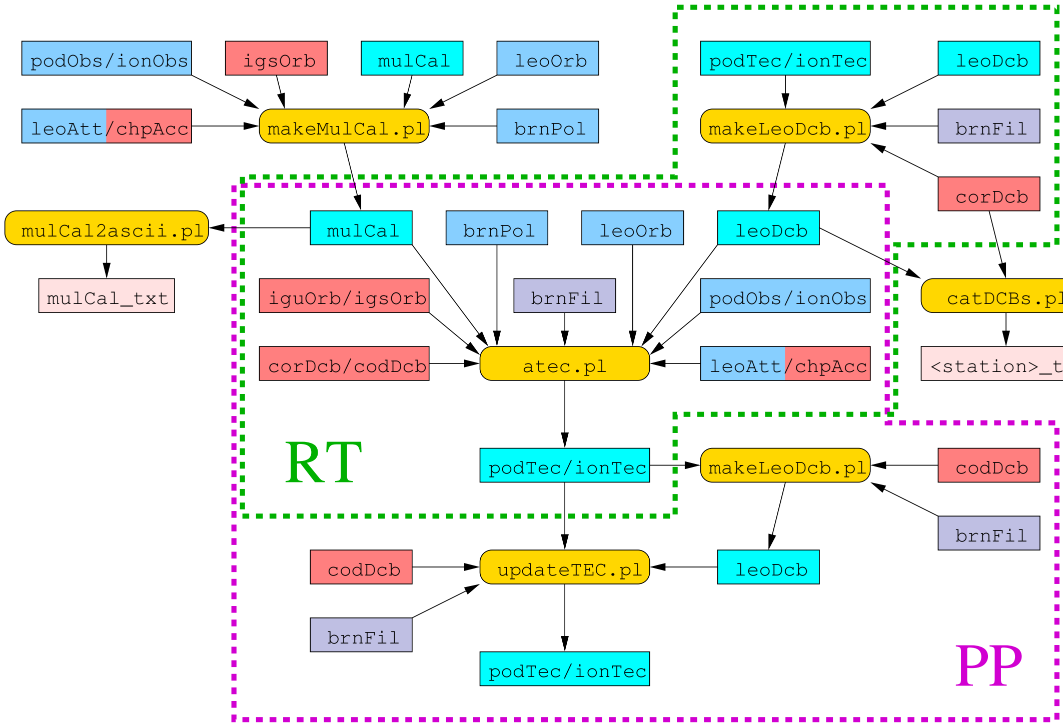
# Algorithms for processing GPS data into absolute Total Electron Content (aTEC) along LEO–GPS links

## Overview

This document describes the programs, scripts, and algorithms for processing GPS data collected in low earth orbit (LEO) into absolute total electron content (TEC) along LEO–GPS links. This includes estimation of the LEO differential code biases (DCBs). The document also describes the computation of pseudo-range local multipath patterns, which are used to calibrate the pseudo-range data for the TEC estimation. The pseudo-range local multipath calibration helps the TEC estimation partly through improved cycle-slip detection and correction, and partly through improved phase/pseudo-range leveling. Within CDAAC, the module has been named `atec` and consists of several Perl scripts and Fortran programs/subroutines. The Perl scripts can be considered the main drivers, matching up different file types, performing minor computational tasks, and writing output to NetCDF files. The Fortran programs and related subroutines are called from Perl scripts when more heavy-duty computations are necessary.

Below is an overview of the `atec` module architecture for both real-time (RT) processing and post-processing (PP).



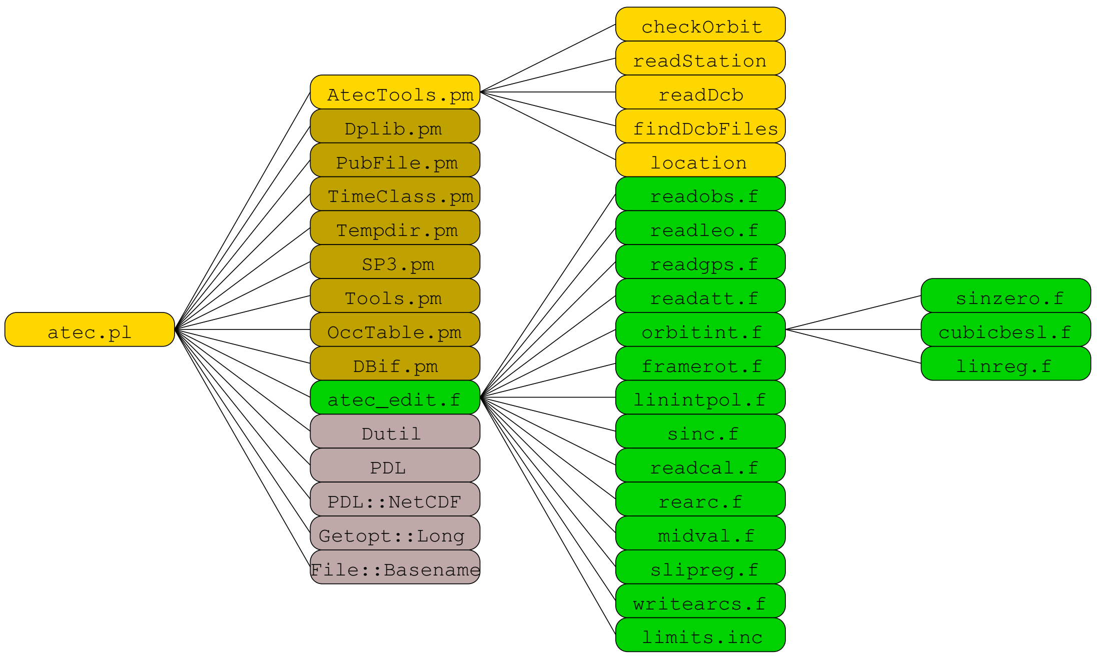
The yellow rounded boxes are Perl scripts within the `atec` module. The rectangular boxes denotes different file types. The cyan and pink boxes are file types generated by the `atec` module, but the pink file types are not generated/used in the CDAAC production. The blue/violet boxes are other file types/directory generated within CDAAC and used by the `atec` module. The red boxes are file types generated outside CDAAC (fetched by CDAAC from various sources around the world) and used by the `atec` module.

The Perl script named `atec.pl` can be considered the core of the `atec` module. It takes care of the processing of raw GPS data into TEC arcs. The script named `makeMulCal.pl` is not used in daily operations, but is used for calculation of pseudo-range multipath patterns. These patterns are used in the daily operations. Once in a while (usually months in between) the patterns can be updated by running `makeMulCal.pl`. The estimation of the daily averaged LEO DCBs are done by the script named `makeLeoDcb.pl`. In the real-time processing, this script is kicked off automatically once a day. In post-processing, the `makeLeoDcb.pl` script is run after running `atec.pl`. The output files of `atec.pl`, either `podTec` or `ionTec` files, are then updated by subsequently running the script named `updateTEC.pl`. Two other scripts, `catDCBs.pl` and `mulCal2ascii.pl` are not essential scripts, and they are not used in the daily processing. They can be used to convert `leoDcb` files and `mulCal` files, respectively, into columnar ASCII files which are easy to plot using the interactive plotting program named gnuplot.

Below, each of these scripts and related Fortran codes are described in detail.

## 1 atec.pl

### 1.1 Dependency chart:



The yellow boxes indicate Perl script/module and subroutines within the `atec` module.

The green boxes indicate Fortran code and subroutines within the `atec` module. The brown boxes indicate other Perl modules, not within the `atec` module. Some of these may have underlying dependencies not included in the diagram. The `AtecTools.pm` module also depends on some of the brown modules. The gray boxes indicate open-source code, not within the `atec` module. Some of these may have underlying dependencies not included in the diagram.

## 1.2 Short description:

The `atec.pl` script generates absolute Total Electron Content (`podTec` or `ionTec`) files for a whole dump/day given a RINEX 2.20 (`podObs` or `ionObs`) observation file. It matches up the RINEX observations with SP3 orbit files (`leoOrb`/`leoPor` and `igsOrb`/`iguOrb`) and attitude files (`leoAtt` or `chpAcc`). It also uses files from the previous dump/day to ensure continuity across the dump/day boundary. The detection of outliers and cycle-slip correction is handed over to a Fortran code `atec_edit.f` executed from the `atec.pl` script. The output is a bunch of NetCDF files containing continuous arcs of TEC, elevation viewing angle, SNRs, and satellite positions in ECF Cartesian coordinates.

