally, provide the antenna serial number.

  (default: None)

com.DRA module

Project: Parallel.Stacker Date: 6/12/18 10:28 AM Author: Demian D. Gomez

class com.DRA.**DRA**(*cnn*, *project*, *start_date*, *end_date*, *verbose=False*)

Bases: list

get_station(*NetworkCode*, *StationCode*)

Obtains the time series for a given station :param NetworkCode: :param StationCode: :return: a numpy array with the time series [x, y, z, yr, doy, fyear]

stack_dra()

to_json(*json_file*)

com.DRA.**callback_handler**(*job*)

com.DRA.**compute_dra**(*ts*, *NetworkCode*, *StationCode*, *pdates*, *project*, *histogram=False*)

com.DRA.**main**()

com.DRA.**sql_select**(*project*, *fields*, *date2*)

DRA.py - CLI interface

GNSS daily repetitivities analysis (DRA)

```
DRA.py [-h] [-d date [date ...]] [-w date [date ...]] [-hist] [-v] [-np] {project name}
```

DRA.py positional arguments

- ***{project name}*** -

  the GAMIT solutions in Parallel.GAMIT.

  (default: None)

DRA.py options

- *-h*, *--help* - show this help message and exit
- *-d* DATE, *--date_filter* DATE -

  yyyy/mm/dd yyyy_doy wwww-d format

  (default: None)
- *-w* DATE, *--plot_window* DATE -

  in yyyy/mm/dd yyyy_doy wwww-d format

  (default: None)
- *-hist*, *--histogram* - Plot a histogram of the daily repetitivities
- *-v*, *--verbose* -

  alignment process (for debugging purposes)
- *-np*, *--noparallel* - Execute command without parallelization.

com.DownloadSources module

Project: Parallel.GAMIT Date: 11/8/17 9:24 AM Author: Demian D. Gomez

Script to download stations Rinex files from external servers for a given date range to the directory specified in:

[Config.repository]/data_in

Runs scripts stored in:
[Config.format_scripts_path]

class com.DownloadSources.**Client**(*on_download_result*, *on_client_stopped*, *server_id: int*, *protocol*, *host*, *port*, *username*, *password*)

Bases: object

class NextDownload(*urlpath_file*, *abspath_down_file*)

Bases: NamedTuple

abspath_down_file: str

Alias for field number 1

urlpath_file: str

Alias for field number 0

cond: Condition

finish()

next_download: *NextDownload* **| None**

proto: *IProtocol*

server_id: int

set_next_download(*urlpath_file: str*, *abspath_down_file: str*)

start_thread()

state: str

stop()

class com.DownloadSources.**File**(*stn_idx*, *src_idx*, *date_mjd*, *station*, *source*, *date*, *urlpath_file*, *filename*, *abspath_down_file*, *desc*, *url*, *src_desc*)

Bases: NamedTuple

abspath_down_file: str

Alias for field number 8

date: *Date*

Alias for field number 5

date_mjd: int

Alias for field number 2

desc: str

Alias for field number 9

filename: str

Alias for field number 7

static from_descriptor(*stations*, *fd:* FileDescriptor)

static from_params(*stations*, *stn_idx: int*, *date_mjd: int*, *src_idx: int*)

source: *Source*

Alias for field number 4

src_desc: str

Alias for field number 11

src_idx: int

Alias for field number 1

station: *Station*

Alias for field number 3

stn_idx: int

Alias for field number 0

to_descriptor()

url: str

Alias for field number 10

urlpath_file: str

Alias for field number 6

class com.DownloadSources.**FileDescriptor**(*stn_idx*, *src_idx*, *date_mjd*)

Bases: NamedTuple

date_mjd: int

Alias for field number 2

src_idx: int

Alias for field number 1

stn_idx: int

Alias for field number 0

class com.DownloadSources.**FilesBag**

Bases: object

class Dates

Bases: object

CHUNK_SIZE = 4096

is_empty()

pop()

push(*date_mjd: int*)

is_empty()

pop() → *FileDescriptor*

push(*f:* FileDescriptor)

class com.DownloadSources.**IProtocol**(*protocol: str*, *fqdn: str*, *port: int*, *username: str | None*, *password: str | None*)

Bases: ABC

abstract connect()

desc()

abstract disconnect()

abstract download(*server_path: str*, *dest_path: str*) → bool

abstract list_dir(*server_path: str*)

abstract refresh()

class com.DownloadSources.**JobsManager**(*job_server:* JobServer, *abspath_scripts_dir: str*)

Bases: object

Submits PROCESS jobs to cluster while minimizing dispy queue usage

on_job_result(*job*)

called by dispy

on_nodes_changed(*nodes: list*)

called by dispy

queue_process(*f:* File)

class com.DownloadSources.**Msg**

Bases: object

Messages to main thread

class CLIENT_STOPPED(*server_id*)

Bases: NamedTuple

server_id: int

Alias for field number 0

class DOWNLOAD_RESULT(*server_id*, *elapsed_time*, *size*, *error*)

Bases: NamedTuple

elapsed_time: int

Alias for field number 1

error: str | None

Alias for field number 3

server_id: int

Alias for field number 0

size: int

Alias for field number 2

class FILE_IGNORED_EXISTS_IN_DB(*file*)

Bases: NamedTuple

file: *FileDescriptor*

Alias for field number 0

class FILE_SKIPPED_INACTIVE_STATION(*file*)

Bases: NamedTuple

file: *FileDescriptor*

Alias for field number 0

class NEW_FILE(*file*)

Bases: NamedTuple

file: *FileDescriptor*

Alias for field number 0

class PROCESS_RESULT(*file*, *error*)

Bases: NamedTuple

error: str | None

Alias for field number 1

file: *FileDescriptor* | None

Alias for field number 0

class com.DownloadSources.ProtocolFTP(**args*, ***kargs*)

Bases: *IProtocol*

DEFAULT_PORT = 21

connect()

disconnect()

download(*server_path: str*, *dest_path: str*)

list_dir(*server_path: str*)

refresh()

class com.DownloadSources.ProtocolFTPA(**args*, ***kargs*)

Bases: *ProtocolFTP*

DEFAULT_PORT = 21

connect()

class com.DownloadSources.ProtocolHTTP(*args, protocol='http', **kargs)

Bases: IProtocol

DEFAULT_PORT = 80

connect()

disconnect()

download(server_path: str, dest_path: str)

list_dir(server_path: str)

refresh()

class com.DownloadSources.ProtocolHTTPS(*args, **kargs)

Bases: ProtocolHTTP

DEFAULT_PORT = 443

class com.DownloadSources.ProtocolSFTP(*args, **kargs)

Bases: IProtocol

DEFAULT_PORT = 22

connect()

disconnect()

download(server_path: str, dest_path: str)

list_dir(server_path: str)

refresh()

class com.DownloadSources.Source(server_id, protocol, fqdn, username, password, path, format)

Bases: NamedTuple

format: str | None

Alias for field number 6

fqdn: str

Alias for field number 2

password: str

Alias for field number 4

path: str

Alias for field number 5

protocol: str

Alias for field number 1

server_id: int

Alias for field number 0

username: str

Alias for field number 3

class com.DownloadSources.**Station**(*stationID*, *NetworkCode*, *StationCode*, *Marker*, *CountryCode*, *sources*, *abspath_station_dir*)

Bases: NamedTuple

CountryCode: str

Alias for field number 4

Marker: int

Alias for field number 3

NetworkCode: str

Alias for field number 1

StationCode: str

Alias for field number 2

abspath_station_dir: str

Alias for field number 6

sources: List[*Source*]

Alias for field number 5

stationID: str

Alias for field number 0

com.DownloadSources.**db_get_sources_for_station**(*cnn*, *NetworkCode*, *StationCode*) → List[*Source*]

com.DownloadSources.**db_migrate_if_needed**(*cnn*)

com.DownloadSources.**download_all_stations_data**(*cnn: Cnn*, *jobs_manager: JobsManager*, *abspath_repository_dir: str*, *stnlist: List[Any]*, *drange*)

com.DownloadSources.**main**()

com.DownloadSources.**process_file**(*abspath_scripts_dir: str*, *abspath_down_file: str*, *src_format: str*, *StationCode: str*)

com.DownloadSources.**source_host_desc**(*src: Source*)

com.DownloadSources.**thread_queue_all_files**(*cnn*, *drange*, *stations*, *msg_outbox*)

DownloadSources.py - CLI interface

Archive operations Main Program

```
DownloadSources.py [-h] [-date date_start|date_end [date_start|date_end ...]] [-win days]
                   [-np]
                   all|net.stnm [all|net.stnm ...]
```

DownloadSources.py positional arguments

- ***all|net.stnm*** -

[net].[stnm] format or just [stnm] (separated by spaces; if [stnm] is not unique in the database, all stations with that name will be processed). Use keyword `'all'` to process all stations in the database. If [net].all is given, all stations from network [net] will be processed. Alternatively, a file with the station list can be provided.

(default: `None`)

DownloadSources.py options

- ***-h***, ***--help*** - show this help message and exit
- ***-date*** `DATE_START|DATE_END`, ***--date_range*** `DATE_START|DATE_END` -

  or [date_start] and [date_end]. Allowed formats are yyyy.doy or yyyy/mm/dd..

  (default: `None`)
- ***-win*** `DAYS`, ***--window*** `DAYS` -

  determined by today - `{days}`.

  (default: `None`)
- ***-np***, ***--noparallel*** - Execute command without parallelization.

com.DownloadSourcesFill module

Project: Parallel.GAMIT Date: 11/23/2023 11:09 AM Author: Demian D. Gomez

Program to fill the sources_stations table using a probe to specific FTP servers

`com.DownloadSourcesFill.`**`main`**`()`

`com.DownloadSourcesFill.`**`query_yes_no`**`(`*question*, *default='yes'*`)`

Ask a yes/no question via raw_input() and return their answer.

"question" is a string that is presented to the user. "default" is the presumed answer if the user just hits <Enter>.

It must be "yes" (the default), "no" or None (meaning an answer is required of the user).

The "answer" return value is True for "yes" or False for "no". code obtained from https://stackoverflow.com/questions/3041986/

apt-command-line-interface-like-yes-no-input

`com.DownloadSourcesFill.`**`search_by_date`**`(`*stnlist*, *drange*, *svr*, *client*`)`

`com.DownloadSourcesFill.`**`search_by_station`**`(`*stnlist*, *drange*, *svr*, *client*`)`

DownloadSourcesFill.py - CLI interface

Probe FTP server to find stations

```
DownloadSourcesFill.py [-h] [-date date_start | date_end [date_start | date_end ...]]
                       [-source fqdm | server_id] [-skip] [-yes]
                       all|net.stnm [all|net.stnm ...]
```

DownloadSourcesFill.py positional arguments

- ***all|net.stnm*** - List of networks/stations to process given in [net].[stnm] format or just [stnm] (separated by spaces; if [stnm] is not unique in the database, all stations with that name will be processed). Use keyword `'all'` to process all stations in the database. If [net].all is given, all stations from network [net] will be processed. Three letter ISO 3166 international standard codes can be provided (always in upper case) to select all stations within

a country. If a station name is given using a * in front (e.g. *igs.pwro or *pwro) then the station will be removed from the list. If *net.all or ISO country code was used (e.g. *igs.all or *ARG), then remove the stations within this group. Wildcards are accepted using the regex postgres convention. Use [] to provide character ranges (e.g. ars.at1[3-5] or ars.[a-b]x01). Char %% matches any string (e.g. ars.at%%). Char | represents the OR operator that can be used to select one string or another (e.g. ars.at1[1|2] to choose at11 and at12). To specify a wildcard using a single character, use _ (equivalent to ? in POSIX regular expressions). Alternatively, a file with the station list can be provided (using all the same conventions described above). When using a file, * can be replaced with - for clarity in removing stations from .all lists (default: `None`)

DownloadSourcesFill.py options

- ***-h***, ***--help*** - show this help message and exit
- ***-date*** `DATE_START | DATE_END`, ***--date_range*** `DATE_START | DATE_END` -

  or [date_start] and [date_end]. If only [date_start] is given, then [date_end] is today. If an integer is provided, then [date_start] is today minus value provided. Allowed formats are wwww-d, yyyy_ddd, yyyy/mm/dd or fyear.

  (default: `None`)
- ***-source*** `FQDM | SERVER_ID`, ***--data_source*** `FQDM | SERVER_ID` -

  server_id existing in the sources_servers table to probe. If multiple sources use the same FQDM, then all will be probed to search for RINEX data matching the station list provided. Be careful with hitting a single server with too many requests!

  (default: `0`)
- ***-skip***, ***--skip_stations_with_source*** - Remove stations with sources from the search.
- ***-yes***, ***--force_yes*** -

  (without prompting yes/no).

com.FixPlate module

Project: Parallel.GAMIT Date: 6/15/24 10:29 AM Author: Demian D. Gomez

Description goes here

com.FixPlate.**build_design**(*hdata*, *vdata*)

com.FixPlate.**build_design_href**(*data*)

com.FixPlate.**build_design_vref**(*data*)

com.FixPlate.**main**()

FixPlate.py - CLI interface

Script to fix plate given a set of stations. Program outputs Euler vector parameters and optionally produces timee series in such plate-fixed frame.

```
FixPlate.py [-h] [-include {net.stnm} [{net.stnm} ...]] [-vref station [[mm/yr]] ...]]
            [-plot] [-dir {dir name}] [-save {new stack name}]
            {stack name}
```

FixPlate.py positional arguments

- ***{stack name}*** -

  will be calculated so as to fix the velocities of the selected sites in this stack.

  (default: None)

FixPlate.py options

- ***-h***, ***--help*** - show this help message and exit
- ***-include*** {NET.STNM}, ***--include_stations*** {NET.STNM} -

  to use for Euler pole computation.

  (default: None)

- ***-vref*** STATION, ***--vertical_ref*** STATION -

  frame using the provided station list and velocities, given as [net.stnm] [vu], where vu is the vertical velocity in mm/yr.

  (default: [])

- ***-plot***, ***--plot_etms*** -

  after computation is done.

- ***-dir*** {DIR NAME}, ***--directory*** {DIR NAME} -

  If not specified, assumed to be the production directory.

  (default: None)

- ***-save*** {NEW STACK NAME}, ***--save_stack*** {NEW STACK NAME} -

  frame as new stack. Switch requires a stack name to use. WARNING! If stack exists it will be overwritten.

  (default: None)

com.GenerateKml module

Project: Parallel.GAMIT Date: 7/18/18 10:28 AM Author: Demian D. Gomez

Program to generate a KML with the stations in a project and the stations out of a project

com.GenerateKml.**callback_handle**(*job*)

com.GenerateKml.**description_content**(*stn*, *DateS*, *DateE*, *count*, *completion*, *stn_issues*, *stninfo*, *stninfo_records*, *data*, *style*)

com.GenerateKml.**generate_kml**(*cnn*, *project*, *data=False*)

com.GenerateKml.**generate_kml_stninfo**(*JobServer*, *cnn*, *project*, *data=False*, *run_stninfo_check=True*, *stnonly=()*, *kmz_filename='PG.kmz'*)

com.GenerateKml.**main**()

com.GenerateKml.**plot_rinex**(*cnn*, *NetworkCode*, *StationCode*)

com.GenerateKml.**plot_station_info_rinex**(*cnn*, *NetworkCode*, *StationCode*, *stninfo_records*)

GenerateKml.py - CLI interface

Generate KML file to inspect archive in Google Earth

```
GenerateKml.py [-h] [-stn [net.stnm] [[net.stnm] ...]] [-stninfo] [-data]
               [-kmz {filename}.kmz] [-np]
               {project cfg file}
```

GenerateKml.py positional arguments

- ***{project cfg file}*** -

  being processed in Parallel.GAMIT

  (default: None)

GenerateKml.py options

- ***-h***, ***--help*** - show this help message and exit
- ***-stn*** [NET.STNM], ***--station_list*** [NET.STNM] -

  Default returns all stations. Network and station codes allow using wildcards.

  (default: [])

- ***-stninfo***, ***--station_info*** -

  and output results to kmz file. The icons of the stations will represent any problems in the station info records.

- ***-data***, ***--available_data*** - Produce detailed plots with available data.
- ***-kmz*** {FILENAME}.KMZ, ***--kmz_filename*** {FILENAME}.KMZ -

  append the extension). Default uses production/{project_name} where {project_name} is the project name declared in the PG cfg file.