## 1.3 Usage and command-line options:

```
atec.pl podObs_YYYY.DDD.LLL.NN.TT_rnx [--(no)att] [--(no)pod] [--(no)flip] [--(no)db]
      [--tmpdir=dirpath] [--parmsfile=filename] [--calfile=filename]

atec.pl ionObs_YYYY.DDD.LLL.NN.TT_rnx [--(no)att] [--(no)pod] [--(no)flip] [--(no)db]
      [--tmpdir=dirpath] [--parmsfile=filename] [--calfile=filename]
```

| | |
|---|---|
| `--att` | specifies that attitude data should be used (default is `--att`) |
| `--pod` | specifies that precise LEO orbits should be used or … |
| `--nopod` | to use predicted LEO orbits in real-time processing (default is `--nopod` for real-time and `--pod` for post processing) |
| `--flip` | specifies to turn the definition of the s/c frame 180 deg when the `--noatt` option is set (default is `--noflip`) |
| `--db` | specifies that the database should be populated (default is `--db`) |
| `--tmpdir=dirpath` | specifies the full path where to create (possibly overwrite) a "temporary" directory named `localtmp` which will not be deleted after execution. Output NetCDF files are put in `dirpath/MISSION/...` (default is to create a temporary directory in `/tmp` that will be deleted automatically after execution, and to put NetCDF output files in `/PUB/MISSION/...`) |
| `--parmsfile=filename` | gives the name of the parameter filename to be passed on to the Fortran program (default is `--parmsfile=edit.parms2`) |
| `--calfile=filename` | gives the name of the calibration (`mulCal`) filename to be passed on to the Fortran program (default is the most recently generated `mulCal` file) |

## 1.4 Input data:

- `podObs_YYYY.DDD.LLL.NN.TT_rnx` (low rate data)
  `ionObs_YYYY.DDD.LLL.NN.TT_rnx` (medium rate data)

  RINEX 2.20 level 1a file including raw GPS observations between LEO and GPS satellites. The `atec.pl` script reads:

  - Filename information (via `PubFile.pm`; cf. section 1.1)
    - File type (`podObs` or `ionObs`)
    - `YYYY`: year
    - `DDD`: day of year
    - `LLL`: LEO ID
    - `NN`: dump number
    - `TT`: antenna ID
  - RINEX header information
    - `ANTENNA: B.SIGHT XYZ`
    - `TIME OF FIRST OBS`
    - `TIME OF LAST OBS`

  The `readobs.f` code (cf. section 1.1) reads:

  - RINEX header information
    - `# / TYPES OF OBSERV`
    - `OBS SCALE FACTOR`
    - `TIME OF FIRST OBS`
  - Epoch and GPS satellites
  - `L1` carrier phase (based on P code)
  - `L2` carrier phase (based on P code)
  - `C1` pseudo-range (based on C/A code)
  - `P1` pseudo-range
  - `P2` pseudo-range
  - `LA` carrier phase (based on C/A code)
  - `SA` signal-to-noise ratio (based on C/A code)
  - `S2` signal-to-noise ratio (based on P code)

- `chpAcc_YYYY.DDD.HH.MM.UUUU_txt` (for CHAMP only; level 0 file)
  `leoAtt_YYYY.DDD.LLL.NN_txt`

  ASCII level 1a file including LEO attitude information and navigation solution. The `Tools.pm` module (cf. section 1.1) reads:

  - `tim`: Epoch (GPS time)
  - `att`: Quaternion in inertial (ITOE) coordinate system
  - `sad`: Solar array drive angle (deg)
  - `pve`: Navigation solution in ECF coordinates (km)

- `iguOrb_YYYY.DDD.HH.MM.UUUU_sp3` (for real-time processing)
  `igsOrb_YYYY.DDD.HH.MM.UUUU_sp3` (for post-processing)
  `codOrb_YYYY.DDD.HH.MM.UUUU_sp3` (alternative for post-processing)

  SP3 level 1a file including GPS orbit information. The `atec.pl` script reads:

  - SP3 header information
    - First epoch (GPS time)
    - Number of epochs
    - Interval between epochs (s)

  The `readgps.f` code (cf. section 1.1) reads:

  - SP3 header information
    - First epoch (GPS time)
  - Epoch (GPS time)
  - GPS position in ECF coordinates for each PRN (km)

- `leoOrb_YYYY.DDD.LLL.NN_SSSS.VVVV_sp3` (precise orbits)
  `leoPor_YYYY.DDD.LLL.NN.PP_SSSS.VVVV_sp3` (predicted orbits)

  SP3 level 1b file including LEO orbit information. The `atec.pl` script reads:

  - Filename information (via `PubFile.pm`; cf. section 1.1)
    - `PP`: age of prediction in days
  - SP3 header information
    - First epoch (GPS time)
    - Number of epochs
    - Interval between epochs (s)

  The `SP3.pm` module (cf. section 1.1) reads:

  - Epoch (GPS time)
  - LEO position in ECF coordinates (km)
  - LEO velocity in ECF coordinates (dm/s)

  The `readleo.f` code (cf. section 1.1) reads:

  - SP3 header information
    - First epoch (GPS time)
  - Epoch (GPS time)
  - LEO position in ECF coordinates (km)

- `brnPol_YYYY.DDD.HH.MM.UUUU_SSSS.VVVV_erp`

  ASCII level 1b file including information about Earth's nutation. The `Dutil` code (cf. section 1.1) reads:

  - Epoch (GPS time)
  - Parameters for conversion between ECF and ITOE coordinate systems

- `brnFil/MISSION.NN.STA`

  ASCII 'config' file including information about Bernese satellite and antenna names. The `AtecTools.pm` module (`readStation`; cf. section 1.1) reads:

  - CDAAC LEO name
  - Bernese LEO ID and antenna ID

- `leoDcb_YYYY.DDD_SSSS.VVVV_txt`

  ASCII level 1b file including LEO DCB values for all antennas. The `atec.pl` script reads:

  - Filename information (via `PubFile.pm`; cf. section 1.1)
    - `YYYY`: year
    - `DDD`: day of year

  The `AtecTools.pm` module (`readDcb`; cf. section 1.1) reads:

  - Bernese LEO ID and antenna ID
  - LEO DCB value (ns)
  - RMS of LEO DCB value (ns)

- `corDcb_YYYY.DDD_txt` (for real-time processing)
  `codDcb_YYYY.DDD_txt` (for post-processing)

  ASCII level 0 file including GPS DCB values for all PRNs. The `atec.pl` script reads:

  - Filename information (via `PubFile.pm`; cf. section 1.1)
    - `YYYY`: year
    - `DDD`: day of year

  The `AtecTools.pm` module (`readDcb`; cf. section 1.1) reads::

  - PRN
  - GPS DCB value (ns)
  - RMS of GPS DCB value (ns)

- `mulCal_YYYY.DDD.LLL.TT_nc`

  NetCDF 'config' file including 3D data for pseudo-range local multipath calibration. The `readcal.f` code (cf. section 1.1) reads:

  - `X_grid`: Horizontal component of off boresight angle (deg)
  - `Y_grid`: Vertical component of off boresight angle (deg)
  - `Z_grid`: Solar array drive angle (deg)
  - `MC1`: Mean local multipath on C/A pseudo-range (m)
  - `MP1`: Mean local multipath on P1 pseudo-range (m)
  - `MP2`: Mean local multipath on P2 pseudo-range (m)

- `edit.parms#`

  Fortran namelist file including tunable parameters for processing the data. The `atec_edit.f` code (cf. section 1.1) reads:

  - `&edit` tunable parameters
    - `dtleo`: Expected LEO orbit sampling in SP3 files (default is 60 s)
    - `dtgps`: Expected GPS orbit sampling in SP3 files (default is 900 s)
    - `dtatt`: Maximum threshold for attitude sampling (default is 300 s)
    - `snrmin`: Minimum SNR threshold (default is 5 volts/volt)
    - `snrmax`: Maximum SNR threshold (default is 10000 volts/volt)
    - `altmin`: Minimum altitude threshold (default is 50 km)
    - `gapmax`: Maximum acceptable gap in arcs (default is 120 s)
    - `minarclen`: Minimum acceptable length of arcs (default is 300 data points)
    - `minsublen`: Minumum acceptable length of subarcs (default is 30 data points)
    - `rms0`: Initial assumed RMS of 'wide-lane' noise (default is 0.05 cycles)
    - `rmsfac`: Threshold of the number of standard deviations of the 'wide-lane' noise (default is 8)
    - `rmsmax`: Maximum threshold of the 'wide-lane' RMS (default is 0.1 cycles)
    - `errmax`: Maximum threshold of the 'wide-lane' mean error of subarcs (default is 0.5 cycles)
    - `slipmax`: Maximum threshold of the number of cycles for cycle-slip detection in the 'Q-lane' (default is 1.5 cycles)
    - `slipmin`: Minimum threshold of the number of cycles in the 'Q-lane' between two consecutive data points at the beginning of an arc (default is 0.1 cycles)
    - `intslip`: Indication of whether cycle-slip correction should be to the nearest integer (default is .false., i.e., float correction)
    - `wreg0`: Basic regularization parameter for smoothing arcs in the cycle-slip correction algorithm (default is $10^8$)
    - `sigma0`: Assumed standard deviation of L1-L2 thermal noise corresponding to an L2 SNR of 100 volts/volt (default is 0.02 cycles)
    - `mfit`: Number of points to be fitted across a cycle-slip in the cycle-slip detection algorithm (default is 20 data points)
    - `jderiv`: The derivative order of the regularization (default is 2)
    - `postfac`: Threshold of the number of standard deviations of the post-fit residual P2-P1 noise during cycle-slip correction (default is 150)
    - `sigmax`: Maximum threshold for the post-fit RMS error (default is 1 TECU)
    - `calphase`: The L1 phase observable used ('L1' or 'LA') (default is 'LA')
    - `calcode`: The P1 pseudo-range observable used in the 'wide-lane' ('C1' or 'P1') (default is 'C1')
  - `&output` tunable parameters
    - `mulcal`: Write out multipath calibration data file (default is .false.)
    - `abstec`: Write out TEC arc files (default is true.)
    - `rawobs`: Write read-in RINEX observations to standard output (default is .false.),
    - `orbits`: Write read-in and interpolated orbits to standard output (default is .false.)
    - `wqlane`: Write the 'wide-lane' and the 'Q-lane' to standard output for all arcs (default is .false.)
    - `crvfit`: Write the ionospheric combinations and the fitted curves across cycle-slips to standard output (default is .false.)
    - `cslips`: Write the location and number of cycles for each cycle-slip to standard output (default is .false.)
    - `tecsmp`: Write all the TEC arcs, pseudo-range local multipath, and off-boresight angles to standard output (default is .false.)

- `podTec_IIII.YYYY.DDD.HH.MM.UUUU.GGG.TT_SSSS.VVVV_nc` (low rate data)
  `ionTec_IIII.YYYY.DDD.HH.MM.UUUU.GGG.TT_SSSS.VVVV_nc` (medium rate data)

  NetCDF level 1b file including total electron content (TEC) data between LEO and GPS satellites. The `atec.pl` script reads:

  - Filename information (via `PubFile.pm`; cf. section 1.1)
    - `YYYY`: year
    - `DDD`: day of year
    - `HH`: hour
    - `MM`: minute
    - `UUUU`: duration in minutes
  - Global attributes (via `PDL::NetCDF`; cf. section 1.1)
    - `start_time`: Starttime of data arc in GPS sec
    - `stop_time`: Stoptime of data arc in GPS sec

## 1.5 Output data:

- `podTec_IIII.YYYY.DDD.HH.MM.UUUU.GGG.TT_SSSS.VVVV_nc` (low rate data)
  `ionTec_IIII.YYYY.DDD.HH.MM.UUUU.GGG.TT_SSSS.VVVV_nc` (medium rate data)

  NetCDF level 1b file including total electron content (TEC) data between LEO and GPS satellites. The `atec.pl` script (via `PDL::NetCDF`; cf. section 1.1) writes:

  - `time`: Time of GPS measurement (GPS seconds)
  - `TEC`: Total Electron Content along LEO-GPS link (TECU)
  - `elevation`: Elevation angle of LEO-GPS link (deg)
  - `caL1_SNR`: Signal to Noise Ratio on the L1 channel, C/A code (volts/volt)
  - `pL2_SNR`: Signal to Noise Ratio on the L2 channel, P code (volts/volt)
  - `x_LEO`: LEO x position (ECF) at time of signal reception (km)
  - `y_LEO`: LEO y position (ECF) at time of signal reception (km)
  - `z_LEO`: LEO z position (ECF) at time of signal reception (km)
  - `x_GPS`: GPS x position (ECF) at time of signal transmission (km)
  - `y_GPS`: GPS y position (ECF) at time of signal transmission (km)
  - `z_GPS`: GPS z position (ECF) at time of signal transmission (km)
  - Global attributes
    - `processing_center`: Processing center (e.g., UCAR/CDAAC)
    - `creation_time`: Time stamp indicating when file was created
    - `mission`: CDAAC proto mission (e.g., COSMIC)
    - `dump_id`: ID of the dump where the arc ends
    - `leo_id`: Id of the LEO satellite
    - `antenna_id`: Id of the LEO antenna
    - `prn_id`: Id of the GPS satellite
    - `start_time`: Starttime of data arc in GPS sec
    - `stop_time`: Stoptime of data arc in GPS sec
    - `year`: Year of arc start (GPS time)
    - `month`: Month of year of arc start (GPS time)

- `day`: Day of month of arc start (GPS time)
- `hour`: Hour of day of arc start (GPS time)
- `minute`: Minute of hour of arc start (GPS time)
- `second`: Second of day of arc start (GPS time)
- `duration`: Duration of arc in seconds
- `attflag`: Flag indicating if attitude data were used in processing
- `podflag`: Flag indicating if precise or predicted LEO orbits were used
- `predorb_age`: Age in days of predicted orbit (-999 if precise orbits)
- `predorb_rms`: RMS of predicted orbit (km; -999 if precise orbits)
- `parmsfile`: Name of input parameter file
- `calfile`: Name of multipath calibration file
- `dcb_units`: Indicating the unit of the following DCB values
- `leodcb_flag`: Flag indicating if arc is LEO DCB calibrated
- `leodcb_age`: Age in days of LEO DCB value (-999 if not calibrated)
- `leodcb`: LEO DCB value (TECU; -999 if not calibrated)
- `leodcb_rms`: RMS of LEO DCB value (TECU; -999 if not calibrated)
- `gpsdcb_flag`: Flag indicating if arc is GPS DCB calibrated
- `gpsdcb_age`: Age in days of GPS DCB value (-999 if not calibrated)
- `gpsdcb`: GPS DCB value (TECU; -999 if not calibrated)
- `gpsdcb_rms`: RMS of GPS DCB value (TECU; -999 if not calibrated)
- `leveling_err`: Standard error of phase-to-code leveling (TECU)
- `tecmin`: Minimum TEC in arc (TECU)
- `tecmax`: Maximum TEC in arc (TECU)
- `elevmin`: Minimum elevation angle (TECU)
- `elevmax`: Maximum elevation angle (TECU)
- `tecsinmax`: TEC times the sine of elevation at maximum elevation (TECU)
- `elev_tecmax`: Elevation at maximum TEC (deg)
- `alt_elevmax`: Altitude of LEO at maximum elevation (km)
- `lat_elevmax`: Latitude of LEO at maximum elevation (deg)
- `lon_elevmax`: Longitude of LEO at maximum elevation (deg)
- `lct_elevmax`: Local time of LEO at maximum elevation (hr)
- `alt_tecmax`: Altitude of LEO at maximum TEC (km)
- `lat_tecmax`: Latitude of LEO at maximum TEC (deg)
- `lon_tecmax`: Longitude of LEO at maximum TEC (deg)
- `lct_tecmax`: Local time of LEO at maximum TEC (hr)
- `alt_start`: Altitude of LEO at starttime (km)
- `lat_start`: Latitude of LEO at starttime (deg)
- `lon_start`: Longitude of LEO at starttime (deg)
- `lct_start`: Local time of LEO at starttime (hr)
- `alt_stop`: Altitude of LEO at stoptime (km)
- `lat_stop`: Latitude of LEO at stoptime (deg)
- `lon_stop`: Longitude of LEO at stoptime (deg)
- `lct_stop`: Local time of LEO at stoptime (hr)

## 1.6 Processing steps for `atec.pl`:

1. Initial steps
2. Calculation of the spacecraft-to-boresight quaternion
3. Determining which RINEX files to use from previous dumps

4. Identifying corresponding orbit and attitude files
5. Checking predicted orbits against navigation solution
6. Calculation of the ECF-to-boresight quaternions
7. Setting up input for `atec_edit.f`
8. Running `atec_edit`
9. Calibrating data for the LEO and GPS DCBs
10. Checking for arc overlap
11. Calculating the extrema of the TEC and other useful output parameters
12. Writing data and updating the data base

### 1.6.1 Initial steps

The first steps of `atec.pl` are to determine, from system environment variables, the mission ID and whether the script is running in real-time or in post-processing mode.

The command-line options are obtained and it is determined whether some combinations may not be possible. The program is imediately terminated if one of the following combinations were chosen:

- Trying to use attitude data for either GPS/MET or SAC-C - attitude data are not available for these missions
- Trying to use attitude data for CHAMP in near real-time processing - attitude data are not available for CHAMP in near real-time
- Trying to use predicted orbits in post-processing mode - predicted orbits are not available for post processing

After this check, the `mulCal` file for the given LEO and antenna ID is indentified and checked for existence, the `edit.parms#` file is identified and checked for existence, and the subtype (`SSSS`) and version (`VVVV`) numbers to be attached to the output filenames are determined.

A temporary directory is created to serve as the platform for file exchange between `atec.pl` and `atec_edit.f`.

The start time, end time, and antenna boresight vector are read from the RINEX file. The program is terminated if the boresight vector is missing.

**Subroutines:**

`Dplib::protoMission` – Provides the proto mission name given a CDAAC mission.

`GetOptions` – Reads command-line options.

`PubFile->new` – Creates a new `PubFile` object. Used to handle CDAAC filename structures.

`PubFile->parse` – Parses a CDAAC filename and returns a `PubFile` object.

`Dplib::simplefn` – Provides the filename for a given CDAAC path name.

`GetVersion::GetVersion` – Determines the subtype and version of output files.

`Tempdir->new` – Creates a temporary directory and returns the directory name.

### 1.6.2 Calculation of the spacecraft-to-boresight quaternion

The spacecraft (S/C) to boresight (B/S) quaternion is calculated given the antenna boresight vector in the spacecraft frame. This quaternion relates the orientation of the B/S frame to the S/C frame, and is used later on to determine off-boresight angles of LEO-GPS links – which in turn is necessary for monitoring and calibration of pseudo-range local multipath. The B/S frame, given by unit vectors ($\mathbf{x}_{\mathrm{bs}}$, $\mathbf{y}_{\mathrm{bs}}$, $\mathbf{z}_{\mathrm{bs}}$) with coordinates in the S/C frame, is defined for a given antenna as follows:

$\mathbf{x}_{\mathrm{bs}}$ is a unit vector in the direction of the boresight vector.

$\mathbf{y}_{\mathrm{bs}}$ is a unit vector in the direction of the vector product between the nominal nadir of the S/C (which is $\mathbf{z}_{\mathrm{sc}}$ for COSMIC and CHAMP) and $\mathbf{x}_{\mathrm{bs}}$. If $\mathbf{x}_{\mathrm{bs}}$ is parallel to the nadir/zenith (e.g., CHAMP has a zenith antenna), then $\mathbf{y}_{\mathrm{bs}} \equiv -\mathbf{y}_{\mathrm{sc}}$.