  (default: None)

- ***-np***, ***--noparallel*** - Execute command without parallelization.

com.GenerateSinex module

Project: Parallel.GAMIT Date: 7/18/18 10:28 AM Author: Demian D. Gomez

Program to load a SINEX solution created using glbtosnx and add the missing unknown parameters (tropospheric delays and gradients)

com.GenerateSinex.**add_domes**(*sinex*, *stations*)

com.GenerateSinex.**main**()

com.GenerateSinex.**process_sinex**(*cnn*, *project*, *dates*, *sinex*)

com.GenerateSinex.**replace_in_sinex**(*sinex*, *observations*, *unknowns*, *new_val*)

GenerateSinex.py - CLI interface

GNSS time series stacker

```
GenerateSinex.py [-h] [-d date [date ...]] {project name} {project name}
```

GenerateSinex.py positional arguments

- ***{project name}*** - Specify the project name used to process the GAMIT solutions in Parallel.GAMIT. (default: None)
- ***{project name}*** - SINEX file to update. (default: None)

GenerateSinex.py options

- ***-h***, ***--help*** - show this help message and exit
- ***-d*** DATE, ***--date_filter*** DATE - Date range filter can be specified in yyyy/mm/dd yyyy_doy wwww-d format (default: None)

com.IntegrityCheck module

Project: Parallel.Archive Date: 3/27/17 11:54 AM Author: Demian D. Gomez

Integrity check utility of the database. Checks the following:

- Station info consistency
- Proposes a station info based on RINEX data
- Searches for data gaps in the rinex table
- Prints the station info records
- renames or merges two stations into one

com.IntegrityCheck.**CheckRinexIntegrity**(*cnn*, *Config*, *stnlist*, *start_date*, *end_date*, *operation*, *JobServer*)

com.IntegrityCheck.**CheckSpatialCoherence**(*cnn*, *stnlist*, *start_date*, *end_date*)

com.IntegrityCheck.**DeleteRinex**(*cnn*, *stnlist*, *start_date*, *end_date*, *completion_limit=0.0*)

com.IntegrityCheck.**ExcludeSolutions**(*cnn*, *stnlist*, *start_date*, *end_date*)

com.IntegrityCheck.**GetGaps**(*cnn*, *NetworkCode*, *StationCode*, *start_date*, *end_date*)

com.IntegrityCheck.**GetStnGaps**(*cnn*, *stnlist*, *ignore_val*, *start_date*, *end_date*)

com.IntegrityCheck.**PrintStationInfo**(*cnn*, *stnlist*, *short=False*)

com.IntegrityCheck.**RenameStation**(*cnn*, *NetworkCode*, *StationCode*, *DestNetworkCode*, *DestStationCode*, *start_date*, *end_date*, *archive_path*, *delete_if_empty=True*)

com.IntegrityCheck.**RinexCount**(*cnn*, *stnlist*, *start_date*, *end_date*)

com.IntegrityCheck.**StnInfoCheck**(*cnn*, *stnlist*, *Config*)

com.IntegrityCheck.**StnInfoRinexIntegrity**(*cnn*, *stnlist*, *start_date*, *end_date*, *JobServer*)

com.IntegrityCheck.**VisualizeGaps**(*cnn*, *stnlist*, *start_date*, *end_date*)

com.IntegrityCheck.**check_rinex_stn**(*NetworkCode*, *StationCode*, *start_date*, *end_date*)

com.IntegrityCheck.**compare_stninfo_rinex**(*NetworkCode*, *StationCode*, *STime*, *ETime*, *rinex_serial*)

com.IntegrityCheck.**get_differences**(*differences*)

com.IntegrityCheck.**main**()

com.IntegrityCheck.**stnrnx_callback**(*job*)

IntegrityCheck.py - CLI interface

Database integrity tools, metadata check and fixing tools program

```
IntegrityCheck.py [-h] [-d date [date ...]] [-rinex {fix,report}] [-rnx_count] [-stnr]
                  [-stns] [-stnp [ignore_days]] [-stnc] [-g [ignore_days]] [-gg]
                  [-sc {exclude,delete,noop}] [-print {long,short}] [-r net.stnm] [-del_
↪stn]
                  [-es {start_date} {end_date}] [-del {start_date} {end_date}
↪{completion}]
                  [-np]
                  all|net.stnm [all|net.stnm ...]
```

IntegrityCheck.py positional arguments

- ***all|net.stnm*** - List of networks/stations to process given in [net].[stnm] format or just [stnm] (separated by spaces; if [stnm] is not unique in the database, all stations with that name will be processed). Use keyword `'all'` to process all stations in the database. If [net].all is given, all stations from network [net] will be processed. Three letter ISO 3166 international standard codes can be provided (always in upper case) to select all stations within a country. If a station name is given using a * in front (e.g. *igs.pwro or *pwro) then the station will be removed from the list. If *net.all or ISO country code was used (e.g. *igs.all or *ARG), then remove the stations within this group. Wildcards are accepted using the regex postgres convention. Use [] to provide character ranges (e.g. ars.at1[3-5] or ars.[a-b]x01). Char %% matches any string (e.g. ars.at%%). Char | represents the OR operator that can be used to select one string or another (e.g. ars.at1[1|2] to choose at11 and at12). To specify a wildcard using a single character, use _ (equivalent to ? in POSIX regular expressions). Alternatively, a file with the station list can be provided (using all the same conventions described above). When using a file, * can be replaced with - for clarity in removing stations from .all lists (default: `None`)

IntegrityCheck.py options

- ***-h***, ***--help*** - show this help message and exit
- ***-d*** `DATE`, ***--date_filter*** `DATE` - Date range filter for all operations. Can be specified in wwww-d, yyyy_ddd, yyyy/mm/dd or fyear format (default: `None`)
- ***-rinex*** `CHECK_RINEX`, ***--check_rinex*** `CHECK_RINEX` - Check the RINEX integrity of the archive-database by verifying that the RINEX files reported in the rinex table exist in the archive. If argument = `"fix"` and a RINEX file does not exist, remove the record. PPP records or gamit_soln are deleted. If argument = `"report"` then just list the missing files. (default: `None`)
- ***-rnx_count***, ***--rinex_count*** - Count the total number of RINEX files (unique station-days) per day for a given time interval.
- ***-stnr***, ***--station_info_rinex*** - Check that the receiver serial number in the rinex headers agrees with the station info receiver serial number.
- ***-stns***, ***--station_info_solutions*** - Check that the PPP hash values match the station info hash.
- ***-stnp*** `IGNORE_DAYS`, ***--station_info_proposed*** `IGNORE_DAYS` - Output a proposed station.info using the RINEX metadata. Optional, specify [ignore_days] to ignore station.info records <= days. (default: `None`)
- ***-stnc***, ***--station_info_check*** - Check the consistency of the station information records in the database. Date range does not apply. Also, check that the RINEX files fall within a valid station information record.
- ***-g*** `IGNORE_DAYS`, ***--data_gaps*** `IGNORE_DAYS` - Check the RINEX files in the database and look for gaps (missing days). Optional, [ignore_days] with the smallest gap to display. (default: `None`)
- ***-gg***, ***--graphical_gaps*** - Visually output RINEX gaps for stations.

- ***-sc*** SPATIAL_COHERENCE, ***--spatial_coherence*** SPATIAL_COHERENCE - Check that the RINEX files correspond to the stations they are linked to using their PPP coordinate. If keyword [exclude] or [delete], add the PPP solution to the excluded table or delete the PPP solution. If [noop], then only report but do not exlude or delete. (default: None)
- ***-print*** PRINT_STNINFO, ***--print_stninfo*** PRINT_STNINFO - Output the station info to stdout. [long] outputs the full line of the station info. [short] outputs a short version (better for screen visualization). (default: None)
- ***-r*** NET.STNM, ***--rename*** NET.STNM - Takes the data from the station list and renames (merges) it to net.stnm. It also changes the rinex filenames in the archive to match those of the new destiny station. Only a single station can be given as the origin and destiny. Limit the date range using the -d option. If origin station is empty (no RINEX files) after rename/merge, the station is deleted from the database if –delete_station is invoked. (default: None)
- ***-del_stn***, ***--delete_station*** - Switch to enable station deletion after rename/merge operation. Only works when invoking –rename.
- ***-es*** {START_DATE}, ***--exclude_solutions*** {START_DATE} - Exclude PPP solutions (by adding them to the excluded table) between {start_date} and {end_date} (default: None)
- ***-del*** {START_DATE}, ***--delete_rinex*** {START_DATE} - Delete RINEX files (and associated solutions, PPP and GAMIT) from archive between {start_date} and {end_date} with completion <= {completion}. Completion ranges form 1.0 to 0.0. Use 1.0 to delete all data. Operation cannot be undone! (default: None)
- ***-np***, ***--noparallel*** - Execute command without parallelization.

com.LocateRinex module

com.LocateRinex.**execute_ppp**(*rinexinfo*, *args*, *stnm*, *options*, *sp3types*, *sp3altrn*, *brdc_path*, *erase*, *apply_met=True*, *decimate=True*, *fix_coordinate=None*, *solve_troposphere=105*, *copy_results=None*, *backward_substitution=False*, *elevation_mask=5*, *code_only=False*)

com.LocateRinex.**main**()

LocateRinex.py - CLI interface

Simple PPP python wrapper. Calculate a coordinate for a RINEX file. Output one line per file with stnm epoch x y z lat lon h

```
LocateRinex.py [-h] [-otl] [-ns] [-no_met] [-dec] [-rnx] [-ins [[net]]] [-find] [-ne] [-
↪back]
               [-fix coordinate_file | x y z [coordinate_file | x y z ...]]
               [-st {1,2,3,4,5,102,103,104,105}] [-elv ELEVATION_MASK]
               [-min MIN_TIME_SECONDS] [-code] [-c storage_dir]
               [-nocfg sp3_directory sp3_types brdc_directory]
               files [files ...]
```

LocateRinex.py positional arguments

- ***files*** - List of files, directories or wildcards to process. If directories are given, searches for .Z files. Individual files or wildcards can be either .Z or ??o. Eg: LocationRinex.py ./igm10010.10d.Z ./igm1002a.10o ./cs*.Z ./rinex2process/ (default: None)

LocateRinex.py options

- **-h**, **--help** - show this help message and exit
- **-otl**, **--ocean_loading** - Apply ocean loading coefficients (obtained from grdtab).
- **-ns**, **--no_split** - Do not split multiday RINEX files and obtain a single coordinate.
- **-no_met**, **--no_met** - Do not apply the GPT2 model to correct tropospheric delays (use GPT).
- **-dec**, **--decimate** - Decimate RINEX to 30 s if interval < 15.
- **-rnx**, **--load_rinex** - Fix RINEX using pyRinex, create a local copy (with session number+1) and exit. Do not run PPP.
- **-ins** [NET], **--insert_sql** [NET] - Produce a SQL INSERT statement for this station including OTL and coordinates. If a network code [net] is specified, then produce the insert in the database. Network code will be inserted if it does not exist. If [net] is not specified, then the command will only output the insert statement with ??? as the network code. (default: False)
- **-find**, **--find** - Find the matching station in the db using the spatial location algorithm.
- **-ne**, **--no_erase** - Do not erase PPP folder structure after completion.
- **-back**, **--backward_substitution** - Run PPP with backward substitution.
- **-fix** COORDINATE_FILE | X Y Z, **--fix_coordinate** COORDINATE_FILE | X Y Z - Do not solve for station coordinates, fix station position as given in [coordinate_file] or provide a list of X Y Z coordinates. File should contain the apriori coordinates as a list starting with the station name and the X Y Z coordinates. For example: OSU1 595355.1776 -4856629.7091 4077991.9857 (default: None)
- **-st** SOLVE_TROPOSPHERE, **--solve_troposphere** SOLVE_TROPOSPHERE - Solve for the tropospheric wet delay. Possible options are 1: do not solve, 2-5: solve without gradients (number determine the random walk in mm/hr), +100: solve gradients. (default: 105)
- **-elv** ELEVATION_MASK, **--elevation_mask** ELEVATION_MASK - Elevation mask (default=10). (default: 10)
- **-min** MIN_TIME_SECONDS, **--min_time_seconds** MIN_TIME_SECONDS - Minimum observation time in seconds for observations (default=3600). (default: 3600)
- **-code**, **--code_only** - Run PPP using only code (C1) observations.
- **-c** STORAGE_DIR, **--copy_results** STORAGE_DIR - Copy the output files (.ses, .sum, .res, .pos) to [storage_dir]. A folder with the station name will be created in [storage_dir]. (default: None)
- **-nocfg** SP3_DIRECTORY, **--no_config_file** SP3_DIRECTORY - Do not attempt to open gnss_data.cfg. Append [sp3_directory], [sp3_types] and [brdc_directory] to access the precise and broadcast orbit files. Use the keywords $year, $doy, $month, $day, $gpsweek, $gpswkday to dynamically replace with the appropriate values (based on the date in the RINEX file). Grdtab and otl_grid should have the standard names if -otl is invoked and ppp should be in the PATH (with executable name = ppp). (default: None)

com.NEQStack module

Project: Parallel.Stacker Date: 6/12/18 10:28 AM Author: Demian D. Gomez

com.NEQStack.**adjust_lsq**(*A*, *L*, *P=None*)

com.NEQStack.**dra**(*cnn*, *project*, *dates*)

com.NEQStack.**main**()

com.NEQStack.**rotate_sigmas**(*ecef*, *lat*, *lon*)

com.NEQStack.**sql_select**(*project*, *fields*, *date2*)

com.NEQStack.**sql_select_union**(*project*, *fields*, *date1*, *date2*, *stn_filter=None*)

NEQStack.py - CLI interface

GNSS time series stacker

```
NEQStack.py [-h] [-d date [date ...]] {project name}
```

NEQStack.py positional arguments

- ***{project name}*** - Specify the project name used to process the GAMIT solutions in Parallel.GAMIT. (default: None)

NEQStack.py options

- ***-h***, ***--help*** - show this help message and exit
- ***-d*** DATE, ***--date_filter*** DATE - Date range filter Can be specified in yyyy/mm/dd yyyy_doy wwww-d format (default: None)

com.OTL_FES2014b module

Project: Parallel.Archive Date: 02/16/2017 Author: Demian D. Gomez

Ocean loading coefficients class. It runs and reads grdtab (from GAMIT).

Subject: Ocean Loading Tides Header = —- Ocean loading values follow next: —- Model = FES2014b Loading-Type = displacement GreensF = mc00egbc CMC = 1 Plot = 0 OutputFormat = BLQ Stations = %s MyEmail = demi-ang@gmail.com

com.OTL_FES2014b.**create_files**(*stnlist*)

com.OTL_FES2014b.**import_blq**(*filename*)

com.OTL_FES2014b.**import_harpos**(*filename*)

com.OTL_FES2014b.**load_blq**(*header*, *otl*)

com.OTL_FES2014b.**load_harpos**(*header*, *otl*)

com.OTL_FES2014b.**main**()

com.ParallelGamit module

Project: Parallel.GAMIT Date: 3/31/17 6:33 PM Author: Demian D. Gomez

class com.ParallelGamit.**DbAlive**(*cnn*, *increment*)

Bases: object

run()

stop()

com.ParallelGamit.**ExecuteGamit**(*cnn*, *JobServer*, *GamitConfig*, *stations*, *check_stations*, *ignore_missing*, *dates*, *dry_run=False*, *create_kml=False*)

com.ParallelGamit.**ExecuteGlobk**(*cnn*, *JobServer*, *GamitConfig*, *sessions*, *dates*)

com.ParallelGamit.**ParseZTD**(*project*, *dates*, *Sessions*, *GamitConfig*, *JobServer*)

com.ParallelGamit.**check_station_alias**(*cnn*)

com.ParallelGamit.**gamit_callback**(*job*)

com.ParallelGamit.**generate_kml**(*dates*, *sessions*, *GamitConfig*)

com.ParallelGamit.**id_generator**(*size=4*, *chars='abcdefghijklmnopqrstuvwxyz0123456789'*)

com.ParallelGamit.**job_callback**(*job*)

com.ParallelGamit.**main**()

com.ParallelGamit.**prGreen**(*skk*)

com.ParallelGamit.**prRed**(*skk*)

com.ParallelGamit.**prYellow**(*skk*)

com.ParallelGamit.**print_datetime**()

com.ParallelGamit.**print_summary**(*stations*, *sessions*, *dates*)

com.ParallelGamit.**purge_solution**(*pwd*, *project*, *date*)

com.ParallelGamit.**purge_solutions**(*JobServer*, *args*, *dates*, *GamitConfig*)

com.ParallelGamit.**run_gamit_session**(*gamit_task*, *dir_name*, *year*, *doy*, *dry_run*)

com.ParallelGamit.**run_globk**(*globk_task*, *project*, *date*)

com.ParallelGamit.**run_parse_ztd**(*parse_task*)

com.ParallelGamit.**station_list**(*cnn*, *stations*, *dates*)

ParallelGamit.py - CLI interface

Parallel.GAMIT main execution program

```
ParallelGamit.py [-h] [-d {date} {date}] [-dp {year} {doys} {year} {doys}]
                 [-e {station} [{station} ...]] [-c {station} [{station} ...]] [-i] [-p]
                 [-dry] [-kml] [-np]
                 session.cfg
```

ParallelGamit.py positional arguments

- ***session.cfg*** - Filename with the session configuration to run Parallel.GAMIT (default: None)

ParallelGamit.py options

- ***-h***, ***--help*** - show this help message and exit
- ***-d*** {DATE}, ***--date*** {DATE} - Date range to process. Can be specified in yyyy/mm/dd yyyy_doy wwww-d format (default: None)

- ***-dp*** {YEAR} {DOYS}, ***--date_parser*** {YEAR} {DOYS} - Parse date using ranges and commas (e.g. 2018 1,3-6). Cannot cross year boundaries (default: None)
- ***-e*** {STATION}, ***--exclude*** {STATION} - List of stations to exclude from this processing (e.g. -e igm1 lpgs vbca) (default: None)
- ***-c*** {STATION}, ***--check_mode*** {STATION} - Check station(s) mode. If station(s) are not present in the GAMIT polyhedron, (i.e. the RINEX file(s) were missing at the time of the processing) Parallel.GAMIT will add the station to the closest subnetwork(s) and reprocess them. If station(s) were present at the time of the processing but failed to process (i.e. they are in the missing stations list), these subnetworks will be reprocessed to try to obtain a solution. Station list provided in the cfg is ignored in this mode. Therefore, changes in the station list will not produce any changes in network configuration. Purge not allowed when using this mode. (Syntax: -c igm1 lpgs rms.vbca) (default: None)
- ***-i***, ***--ignore_missing*** - When using check mode or processing existing sessions, ignore missing stations. In other words, do not try to reprocess sessions that have missing solutions.
- ***-p***, ***--purge*** - Purge year doys from the database and directory structure and re-run the solution.
- ***-dry***, ***--dry_run*** - Generate the directory structures (locally) but do not run GAMIT. Output is left in the production directory.
- ***-kml***, ***--create_kml*** - Create a KML with everything processed in this run.
- ***-np***, ***--noparallel*** - Execute command without parallelization.