$\mathbf{z}_{\mathrm{bs}}$ is a unit vector given by the vector product between $\mathbf{x}_{\mathrm{bs}}$ and $\mathbf{y}_{\mathrm{bs}}$.

The spacecraft-to-boresight quaternion, $q_{\mathrm{sc2bs}}$, can be interpreted as describing a so-called frame rotation from the S/C frame to the B/S frame. That is, a vector $\mathbf{v}$ in the S/C frame has coordinates $\mathbf{v}'$ in the B/S frame when applying the quaternion operator $\mathbf{v}' = q^*_{\mathrm{sc2bs}}\mathbf{v}q_{\mathrm{sc2bs}}$, where the asterix denotes the conjugate. This definition for a frame rotation is in accordance with the definition in (Kuipers, 2002).

The corresponding matrix, $\mathbf{R}_{\mathrm{sc2bs}}$, describing this rotation, i.e., $\mathbf{v}' = \mathbf{R}_{\mathrm{sc2bs}}\mathbf{v}$, is given by

$$\mathbf{R}_{\mathrm{sc2bs}} = \begin{pmatrix} \mathbf{x}_{\mathrm{bs}} & \mathbf{y}_{\mathrm{bs}} & \mathbf{z}_{\mathrm{bs}} \end{pmatrix}^{\mathrm{T}} = \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{pmatrix} . \tag{1}$$

The components ($q_w$, $q_x$, $q_y$, $q_z$) of $q_{\mathrm{sc2bs}} = q_{\mathrm{w}} + \mathbf{i}q_{\mathrm{x}} + \mathbf{j}q_{\mathrm{y}} + \mathbf{k}q_{\mathrm{z}}$ is then calculated as

$$q_{\mathrm{w}} = \frac{1}{2}\sqrt{1 + x_1 + y_2 + z_3} \ , \tag{2}$$

$$q_{\mathrm{x}} = \frac{1}{2}\mathrm{sign}(y_3 - z_2)\sqrt{1 + x_1 - y_2 - z_3} \ , \tag{3}$$

$$q_{\mathrm{y}} = \frac{1}{2}\mathrm{sign}(z_1 - x_3)\sqrt{1 - x_1 + y_2 - z_3} \ , \tag{4}$$

$$q_{\mathrm{z}} = \frac{1}{2}\mathrm{sign}(x_2 - y_1)\sqrt{1 - x_1 - y_2 + z_3} \ . \tag{5}$$

**Subroutines:**

`Dutil::mat2quat` – Converts a rotation matrix to a quaternion.

`Dutil::quatinv` – Gives the conjugate of a quaternion.

### 1.6.3 Determining which RINEX files to use from previous dumps

RINEX files from previous dumps containing data within 90 minutes before the start time of the input RINEX file is identified to make sure to include the beginning of data arcs that may cross the dump/day boundary. Up to four RINEX files are allowed. Having more than four should happen only if the combined duration of the three first ones is less than 90 minutes. That should not happen very often (if ever) with current CDAAC missions. The start time and end time of each RINEX file are read and stored for later use. The `atec.pl` script is terminated if any of the RINEX files are longer than 24 hours.

**Subroutines:**

`TimeClass->new` – Creates a new `TimeClass` object. Used to convert date and time to GPS seconds since January 6, 1980, and vice versa.

`Tools::findPrevDump` – Finds the dump ID of a previous dump given a current dump.

`PubFile->new` – Creates a new `PubFile` object. Used to handle CDAAC filename structures.

### 1.6.4 Identifying corresponding orbit and attitude files

Orbit and attitude files corresponding to the RINEX files are identified. In the case that predicted LEO orbits are to be used, the latest available predictions are chosen. Start and end times are read and stored for later use. If there are gaps of more than one minute between LEO orbit files, those before the gap(s) are discarded.

**Subroutines:**

`PubFile->new` – Creates a new `PubFile` object. Used to handle CDAAC filename structures.

`PubFile->parse` – Parses a CDAAC filename and returns a `PubFile` object.

`TimeClass->new` – Creates a new `TimeClass` object. Used to convert date and time to GPS seconds since January 6, 1980, and vice versa.

`OccTable::findGpsFiles` – Finds GPS orbit files covering the time span of a given dump.

### 1.6.5 Checking predicted orbits against navigation solution

The predicted LEO orbits (if chosen by the user instead of precise orbits) are checked against the raw navigation solution. This is done to avoid completely wrong orbits in case of orbit maneuvering which is not taken into account in the LEO orbit predictions. The 3D RMS of predicted positions relative to navigation solutions (where navigation solutions are available) are calculated based on simple linear interpolation. Outliers more than 3 times the RMS are discarded. LEO orbit prediction files with a resulting RMS larger than 5 km are discarded. The `atec.pl` script is terminated if the navigation solution is not available, or if all `leoPor` files fails the test.

**Subroutines:**

`Tools::readAtt` – Reads and returns data from a `leoAtt` or `chpAcc` file.

`AtecTools::checkOrbit` – Checks a LEO SP3 file against a set of valid navigation solutions (i.e., gaps are allowed but not -999 values).

`PubFile->parse` – Parses a CDAAC filename and returns a `PubFile` object.

`SP3::readSP3` – Reads an SP3 file and returns times, positions, and velocities.

### 1.6.6 Calculation of the ECF-to-boresight quaternions

The quaternions read from either `leoAtt` or `chpAcc` files are so-called ITOE-to-S/C quaternions. An ITOE-to-S/C quaternion, $q_{\mathrm{itoe2sc}}$, describes a frame rotation from the ITOE (Inertial True-of-Epoch) frame to the S/C (spacecraft) frame. That is, a vector $\mathbf{v}$ in the ITOE frame has coordinates $\mathbf{v}'$ in the S/C frame when applying the quaternion operator $\mathbf{v}' = q^*_{\mathrm{itoe2sc}}\mathbf{v}q_{\mathrm{itoe2sc}}$. Since the satellite orbits in the SP3 files are in Earth Centered Fixed (ECF) coordinates, and we are interested in the lines of sight of GPS satellites as seen from the B/S (boresight) frame, these quaternions are converted to ECF-to-B/S quaternions. With such quaternions, the coordinates of vectors along LEO-GPS links in the ECF frame can be easily transformed to coordinates in the B/S frame. Directions of lines of sight in the B/S frame are used for monitoring and calibration of local multipath.

For each attitude sampling time (disregarding missing or bad values) the ITOE-to-B/S quaternion, $q_{\mathrm{itoe2bs}}$, is obtained by a quaternion multiplication between the ITOE-to-S/C quaternion from the `leoAtt` or `chpAcc` file and the S/C-to-B/S quaternion found in section 1.6.2:

$$q_{\mathrm{itoe2bs}} = q_{\mathrm{itoe2sc}}q_{\mathrm{sc2bs}} \; . \tag{6}$$

The ITOE-to-B/S quaternion is then converted to an ECF-to-B/S quaternion by a coordinate transformation from the ITOE frame to the ECF frame. This coordinate transformation is not trivial and is performed by a Bernese routine using the most recent `brnPol` file containing information about Earth's nutation. The Bernese routine allows the calculation of the ITOE axes unit vectors ($\mathbf{x}_{\mathrm{itoe}}$, $\mathbf{y}_{\mathrm{itoe}}$, $\mathbf{z}_{\mathrm{itoe}}$) in ECF coordinates. The matrix, $\mathbf{R}_{\mathrm{ecf2itoe}}$, describing the rotation from ITOE coordinates to ECF coordinates is then given by

$$\mathbf{R}_{\mathrm{ecf2itoe}} = \begin{pmatrix} \mathbf{x}_{\mathrm{itoe}} & \mathbf{y}_{\mathrm{itoe}} & \mathbf{z}_{\mathrm{itoe}} \end{pmatrix}^{\mathrm{T}} = \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{pmatrix} \; . \tag{7}$$

The components ($q_w$, $q_x$, $q_y$, $q_z$) of the corresponding quaternion, $q_{\mathrm{ecf2itoe}} = q_{\mathrm{w}} + \mathbf{i}q_{\mathrm{x}} + \mathbf{j}q_{\mathrm{y}} + \mathbf{k}q_{\mathrm{z}}$, is then calculated as

$$q_{\mathrm{w}} = \frac{1}{2}\sqrt{1 + x_1 + y_2 + z_3} \; , \tag{8}$$

$$q_{\mathrm{x}} = \frac{1}{2}\mathrm{sign}(y_3 - z_2)\sqrt{1 + x_1 - y_2 - z_3} \; , \tag{9}$$

$$q_{\mathrm{y}} = \frac{1}{2}\mathrm{sign}(z_1 - x_3)\sqrt{1 - x_1 + y_2 - z_3} \; , \tag{10}$$

$$q_{\mathrm{z}} = \frac{1}{2}\mathrm{sign}(x_2 - y_1)\sqrt{1 - x_1 - y_2 + z_3} \; . \tag{11}$$

The ECF-to-B/S quaternion, $q_{\text{ecf2bs}}$ is now obtained by a quaternion multiplication between the ECF-to-ITOE quaternion and the ITOE-to-B/S quaternion:

$$q_{\text{ecf2bs}} = q_{\text{ecf2itoe}} q_{\text{itoe2bs}} \; . \tag{12}$$

If no attitude information is available, e.g., for CHAMP in real-time processing, ECF-to-B/S quaternions are estimated assuming that the spacecraft attitude does not vary. This assumption is approximately true for CHAMP most of the time. It allows a simple relation between the S/C frame and the so-called Local Level (LL) frame. For COSMIC (where we usually do have attitude data) the S/C frame may or may not be turned about 180 degrees with respect to the LL frame, so using the `--noatt` option for COSMIC may give spurious results. One can use the `--flip` option if it is known that the S/C frame has been turned for all days/dumps under consideration. For CHAMP (which has opposite definition than COSMIC of S/C frame with respect to LL frame) the `--flip` should always be set when the `--noatt` is set.