com.PlotETM module

Project: Parallel.PPP Date: 10/10/17 9:10 AM Author: Demian D. Gomez

User interface to plot and save JSON files of ETM objects. Type python pyPlotETM.py -h for usage help

com.PlotETM.**from_file**(*args*, *cnn*, *stn*)

com.PlotETM.**gamit_soln**(*args*, *cnn*, *stn*)

com.PlotETM.**main**()

PlotETM.py - CLI interface

Plot ETM for stations in the database

```
PlotETM.py [-h] [-nop] [-nom] [-nm] [-r] [-dir DIRECTORY] [-json JSON] [-gui] [-rj] [-rp]
           [-win interval [interval ...]] [-q {type} {date} {type} {date}] [-gamit
↪{stack}]
           [-lang LANGUAGE] [-hist] [-file FILENAME] [-format FORMAT [FORMAT ...]]
           [-outliers] [-vel] [-seasonal] [-quiet]
           stnlist [stnlist ...]
```

PlotETM.py positional arguments

- ***stnlist*** - List of networks/stations to process given in [net].[stnm] format or just [stnm] (separated by spaces; if [stnm] is not unique in the database, all stations with that name will be processed). Use keyword 'all' to process all stations in the database. If [net].all is given, all stations from network [net] will be processed. Three letter ISO 3166 international standard codes can be provided (always in upper case) to select all stations within a country. If a station name is given using a * in front (e.g. *igs.pwro or *pwro) then the station will be removed from the list. If *net.all or ISO country code was used (e.g. *igs.all or *ARG), then remove the stations within this group. Wildcards are accepted using the regex postgres convention. Use [] to provide character ranges (e.g. ars.at1[3-5] or ars.[a-b]x01). Char %% matches any string (e.g. ars.at%%). Char | represents the OR operator

that can be used to select one string or another (e.g. ars.at1[1|2] to choose at11 and at12). To specify a wildcard using a single character, use _ (equivalent to ? in POSIX regular expressions). Alternatively, a file with the station list can be provided (using all the same conventions described above). When using a file, * can be replaced with - for clarity in removing stations from .all lists (default: None)

PlotETM.py options

- ***-h***, ***--help*** - show this help message and exit
- ***-nop***, ***--no_plots*** - Do not produce plots
- ***-nom***, ***--no_missing_data*** - Do not show missing days
- ***-nm***, ***--no_model*** - Plot time series without fitting a model
- ***-r***, ***--residuals*** - Plot time series residuals
- ***-dir*** DIRECTORY, ***--directory*** DIRECTORY - Directory to save the resulting PNG files. If not specified, assumed to be the production directory (default: None)
- ***-json*** JSON, ***--json*** JSON - Export ETM adjustment to JSON. Append '0' to just output the ETM parameters, '1' to export time series without model and '2' to export both time series and model. (default: None)
- ***-gui***, ***--interactive*** - Interactive mode: allows to zoom and view the plot interactively
- ***-rj***, ***--remove_jumps*** - Remove jumps from model and time series before plotting
- ***-rp***, ***--remove_polynomial*** - Remove polynomial terms from model and time series before plotting
- ***-win*** INTERVAL, ***--time_window*** INTERVAL - Date range to window data. Can be specified in yyyy/mm/dd, yyyy.doy or as a single integer value (N) which shall be interpreted as last epoch-N (default: None)
- ***-q*** {TYPE} {DATE}, ***--query*** {TYPE} {DATE} - Dates to query the ETM. Specify "model" or "solution" to get the ETM value or the value of the daily solution (if exists). Output is in XYZ. (default: None)
- ***-gamit*** {STACK}, ***--gamit*** {STACK} - Plot the GAMIT time series specifying which stack name to plot. (default: None)
- ***-lang*** LANGUAGE, ***--language*** LANGUAGE - Change the language of the plots. Default is English. Use ESP to select Spanish. To add more languages, include the ISO 639-1 code in pyETM.py (default: ENG)
- ***-hist***, ***--histogram*** - Plot histogram of residuals
- ***-file*** FILENAME, ***--filename*** FILENAME - Obtain data from an external source (filename). This name accepts variables for {net} and {stn} to specify more than one file based on a list of stations. If a single file is used (no variables), then only the first station is processed. File column format should be specified with -format (required). (default: None)
- ***-format*** FORMAT, ***--format*** FORMAT - To be used together with –filename. Specify order of the fields as found in the input file. Format strings are gpsWeek, gpsWeekDay, year, doy, fyear, month, day, mjd, x, y, z, na. Use 'na' to specify a field that should be ignored. If fields to be ignored are at the end of the line, then there is no need to specify those. (default: None)
- ***-outliers***, ***--plot_outliers*** - Plot an additional panel with the outliers
- ***-vel***, ***--velocity*** - During query, output the velocity in XYZ.
- ***-seasonal***, ***--seasonal_terms*** - During query, output the seasonal terms in NEU.
- ***-quiet***, ***--suppress_messages*** - Quiet mode: suppress information messages

com.PlotMapView module

Project: Parallel.GAMIT Date: 6/12/18 10:28 AM Author: Demian D. Gomez

com.PlotMapView.**generate_kmz**(*kmz*, *stations*)

com.PlotMapView.**main**()

com.PlotMapView.**plot_map_view**(*pngfile*, *etm*, *lneu*, *fil*, *event*, *co_jump*)

PlotMapView.py - CLI interface

Archive operations Main Program

```
PlotMapView.py [-h] [-stn {station list} [{station list} ...]] [-inter {interseismic_
↪model}]
               [-post {event_date} [event_date_1] [event_date_2] ... [event_grid_1]
               [event_grid_2] ... [{event_date} [event_date_1] [event_date_2] ...
               [event_grid_1] [event_grid_2] ... ...]] [-co {coseismic_grid}]
               [-dir DIRECTORY] [-missing]
               {stack name}
```

PlotMapView.py positional arguments

- ***{stack name}*** - Name of the GAMIT stack to use for the trajectories (default: None)

PlotMapView.py options

- ***-h***, ***--help*** - show this help message and exit
- ***-stn*** {STATION LIST}, ***--stations*** {STATION LIST} - Specify the list of networks/stations given in [net].[stnm] format or just [stnm] that will be filtered using the selected field specifications. If [stnm] is not unique in the database, all stations with that name will be processed.Alternatively, a file with the station list can be provided. (default: [])
- ***-inter*** {INTERSEISMIC_MODEL}, ***--interseismic_model*** {INTERSEISMIC_MODEL} - Interseismic removal is done using {interseismic_grid}. (default: None)
- ***-post*** {EVENT_DATE} [EVENT_DATE_1] [EVENT_DATE_2] ... [EVENT_GRID_1] [EVENT_GRID_2] ..., ***--postseismic*** {EVENT_DATE} [EVENT_DATE_1] [EVENT_DATE_2] ... [EVENT_GRID_1] [EVENT_GRID_2] ... - Interseismic removal is done using the grid provided in –interseismic_model. The event postseimic to be plotted correspond to seismic event given in {event_date}. Additionally, provide [event_date_n] and [event_grid_n] if a previous postsiesmic processes should be removed from the ETMs. If no [event_date_n] are given, then any previous events are ignored (but they could still be present in the ETM fit). (default: None)
- ***-co*** {COSEISMIC_GRID}, ***--coseismic*** {COSEISMIC_GRID} - Coseismic grid to use for stations that were not active by the time of the event. This assumes that you are providing the grid of the event specific in -post (default: None)
- ***-dir*** DIRECTORY, ***--directory*** DIRECTORY - Directory to save the resulting PNG files. If not specified, assumed to be the production directory (default: None)
- ***-missing***, ***--plot_missing_solutions*** - Plot the missing solutions in the ETMs stored in the KMZ file (might take longer to produce the file).

com.QueryETM module

Project: Parallel.PPP Date: 10/10/17 9:10 AM Author: Demian D. Gomez

User interface to plot and save JSON files of ETM objects. Type python pyPlotETM.py -h for usage help

com.QueryETM.**from_file**(*args*, *cnn*, *stn*)

com.QueryETM.**main**()

QueryETM.py - CLI interface

Query ETM for stations in the database. Default is PPP ETMs.

```
QueryETM.py [-h] [-q {type} {date} {type} {date}] [-gamit {stack}] [-file FILENAME]
            [-format FORMAT [FORMAT ...]] [-quiet] [-vel] [-seasonal]
            stnlist [stnlist ...]
```

QueryETM.py positional arguments

- ***stnlist*** - List of networks/stations to plot given in [net].[stnm] format or just [stnm] (separated by spaces; if [stnm] is not unique in the database, all stations with that name will be plotted). Use keyword `'all'` to plot all stations in all networks. If [net].all is given, all stations from network [net] will be plotted (default: `None`)

QueryETM.py options

- ***-h***, ***--help*** - show this help message and exit
- ***-q*** `{TYPE} {DATE}`, ***--query*** `{TYPE} {DATE}` - Dates to query the ETM. Specify `"model"` or `"solution"` to get the ETM value or the value of the daily solution (if exists). Output is in XYZ. (default: `None`)
- ***-gamit*** `{STACK}`, ***--gamit*** `{STACK}` - Plot the GAMIT time series specifying which stack name to plot. (default: `None`)
- ***-file*** `FILENAME`, ***--filename*** `FILENAME` - Obtain data from an external source (filename). Format should be specified with -format. (default: `None`)
- ***-format*** `FORMAT`, ***--format*** `FORMAT` - To be used together with –filename. Specify order of the fields as found in the input file. Format strings are gpsWeek, gpsWeekDay, year, doy, fyear, month, day, mjd, x, y, z, na. Use `'na'` to specify a field that should be ignored. If fields to be ignored are at the end of the line, then there is no need to specify those. (default: `None`)
- ***-quiet***, ***--quiet*** - Do not print message when no solutions are available.
- ***-vel***, ***--velocity*** - Output the velocity in XYZ.
- ***-seasonal***, ***--seasonal_terms*** - Output the seasonal terms in NEU.

com.ReadPackage module

com.ReadPackage.**extract_json**(*zipfile*)

com.ReadPackage.**main**()

com.ReadPackage.**print_insert_sql**(*station*)

com.ReadPackage.**print_station_info**(*NetworkCode*, *StationCode*, *stninfo*)

ReadPackage.py - CLI interface

Program to read package contents created by another Parallel.GAMIT system

```
ReadPackage.py [-h] [-stninfo] [-ins] zipfiles [zipfiles ...]
```

ReadPackage.py positional arguments

- ***zipfiles*** - List of zipfiles to extract information from and print to screen. See option switches to see printing options. (default: None)

ReadPackage.py options

- ***-h***, ***--help*** - show this help message and exit
- ***-stninfo***, ***--station_info*** - Print the station information content (in GAMIT format) to the screen.
- ***-ins***, ***--insert_sql*** - Produce a SQL INSERT statement for this station including OTL and coordinates.

com.S-score module

com.ScanArchive module

Project: Parallel.Archive Date: 02/16/2017 Author: Demian D. Gomez

Main routines to load the RINEX files to the database, load station information, run PPP on the archive files and obtain the OTL coefficients

usage: pyScanArchive.py [-h] [-rinex] [-otl]
[-stninfo [argument [argument . . .]]] [-ppp [argument [argument . . .]]] [-rehash [argument [argument . . .]]] [-np] all|net.stnm [all|net.stnm . . .]

Archive operations Main Program

positional arguments:

all|net.stnm List of networks/stations to process given in
[net].[stnm] format or just [stnm] (separated by spaces; if [stnm] is not unique in the database, all stations with that name will be processed). Use keyword 'all' to process all stations in the database. If [net].all is given, all stations from network [net] will be processed. Alternatevily, a file with the station list can be provided.

optional arguments:

-h, --help	show this help message and exit
-rinex, --rinex	Scan the current archive for RINEX 2/3 files.
-otl, --ocean_loading	Calculate ocean loading coefficients.

-stninfo [argument [argument . . .]], –station_info [argument [argument . . .]]
Insert station information to the database. If no arguments are given, then scan the archive for station info files and use their location (folder) to determine the network to use during insertion. Only stations in the station list will be processed. If a filename is provided, then scan that file only, in which case a second argument specifies the network to use during insertion. Eg: -stninfo ~/station.info arg. In cases where multiple networks are being processed, the network argument will be used to desambiguate station code conflicts. Eg: pyScanArchive all -stninfo ~/station.info arg -> if a station named igm1 exists in networks 'igs' and 'arg', only 'arg.igm1' will get the station information insert. Use keyword 'stdin' to read the station information data from the pipeline.

-ppp [argument [argument ...]], –ppp [argument [argument ...]]
Run ppp on the rinex files in the database. Append [date_start] and (optionally) [date_end] to limit the range of the processing. Allowed formats are yyyy.doy or yyyy/mm/dd. Append keyword 'hash' to the end to check the PPP hash values against the station information records. If hash doesn't match, recalculate the PPP solutions.

-rehash [argument [argument ...]], –rehash [argument [argument ...]]
Check PPP hash against station information hash. Rehash PPP solutions to match the station information hash without recalculating the PPP solution. Optionally append [date_start] and (optionally) [date_end] to limit the rehashing time window. Allowed formats are yyyy.doy or yyyy/mm/dd.

-np, --noparallel Execute command without parallelization.

class com.ScanArchive.**Encoder**(*, *skipkeys=False*, *ensure_ascii=True*, *check_circular=True*, *allow_nan=True*, *sort_keys=False*, *indent=None*, *separators=None*, *default=None*)

Bases: JSONEncoder

default(*o*)

class com.ScanArchive.**callback_class**(*pbar*)

Bases: object

callbackfunc(*args*)

com.ScanArchive.**callback_handle**(*job*)

com.ScanArchive.**execute_ppp**(*record*, *rinex_path*, *h_tolerance*)

com.ScanArchive.**export_station**(*cnn*, *stnlist*, *pyArchive*, *archive_path*, *dataless*)

com.ScanArchive.**get_rinex_file**(*cnn*, *stnlist*, *date*, *Archive_path*)

com.ScanArchive.**hash_check**(*cnn*, *master_list*, *sdate*, *edate*, *rehash=False*, *h_tolerant=0*)

com.ScanArchive.**import_station**(*cnn*, *args*)

com.ScanArchive.**insert_station**(*cnn*, *network*, *station*)

com.ScanArchive.**insert_stninfo**(*NetworkCode*, *StationCode*, *stninfofile*)

com.ScanArchive.**main**()

com.ScanArchive.**obtain_otl**(*NetworkCode*, *StationCode*)

com.ScanArchive.**post_scan_rinex_job**(*cnn*, *Archive*, *rinex_file*, *rinexpath*, *master_list*, *JobServer*, *ignore*)

com.ScanArchive.**print_scan_archive_summary**(*cnn*)

com.ScanArchive.**process_otl**(*cnn*, *JobServer*, *master_list*)

com.ScanArchive.**process_ppp**(*cnn*, *Config*, *pyArchive*, *archive_path*, *JobServer*, *master_list*, *sdate*, *edate*, *h_tolerance*)

com.ScanArchive.**remove_from_archive**(*cnn*, *record*, *Rinex*, *Config*)

com.ScanArchive.**scan_rinex**(*cnn*, *JobServer*, *pyArchive*, *archive_path*, *master_list*, *ignore*)

com.ScanArchive.**scan_station_info**(*JobServer*, *pyArchive*, *archive_path*, *master_list*)

com.ScanArchive.**scan_station_info_man**(*cnn*, *pyArchive*, *stn_info_path*, *stations*, *stn_info_net*, *stdin=None*)

com.ScanArchive.**try_insert**(*NetworkCode*, *StationCode*, *year*, *doy*, *rinex*)

com.ScanArchive.**try_insert_files**(*cnn*, *archive*, *station*, *NetworkCode*, *StationCode*, *rinex*)

com.ScanArchive.**verify_rinex_date_multiday**(*cnn*, *date*, *rinexinfo*, *Config*)

ScanArchive.py - CLI interface

Archive operations Main Program

```
ScanArchive.py [-h] [-rinex {ignore_stnlist}] [-otl] [-stninfo [argument ...]]
               [-export [[dataless seed]]] [-import {default net} [{zipfiles} ...]]
               [-get {date}] [-ppp [argument ...]] [-rehash [argument ...]] [-tol {hours}
↪]
               [-np]
               all|net.stnm [all|net.stnm ...]
```