The LL frame, given by unit vectors ($\mathbf{x}_{\text{ll}}$, $\mathbf{y}_{\text{ll}}$, $\mathbf{z}_{\text{ll}}$) with coordinates in the ITOE frame, is defined as follows:

$\mathbf{z}_{\text{ll}}$ is a unit vector in the opposite direction of the satellite position vector.

$\mathbf{y}_{\text{ll}}$ is a unit vector in the direction of the vector product between the satellite position vector and its velocity vector (or opposite if the `--flip` options is set).

$\mathbf{x}_{\text{ll}}$ is a unit vector given by the vector product between $\mathbf{y}_{\text{ll}}$ and $\mathbf{z}_{\text{ll}}$.

Having the LL frame in the coordinates of the ITOE frame, the LL frame in the coordinates of the ECF frame could be obtained by a frame rotation from ITOE to ECF coordinates. However, a simpler approach, although an approximation, is to define the LL frame as above, but with reference to the ECF frame instead of the ITOE frame. This would not work for a geostationary satellite, where the velocity of the satellite in ECF coordinates is – naturally – zero, and $\mathbf{y}_{\text{ll}}$ would be erroneously undefined, but the error we make is small for a LEO satellite. Thus, in the case where attitude information is not available (or deliberately not used) we assume that

$$\mathbf{z}_{\text{sc}} = -\frac{\mathbf{r}_{\text{leo}}}{|\mathbf{r}_{\text{leo}}|} \; , \tag{13}$$

$$\mathbf{y}_{\text{sc}} = \frac{\mathbf{r}_{\text{leo}} \times \dot{\mathbf{r}}_{\text{leo}}}{|\mathbf{r}_{\text{leo}} \times \dot{\mathbf{r}}_{\text{leo}}|} \qquad \text{(or opposite if \texttt{--flip})} \; , \tag{14}$$

$$\mathbf{x}_{\text{sc}} = \mathbf{y}_{\text{sc}} \times \mathbf{z}_{\text{sc}} \; , \tag{15}$$

where the coordinates of the LEO position, $\mathbf{r}_{\text{leo}}$, and the velocity, $\dot{\mathbf{r}}_{\text{leo}}$, refer to the ECF frame.

The matrix, $\mathbf{R}_{\text{ecf2sc}}$, describing the rotation from ECF coordinates to S/C coordinates is given by

$$\mathbf{R}_{\text{ecf2sc}} = \begin{pmatrix} \mathbf{x}_{\text{sc}} & \mathbf{y}_{\text{sc}} & \mathbf{z}_{\text{sc}} \end{pmatrix}^{\text{T}} = \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{pmatrix} \; . \tag{16}$$

The components $(q_w, q_x, q_y, q_z)$ of $q_{\mathrm{ecf2sc}} = q_{\mathrm{w}} + \mathbf{i}q_{\mathrm{x}} + \mathbf{j}q_{\mathrm{y}} + \mathbf{k}q_{\mathrm{z}}$ is then calculated as

$$q_{\mathrm{w}} = \frac{1}{2}\sqrt{1 + x_1 + y_2 + z_3}\ , \tag{17}$$

$$q_{\mathrm{x}} = \frac{1}{2}\mathrm{sign}(y_3 - z_2)\sqrt{1 + x_1 - y_2 - z_3}\ , \tag{18}$$

$$q_{\mathrm{y}} = \frac{1}{2}\mathrm{sign}(z_1 - x_3)\sqrt{1 - x_1 + y_2 - z_3}\ , \tag{19}$$

$$q_{\mathrm{z}} = \frac{1}{2}\mathrm{sign}(x_2 - y_1)\sqrt{1 - x_1 - y_2 + z_3}\ . \tag{20}$$

Finally, the ECF-to-B/S quaternion, $q_{\mathrm{ecf2bs}}$ is obtained by a quaternion multiplication between the ECF-to-S/C quaternion and the S/C-to-B/S quaternion found in section 1.6.2:

$$q_{\mathrm{ecf2bs}} = q_{\mathrm{ecf2sc}}q_{\mathrm{sc2bs}}\ . \tag{21}$$

To summarize, when attitude information is available, the ECF-to-B/S quaternion is calculated via the steps leading to eq. (12). In cases whithout attitude information (i.e., when the `--noatt` option is set), the ECF-to-B/S quaternion is calculated via the steps leading to eq. (21).

**Subroutines:**

`SP3::readSP3` – Reads an SP3 file and returns times, positions, and velocities.

`Dutil::mat2quat` – Converts a rotation matrix to a quaternion.

`Dutil::quatinv` – Gives the conjugate of a quaternion.

`Dutil::quatmult` – Performs a quaternion multiplication.

`Tools::readAtt` – Reads and returns data from a `leoAtt` or `chpAcc` file.

`Tools::cachefid` – Finds `brnPol` files falling withing a given time span, and returns a data structure giving all files along with the start and stop times.

`Dutil::ITOEtoECF` – Makes an ITOE to ECF coordinate transformation. Returns the coordinates of a position vector (and optionally the velocity vector) in the ECF frame given the coordinates in the ITOE frame.

### 1.6.7 Setting up input for `atec_edit.f`

The ECF-to-B/S quaternions together with the solar array drive angle (which indicate the orientation of solar panels) are written to a temporary file named `leoAtt_quat`. Start and end times in GPS seconds for each file to be read by `atec_edit.f` is written to another temporary file named `infile_list`. An example of the content of `infile_list` is given here:

```
869788800.000 869875199.000 /pub/cosmic/level1a/podObs/2007.211/podObs_2007.211.004.01.01_rnx
869875200.000 869961599.000 /pub/cosmic/level1a/podObs/2007.212/podObs_2007.212.004.01.01_rnx
869788800.000 869886000.000 /pub/cosmic/level1b/leoOrb/leoOrb_2007.211.004.01_2007.3200_sp3
869875200.000 869961060.000 /pub/cosmic/level1b/leoOrb/leoOrb_2007.212.004.01_2007.3200_sp3
869788800.000 869874300.000 /pub/cosmic/level1a/igsOrb/igsOrb_2007.211.00.00.1440_sp3
869875200.000 869960700.000 /pub/cosmic/level1a/igsOrb/igsOrb_2007.212.00.00.1440_sp3
```

```
869961600.000 870047100.000 /pub/cosmic/level1a/igsOrb/igsOrb_2007.213.00.00.1440_sp3
869788804.004 869961585.004 /tmp/atec.pl12873/leoAtt_quat
```

### 1.6.8 Running `atec_edit`

The `atec_edit.f` program (usually invoked by either `atec.pl` or `makeMulCal.pl`) takes as input a list of files, among them RINEX `podObs` or `ionObs` files, and detects/removes cycle-slips and outliers in seperate data arcs based on the principles described by Blewitt (1990). A previously generated multipath calibration file is used to mitigate the effects of local multipath. In the end, the TEC based on the phase measurements are leveled to the pseudo-range (code) measurements. The program also estimates the MC1, MP1, and MP2 multipath to generate a new multipath calibration file. Parameters in the parmsfile determines whether the program generates data arcs with TEC or data for a new calibration file.

**Usage and command-line arguments:**

```
atec_edit tmpdir parmsfile calfile
```

**Input files:**

- The name of a temporary directory where the program will look for a file named `infile_list`. This file lists the start and end times and full path names for the following files to be read (cf. section 1.6.7)
  - Up to four level1a RINEX observation files (`podObs` or `ionObs`)
  - Up to four level1b LEO orbit files (`leoOrb` or `leoPor`)
  - Up to four level1a GPS orbit files (`igsOrb`, `codOrb`, or `iguOrb`)
  - Temporary file (`leoAtt_quat`) with quaternions and Solar Array Drive angle
- Parameters in an input parmsfile (`edit.parms#`)
- A previously generated multipath calibration file (`mulCal`)

**Output files:**

- Either a bunch of files (to the temporary directory) containing arcs of data with calibrated TEC, SNRs, and orbits
- Or five files (also to the temporary directory; one file for each of C1, P1, P2, S1, SA) with one dump/day worth of multipath calibration data and average SNRs

The output files are in ASCII format and read by `atec.pl` for further processing.

### 1.6.9 Calibrating data for the LEO and GPS DCBs

The most recent LEO DCB and its RMS for the given LEO ID and antenna ID are read from a previous `leoDcb` file. If the most recent LEO DCB is more than five days old, data are not calibrated, and the `leodcb_flag` is set to 0. Similarly, the GPS DCB and its RMS for each PRN are read from a previous `corDcb/codDcb` file. For a given PRN, if the most recent GPS DCB is more than five days old, data are not calibrated, and the `gpsdcb_flag` is set to 0.