ScanArchive.py positional arguments

- ***all|net.stnm*** - List of networks/stations to process given in [net].[stnm] format or just [stnm] (separated by spaces; if [stnm] is not unique in the database, all stations with that name will be processed). Use keyword `'all'` to process all stations in the database. If [net].all is given, all stations from network [net] will be processed. Three letter ISO 3166 international standard codes can be provided (always in upper case) to select all stations within a country. If a station name is given using a * in front (e.g. *igs.pwro or *pwro) then the station will be removed from the list. If *net.all or ISO country code was used (e.g. *igs.all or *ARG), then remove the stations within this group. Wildcards are accepted using the regex postgres convention. Use [] to provide character ranges (e.g. ars.at1[3-5] or ars.[a-b]x01). Char %% matches any string (e.g. ars.at%%). Char | represents the OR operator that can be used to select one string or another (e.g. ars.at1[1|2] to choose at11 and at12). To specify a wildcard using a single character, use _ (equivalent to ? in POSIX regular expressions). Alternatively, a file with the station list can be provided (using all the same conventions described above). When using a file, * can be replaced with - for clarity in removing stations from .all lists (default: `None`)

ScanArchive.py options

- ***-h***, ***--help*** - show this help message and exit
- ***-rinex*** `{IGNORE_STNLIST}`, ***--rinex*** `{IGNORE_STNLIST}` - Scan the current archive for RINEX 2/3 files and add them to the database if missing. Station list will be used to filter specific networks and stations if `{ignore_stnlist}` = 0. For example: ScanArchive [net].all -rinex 0 will process all the stations in network [net], but networks and stations have to exist in the database. If ScanArchive [net].all -rinex 1 the station list will be ignored and everything in the archive will be checked (and added to the db if missing) even if networks and stations don```t exist. Networks and stations will be added if they don```t exist. (default: `None`)
- ***-otl***, ***--ocean_loading*** - Calculate ocean loading coefficients using FES2004. To calculate FES2014b coefficients, use OTL_FES2014b.py
- ***-stninfo*** `ARGUMENT`, ***--station_info*** `ARGUMENT` - Insert station information to the database. If no arguments are given, then scan the archive for station info files and use their location (folder) to determine the network to use during insertion. Only stations in the station list will be processed. If a filename is provided, then scan that file only, in which case a second argument specifies the network to use during insertion. Eg: -stninfo ~/station.info arg. In cases where multiple networks are being processed, the network argument will be used to desambiguate station code conflicts. Eg: ScanArchive all -stninfo ~/station.info arg -> if a station named igm1 exists in networks `'igs'` and `'arg'`, only `'arg.igm1'` will get the station information insert. Use keyword `'stdin'` to read the station information data from the pipeline. (default: `None`)

- ***-export*** [DATALESS SEED], ***--export_station*** [DATALESS SEED] - Export a station from the local database that can be imported into another Parallel.GAMIT system using the -import option.One file is created per station in the current directory. If the [dataless seed] switch is passed (e.g. -export true), then the export seed is created without data (only metadata included, i.e. station info, station record, etc). (default: None)
- ***-import*** {DEFAULT NET}, ***--import_station*** {DEFAULT NET} - Import a station from zipfiles produced by another Parallel.GAMIT system. Wildcards are accepted to import multiple zipfiles. If station does not exist, use {default net} to specify the network where station should be added to. If {default net} does not exit, it will be created. Station list is ignored. (default: None)
- ***-get*** {DATE}, ***--get_from_archive*** {DATE} - Get the specified station from the archive and copy it to the current directory. Fix it to match the station information in the database. (default: None)
- ***-ppp*** ARGUMENT, ***--ppp*** ARGUMENT - Run ppp on the rinex files in the database. Append [date_start] and (optionally) [date_end] to limit the range of the processing. Allowed formats are yyyy_doy, wwww-d, fyear or yyyy/mm/dd. Append keyword 'hash' to the end to check the PPP hash values against the station information records. If hash doesn't match, recalculate the PPP solutions. (default: None)
- ***-rehash*** ARGUMENT, ***--rehash*** ARGUMENT - Check PPP hash against station information hash. Rehash PPP solutions to match the station information hash without recalculating the PPP solution. Optionally append [date_start] and (optionally) [date_end] to limit the rehashing time window. Allowed formats are yyyy.doy or yyyy/mm/dd. (default: None)
- ***-tol*** {HOURS}, ***--stninfo_tolerant*** {HOURS} - Specify a tolerance (in hours) for station information gaps (only use for early survey data). Default is zero. (default: [0])
- ***-np***, ***--noparallel*** - Execute command without parallelization.

com.Stacker module

Project: Parallel.Stacker Date: 6/12/18 10:28 AM Author: Demian D. Gomez

com.Stacker.**calculate_etms**(*cnn*, *stack*, *JobServer*, *iterations*, *create_target=True*, *exclude_stn=()*)

Parallel calculation of ETMs to save some time :param cnn: connection to the db :param stack: object with the list of polyhedrons :param JobServer: parallel.python object :param iterations: current iteration number :param create_target: indicate if function should create and return target polyhedrons :param exclude_stn: list of stations to exclude from the stacking process :return: the target polyhedron list that will be used for alignment (if create_target = True)

com.Stacker.**callback_handler**(*job*)

com.Stacker.**load_constrains**(*constrains_file*, *exclude_stn=()*)

Load the frame parameters :param constrains_file: file with the parameters to inherit from primary frame :param exclude_stn: station list to exclude from the inheritance process :return: dictionary with the parameters for the given frame

com.Stacker.**load_periodic_space**(*periodic_file*)

Load the periodic space parameters from an ITRF file :param periodic_file: :return: dictionary with the periodic terms

com.Stacker.**main**()

com.Stacker.**plot_etm**(*cnn*, *stack*, *station*, *directory*)

com.Stacker.**station_etm**(*station*, *stn_ts*, *stack_name*, *iteration=0*)

Stacker.py - CLI interface

GNSS time series stacker

```
Stacker.py [-h] [-max {max_iter}] [-exclude {net.stnm} [{net.stnm} ...]] [-dir DIRECTORY]
           [-redo] [-plot] [-constraints EXTERNAL_CONSTRAINTS [EXTERNAL_CONSTRAINTS ...]]
           [-d date] [-np]
           {project name} {stack name}
```

Stacker.py positional arguments

- ***{project name}*** - Specify the project name used to process the GAMIT solutions in Parallel.GAMIT. (default: None)
- ***{stack name}*** - Specify a name for the stack: eg. itrf2014 or posgar07b. This name should be unique and cannot be repeated for any other solution project (default: None)

Stacker.py options

- ***-h***, ***--help*** - show this help message and exit
- ***-max*** {MAX_ITER}, ***--max_iters*** {MAX_ITER} - Specify maximum number of iterations. Default is 4. (default: None)
- ***-exclude*** {NET.STNM}, ***--exclude_stations*** {NET.STNM} - Manually specify stations to remove from the stacking process. (default: None)
- ***-dir*** DIRECTORY, ***--directory*** DIRECTORY - Directory to save the resulting PNG files. If not specified, assumed to be the production directory (default: None)
- ***-redo***, ***--redo_stack*** - Delete the stack and redo it from scratch
- ***-plot***, ***--plot_stack_etms*** - Plot the stack ETMs after computation is done
- ***-constraints*** EXTERNAL_CONSTRAINTS, ***--external_constraints*** EXTERNAL_CONSTRAINTS - File with external constraints parameters (position, velocity and periodic). These may br from a parent frame such as ITRF. Inheritance will occur with stations on the list whenever a parameter exists. Example: -constraints itrf14.txt Format is: net.stn x y z epoch vx vy vz sn_1y sn_6m cn_1y cn_6m se_1y se_6m ce_1y ce_6m su_1y su_6m cu_1y cu_6m (default: None)
- ***-d*** DATE, ***--date_end*** DATE - Limit the polyhedrons to the specified date. Can be in wwww-d, yyyy_ddd, yyyy/mm/dd or fyear format (default: None)
- ***-np***, ***--noparallel*** - Execute command without parallelization.

com.StationInfoEdit module

Project: Date: 10/10/17 3:07 PM Author: Demian D. Gomez

class com.StationInfoEdit.**Menu**(*cnn*, *items*, *stdscreen*, *title=''*, *type='main'*, *record_index=None*)

Bases: object

ShowError(*error*)

display()

enter_edit_mode(*value=None*)

navigate(*n*)

validate(*edit_field*)

class com.StationInfoEdit.**MyApp**(*stdscreen*)

Bases: object

com.StationInfoEdit.**delete_record**(*menu*)

com.StationInfoEdit.**edit_record**(*position*)

com.StationInfoEdit.**get_fields**(*position*)

com.StationInfoEdit.**get_records**()

com.StationInfoEdit.**main**()

com.StationInfoEdit.**save_changes**(*menu*)

com.StationInfoEdit.**selection_main_menu**(*menu*)

StationInfoEdit.py - CLI interface

Edit Stations info in the database

```
StationInfoEdit.py [-h] stn
```

StationInfoEdit.py positional arguments

- ***stn*** - Station name given in net.stnm format. (default: None)

StationInfoEdit.py options

- ***-h***, ***--help*** - show this help message and exit

com.SyncOrbits module

Project: Parallel.GAMIT Date: 11/8/17 9:24 AM Author: Demian D. Gomez

Script to synchronize sp3 and brdc orbits using the CDDIS FTP server.

com.SyncOrbits.**main**()

SyncOrbits.py - CLI interface

Synchronize orbit archive

```
SyncOrbits.py [-h] [-date date_start|date_end [date_start|date_end ...]] [-win days]
```

SyncOrbits.py options

- ***-h***, ***--help*** - show this help message and exit
- ***-date*** DATE_START|DATE_END, ***--date_range*** DATE_START|DATE_END - Date range to check given as [date_start] or [date_start] and [date_end]. Allowed formats are yyyy.doy or yyyy/mm/dd.. (default: None)
- ***-win*** DAYS, ***--window*** DAYS - Download data from a given time window determined by today - {days}. (default: None)

com.TrajectoryFit module

Project: Parallel.Archive Date: 02/16/2017 Author: Demian D. Gomez

com.TrajectoryFit.**generate_kmz**(*kmz*, *stations*, *discarded*, *deformation_type='interseismic'*, *units='mm/yr'*)

com.TrajectoryFit.**main**()

com.TrajectoryFit.**plot_station_param**(*NetworkCode*, *StationCode*, *parameter_name*, *unit*, *pn*, *pe*)

com.TrajectoryFit.**process_interseismic**(*cnn*, *stnlist*, *force_stnlist*, *stack*, *sigma_cutoff*, *vel_cutoff*, *lat_lim*, *filename*, *kmz*)

com.TrajectoryFit.**process_postseismic**(*cnn*, *stnlist*, *force_stnlist*, *stack*, *interseimic_filename*, *event*, *filename*, *kmz*, *prev_events=None*, *sigma_cutoff=0*, *lat_lim=0*)

TrajectoryFit.py - CLI interface

Archive operations Main Program

```
TrajectoryFit.py [-h] [-stn {station list} [{station list} ...]]
                 [-force_stn {station list} [{station list} ...]]
                 [-lat_lim {min_lat max_lat} {min_lat max_lat}] [-sigma {mm}] [-vel {mm/
↪yr}]
                 [-interseismic [[velocity_cutoff] [output_filename] [kmz_filename] ...]]
                 [-postseismic {interseismic_grid} {event_date} {output_filename} {kmz_
↪filename} [event_date_1]
                 [event_date_2] ... [event_grid_1] [event_grid_2] ...
                 [{interseismic_grid} {event_date} {output_filename} {kmz_filename}␣
↪[event_date_1]
                 [event_date_2] ... [event_grid_1] [event_grid_2] ... ...]]
                 {stack name}
```

TrajectoryFit.py positional arguments

- ***{stack name}*** - Name of the GAMIT stack to use for the trajectories (default: None)

TrajectoryFit.py options

- ***-h***, ***--help*** - show this help message and exit
- ***-stn*** {STATION LIST}, ***--stations*** {STATION LIST} - Specify the list of networks/stations given in [net].[stnm] format or just [stnm] that will be filtered using the selected field specifications. If [stnm] is not unique in the database, all stations with that name will be processed.Alternatively, a file with the station list can be provided. (default: [])
- ***-force_stn*** {STATION LIST}, ***--force_stations*** {STATION LIST} - Force stations to be included in the selected field. Specify the list of networks/stations given in [net].[stnm] format or just [stnm]. If [stnm] is not unique in the database, all stations with that name will be processed.Alternatively, a file with the station list can be provided. (default: [])
- ***-lat_lim*** {MIN_LAT MAX_LAT}, ***--latitude_limits*** {MIN_LAT MAX_LAT} - Latitude limits (decimal degrees, stations discarded outside of this limit) provided as south, north limit. Default is -90 90 (default: [-90, 90])
- ***-sigma*** {MM}, ***--sigma_cutoff*** {MM} - Reject stations based on the ETM's wrms (in mm). This filter is not applied for the forced station list. (default: [2.5])

- ***-vel*** {MM/YR}, ***--velocity_cutoff*** {MM/YR} - ETM velocity cutoff value to reject stations for velocity interpolation (norm of NE in mm/yr). (default: [50])
- ***-interseismic*** [VELOCITY_CUTOFF] [OUTPUT_FILENAME] [KMZ_FILENAME], ***--interseismic_process*** [VELOCITY_CUTOFF] [OUTPUT_FILENAME] [KMZ_FILENAME] - Process stations for interseismic velocity field computation. Reject stations with interseismic velocity > {velocity_cutoff} (default 50 mm/yr). Filename to output the selected stations (default filename interseismic.txt). Optionally, specify a kmz filename to output the selected and rejected stations with their velocity components and ETMs embedded in the kmz (default no kmz).
- ***-postseismic*** {INTERSEISMIC_GRID} {EVENT_DATE} {OUTPUT_FILENAME} {KMZ_FILENAME} [EVENT_DATE_1] [EVENT_DATE_2] ... [EVENT_GRID_1] [EVENT_GRID_2] ..., ***--postseismic_process*** {INTERSEISMIC_GRID} {EVENT_DATE} {OUTPUT_FILENAME} {KMZ_FILENAME} [EVENT_DATE_1] [EVENT_DATE_2] ... [EVENT_GRID_1] [EVENT_GRID_2] ... - Process stations for postseismic field computation. Interseismic removal is done using {interseismic_grid}. The event parameters to be extracted from the ETMs correspond to seismic event given in {event_date}. Resulting parameters will be written to {output_filename}. Additionally, provide [event_date_n] and [event_grid_n] if a previous postsiesmic processes should be removed from the ETMs. If no [event_date_n] are given, then any previous events are ignored (but they could still be present in the ETM fit). (default: None)

com.UpdateEarthquakes module

Project: Parallel.Archive Date: 3/3/17 9:56 AM Author: Demian D. Gomez

This class is based on the USGS Neicio: the USGS NEIC Python interface and its purpose is to update the table "earthquakes" in the db with the latest events (M >= 6)

class com.UpdateEarthquakes.**AddEarthquakes**(*cnn*)

Bases: object

com.UpdateEarthquakes.**getTimeSegments2**(*starttime*, *endtime*)

com.UpdateEarthquakes.**main**()

com.WeeklyCombination module

Project: Parallel.GAMIT Date: 05/30/19 10:44 AM Author: Demian D. Gomez

class com.WeeklyCombination.**Globk**(*pwd_comb*, *org*, *glx_list*, *gpsweek*, *gpsweekday*, *sites*)

Bases: object

create_combination_script(*org*, *gpsweek*, *gpsweekday*, *sites*)

execute()

exception com.WeeklyCombination.**GlobkException**(*value*)

Bases: Exception

com.WeeklyCombination.**add_domes**(*sinex*, *stations*)

com.WeeklyCombination.**id_generator**(*size=4*, *chars='abcdefghijklmnopqrstuvwxyz0123456789'*)

com.WeeklyCombination.**main**()

com.WeeklyCombination.**process_sinex**(*cnn*, *project*, *dates*, *sinex*)

com.WeeklyCombination.**replace_in_sinex**(*sinex*, *observations*, *unknowns*, *new_val*)

WeeklyCombination.py - CLI interface

Program to perform weekly loosely-constrained solutions. Combination is performed using GLOBK. Result is output in SINEX format.

```
WeeklyCombination.py [-h] [-s session.cfg] [-w GPSWEEK] [-e station [station ...]]
                     all|net.stnm [all|net.stnm ...]
```

WeeklyCombination.py positional arguments

- ***all|net.stnm*** - List of networks/stations to include in the solution. (default: None)

WeeklyCombination.py options

- ***-h***, ***--help*** - show this help message and exit
- ***-s*** SESSION.CFG, ***--session_config*** SESSION.CFG - Filename with the session configuration to run Parallel.GAMIT (default: None)
- ***-w*** GPSWEEK, ***--gpsweek*** GPSWEEK - GPS week to combine. (default: None)
- ***-e*** STATION, ***--exclude*** STATION - List of stations to exclude (e.g. -e igm1 lpgs vbca) (default: None)

com.Ztd2trp module

Original author: Federico Fernandez (IGN) Project: Parallel.GAMIT Date: Jul 31 2023 12:24 PM Modified by: Demian D. Gomez

Script to download zenith tropospheric delays from database and save them in TRP BERNESE format. Limited functionality, need to implement compression, read of model types (codes), etc.

com.Ztd2trp.**main**()

Ztd2trp.py - CLI interface

Zenith tropospheric delays to BERNESE TRP format

```
Ztd2trp.py [-h] [-z] [-c] [-date date_start|date_end [date_start|date_end ...]]
           [-dir DIRECTORY]
           project_name
```

Ztd2trp.py positional arguments

- ***project_name*** - GAMIT processing project name (default: None)