Since the DCBs in the `leoDcb` and `corDcb/codDcb` files are in nano-seconds, a conversion to TECU is necessary. The conversion factor is given by $\frac{cf_1^2 f_2^2}{C_0(f_1^2 - f_2^2)}$, where $f_1$ and $f_2$ are the two GPS frequencies for the L1 and L2 signals, respectively, $c$ is the speed of light, and $C_0 = e^2/(8\pi^2 m\varepsilon_0) \approx 40.3082\,\mathrm{m}^3/\mathrm{s}^2$ is a constant where $\varepsilon_0$ is the permittivity of vacuum, $m$ is the electron mass, and $e$ is the elementary charge. That is:

$$1\,\mathrm{ns} = 2.853336681\,\mathrm{TECU}\ . \qquad (22)$$

The LEO DCB (if `leodcb_flag` = 1) and the GPS DCB (if `gpsdcb_flag` = 1) are then added to the uncalibrated TEC. For each arc of data, the `leodcb_age` and the `gpsdcb_age` indicates the age in days of the LEO DCB and the GPS DCB, respectively.

**Subroutines:**

`AtecTools::readStation` – Reads and returns the Bernese satellite and antenna name, given a CDAAC LEO ID and antenna ID.

`AtecTools::findDcbFiles` – Returns `leoDcb` or `corDcb/codDcb` files not more than a given number of days older than a given date.

`AtecTools::readDcb` – Reads DCBs and their associated RMS values as a function of `PRN / STATION NAME` (a 22 character string) from a `leoDcb` or `corDcb/codDcb` file.

`PubFile->parse` – Parses a CDAAC filename and returns a `PubFile` object.

`TimeClass->new` – Creates a new `TimeClass` object. Used to convert date and time to GPS seconds since January 6, 1980, and vice versa.

### 1.6.10 Checking for arc overlap

Each new data arc (filename) is checked for arc overlap with previously generated arcs (processed in a previous dump). In some cases overlap is allowed to avoid discontinuities between dumps/days, e.g., because of a different LEO or GPS DCB used for two consequtive dumps/days, or because the leveling of phases to pseudo-ranges could be different when the last part of an arc (in the new dump/day) is taken into account. The following rule applies: Filenames containing different data from the same arc (because the first piece of the arc has already been processed earlier) must have the same start time in the filename, and the latest arc must have a longer duration in the filename. This should automatically be the case under normal circumstances where there are no differences in the processing from one dump/day to another. However, when, e.g., a new parmsfile is taken into use, and successive dumps/days are processed with different parmsfiles, the rule of having the same start time in the filename (when there is overlap) could potentially be broken (the rule of the latest arc having a longer duration in the filename is ensured in `atec_edit.f`). A given arc with a given `start_time`, is checked against all previously generated files starting within 90 minutes before the `start_time` of the given arc. If arc overlap is found that does not comply with the rule, the newly generated arc is discarded. If a new arc is longer than 90 minutes duration it is discarded (this probably never happens in practice, but is necessary to ensure that such very long arcs are not overlooked when arcs from the next dump/day are checked).

The checking for arc overlap in this way is done in an effort to provide the best quality of data arcs that crosses the dump/day boundary, while at the same time ensuring that previously processed files are not overwritten, and that overlap (for arcs where the first piece has already been processed in a previous dump/day) is easy to identify from the filenames: In case of arc overlap, there should only be a difference in the file duration (the `UUUU` field), and the file with the longest duration should generally contain the more accurate data.

**Subroutines:**

`TimeClass->new` – Creates a new `TimeClass` object. Used to convert date and time to GPS seconds since January 6, 1980, and vice versa.

`PubFile->new` – Creates a new `PubFile` object. Used to handle CDAAC filename structures.

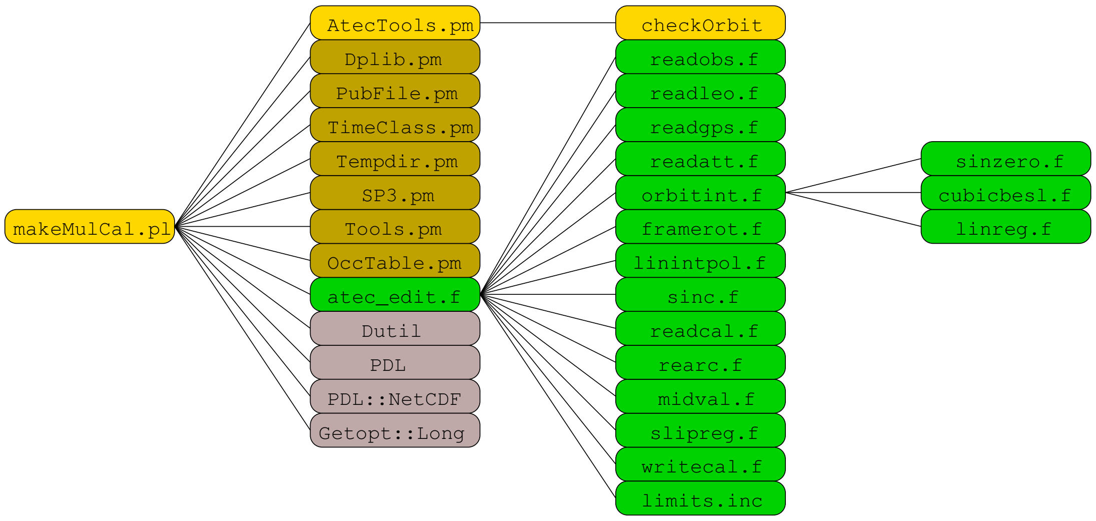
`Dplib::simplefn` – Provides the filename for a given CDAAC path name.

`PubFile->parse` – Parses a CDAAC filename and returns a `PubFile` object.

`PDL::NetCDF->new` – Opens a NetCDF file and returns an object that can be used for reading and writing.

## 2 makeMulCal.pl

### 2.1 Dependency chart:



The yellow boxes indicate Perl script/module and subroutines within the `atec` module. The green boxes indicate Fortran code and subroutines within the `atec` module. The brown boxes indicate other Perl modules, not within the `atec` module. Some of these may have underlying dependencies not included in the diagram. The `AtecTools.pm` module also depends on some of the brown modules. The grey boxes indicate open-source code, not within the `atec` module. Some of these may have underlying dependencies not included in the diagram.

## 2.2 Short description:

The `makeMulCal.pl` script creates a multipath calibration file based on RINEX (`podObs` or `ionObs`) observation files, SP3 (`leoOrb/leoPor` and `igsOrb/iguOrb`) orbit files, and attitude (`leoAtt` or `chpAcc`) files, given a timespan and mission on the command-line as well as a series of options. The output is a NetCDF file containing information to reduce multipath effects on C1, P1 and P2 for a given LEO and antenna ID. The detection of outliers and cycle-slip correction of the RINEX data is handed over to a Fortran code `atec_edit.f` executed from the `makeMulCal.pl` script.

## 2.3 Usage and command-line options:

```
makeMulCal.pl YYYY.DDD[-[YYYY.]DDD] MISSION [--prefix=filetype] [--leo=#] [--ant=#]
              [--(post|real)] [--(no)att] [--(no)pod] [--(no)flip] [--tmpdir=dirpath]
              [--parmsfile=filename] [--calfile=filename] [--datesfile=filename]
```

| | |
|---|---|
| `YYYY.DDD-YYYY.DDD` | is the time span used to generate the multipath calibration file |
| `MISSION` | is the name of the CDAAC mission |
| `--prefix=filetype` | specifies the filetype `podObs` or `ionObs`, but only one of them (default is `--prefix=podObs`) |
| `--leo=#` | specifies the leo ID (default is `--leo=1`) |
| `--ant=#` | specifies the antenna ID (default is `--ant=0`) |
| `--post` | specifies that post-processed files should be used or ... |
| `--real` | to use real-time generated files (default is `--post`) |
| `--att` | specifies that attitude data should be used (default is `--att`) |
| `--pod` | specifies that precise LEO orbits should be used or ... |
| `--nopod` | to use predicted LEO orbits, but only if `--real` is set (default is `--pod`) |
| `--flip` | specifies to turn the definition of the s/c frame 180 deg when the `--noatt` option is set (default is `--noflip`) |
| `--tmpdir=dirpath` | specifies the full path where to create (possibly overwrite) a “temporary” directory named localtmp which will not be deleted after execution. Output NetCDF file is put in `dirpath/MISSION/...` (default is to create a temporary directory in `/tmp` that will be deleted automatically after execution, and to put NetCDF output file in `/PUB/MISSION/...`) |
| `--parmsfile=filename` | gives the name of the parameter filename to be passed on to the Fortran program (default is `--parmsfile=edit.parms1`) |

`--calfile=filename` gives the name of the input calibration filename to be passed on to the Fortran program (default is `--calfile=mulCal_1980.006.001.00_nc`)

`--datesfile=filename` gives the name of a file containing dates to avoid for whatever reason (default is no file)

## 2.4 Input data:

- `podObs_YYYY.DDD.LLL.NN.TT_rnx` (low rate data)
  `ionObs_YYYY.DDD.LLL.NN.TT_rnx` (medium rate data)

  RINEX 2.20 level 1a file including raw GPS observations between LEO and GPS satellites. The `makeMulCal.pl` script reads:

  - Filename information (via `PubFile.pm`; cf. section 2.1)
    - `YYYY`: year
    - `DDD`: day of year
    - `LLL`: LEO ID
    - `NN`: dump number
  - RINEX header information
    - `ANTENNA: B.SIGHT XYZ`
    - `TIME OF FIRST OBS`
    - `TIME OF LAST OBS`

  The `readobs.f` code (cf. section 2.1) reads:

  - RINEX header information
    - `# / TYPES OF OBSERV`
    - `OBS SCALE FACTOR`
    - `TIME OF FIRST OBS`
  - Epoch and GPS satellites
  - `L1` carrier phase (based on P code)
  - `L2` carrier phase (based on P code)
  - `C1` pseudo-range (based on C/A code)
  - `P1` pseudo-range
  - `P2` pseudo-range
  - `LA` carrier phase (based on C/A code)
  - `SA` signal-to-noise ratio (based on C/A code)
  - `S2` signal-to-noise ratio (based on P code)

- `chpAcc_YYYY.DDD.HH.MM.UUUU_txt` (for CHAMP only; level 0 file)
  `leoAtt_YYYY.DDD.LLL.NN_txt`

  ASCII level 1a file including LEO attitude information and navigation solution. The `Tools.pm` module (cf. section 2.1) reads:

  - `tim`: Epoch (GPS time)
  - `att`: Quaternion in inertial (ITOE) coordinate system
  - `sad`: Solar array drive angle (deg)

  - pve: Navigation solution in ECF coordinates (km)

- iguOrb_YYYY.DDD.HH.MM.UUUU_sp3 (when --real is specified)
  igsOrb_YYYY.DDD.HH.MM.UUUU_sp3 (when --post is specified)
  codOrb_YYYY.DDD.HH.MM.UUUU_sp3 (when --post is specified)

  SP3 level 1a file including GPS orbit information. The makeMulCal.pl script reads:

  - Filename information (via PubFile.pm; cf. section 2.1)
    * YYYY: year
    * DDD: day of year
    * HH: hour
    * MM: minute
    * UUUU: duration in minutes
  - SP3 header information
    * First epoch (GPS time)
    * Number of epochs
    * Interval between epochs (s)

  The readgps.f code (cf. section 2.1) reads:

  - SP3 header information
    * First epoch (GPS time)
  - Epoch (GPS time)
  - GPS position in ECF coordinates for each PRN (km)

- leoOrb_YYYY.DDD.LLL.NN_SSSS.VVVV_sp3 (precise orbits)
  leoPor_YYYY.DDD.LLL.NN.PP_SSSS.VVVV_sp3 (predicted orbits)

  SP3 level 1b file including LEO orbit information. The makeMulCal.pl script reads:

  - Filename information (via PubFile.pm; cf. section 2.1)
    * PP: age of prediction in days
  - SP3 header information
    * First epoch (GPS time)
    * Number of epochs
    * Interval between epochs (s)

  The SP3.pm module (cf. section 2.1) reads:

  - Epoch (GPS time)
  - LEO position in ECF coordinates (km)
  - LEO velocity in ECF coordinates (dm/s)

  The readleo.f code (cf. section 2.1) reads:

  - SP3 header information
    * First epoch (GPS time)
  - Epoch (GPS time)

  - LEO position in ECF coordinates (km)

- `brnPol_YYYY.DDD.HH.MM.UUUU_SSSS.VVVV_erp`

  ASCII level 1b file including information about Earth's nutation. The `Dutil` code (cf. section 2.1) reads:

  - Epoch (GPS time)
  - Parameters for conversion between ECF and ITOE coordinate systems

- `mulCal_YYYY.DDD.LLL.TT_nc`

  NetCDF 'config' file including 3D data for pseudo-range local multipath calibration. The `readcal.f` code (cf. section 2.1) reads:

  - `X_grid`: Horizontal component of off boresight angle (deg)
  - `Y_grid`: Vertical component of off boresight angle (deg)
  - `Z_grid`: Solar array drive angle (deg)
  - `MC1`: Mean local multipath on C/A pseudo-range (m)
  - `MP1`: Mean local multipath on P1 pseudo-range (m)
  - `MP2`: Mean local multipath on P2 pseudo-range (m)

- `edit.parms#`

  Fortran namelist file including tunable parameters for processing the data. The `atec_edit.f` code (cf. section 2.1) reads:

  - `&edit` tunable parameters
    * `dtleo`: Expected LEO orbit sampling in SP3 files (default is 60 s)
    * `dtgps`: Expected GPS orbit sampling in SP3 files (default is 900 s)
    * `dtatt`: Maximum threshold for attitude sampling (default is 300 s)
    * `snrmin`: Minimum SNR threshold (default is 5 volts/volt)
    * `snrmax`: Maximum SNR threshold (default is 10000 volts/volt)
    * `altmin`: Minimum altitude threshold (default is 50 km)
    * `gapmax`: Maximum acceptable gap in arcs (default is 120 s)
    * `minarclen`: Minimum acceptable length of arcs (default is 300 data points)
    * `minsublen`: Minumum acceptable length of subarcs (default is 30 data points)
    * `rms0`: Initial assumed RMS of 'wide-lane' noise (default is 0.05 cycles)
    * `rmsfac`: Threshold of the number of standard deviations of the 'wide-lane' noise (default is 8)
    * `rmsmax`: Maximum threshold of the 'wide-lane' RMS (default is 0.1 cycles)
    * `errmax`: Maximum threshold of the 'wide-lane' mean error of subarcs (default is 0.5 cycles)
    * `slipmax`: Maximum threshold of the number of cycles for cycle-slip detection in the 'Q-lane' (default is 1.5 cycles)
    * `slipmin`: Minimum threshold of the number of cycles in the 'Q-lane' between two consecutive data points at the beginning of an arc (default is 0.1 cycles)
    * `intslip`: Indication of whether cycle-slip correction should be to the nearest integer (default is .false., i.e., float correction)
    * `wreg0`: Basic regularization parameter for smoothing arcs in the cycle-slip correction algorithm (default is $10^8$)
    * `sigma0`: Assumed standard deviation of L1-L2 thermal noise corresponding to an L2 SNR of 100 volts/volt (default is 0.02 cycles)

    * `mfit`: Number of points to be fitted across a cycle-slip in the cycle-slip detection algorithm (default is 20 data points)
    * `jderiv`: The derivative order of the regularization (default is 2)
    * `postfac`: Threshold of the number of standard deviations of the post-fit residual P2-P1 noise during cycle-slip correction (default is 150)
    * `sigmax`: Maximum threshold for the post-fit RMS error (default is 1 TECU)
    * `calphase`: The L1 phase observable used ('L1' or 'LA') (default is 'LA')
    * `calcode`: The P1 pseudo-range observable used in the 'wide-lane' ('C1' or 'P1') (default is 'C1')
  - `&output` tunable parameters
    * `mulcal`: Write out multipath calibration data file (default is .false.)
    * `abstec`: Write out TEC arc files (default is true.)
    * `rawobs`: Write read-in RINEX observations to standard output (default is .false.),
    * `orbits`: Write read-in and interpolated orbits to standard output (default is .false.)
    * `wqlane`: Write the 'wide-lane' and the 'Q-lane' to standard output for all arcs (default is .false.)
    * `crvfit`: Write the ionospheric combinations and the fitted curves across cycle-slips to standard output (default is .false.)
    * `cslips`: Write the location and number of cycles for each cycle-slip to standard output (default is .false.)
    * `tecsmp`: Write all the TEC arcs, pseudo-range local multipath, and off-boresight angles to standard output (default is .false.)
  - `&grid` tunable parameters
    * `grid0`: The 'lower left' corner of the 3D grid for the pseudo-range local multipath calibration file to be created (default is -100, -100, 0 (deg))
    * `lgrid`: The number of grid points in each dimension (default is 201, 201, 1)
    * `dgrid`: The grid resolution (default is 1, 1, 360 (deg))

## 2.5 Output data:

- `mulCal_YYYY.DDD.LLL.TT_nc`

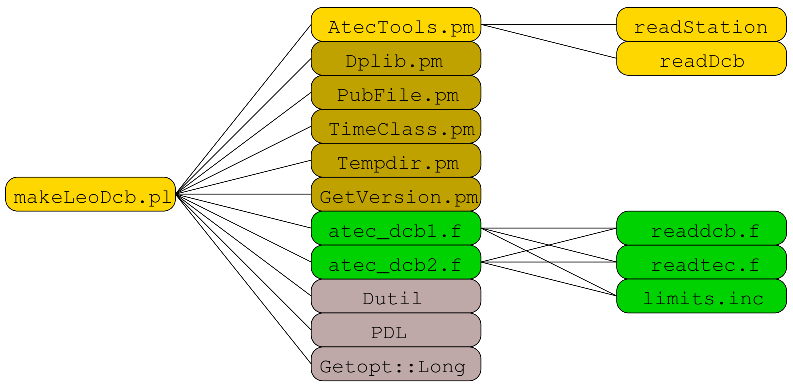
  NetCDF 'config' file including 3D data for pseudo-range local multipath calibration. The `makeMulCal.pl` script (via `PDL::NetCDF`; cf. section 2.1) writes:

  - `X_grid:` Horizontal component of off boresight angle (deg)
  - `Y_grid:` Vertical component of off boresight angle (deg)
  - `Z_grid:` Solar array drive angle (deg)
  - `MC1:` Mean local multipath on C/A pseudo-range (m)
  - `MP1:` Mean local multipath on P1 pseudo-range (m)
  - `MP2:` Mean local multipath on P2 pseudo-range (m)
  - `MSA:` Mean signal-to-noise ratio on L1 channel, C/A code (volts/volt)
  - `MS2:` Mean signal-to-noise ratio on L2 channel, P code (volts/volt)
  - Global attributes
    * `processing_center`: Processing center (e.g., UCAR/CDAAC)
    * `creation_time`: Time stamp indicating when file was created
    * `mission`: CDAAC proto mission (e.g., COSMIC)
    * `timespan`: Time-span used for generating this multipath calibration file
    * `leo_id`: Id of the LEO satellite

* **antenna_id**: Id of the LEO antenna
* **attflag**: Flag indicating if attitude data were used in processing
* **podflag**: Flag indicating if precise or predicted LEO orbits were used
* **parmsfile**: Name of input parameter file
* **calfile**: Name of input multipath calibration file

## 3 makeLeoDcb.pl

### 3.1 Dependency chart:



The yellow boxes indicate Perl script/module and subroutines within the **atec** module. The green boxes indicate Fortran code and subroutines within the **atec** module. The brown boxes indicate other Perl modules, not within the **atec** module. Some of these may have underlying dependencies not included in the diagram. The **AtecTools.pm** module also depends on some of the brown modules. The grey boxes indicate open-source code, not within the **atec** module. Some of these may have underlying dependencies not included in the diagram.