Ztd2trp.py options

- ***-h***, ***--help*** - show this help message and exit
- ***-z***, ***--compress*** - Compress resulting files (gz format)
- ***-c***, ***--produce_here*** - Bandera para indicar que se produzcan los archivos en el mismo directorio
- ***-date*** DATE_START|DATE_END, ***--date_range*** DATE_START|DATE_END - Date range to get from database given as [date_start] or [date_start] and [date_end]. Allowed formats are yyyy/mm/dd, yyyy_doy, or wwww-d. (default: None)

- ***-dir*** DIRECTORY, ***--directory*** DIRECTORY - Directory to save the resulting ZTD files. If not specified, assumed to be the production directory (default: None)

com.gamit_stats module

Project: Date: 10/27/17 12:40 PM Author: Demian D. Gomez

com.gamit_stats.**main**()

com.gamit_stats.**parse_monitor**(*cnn*, *monitor*)

Module contents

2.1.2 pgamit package

Subpackages

pgamit.tests package

Submodules

pgamit.tests.common module

Common utilities for testing clustering.

pgamit.tests.common.**gen_variable_density_clusters**(*n_points_per_cluster=250*, *seed=0*)

Generates 6 cluster blobs of varying density as synthetic continents

Modified from the sklearn OPTICS example and unit tests. (see https://scikit-learn.org/stable/auto_examples/cluster/plot_optics.html)

pgamit.tests.common.**generate_clustered_data**(*seed=0*, *n_clusters=3*, *n_features=2*, *n_samples_per_cluster=20*, *std=0.4*)

Generic cluster generator (taken from sklearn common cluster tests)

pgamit.tests.test_make_clusters module

Module contents

Submodules

pgamit.StationList module

Project: Parallel.GAMIT Date: Dic-03-2016 Author: Demian D. Gomez

class pgamit.StationList.**StationList**(*src=None*)

Bases: object

addStation(*stn*)

to_string()

exception pgamit.StationList.**StationListException**(*value*)

Bases: Exception

pgamit.Utils module

exception pgamit.Utils.**UtilsException**(*value*)

Bases: Exception

pgamit.Utils.**cart2euler**(*x*, *y*, *z*)

pgamit.Utils.**chmod_exec**(*path*)

pgamit.Utils.**copyfile**(*src*, *dst*, *rnx_ver=2*)

Copies a file from path src to path dst. If a file already exists at dst, it will not be overwritten, but:

- If it is the same as the source file, do nothing
- If it is different to the source file, pick a new name for the copy that is different and unused, then copy the file there (if rnx_ver=2)
- If because rinex 3 files have names that are more comprehensive (include start time and duration) if a rnx_ver == 3 then copy the file unless it already exists (in which case it does nothing)

Returns the path to the copy.

pgamit.Utils.**crc32**(*s*)

pgamit.Utils.**create_empty_cfg**()

function to create an empty cfg file with all the parts that are needed

pgamit.Utils.**ct2lg**(*dX*, *dY*, *dZ*, *lat*, *lon*)

pgamit.Utils.**determine_frame**(*frames*, *date*)

pgamit.Utils.**dir_try_remove**(*path*, *recursive=False*)

pgamit.Utils.**do_copy_op**(*src*, *dst*)

pgamit.Utils.**ecef2lla**(*ecefArr*)

pgamit.Utils.**file_append**(*path*, *data*)

pgamit.Utils.**file_open**(*path*, *mode='r'*)

pgamit.Utils.**file_read_all**(*path*)

pgamit.Utils.**file_readlines**(*path*)

pgamit.Utils.**file_try_remove**(*path*)

pgamit.Utils.**file_write**(*path*, *data*)

pgamit.Utils.**fix_gps_week**(*file_path*)

pgamit.Utils.**fqdn_parse**(*fqdn*, *default_port=None*)

pgamit.Utils.**get_field_or_attr**(*obj*, *f*)

pgamit.Utils.**get_norm_doy_str**(*doy*)

pgamit.Utils.**get_norm_year_str**(*year*)

pgamit.Utils.**get_platform_id**()

pgamit.Utils.**get_processor_count**()

pgamit.Utils.**get_resource_delimiter**()

pgamit.Utils.**get_stack_stations**(*cnn*, *name*)

pgamit.Utils.**human_readable_time**(*secs*)

pgamit.Utils.**indent**(*text*, *amount*, *ch=' '*)

pgamit.Utils.**json_converter**(*obj*)

pgamit.Utils.**lg2ct**(*dN*, *dE*, *dU*, *lat*, *lon*)

pgamit.Utils.**ll2sphere_xyz**(*ell*)

pgamit.Utils.**move**(*src*, *dst*)

Moves a file from path src to path dst. If a file already exists at dst, it will not be overwritten, but:

- If it is the same as the source file, do nothing
- If it is different to the source file, pick a new name for the copy that is distinct and unused, then copy the file there.

Returns the path to the new file.

pgamit.Utils.**parseIntSet**(*nputstr=''*)

pgamit.Utils.**parse_atx_antennas**(*atx_file*)

pgamit.Utils.**parse_crinex_rinex_filename**(*filename*)

pgamit.Utils.**print_columns**(*l*)

pgamit.Utils.**process_date**(*arg*, *missing_input='fill'*, *allow_days=True*)

pgamit.Utils.**process_date_str**(*arg*, *allow_days=False*)

pgamit.Utils.**process_stnlist**(*cnn*, *stnlist_in*, *print_summary=True*, *summary_title=None*)

Now the station list parser handles postgres regular expressions in the station list everything behaves as before, but also now support * and - to remove a station from the list (rather than only -) this is to support removal from the command line DDG June 21 2024: Now the file read accepts additional parameters in the station lines that are passed back in the

params key of the dictionary. This is to support, for example, passing velocities. Example: arg.igm1 1.00 arg.lpgs 1.20 …

pgamit.Utils.**required_length**(*nmin*, *nmax*)

pgamit.Utils.**rotct2lg**(*lat*, *lon*, *n=1*)

pgamit.Utils.**rotlg2ct**(*lat*, *lon*, *n=1*)

pgamit.Utils.**smallestN_indices**(*a*, *N*)

Function to return the row and column of the N smallest values :param a: array to search (any dimension) :param N: number of values to search :return: array with the rows-cols of min values

pgamit.Utils.**split_string**(*str*, *limit*, *sep=' '*)

pgamit.Utils.**stationID**(*s*)

pgamit.Utils.**station_list_help**()

pgamit.Utils.**struct_unpack**(*fs*, *data*)

pgamit.cluster module

Various utilities and functions to help ParallelGamit.

pgamit.cluster.**over_cluster**(*labels*, *coordinates*, *metric='haversine'*, *neighborhood=5*, *overlap_points=2*, *rejection_threshold=None*)

Expand cluster membership to include edge points of neighbor clusters

Expands an existing clustering to create overlapping membership between clusters. Existing clusters are processed sequentially by looking up nearest neighbors as an intersection of the current cluster membership and all cluster's point membership. Once the *overlap_points* for a given neighbor cluster have been determined and added to current cluster, the remainder of that neighboring cluster is removed from consideration and distance query is rerun, with the process repeating until a number of clusters equal to the *neighborhood* parameter is reached. For stability, only original points are included for subsequent neighborhood searches; that is, Nearest neighbor distances are run as the shortest distance from all **original** members of the current cluster.

Function requires an initial vector of cluster labels from a prior clustering, and coordinates in an ordering that matches the labels. This function also assumes that all points have been assigned a label (i.e., there are no unlabeled points, or points labeled as 'noise').

Parameters

- **labels** (*ndarray of type int, and shape (n_samples,)*) – Cluster labels for each point in the dataset from prior clustering.
- **coordinates** (*ndarray of shape (n_samples, n_features)*) – Coordinates do not need to match what was used for the prior clustering; i.e., if 'Euclidean' was used to calculate the prior clustering in an X,Y,Z projection, those coordinates can be provided in spherical coordinates, provided that 'haversine' is selected for the *metric* parameter.
- **metric** (*str or callable, default='haversine'*) – Metric to use for distance computation. Any metric from scikit-learn or scipy.spatial.distance can be used. Note that latitude and longitude values will need to be converted to radians if using the default 'haversine' distance metric.

  If metric is a callable function, it is called on each pair of instances (rows) and the resulting value recorded. The callable should take two arrays as input and return one value indicating the distance between them. This works for Scipy's metrics, but is less efficient than passing the metric name as a string.

  Use of "precomputed", i.e., a N-by-N distance matrix, has not been tested, nor have sparse matrices. These may or may not work.

  Valid values for metric are:

  – from scikit-learn: ['cityblock', 'cosine', 'euclidean', 'l1', 'l2', 'manhattan']

  – from scipy.spatial.distance: ['braycurtis', 'canberra', 'chebyshev', 'correlation', 'dice', 'hamming', 'jaccard', 'mahalanobis', 'minkowski', 'rogerstanimoto', 'russellrao', 'seuclidean', 'sokalmichener', 'sokalsneath', 'sqeuclidean', 'yule']

  Sparse matrices are only supported by scikit-learn metrics. See the documentation for scipy.spatial.distance for details on these metrics.
- **neighborhood** (*int greater than or equal to 1, default=3*) – Number of adjacent clusters to include when adding cluster membership overlap. Should be less than the number of unique cluster labels - 1.
- **overlap_points** (*int greater than or equal to 1, default=2*) – Should not exceed the size of the smallest cluster in *labels*.

- **rejection_threshold** (*float, default=None*) – Determines if any potential overlapping points should be rejected for being too far (from source centroid or nearest source edge point). Default of 'None' is equivalent to setting the threshold to infinity. Note that if value other than 'None' is used, there is no guarantee that all clusters will have overlap points added.

Returns

expanded_clusters – The updated labels, one-hot encoded. Each row is a boolean index to extract cluster membership for a given label. If labels are continuous integers starting at 0, then the row number will match the cluster label; if not, rows are ordered to monotonically increase from the smallest cluster label.

Return type

bool array of shape (n_clusters, n_coordinates)

pgamit.cluster.**select_central_point**(*labels*, *coordinates*, *centroids*, *metric='euclidean'*)

Select the nearest central point in a given neighborhood

Note this code explicitly assumes that centroids are passed from a sklearn clustering result (i.e., kmeans, or bisecting kmeans); those centroids are ordered as monotonically increasing labels. In other words, the output indices will match the labeling order of the input centroids. Note that *n_features* refers to the dimensionality of coordinate system, i.e., 2 for lat/lon, 3 for ECEF (Earth-Centered Earth-Fixed), etc.

Parameters

- **labels** (*ndarray of type int, and shape (n_samples,)*) – Cluster labels for each point in the dataset from prior clustering.
- **coordinates** (*ndarray of shape (n_samples, n_features)*) – Coordinates do not need to match what was used for the prior clustering; i.e., if 'Euclidean' was used to calculate the prior clustering in an X,Y,Z projection, those coordinates can be provided in spherical coordinates, provided that 'haversine' is selected for the *metric* parameter.
- **centroids** (*ndarray of shape (n_clusters, n_features)*) – Coordinates of the cluster centroids; distance metric for centroids should match both *coordinates* and the *metric* parameter.
- **metric** (*str or callable, default='euclidean'*) – Metric to use for distance computation. Any metric from scikit-learn or scipy.spatial.distance can be used. If metric is a callable function, it is called on each pair of instances (rows) and the resulting value recorded. See scikit-learn documentation for additional details.

Returns

central_points_idxs – Indices of the central most point for each cluster; indices match the *labels* ordering.

Return type

int array of shape (n_clusters,)

pgamit.dbConnection module

Project: Parallel.Archive Date: 02/16/2017 Author: Demian D. Gomez

This class is used to connect to the database and handles inserts, updates and selects It also handles the error, info and warning messages

class pgamit.dbConnection.**Cnn**(*configfile*, *use_float=False*, *write_cfg_file=False*)

Bases: object

begin_transac()

close()

commit_transac()

delete(*table*, ***kw*)

Deletes row(s) from the specified table based on the provided keyword arguments.

Parameters: table (str): The table to delete from. kw: Keywords to identify the row(s) to be deleted.

get(*table*, *filter_fields*, *return_fields=None*)

Selects from the given table the records that match filter_fields and returns ONE dictionary. Method should not be used to retrieve more than one single record. Parameters: table (str): The table to select from. filter_fields (dict): The dictionary where the keys are the field names and the values are the filter values. return_fields (list of str): The fields to return. If empty return all columns

Returns: list: A list of dictionaries, each representing a record that matches the filter.

get_columns(*table*)

insert(*table*, ***kw*)

insert_error(*desc*)

insert_event(*event*)

insert_event_bak(*type*, *module*, *desc*)

insert_info(*desc*)

insert_warning(*desc*)

query(*command*)

query_float(*command*, *as_dict=False*)

rollback_transac()

update(*table*, *set_row*, ***kwargs*)

Updates the specified table with new field values. The row(s) are updated based on the primary key(s) indicated in the 'row' dictionary. New values are specified in kwargs. Field names must be enclosed with double quotes to handle camel case names.

Parameters: table (str): The table to update. set_row (dict): New field values for the row. kwargs: The dictionary where the keys are the primary key fields and the values are the row's identifiers.

pgamit.dbConnection.**cast_array_to_float**(*recordset*)

exception pgamit.dbConnection.**dbErrConnect**

Bases: Exception

exception pgamit.dbConnection.**dbErrDelete**

Bases: Exception

exception pgamit.dbConnection.**dbErrUpdate**

Bases: Exception

pgamit.dbConnection.**debug**(*s*)

class pgamit.dbConnection.**query_obj**(*cursor*)

Bases: object

dictresult()

getresult()

ntuples()

pgamit.dbConnection_old module

Project: Parallel.Archive Date: 02/16/2017 Author: Demian D. Gomez

This class is used to connect to the database and handles inserts, updates and selects It also handles the error, info and warning messages

exception pgamit.dbConnection_old.**dbErrConnect**

Bases: Exception

exception pgamit.dbConnection_old.**dbErrDelete**

Bases: Exception

exception pgamit.dbConnection_old.**dbErrInsert**

Bases: Exception

exception pgamit.dbConnection_old.**dbErrUpdate**

Bases: Exception

pgamit.dbConnection_old.**debug**(*s*)

pgamit.network module

Project: Parallel.GAMIT Date: Mar-31-2017 Author: Demian D. Gomez

pom170921: Added new global subnet methods. pom210921: - Added comments - Changed the brackets around the dist variable, instead of checking the

truth of dist it was checking that the list contained a value instead. Should fix far away stations from being added.

- Added a distance check to the tie station assignment. Only stations within 1 Earth radius linear distance will be added to the ties.
- When checking for stations that weren't assigned to a subnet, I mistakenly included the backbone stations which caused duplicate entries in some subnets.

pom220921: - Changed 'if not iterate or len(points) <= BACKBONE_NET:' to

'if not iterate or len(points[n_mask]) <= BACKBONE_NET:' so only the masked stations are counted when comparing against the desired number of backbone stations.

- Added the geocenter to the backbone delaunay triangulation.

pom011021: - Set a minimum number of subnets, the program will reduce the minimum number

of stations per subnet until the minimum number of subnets is created. All changes are in the subnets_delaunay routine.

class pgamit.network.**Network**(*cnn*, *archive*, *GamitConfig*, *stations*, *date*, *check_stations=None*, *ignore_missing=False*)

Bases: object

add_missing_station(*cnn*, *clusters*, *add_station*)

static backbone_delauney(*points*, *stations*)

create_gamit_sessions(*cnn*, *archive*, *clusters*, *backbone*, *ties*, *date*)

make_clusters(*points*, *stations*, *net_limit=40*)

static recover_subnets(*db_subnets*, *stations*)

exception pgamit.network.**NetworkException**(*value*)

Bases: Exception

pgamit.network.**tic**()

pgamit.network.**toc**(*text*)

pgamit.plots module

pgamit.plots.**plot_global_network**(*central_points*, *OC*, *labels*, *points*, *output_path*, *lat_lon=False*)

Plot global GNSS station clustering and subnetwork connections

Plots six views of the GNSS station clustering segmentation: polar stereographic north & south, and four views of the goid disk centered on the equator and rotated in 90 degree increments. Label colors for clusters are stable between subplots, and always refer to the same subnetwork. Overlapping tie points plotted twice, so that whichever subnetwork was plotter first will have that tie station over-plotted when the later subnetwork is plotted.

Parameters

- **central_points** (*ndarray of type int, and shape (n_clusters,)*) – Indices of the central most point (station) for each subnetwork. This is the output from *utils.select_central_point*
- **OC** (*bool array of shape (n_clusters, n_coordinates)*) – The expanded cluster labels, one-hot encoded. Each row is a boolean index to extract cluster membership for a given label. This is the output from *utils.over_cluster*
- **labels** (*ndarray of type int, and shape (n_samples,)*) – Cluster labels for each point in the dataset from prior clustering.
- **points** (*ndarray of shape (n_samples, n_features)*) – Coordinates of the station data, as either ECEF or Lat / Lon
- **output_path** (*str*) – File path to write the output plot to. Note that matplotlib infers the image output format from the file path suffix, so a path ending in '.png' will produce a png, and a '.jpeg' suffix will produce a jpg.
- **lat_lon** (*boolean, default=False*) – If the input parameter *points* is already in lat / lon coordinates. When set to (the default) *False*, input ECEF coordinates are assumed for *points*, and these are transformed to lat / lon for plotting.

pgamit.proto_download module

Project: Parallel.GAMIT Date: 10/18/24 11:53AM Author: Demian D. Gomez

Description goes here

class pgamit.proto_download.**Client**(*on_download_result*, *on_client_stopped*, *server_id: int*, *protocol*, *host*, *port*, *username*, *password*)

Bases: object

class **NextDownload**(*urlpath_file*, *abspath_down_file*)

Bases: NamedTuple

abspath_down_file: str

Alias for field number 1

urlpath_file: str

Alias for field number 0

cond: Condition

finish()

next_download: *NextDownload* | None

proto: *IProtocol*

server_id: int

set_next_download(*urlpath_file: str*, *abspath_down_file: str*)

start_thread()

state: str

stop()

class pgamit.proto_download.**IProtocol**(*protocol: str*, *fqdn: str*, *port: int*, *username: str | None*, *password: str | None*)

Bases: ABC

abstract connect()

desc()

abstract disconnect()

abstract download(*server_path: str*, *dest_path: str*) → bool

abstract list_dir(*server_path: str*)

abstract refresh()

class pgamit.proto_download.**ProtocolFTP**(**args*, ***kargs*)

Bases: *IProtocol*

DEFAULT_PORT = 21

connect()

disconnect()

download(*server_path: str*, *dest_path: str*)

list_dir(*server_path: str*)

refresh()

class pgamit.proto_download.**ProtocolFTPA**(**args*, ***kargs*)

Bases: *ProtocolFTP*

DEFAULT_PORT = 21

connect()

class pgamit.proto_download.ProtocolHTTP(*args, protocol='http', **kargs)

Bases: IProtocol

DEFAULT_PORT = 80

connect()

disconnect()

download(server_path: str, dest_path: str)

list_dir(server_path: str)

refresh()

class pgamit.proto_download.ProtocolHTTPS(*args, **kargs)

Bases: ProtocolHTTP

DEFAULT_PORT = 443

class pgamit.proto_download.ProtocolSFTP(*args, **kargs)

Bases: IProtocol

DEFAULT_PORT = 22

connect()

disconnect()

download(server_path: str, dest_path: str)

list_dir(server_path: str)

refresh()

pgamit.pyArchiveStruct module

Project: Parallel.Archive Date: 02/16/2017 Author: Demian D. Gomez

This class handles the interface between the directory structure of the rinex archive and the databased records. It can be used to retrieve a rinex path based on a rinex database record It can also scan the dirs of a supplied path for d.Z and station.info files (the directories and files have to match the declared directory structure and {stmn}{doy}{session}.{year}d.Z, respectively)

class pgamit.pyArchiveStruct.RinexStruct(cnn, path_cfg='')

Bases: object

build_rinex_path(NetworkCode, StationCode, ObservationYear, ObservationDOY, with_filename=True, filename=None, rinexobj=None)

Function to get the location in the archive of a rinex file. It has two modes of operation: 1) retrieve an existing rinex file, either specific or the rinex for processing (most complete, largest interval) or a specific rinex file (already existing in the rinex table). 2) To get the location of a potential file (probably used for injecting a new file in the archive. No this mode, filename has no effect. :param NetworkCode: NetworkCode of the station being retrieved :param StationCode: StationCode of the station being retrieved :param ObservationYear: Year of the rinex file being retrieved :param ObservationDOY: DOY of the rinex file

being retrieved :param with_filename: if set, returns a path including the filename. Otherwise, just returns the path :param filename: name of a specific file to search in the rinex table :param rinexobj: a pyRinex object to pull the information from (to fill the achive keys). :return: a path with or without filename

get_rinex_record(***kwargs*)

Retrieve a single or multiple records from the rinex table given a set parameters. If parameters are left empty, it wil return all records matching the specified criteria. Each parameter acts like a filter, narrowing down the records returned by the function. The default behavior is to use tables rinex or rinex_proc depending on the provided parameters. E.g. if Interval, Completion and Filename are all left blank, the function will return the records using rinex_proc. Otherwise, the rinex table will be used. :param NetworkCode: filter :param StationCode: filter :param ObservationYear: filter :param ObservationDOY: filter :param Interval: filter :param Completion: filter :param Filename: filter :return: a dictionary will the records matching the provided parameters

insert_rinex(*record=None*, *rinexobj=None*)

Insert a RINEX record and file into the database and archive. If only record is provided, only insert into db If only rinexobj is provided, then RinexRecord of rinexobj is used for the insert. If both are given, then RinexRecord overrides the passed record. :param record: a RinexRecord dictionary to make the insert to the db :param rinexobj: the pyRinex object containing the file being processed :param rnxaction: accion to perform to rinexobj. :return: True if insertion was successful. False if no insertion was done.

parse_archive_keys(*path_filename*, *key_filter=()*)

based on a path and filename, this function parses the data and organizes the information in a dictionary key_filter allows to select which keys you want to get a hold on. The order of the keys in the path is given by the database table rinex_tank_struct :param path: :param key_filter: :return:

remove_rinex(*record*, *move_to_dir=None*)

scan_archive_struct(*rootdir*, *progress_bar=None*)

scan_archive_struct_stninfo(*rootdir*)

pgamit.pyBunch module

Bunch is a subclass of dict with attribute-style access.

```
>>> b = Bunch()
>>> b.hello = 'world'
>>> b.hello
'world'
>>> b['hello'] += "!"
>>> b.hello
'world!'
>>> b.foo = Bunch(lol=True)
>>> b.foo.lol
True
>>> b.foo is b['foo']
True
```

It is safe to import * from this module:

__all__ = (‘Bunch’, ‘bunchify’,’unbunchify’)

un/bunchify provide dictionary conversion; Bunches can also be converted via Bunch.to/fromDict().

class pgamit.pyBunch.**Bunch**

Bases: dict

A dictionary that provides attribute-style access.

```
>>> b = Bunch()
>>> b.hello = 'world'
>>> b.hello
'world'
>>> b['hello'] += "!"
>>> b.hello
'world!'
>>> b.foo = Bunch(lol=True)
>>> b.foo.lol
True
>>> b.foo is b['foo']
True
```

A Bunch is a subclass of dict; it supports all the methods a dict does...

```
>>> sorted(b.keys())
['foo', 'hello']
```

Including update()...

```
>>> b.update({ 'ponies': 'are pretty!' }, hello=42)
>>> print (repr(b))
Bunch(foo=Bunch(lol=True), hello=42, ponies='are pretty!')
```

As well as iteration...

```
>>> [ (k,b[k]) for k in b ]
[('ponies', 'are pretty!'), ('foo', Bunch(lol=True)), ('hello', 42)]
```

And "splats".

```
>>> "The {knights} who say {ni}!".format(**Bunch(knights='lolcats', ni='can haz'))
'The lolcats who say can haz!'
```

See unbunchify/Bunch.toDict, bunchify/Bunch.fromDict for notes about conversion.

static fromDict(*d*)

Recursively transforms a dictionary into a Bunch via copy.

```
>>> b = Bunch.fromDict({'urmom': {'sez': {'what': 'what'}}})
>>> b.urmom.sez.what
'what'
```

See bunchify for more info.

toDict()

Recursively converts a bunch back into a dictionary.

```
>>> b = Bunch(foo=Bunch(lol=True), hello=42, ponies='are pretty!')
>>> b.toDict()
{'ponies': 'are pretty!', 'foo': {'lol': True}, 'hello': 42}
```

See unbunchify for more info.

toJSON(***options*)

Serializes this Bunch to JSON. Accepts the same keyword options as *json.dumps()*.

```
>>> b = Bunch(foo=Bunch(lol=True), hello=42, ponies='are pretty!')
>>> json.dumps(b)
'{"ponies": "are pretty!", "foo": {"lol": true}, "hello": 42}'
>>> b.toJSON()
'{"ponies": "are pretty!", "foo": {"lol": true}, "hello": 42}'
```

pgamit.pyBunch.**bunchify**(*x*)

Recursively transforms a dictionary into a Bunch via copy.

```
>>> b = bunchify({'urmom': {'sez': {'what': 'what'}}})
>>> b.urmom.sez.what
'what'
```

bunchify can handle intermediary dicts, lists and tuples (as well as their subclasses), but ymmv on custom datatypes.

```
>>> b = bunchify({ 'lol': ('cats', {'hah':'i win again'}),
...         'hello': [{'french':'salut', 'german':'hallo'}] })
>>> b.hello[0].french
'salut'
>>> b.lol[1].hah
'i win again'
```

nb. As dicts are not hashable, they cannot be nested in sets/frozensets.

pgamit.pyBunch.**unbunchify**(*x*)

Recursively converts a Bunch into a dictionary.

```
>>> b = Bunch(foo=Bunch(lol=True), hello=42, ponies='are pretty!')
>>> unbunchify(b)
{'ponies': 'are pretty!', 'foo': {'lol': True}, 'hello': 42}
```

unbunchify will handle intermediary dicts, lists and tuples (as well as their subclasses), but ymmv on custom datatypes.

```
>>> b = Bunch(foo=['bar', Bunch(lol=True)], hello=42,
...         ponies=('are pretty!', Bunch(lies='are trouble!')))
>>> unbunchify(b)
{'ponies': ('are pretty!', {'lies': 'are trouble!'}),
 'foo': ['bar', {'lol': True}], 'hello': 42}
```

nb. As dicts are not hashable, they cannot be nested in sets/frozensets.

pgamit.pyDate module

Project: Parallel.Archive Date: 2/23/17 9:28 AM Author: Abel Brown Modified by: Demian D. Gomez

Class that handles all the date conversions betweem different systems and formats

class pgamit.pyDate.**Date**(***kwargs*)

Bases: object

datetime()

ddd()

first_epoch(*out_format='datetime'*)

iso_date()

last_epoch(*out_format='datetime'*)

strftime()

to_json()

wwww()

wwwwd()

yyyy()

yyyyddd(*space=True*)

yyyymmdd()

pgamit.pyDate.**date2doy**(*year*, *month*, *day*, *hour=12*, *minute=0*, *second=0*)

pgamit.pyDate.**date2gpsDate**(*year*, *month*, *day*)

pgamit.pyDate.**doy2date**(*year*, *doy*)

pgamit.pyDate.**fyear2yeardoy**(*fyear*)

pgamit.pyDate.**gpsDate2mjd**(*gpsWeek*, *gpsWeekDay*)

pgamit.pyDate.**mjd2date**(*mjd*)

pgamit.pyDate.**parse_stninfo**(*stninfo_datetime*)

exception pgamit.pyDate.**pyDateException**(*value*)

Bases: Exception

pgamit.pyDate.**yeardoy2fyear**(*year*, *doy*, *hour=12*, *minute=0*, *second=0*)

pgamit.pyETM module

Project: Parallel.Archive Date: 3/3/17 11:27 AM Author: Demian D. Gomez

class pgamit.pyETM.**CoSeisJump**(*NetworkCode*, *StationCode*, *soln*, *t*, *date*, *relaxation*, *metadata*, *dtype=10*, *magnitude=0.0*, *action='A'*, *fit=True*, *models=()*, *epi_distance=0.0*)

Bases: *Jump*

eval(*t*)

rehash()

class pgamit.pyETM.**DailyRep**(*cnn*, *NetworkCode*, *StationCode*, *plotit=False*, *no_model=False*, *gamit_soln=None*, *project=None*)

Bases: *ETM*

get_residuals_dict()

class pgamit.pyETM.**ETM**(*cnn*, *soln*, *no_model=False*, *FitEarthquakes=True*, *FitGenericJumps=True*, *FitPeriodic=True*, *plotit=False*, *ignore_db_params=False*, *models=()*, *plot_remove_jumps=False*, *plot_polynomial_removed=False*)

Bases: object

adjust_lsq(*Ai*, *Li*)

apply_postseismic_model(*postseismic*)

static autoscale_y(*ax*, *margin=0.1*)

This function rescales the y-axis based on the data that is visible given the current xlim of the axis. ax – a matplotlib axes object margin – the fraction of the total height of the y-data to pad the upper and lower ylims

static chi2inv(*chi*, *df*)

Return prob(chisq >= chi, with df degrees of freedom).

df must be even.

display_postseismic_params(*postseismic*)

enable_picking(*event*)

entropy_sigma()

get_data_segments(*tolerance*)

get_outliers_list()

Function to obtain the outliers based on the ETMs sigma :return: a list containing the network code, station code and dates of the outliers in the time series

get_xyz_s(*year*, *doy*, *jmp=None*, *sigma_h=0.1*, *sigma_v=0.15*, *force_model=False*)

static isPD(*B*)

Returns true when input is positive-definite, via Cholesky

load_parameters(*params*, *l*)

nearestPD(*A*)

Find the nearest positive-definite matrix to input

A Python/Numpy port of John D'Errico's *nearestSPD* MATLAB code [1], which credits [2].

[1] https://www.mathworks.com/matlabcentral/fileexchange/42885-nearestspd

[2] N.J. Higham, "Computing a nearest symmetric positive semidefinite matrix" (1988): https://doi.org/10.1016/0024-3795(88)90223-6

onpick(*event*)

plot(*pngfile=None*, *t_win=None*, *residuals=False*, *plot_missing=True*, *ecef=False*, *plot_outliers=True*, *fileio=None*)

plot_hist(*pngfile=None*, *fileio=None*)

plot_jumps(*ax*)

plot_missing_soln(*ax*)

process_covariance()

rotate_2neu(*ecef*)

rotate_2xyz(*neu*)

rotate_sig_cov(*sigmas=None*, *covar=None*)

run_adjustment(*cnn*, *l*, *soln*, *ignore_db_params=False*)

save_excluded_soln(*cnn*)

save_parameters(*cnn*)

set_lims(*t_win*, *plt*, *ax*)

todictionary(*time_series=False*, *model=False*)

static warn_with_traceback(*message*, *category*, *filename*, *lineno*, *file=None*, *line=None*)

class pgamit.pyETM.**Earthquakes**(*cnn*, *NetworkCode*, *StationCode*, *soln*, *t*, *FitEarthquakes=True*, *models=()*)

Bases: object

class pgamit.pyETM.**EtmFunction**(***kwargs*)

Bases: object

load_parameters(***kwargs*)

class pgamit.pyETM.**FileETM**(*cnn*, *poly_list=None*, *plotit=False*, *no_model=False*, *plot_remove_jumps=False*, *plot_polynomial_removed=False*)

Bases: *ETM*

class pgamit.pyETM.**GamitETM**(*cnn*, *NetworkCode*, *StationCode*, *plotit=False*, *no_model=False*, *gamit_soln=None*, *stack_name=None*, *models=()*, *ignore_db_params=False*, *plot_remove_jumps=False*, *plot_polynomial_removed=False*)

Bases: *ETM*

get_etm_soln_list(*use_ppp_model=False*, *cnn=None*)

class pgamit.pyETM.**GamitSoln**(*cnn*, *polyhedrons*, *NetworkCode*, *StationCode*, *stack_name*)

Bases: object

"class to extract the GAMIT polyhedrons from the database

class pgamit.pyETM.**GenericJumps**(*cnn*, *NetworkCode*, *StationCode*, *soln*, *t*, *FitGenericJumps=True*)

Bases: object

class pgamit.pyETM.**Jump**(*NetworkCode*, *StationCode*, *soln*, *t*, *date*, *metadata*, *dtype=1*, *action='A'*, *fit=True*)

Bases: *EtmFunction*

generic jump (mechanic jump, frame change, etc) class :argument NetworkCode :argument StationCode

eval(*t*)

load_parameters(***kwargs*)

rehash()

remove_from_fit()

class pgamit.pyETM.**JumpTable**(*cnn*, *NetworkCode*, *StationCode*, *soln*, *t*, *FitEarthquakes=True*, *FitGenericJumps=True*, *models=()*)

Bases: object

get_design_ts(*t*)

insert_jump(*jump*)

load_parameters(*params*, *sigmas*)

param_count()

print_parameters()

pgamit.pyETM.**LABEL**(*msg*)

class pgamit.pyETM.**ListSoln**(*cnn*, *polyhedrons*, *NetworkCode*, *StationCode*, *stack_name='file-unknown'*)

Bases: *GamitSoln*

"class to extract the polyhedrons from a list

class pgamit.pyETM.**Model**(*m_type*, ***kwargs*)

Bases: object

LOG = 2

VEL = 1

eval(*t*)

class pgamit.pyETM.**PPPETM**(*cnn*, *NetworkCode*, *StationCode*, *plotit=False*, *no_model=False*, *models=()*, *ignore_db_params=False*, *plot_remove_jumps=False*, *plot_polynomial_removed=False*)

Bases: *ETM*

class pgamit.pyETM.**Periodic**(*cnn*, *NetworkCode*, *StationCode*, *soln*, *t*, *FitPeriodic=True*)

Bases: *EtmFunction*

"class to determine the periodic terms to be included in the ETM

get_design_ts(*ts*)

print_parameters()

class pgamit.pyETM.**Polynomial**(*cnn*, *NetworkCode*, *StationCode*, *soln*, *t*, *t_ref=0*, *models=()*)

Bases: *EtmFunction*

"class to build the linear portion of the design matrix

get_design_ts(*ts*)

load_parameters(*params*, *sigmas*, *t_ref*)

print_parameters(*ref_xyz*, *lat*, *lon*)

class pgamit.pyETM.**PppSoln**(*cnn*, *NetworkCode*, *StationCode*)

Bases: object

"class to extract the PPP solutions from the database

pgamit.pyETM.**distance**(*lon1*, *lat1*, *lon2*, *lat2*)

Calculate the great circle distance between two points on the earth (specified in decimal degrees)

exception pgamit.pyETM.**pyETMException**(*value*)

Bases: Exception

exception pgamit.pyETM.**pyETMException_Model**(*value*)

Bases: *pyETMException*

exception pgamit.pyETM.**pyETMException_NoDesignMatrix**(*value*)

Bases: *pyETMException*

pgamit.pyETM.**tic**()

pgamit.pyETM.**to_list**(*dictionary*)

pgamit.pyETM.**to_postgres**(*dictionary*)

pgamit.pyETM.**toc**(*text*)

pgamit.pyEvents module

Project: Parallel.PPP Date: 10/25/17 8:53 AM Author: Demian D. Gomez

Class to manage (insert, create and query) events produced by the Parallel.PPP wrapper

class pgamit.pyEvents.**Event**(***kwargs*)

Bases: dict

db_dict()

pgamit.pyGamitConfig module

Project: Date: 3/31/17 5:28 PM Author: Demian D. Gomez

class pgamit.pyGamitConfig.**GamitConfiguration**(*session_config_file*, *check_config=True*)

Bases: *ReadOptions*

load_session_config(*session_config_file*, *check_config*)

exception pgamit.pyGamitConfig.**pyGamitConfigException**(*value*)

Bases: Exception

pgamit.pyGamitSession module

Project: Parallel.GAMIT Date: 4/3/17 6:57 PM Author: Demian D. Gomez

class pgamit.pyGamitSession.**GamitSession**(*cnn*, *archive*, *name*, *org*, *subnet*, *date*, *GamitConfig*, *stations*, *ties=()*, *centroid=()*)

Bases: object

copy_sestbl_procdef_atx()

create_apr_sittbl_file()

create_otl_list()

create_sitedef()

create_station_info()

generate_kml()

get_rinex_filenames()

initialize()

link_tables()

parse_sinex()

exception pgamit.pyGamitSession.**GamitSessionException**(*value*)

Bases: Exception

pgamit.pyGamitTask module

Project: Parallel.GAMIT Date: 4/3/17 6:57 PM Author: Demian D. Gomez

class pgamit.pyGamitTask.**GamitTask**(*remote_pwd*, *params*, *solution_pwd*)

Bases: object

combine_systems(*results*)

create_replace_links()

execute(*script_name*, *shell=False*)

fetch_orbits()

fetch_rinex()

finish()

log(*message*, *no_timestamp=False*)

process_tropo(*results*)

run_gamit(*system*, *dummy=False*)

start(*dirname*, *year*, *doy*, *dry_run=False*)

translate_station_alias(*station_alias*)

window_rinex(*Rinex*, *window*)

pgamit.pyGamitTask.**now_str**()

pgamit.pyGamitTask.**replace_vars**(*archive*, *date*)

pgamit.pyGlobkTask module

Project: Parallel.GAMIT Date: Dic-03-2016 Author: Demian D. Gomez

class pgamit.pyGlobkTask.**Globk**(*pwd_comb*, *date*, *Sessions*, *net_type='regional'*)

Bases: object

create_combination_script(*date*, *org*)

execute(*parse_sinex=True*)

linktables(*year*, *eop_type*)

parse_sinex()

exception pgamit.pyGlobkTask.**GlobkException**(*value*)

Bases: Exception

pgamit.pyJobServer module

Project: Parallel.PPP Date: 9/13/17 6:30 PM Author: Demian D. Gomez

This module handles the cluster nodes and checks all the necessary dependencies before sending jobs to each node

class pgamit.pyJobServer.**JobServer**(*Config*, *check_gamit_tables=None*, *check_archive=True*, *check_executables=True*, *check_atx=True*, *run_parallel=True*, *software_sync=()*)

Bases: object

check_cluster(*status*, *node*, *job*)

cleanup()

close_cluster()

cluster_status(*status*, *node*, *job*)

Called by dispy on cluster events

create_cluster(*function*, *deps=()*, *callback=None*, *progress_bar=None*, *verbose=False*, *modules=()*, *on_nodes_changed=None*, *node_setup=None*, *node_cleanup=None*)

submit(**args*)

function to submit jobs to dispy. If run_parallel == False, the jobs are executed :param args: :return:

submit_async(**args*)

wait()

wrapped function to wait for cluster execution :return: none

pgamit.pyJobServer.**setup**(*modules*)

function to import modules in the nodes :return: 0

pgamit.pyJobServer.**test_node**(*check_gamit_tables=None*, *check_archive=True*, *check_executables=True*, *check_atx=True*, *software_sync=()*)

pgamit.pyLeastSquares module

Project: Parallel.Archive Date: 3/3/17 11:27 AM Author: Demian D. Gomez

module for robust least squares operations

pgamit.pyLeastSquares.**adjust_lsq**(*A*, *L*, *limit=2.5*)

pgamit.pyLeastSquares.**rotate_vector**(*ecef*, *lat*, *lon*)

pgamit.pyOTL module

project: Parallel.Archive Date: 02/16/2017 Author: Demian D. Gomez

Ocean loading coefficients class. It runs and reads grdtab (from GAMIT).

class pgamit.pyOTL.**OceanLoading**(*StationCode*, *grdtab*, *otlgrid*, *x=None*, *y=None*, *z=None*)

Bases: object

calculate_otl_coeff(*x=None*, *y=None*, *z=None*)

exception pgamit.pyOTL.**pyOTLException**(*value*)

Bases: Exception

pgamit.pyOkada module

Project: Parallel.GAMIT Date: 5/15/24 9:08 AM Author: Demian D. Gomez

Seismic-score (s-score) class that allows testing if a latitude longitude locations requires co-seismic displacement parameters on its ETM. The class includes the formulations from Wells and Coppersmith (1994) to estimate the fault dimensions of the event and calculate the level-2 s-score. If the nodal planes of the event are not available, then the class returns the level-1 (isotropic) s-score.

Okada code Translated from Matlab to Python. Original by Michael Bevis based on Okada 1985

OKADA surface displacements due to dislocation in an elastic half-space This is a matlab implementation of Okada's fortran code SRECTF except that only surface displacements are computed, not strains or tilts. See document Okada-CoordSystem.pdf by M. Bevis for graphics describing of Okada's (1985) coordinate systems and sign conventions

USAGE: [ux,uy,uz] = okada(alpha,x,y,d,L1,L2,W1,W2,snd,csd,B1,B2,B3)

A rectangular dislocation is located in an elastic halfspace. The right-handed coordinate system {X,Y,Z} has Z positive upwards, and the X and Y axes lie on the surface of the halfspace. The upper and lower edges of the rectangle are horizontal and parallel to the X axis, which constitutes the 'strike direction'. The rectangular dislocation is located within a dipping plane which intersects the Z axis at -d (so d is a positive number coinciding with 'depth'). The dip of this plane is the angle delta which is measured positive anticlockwise from horizontal looking in the -X direction, as seen in Figure 1 of Okada's 1985 BSSA paper. The user must specify sd = sin(delta) and cd = cos(delta) rather than delta itself, since this convention allows a general treatment of the vertical fault (switching sd between +1 and -1 when cd = 0, changes which side of the fault is going up). The position of the dislocation within the dipping plane is specified using a cartesian axis system {L,W} confined to the plane of the dislocation, with the L axis being parallel to and located beneath the X axis. The origin of the {L,W} system is at X=Y=0, Z=-d. The actual dislocation is located in the rectangle L1 <= L <= L2, W1 <= W <= W2.

With reference to Okada (1985) Figure 2, the fault outlined with a solid line has L1=0,L2=L, W1=0, W2=W. To specify the extended fault shown with the dashed line, change L1 to -L.

The Burgers vector B for the dislocation has three components B1,B2 and B3 where B1 is the L component, B2 is the W component, and B3 is the component normal to the dislocation surface.

The elastic half-space (Z<0) is uniform and isotropic. The displacement field produced by the dislocation depends only on scalar alpha, where alpha = mu/(lambda+mu). When the Lamé parameters lambda and mu are equal, the elastic is said to be a Poisson solid and alpha=0.5

The stations where displacememts are to be computed have coordinates x and y (which can be scalars, vectors or matrices, but must have the same sizes. THe output arguments ux, uy and uz, which have the sames sizes as input arguments x and y, are the X, Y and Z components of displacement at each station.

Okada.m uses internal matlab function okadakernel.m to peform the indefinite integrals.

version 1.0 Michael Bevis 4 Nov 99 version 1.1 26 Feb 04 version 1.2 (change argument names, new header) 11 Oct 13 version 1.3 (changed sd,cd too) 3 Apr 14 version 1.0 (PYTHON) translated by Demian Gomez 15 May 24

class pgamit.pyOkada.**Score**(*event_lat*, *event_lon*, *depth_km*, *magnitude*, *strike=()*, *dip=()*, *rake=()*, *event_date=None*, *density=250*, *location=''*)

Bases: object

compute_disp_field(*scale_factor=1.0*, *limit=0.001*)

save_masks(*txt_file=None*, *kmz_file=None*, *include_postseismic=False*)

Function to export coseismic mask. Method returns the kml structure. If txt_file and/or kmz_file are given, then files are saved

score(*lat*, *lon*)

class pgamit.pyOkada.**ScoreTable**(*cnn*, *lat*, *lon*, *sdate*, *edate*)

Bases: object

Given a connection to the database, lat and lon of point of interest, and date range, find all the seismic events with level-2 s-score = 1. If no strike, dip, and rake parameters available, return events with level-1 s-score > 0 Returns a list with [mag, date, lon, lat] ordered by ascending date and descending magnitude.

pgamit.pyOkada.**acosd**(*x*)

pgamit.pyOkada.**asind**(*x*)

pgamit.pyOkada.**atand**(*x*)

pgamit.pyOkada.**cosd**(*x*)

pgamit.pyOkada.**distance**(*lon1*, *lat1*, *lon2*, *lat2*)

Calculate the great circle distance between two points on the earth (specified in decimal degrees)

pgamit.pyOkada.**inv_azimuthal**(*x*, *y*, *lon*, *lat*)

pgamit.pyOkada.**okada**(*alpha*, *x*, *y*, *d*, *L1*, *L2*, *W1*, *W2*, *snd*, *csd*, *B1*, *B2*, *B3*)

pgamit.pyOkada.**okadakernel**(*alp*, *xi*, *et*, *q*, *sd*, *cd*, *disl1*, *disl2*, *disl3*)

pgamit.pyOkada.**sind**(*x*)

pgamit.pyOptions module

Project: Parallel.Archive Date: 3/21/17 5:36 PM Author: Demian D. Gomez

Class with all the configuration information necessary to run many of the scripts. It loads the config file (gnss_data.cfg).

class pgamit.pyOptions.**ReadOptions**(*configfile*, *write_cfg_file=False*)

Bases: object

pgamit.pyPPP module

Project: Parallel.PPP Date: 2/21/17 3:34 PM Author: Demian D. Gomez

Python wrapper for PPP. It runs the NRCAN PPP and loads the information from the summary file. Can be used without a database connection, except for PPPSpatialCheck

class pgamit.pyPPP.**PPPSpatialCheck**(*lat=None*, *lon=None*, *h=None*, *epoch=None*)

Bases: object

verify_spatial_coherence(*cnn*, *StationCode*, *search_in_new=False*)

class pgamit.pyPPP.**RunPPP**(*in_rinex*, *otl_coeff*, *options*, *sp3types*, *sp3altrn*, *antenna_height*, *strict=True*, *apply_met=True*, *kinematic=False*, *clock_interpolation=False*, *hash=0*, *erase=True*, *decimate=True*, *solve_coordinates=True*, *solve_troposphere=105*, *back_substitution=False*, *elev_mask=10*, *x=0*, *y=0*, *z=0*, *observations='2'*)

Bases: *PPPSpatialCheck*

static check_eop(*section*)

static check_otl(*section*)

static check_phase_center(*section*)

cleanup()

config_session()

copyfiles()

exec_ppp()

static get_clock(*section*, *kinematic*)

static get_frame(*section*)

get_orbits(*orbit_type*)

get_pr_observations(*section*, *kinematic*)

static get_sigmas(*section*, *kinematic*)

get_text(*summary*, *start*, *end*)

static get_xyz(*section*)

load_record()

parse_summary()

write_otl()

pgamit.pyPPP.**find_between**(*s*, *first*, *last*)

exception pgamit.pyPPP.**pyRunPPPException**(*value*)

Bases: Exception

exception pgamit.pyPPP.**pyRunPPPExceptionCoordConflict**(*value*)

Bases: *pyRunPPPException*

exception pgamit.pyPPP.**pyRunPPPExceptionEOPError**(*value*)

Bases: *pyRunPPPException*

exception pgamit.pyPPP.**pyRunPPPExceptionNaN**(*value*)

Bases: *pyRunPPPException*

exception pgamit.pyPPP.**pyRunPPPExceptionTooFewAcceptedObs**(*value*)

Bases: *pyRunPPPException*

exception pgamit.pyPPP.**pyRunPPPExceptionZeroProcEpochs**(*value*)

Bases: *pyRunPPPException*

pgamit.pyParseAntex module

Project: Parallel.Archive Date: 2/25/17 7:15 PM Author: Demian D. Gomez

class pgamit.pyParseAntex.**ParseAntexFile**(*filename*)

Bases: object

pgamit.pyParseZTD module

Project: Parallel.GAMIT Date: 7/19/20 6:33 PM Author: Demian D. Gomez

class pgamit.pyParseZTD.**ParseZtdTask**(*GamitConfig*, *project*, *sessions*, *date*)

Bases: object

execute()

pgamit.pyProducts module

Project: Date: 2/23/17 10:12 AM Author: Demian D. Gomez

class pgamit.pyProducts.**GetBrdcOrbits**(*brdc_archive*, *date*, *copyto*, *no_cleanup=False*)

Bases: *OrbitalProduct*

cleanup()

class pgamit.pyProducts.**GetClkFile**(*clk_archive*, *date*, *sp3types*, *copyto*, *no_cleanup=False*)

Bases: *OrbitalProduct*

cleanup()

class pgamit.pyProducts.**GetEOP**(*sp3archive*, *date*, *sp3types*, *copyto*)

Bases: *OrbitalProduct*

class pgamit.pyProducts.**GetSp3Orbits**(*sp3archive*, *date*, *sp3types*, *copyto*, *no_cleanup=False*)

Bases: *OrbitalProduct*

cleanup()

class pgamit.pyProducts.**OrbitalProduct**(*archive*, *date*, *filename*, *copyto*, *short_name=True*)

Bases: object

exception pgamit.pyProducts.**pyBrdcException**(*value*)

Bases: *pyProductsException*

exception pgamit.pyProducts.**pyClkException**(*value*)

Bases: *pyProductsException*

exception pgamit.pyProducts.**pyEOPException**(*value*)

Bases: *pyProductsException*

exception pgamit.pyProducts.**pyProductsException**(*value*)

Bases: Exception

exception pgamit.pyProducts.**pyProductsExceptionUnreasonableDate**(*value*)

Bases: *pyProductsException*

exception pgamit.pyProducts.**pySp3Exception**(*value*)

Bases: *pyProductsException*

pgamit.pyRinex module

Project: Parallel.Archive Date: 02/16/2017 Author: Demian D. Gomez

class pgamit.pyRinex.**ReadRinex**(*NetworkCode*, *StationCode*, *origin_file*, *no_cleanup=False*, *allow_multiday=False*, *min_time_seconds=3600*)

Bases: *RinexRecord*

ConvertRinex(*to_version*)

RunGfzrnx()

RunRinSum()

deprecated function to run RinSum :return:

apply_file_naming_convention()

function to rename a file to make it consistent with the RINEX naming convention gfzrnx now makes things simpler: just use the appropriate command :return:

auto_coord(*brdc*, *chi_limit=3*)

auto_coord_sh_rx2apr(*brdc*, *chi_limit=3*)

check_header()

check_interval()

cleanup()

compress_local_copyto(*path*, *force_filename=None*)

create_script(*name*, *command*)

create_temp_dirs()

decimate(*decimate_rate*, *copyto=None*)

format_record(*record_dict*, *record*, *values*)

get_firstobs()

get_header()

indentify_file(*input_file*)

insert_comment(*header*, *comment*)

log_event(*desc*)

move_origin_file(*path*, *destiny_type=0*)

multiday_handle(*origin_file*)

normalize_header(*NewValues=None*, *brdc=None*, *x=None*, *y=None*, *z=None*)

parse_output(*output*, *min_time_seconds=3600*)

purge_comments()

read_data()

read_fields(*line*, *record*, *format_tuple*)

remove_systems(*systems=('C', 'E', 'I', 'J', 'R', 'S')*, *copyto=None*)

rename(*new_name=None*, *NetworkCode=None*, *StationCode=None*)

replace_record(*header*, *record*, *new_values*)

split_file()

uncompress()

window_data(*start=None*, *end=None*, *copyto=None*)

Window the RINEX data using GFZRNX :param start: a start datetime or self.firstObs if None :param end: a end datetime or self.lastObs if None :return:

write_rinex(*new_header*)

class pgamit.pyRinex.**RinexRecord**(*NetworkCode=None*, *StationCode=None*)

Bases: object

load_record()

exception pgamit.pyRinex.**pyRinexException**(*value*)

Bases: Exception

exception pgamit.pyRinex.**pyRinexExceptionBadFile**(*value*)

Bases: *pyRinexException*

exception pgamit.pyRinex.**pyRinexExceptionNoAutoCoord**(*value*)

Bases: *pyRinexException*

exception pgamit.pyRinex.**pyRinexExceptionSingleEpoch**(*value*)

Bases: *pyRinexException*

pgamit.pyRinexName module

Project: Parallel.GAMIT Date: 7/15/20 5:25 PM Author: Demian D. Gomez

exception pgamit.pyRinexName.**RinexNameException**(*value*)

Bases: Exception

class pgamit.pyRinexName.**RinexNameFormat**(*filename*, *StationCode='XXXX'*, *monument='0'*, *receiver='0'*, *country='UNK'*, *data_source='R'*, *date=None*, *file_period='01D'*, *data_frequency='30S'*, *data_type='MO'*, *version=2*)

Bases: object

filename_base(*override_freq=None*)

filename_no_ext(*no_path=False*, *override_freq=None*)

identify_rinex_type(*filename*)

split_filename(*filename*)

to_rinex_format(*to_type*, *no_path=False*, *override_freq=None*)

pgamit.pyRinexName.**check_year**(*year*)

pgamit.pyRinexName.**path_replace_tags**(*filename*, *date:* Date, *NetworkCode=''*, *StationCode=''*, *Marker=0*, *CountryCode='ARG'*)

function to replace tags / keywords in a string for their corresponding values

pgamit.pyRunWithRetry module

Project: Parallel.Archive Date: 02/16/2017 Author: Demian D. Gomez

class pgamit.pyRunWithRetry.**RunCommand**(*command*, *time_out*, *cwd='/Users/megan/Documents/Code/fork_Parallel.GAMIT/Parallel.GAMIT/docs'*, *cat_file=None*)

Bases: object

run_shell()

exception pgamit.pyRunWithRetry.**RunCommandWithRetryExeception**(*value*)

Bases: Exception

class pgamit.pyRunWithRetry.**command**(*command*, *cwd='/Users/megan/Documents/Code/fork_Parallel.GAMIT/Parallel.GAMIT/docs'*, *cat_file=None*)

Bases: Thread

run()

Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

wait(*timeout=None*)

pgamit.pyStack module

class pgamit.pyStack.**Combination**(*polyhedrons*)

Bases: *Polyhedron*

class pgamit.pyStack.**Polyhedron**(*vertices*, *project*, *date*, *rot=True*, *aligned=False*)

Bases: object

align(*target=None*, *set_aligned=True*, *helmert=None*, *scale=False*, *verbose=False*)

Align to another polyhedron object using a Helmert transformation defined during the initialization of the object :param target: polyhedron object :param set_aligned: determine whether the polyhedron should be marked as aligned or not after performing the Helmert transformation :param helmert: provide an externally calculated helmert transformation :return: before and after residuals and list of stations

ax(*scale=False*)

function to append scale to the design matrix :return: Ax with scale

ay(*scale=False*)

function to append scale to the design matrix :return: Ay with scale

az(*scale=False*)

function to append scale to the design matrix :return: Az with scale

info()

class pgamit.pyStack.**Stack**(*cnn*, *project*, *name*, *redo=False*, *end_date=None*)

Bases: list

align_spaces(*target_dict*)

build_design(*stations*, *scale=False*)

calculate_etms()

Estimates the trajectory models for all stations in the stack :return:

get_station(*NetworkCode*, *StationCode*)

Obtains the time series for a given station :param NetworkCode: :param StationCode: :return: a numpy array with the time series [x, y, z, yr, doy, fyear]

remove_common_modes(*target_periods=None*)

save()

save the polyhedrons to the database :return: nothing

to_json(*json_file*)

pgamit.pyStack.**adjust_lsq**(*A*, *L*, *P=None*)

pgamit.pyStack.**main**()

pgamit.pyStack.**np_array_vertices**(*a*)

pgamit.pyStack.**print_residuals**(*NetworkCode*, *StationCode*, *residuals*, *lat*, *lon*, *components=('N', 'E', 'U')*)

pgamit.pyStatic1d module

Project: Parallel.Archive Date: 03/11/2024 Author: Demian D. Gomez

class pgamit.pyStatic1d.**Static1d**(*cnn*, *max_expansion=5000*, *greens_function_file=None*, *min_depth=0*, *max_depth=10*)

Bases: object

pgamit.pyStation module

Project: Parallel.GAMIT Date: 3/31/17 3:39 PM Author: Demian D. Gomez

Class that holds the station metadata needed to process in GAMIT

class pgamit.pyStation.**Station**(*cnn*, *NetworkCode*, *StationCode*, *dates*, *StationAlias=None*)

Bases: object

check_gamit_soln(*cnn*, *project*, *date*)

Function to check if a gamit solution exists for this station, project and date :param cnn: database connection :param project: name of the project to search :param date: date to check if a solution exists :return: True if solution exists, otherwise False

generate_alias()

static id_generator(*size=4*, *chars='abcdefghijklmnopqrstuvwxyz0123456789'*)

class pgamit.pyStation.**StationCollection**(*stations=None*)

Bases: list

StationCollection object accumulates Station objects verifying there is no collision in StationCodes It is essentially a list with an overloaded append method that triggers verification of the StationCodes and makes changes to StationAliases as needed

append(*station*)

Append object to the end of the list.

compare_aliases(*station*)

get_active_coordinates(*date*)

obtain a numpy array of the coordinates for the active stations in the collection :param date: :return: numpy array

get_active_stations(*date*, *check_aliases=False*)

create a collection with the stations that actually have observations for a given day by default, the aliases are leaved untouched :param date: to check if observations are available or not :param check_aliases: boolean, check the aliases of the stations and change them if necessary :return: a collection with the stations that have observations

ismember(*station*)

determines if a station is part of the collection or not :param station: station object or string :return: boolean

labels_array()

class pgamit.pyStation.**StationInstance**(*cnn*, *archive*, *station*, *date*, *GamitConfig*, *is_tie=False*)

Bases: object

DebugCoord()

GetApr()

GetRinexFilename()

GetSittbl()

GetStationInformation()

exception pgamit.pyStation.**pyStationCollectionException**(*value*)

Bases: *pyStationException*

exception pgamit.pyStation.**pyStationException**(*value*)

Bases: Exception

pgamit.pyStationInfo module

Project: Parallel.Archive Date: 02/16/2017 Author: Demian D. Gomez

class pgamit.pyStationInfo.**StationInfo**(*cnn*, *NetworkCode=None*, *StationCode=None*, *date=None*, *allow_empty=False*, *h_tolerance=0*)

Bases: object

New parameter: h_tolerance makes the station info more tolerant to gaps. This is because station info in the old days had a break in the middle and the average epoch was falling right in between the gap

DeleteStationInfo(*record*)

InsertStationInfo(*record*)

UpdateStationInfo(*record*, *new_record*)

antenna_check(*frames*)

load_stationinfo_records()

overlaps(*qrecord*)

parse_station_info(*stninfo_file_list*)

function used to parse a station information file :param stninfo_file_list: a station information file or list containing station info records :return: a list of StationInformationRecords

static records_are_equal(*record1*, *record2*)

return_stninfo(*record=None*, *no_dharp_translate=False*)

return a station information string to write to a file (without header :param record: to print a specific record, pass a record, otherwise, leave empty to print all records :param no_dharp_translate: specify if DHARP translation should be done or not :return: a string in station information format

return_stninfo_short(*record=None*)

prints a simplified version of the station information to better fit screens :param record: to print a specific record, pass a record, otherwise, leave empty to print all records :return: a string in station information format. It adds the NetworkCode dot StationCode

rinex_based_stninfo(*ignore*)

station_info_gaps()

to_dharp(*record*)

function to convert the current height code to DHARP :return: DHARP height

to_json()

class pgamit.pyStationInfo.**StationInfoRecord**(*NetworkCode=None*, *StationCode=None*, *record=None*)

Bases: *Bunch*

database()

parse_station_record(*record*)

to_json()

exception pgamit.pyStationInfo.**pyStationInfoException**(*value*)

Bases: Exception

exception pgamit.pyStationInfo.**pyStationInfoHeightCodeNotFound**(*value*)

Bases: *pyStationInfoException*

pgamit.pyTerminal module

class pgamit.pyTerminal.**ProgressBar**(*term*, *header*)

Bases: object

A 3-line progress bar, which looks like:

```
                        Header
20% [===========----------------------------------]
                   progress message
```

The progress bar is colored, if the terminal supports color output; and adjusts to the width of the terminal.

BAR = '%3d%% ${GREEN}[${BOLD}%s%s${NORMAL}${GREEN}]${NORMAL}\n'

HEADER = '${BOLD}${CYAN}%s${NORMAL}\n\n'

clear()

cleared

true if we haven't drawn the bar yet.

update(*percent*, *message*)

class pgamit.pyTerminal.**TerminalController**(*term_stream=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='utf-8'>*)

Bases: `object`

A class that can be used to portably generate formatted output to a terminal.

TerminalController defines a set of instance variables whose values are initialized to the control sequence necessary to perform a given action. These can be simply included in normal output to the terminal:

```
>>> term = TerminalController()
>>> print 'This is '+term.GREEN+'green'+term.NORMAL
```

Alternatively, the *render()* method can used, which replaces '${action}' with the string required to perform 'action':

```
>>> term = TerminalController()
>>> print term.render('This is ${GREEN}green${NORMAL}')
```

If the terminal doesn't support a given action, then the value of the corresponding instance variable will be set to ''. As a result, the above code will still work on terminals that do not support color, except that their output will not be colored. Also, this means that you can test whether the terminal supports a given action by simply testing the truth value of the corresponding instance variable:

```
>>> term = TerminalController()
>>> if term.CLEAR_SCREEN:
...     print 'This terminal supports clearning the screen.'
```

Finally, if the width and height of the terminal are known, then they will be stored in the *COLS* and *LINES* attributes.

BG_BLACK = ''

BG_BLUE = ''

BG_CYAN = ''

BG_GREEN = ''

BG_MAGENTA = ''

BG_RED = ''

BG_WHITE = ''

BG_YELLOW = ''

BLACK = ''

BLINK = ''

Turn on blink mode

BLUE = ''

BOL = ''

Move the cursor to the beginning of the line

BOLD = ''

Turn on bold mode

CLEAR_BOL = ''

Clear to the beginning of the line.

CLEAR_EOL = ''

Clear to the end of the line.

CLEAR_EOS = ''

Clear to the end of the screen

CLEAR_SCREEN = ''

Clear the screen and move to home position

COLS = None

Width of the terminal (None for unknown)

CYAN = ''

DIM = ''

Turn on half-bright mode

DOWN = ''

Move the cursor down one line

GREEN = ''

HIDE_CURSOR = ''

Make the cursor invisible

LEFT = ''

Move the cursor left one char

LINES = None

Height of the terminal (None for unknown)

MAGENTA = ''

NORMAL = ''

Turn off all modes

RED = ''

REVERSE = ''

Turn on reverse-video mode

RIGHT = ''

Move the cursor right one char

SHOW_CURSOR = ''

Make the cursor visible

UP = ''

Move the cursor up one line

WHITE = ''

YELLOW = ''

render(*template*)

Replace each $-substitutions in the given template string with the corresponding terminal control string (if it's defined) or '' (if it's not).

pgamit.pyTrimbleT0x module

Project: Parallel.GAMIT Date: 11/28/2023 Author: Demian D. Gomez Module to convert T0x files to RINEX

pgamit.pyTrimbleT0x.**convert_trimble**(*path*, *stnm*, *out_path*, *plain_path=False*, *antenna=None*)

pgamit.pyVoronoi module

Author: Tyler Reddy

The purpose of this Python module is to provide utility code for handling spherical Voronoi Diagrams.

exception pgamit.pyVoronoi.**IntersectionError**

Bases: Exception

pgamit.pyVoronoi.**calculate_Vincenty_distance_between_spherical_points**(*cartesian_array_1*, *cartesian_array_2*, *sphere_radius*)

Apparently, the special case of the Vincenty formula (http://en.wikipedia.org/wiki/Great-circle_distance) may be the most accurate method for calculating great-circle distances.

pgamit.pyVoronoi.**calculate_and_sum_up_inner_sphere_surface_angles_Voronoi_polygon**(*array_ordered_Voronoi_poi*, *sphere_radius*)

Takes an array of ordered Voronoi polygon vertices (for a single generator) and calculates the sum of the inner angles on the sphere surface. The resulting value is theta in the equation provided here: http://mathworld.wolfram.com/SphericalPolygon.html

pgamit.pyVoronoi.**calculate_haversine_distance_between_spherical_points**(*cartesian_array_1*, *cartesian_array_2*, *sphere_radius*)

Calculate the haversine-based distance between two points on the surface of a sphere. Should be more accurate than the arc cosine strategy. See, for example: http://en.wikipedia.org/wiki/Haversine_formula

pgamit.pyVoronoi.**calculate_surface_area_of_a_spherical_Voronoi_polygon**(*array_ordered_Voronoi_polygon_vertices*, *sphere_radius*)

Calculate the surface area of a polygon on the surface of a sphere. Based on equation provided here: http://mathworld.wolfram.com/LHuiliersTheorem.html Decompose into triangles, calculate excess for each

pgamit.pyVoronoi.**calculate_surface_area_of_planar_polygon_in_3D_space**(*array_ordered_Voronoi_polygon_vertices*)

Based largely on: http://stackoverflow.com/a/12653810 Use this function when spherical polygon surface area calculation fails (i.e., lots of nearly-coplanar vertices and negative surface area).

pgamit.pyVoronoi.**convert_cartesian_array_to_spherical_array**(*coord_array*, *angle_measure='radians'*)

Take shape (N,3) cartesian coord_array and return an array of the same shape in spherical polar form (r, theta, phi). Based on StackOverflow response: http://stackoverflow.com/a/4116899 use radians for the angles by default, degrees if angle_measure == 'degrees'

pgamit.pyVoronoi.**convert_spherical_array_to_cartesian_array**(*spherical_coord_array*, *angle_measure='radians'*)

Take shape (N,3) spherical_coord_array (r,theta,phi) and return an array of the same shape in cartesian coordinate form (x,y,z). Based on the equations provided at: http://en.wikipedia.org/wiki/List_of_common_coordinate_transformations#From_spherical_coordinates use radians for the angles by default, degrees if angle_measure == 'degrees'

pgamit.pyVoronoi.**filter_polygon_vertex_coordinates_for_extreme_proximity**(*array_ordered_Voronoi_polygon_vertic*, *sphere_radius*)

Merge (take the midpoint of) polygon vertices that are judged to be extremely close together and return the filtered polygon vertex array. The purpose is to alleviate numerical complications that may arise during surface area calculations involving polygons with ultra-close / nearly coplanar vertices.

pgamit.pyVoronoi.**filter_tetrahedron_to_triangle**(*current_tetrahedron_coord_array*)

pgamit.pyVoronoi.**produce_array_Voronoi_vertices_on_sphere_surface**(*facet_coordinate_array_Delaunay_triangulation*, *sphere_radius*, *sphere_centroid*)

Return shape (N,3) array of coordinates for the vertices of the Voronoi diagram on the sphere surface given a shape (N,3,3) array of Delaunay triangulation vertices.

pgamit.pyVoronoi.**produce_triangle_vertex_coordinate_array_Delaunay_sphere**(*hull_instance*)

Return shape (N,3,3) numpy array of the Delaunay triangle vertex coordinates on the surface of the sphere.

pgamit.pyVoronoi.**test_polygon_for_self_intersection**(*array_ordered_Voronoi_polygon_vertices_2D*)

Test an allegedly properly-ordered numpy array of Voronoi region vertices in 2D for self-intersection of edges based on algorithm described at http://algs4.cs.princeton.edu/91primitives/

pgamit.pyZTD module

Project: Parallel.GAMIT Date: 6/18/20 14:28 Author: Demian D. Gomez

class pgamit.pyZTD.**Ztd**(*cnn*, *NetworkCode*, *StationCode*, *project*, *plotit=False*)

Bases: object

adjust_lsq(*A*, *L*)

static autoscale_y(*ax*, *margin=0.1*)

This function rescales the y-axis based on the data that is visible given the current xlim of the axis. ax – a matplotlib axes object margin – the fraction of the total height of the y-data to pad the upper and lower ylims

plot(*pngfile=None*, *t_win=None*, *residuals=False*, *plot_missing=True*, *ecef=False*, *plot_outliers=True*, *fileio=None*)

set_lims(*t_win*, *plt*, *ax*)

class pgamit.pyZTD.**ZtdSoln**(*cnn*, *NetworkCode*, *StationCode*, *project*)

Bases: object

pgamit.snxParse module

class pgamit.snxParse.**StationData**(*x=None*, *y=None*, *z=None*)

Bases: object

Print()

pgamit.snxParse.**main**()

class pgamit.snxParse.**mergedSinexStationData**(*snxStnData=None*)

Bases: object

Print()

class pgamit.snxParse.**snxFileParser**(*snxFilePath=None*)

Bases: object

Print(*key=None*, *fid=None*)

contains(*key*)

get(*key*)

parse()

size()

class pgamit.snxParse.**snxStationMerger**

Bases: object

Print()

addStation(*level*, *orgName*, *stationName*, *snxStnData*)

addStationsFromSinexObject(*snxObj*)

compareUsingCoordinates(*snxStationDataA*, *snxStationDataB*)

maxLevel()

stationExistsWithNumberOfOccurrences(*stationName*, *numberOfOccurences*)

Module contents

PYTHON MODULE INDEX

c

p

INDEX

D

E

F

G

N

O

P

Q

R

S

T

U

V

W

Y

Z