### 3.2 Short description:

In real-time processing the **makeLeoDcb.pl** script creates a daily file with LEO Differential Code Biases (DCBs) based on **podTec** and/or **ionTec** files generated for one day (say, doy 100). It is supposed to be kicked off every day by a **corDcb** file usually comming in about nine hours (UTC) into the next day (doy 101). The raw DCB estimation for each LEO and antenna ID is handed over to Fortran codes **atec_dcb1.f** and **atec_dcb2.f** (the latter only if there is more than one antenna on a satellite) executed from the **makeLeoDcb.pl** script. The final daily value (for doy 100) is based on this raw (noisy) estimate as well as the already generated DCBs from a number of previous days back (doy 99, 98, 97, etc...). Thus the final daily value is smoothed using a regularization method including previous DCBs. In post-processing the raw daily DCBs are estimated for a whole date range using **codDcb** files instead of **corDcb** files, and the regularization is performed over this range of days.

## 3.3 Usage and command-line options:

```
makeLeoDcb.pl corDcb_YYYY.DDD_txt [--order=#] [--degree=#] [--tmpdir=dirpath]
              [--parmsfile=filename] (for real-time processing)

makeLeoDcb.pl YYYY.DDD[-[YYYY.]DDD] MISSION [--order=#] [--degree=#] [--tmpdir=dirpath]
              [--parmsfile=filename] (for post-processing)
```

| | |
|---|---|
| `YYYY.DDD-YYYY.DDD` | is the time span in post-processing mode |
| `MISSION` | is the name of the CDAAC mission |
| `--order=#` | specifies the order of regularization, i.e, the order of the derivative to be minimized (default is `--order=0`, which means no regularization at all) |
| `--degree=#` | specifes the degree of smoothing, i.e, the regularization parameter = $10^{\#}$ (default is `--degree=0`, which for `order` > 0 corresponds to medium smoothing; choose `degree` < 0 for less smoothing and `degree` > 0 for more) |
| `--tmpdir=dirpath` | specifies the full path where to create (possibly overwrite) a "temporary" directory named `localtmp` which will not be deleted after execution. Output files are put in `dirpath/MISSION/...` (default is to create a temporary directory in `/tmp` that will be deleted automatically after execution, and to put output files in `/PUB/MISSION/...`) |
| `--parmsfile=filename` | gives the name of the parameter filename to be passed on to the Fortran program (default is `--parmsfile=dcb.parms1`) |

## 3.4 Input data:

- `podTec_IIII.YYYY.DDD.HH.MM.UUUU.GGG.TT_SSSS.VVVV_nc` (low rate data)
  `ionTec_IIII.YYYY.DDD.HH.MM.UUUU.GGG.TT_SSSS.VVVV_nc` (medium rate data)

  NetCDF level 1b file including total electron content (TEC) data between LEO and GPS satellites. The `makeLeoDcb.pl` script reads:

  - Filename information (via `PubFile.pm`; cf. section 3.1)
    * `IIII`: CDAAC LEO name
    * `HH`: hour
    * `MM`: minute
    * `GGG`: PRN number
    * `TT`: antenna ID

  The `readtec.f` code (cf. section 3.1) reads:

  - `time`: Time of GPS measurement (GPS seconds)
  - `TEC`: Total Electron Content along LEO-GPS link (TECU)
  - `x_LEO`: LEO x position (ECF) at time of signal reception (km)
  - `y_LEO`: LEO y position (ECF) at time of signal reception (km)
  - `z_LEO`: LEO z position (ECF) at time of signal reception (km)
  - `x_GPS`: GPS x position (ECF) at time of signal transmission (km)
  - `y_GPS`: GPS y position (ECF) at time of signal transmission (km)
  - `z_GPS`: GPS z position (ECF) at time of signal transmission (km)

- Global attributes
  - `prn_id`: Id of the GPS satellite
  - `leodcb_flag`: Flag indicating if arc is LEO DCB calibrated
  - `leodcb`: LEO DCB value (TECU; -999 if not calibrated)
  - `gpsdcb_flag`: Flag indicating if arc is GPS DCB calibrated
  - `gpsdcb`: GPS DCB value (TECU; -999 if not calibrated)

- `brnFil/MISSION.NN.STA`

  ASCII 'config' file including information about Bernese satellite and antenna names. The `AtecTools.pm` module (`readStation`; cf. section 3.1) reads:

  - CDAAC LEO name
  - Bernese LEO ID and antenna ID

- `leoDcb_YYYY.DDD_SSSS.VVVV_txt`

  ASCII level 1b file including LEO DCB values for all antennas. The `makeLeoDcb.pl` script reads:

  - Filename information (via `PubFile.pm`; cf. section 3.1)
    - `YYYY`: year
    - `DDD`: day of year

  The `AtecTools.pm` module (`readDcb`; cf. section 3.1) reads:

  - Bernese LEO ID and antenna ID
  - LEO DCB value (ns)
  - RMS of LEO DCB value (ns)

- `corDcb_YYYY.DDD_txt` (for real-time processing)
  `codDcb_YYYY.DDD_txt` (for post-processing)

  ASCII level 0 file including GPS DCB values for all PRNs. The `makeLeoDcb.pl` script reads:

  - Filename information (via `PubFile.pm`; cf. section 3.1)
    - `YYYY`: year
    - `DDD`: day of year
  - PRN

  The `readdcb.f` code (cf. section 3.1) reads:

  - PRN
  - GPS DCB value (ns)

### 3.5 Output data:

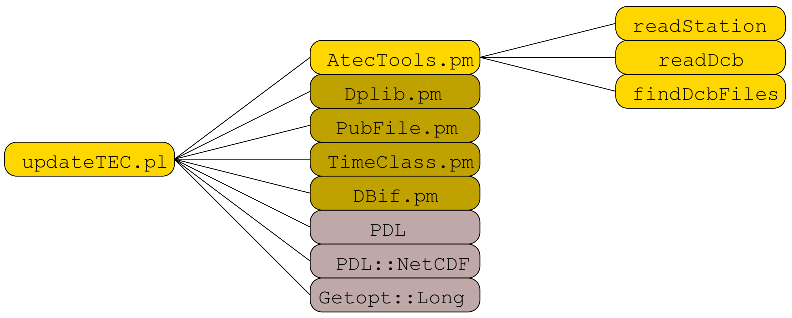
- `leoDcb_YYYY.DDD_SSSS.VVVV_txt`

  ASCII level 1b file including LEO DCB values for all antennas. The `makeLeoDcb.pl` script writes:

- Header information
  - Year and day of year
  - Time stamp indicating when file was created
- Bernese LEO ID and antenna ID
- LEO DCB value (ns)
- RMS of LEO DCB value (ns)

# 4 `updateTEC.pl`

## 4.1 Dependency chart:



The yellow boxes indicate Perl script/module and subroutines within the `atec` module. The brown boxes indicate other Perl modules, not within the `atec` module. Some of these may have underlying dependencies not included in the diagram. The `AtecTools.pm` module also depends on some of the brown modules. The grey boxes indicate open-source code, not within the `atec` module. Some of these may have underlying dependencies not included in the diagram.

## 4.2 Short description:

The `updateTEC.pl` script updates `podTec/ionTec` NetCDF files for a given daterange. It is meant to be used in post-processing where LEO DCB values are calculated after the generation of the `podTec/ionTec` files for a daterange (usually one month). The update affects several global attributes as well as the TEC vector data. Also the database is updated unless the `--nodb` option is specified. If the `--bydump` option is specified (default) then the arcs crossing the midnight right before the daterange (those will have dump `yr/doy` being the first date in the daterange) will be updated as well. If the `--byday` option is specified, those files with arcs crossing the midnight right before the daterange will instead be deleted (including their database entry if any).

## 4.3 Usage and command-line options:

```
updateTec.pl YYYY.DDD[-[YYYY.]DDD] MISSION [--prefix=filetype] [--(no)db] [--(byday|bydump)]
             [--pub=pubdir]
```

| | |
|---|---|
| `YYYY.DDD-YYYY.DDD` | is the time span in post-processing mode |
| `MISSION` | is the name of the CDAAC mission |
| `--prefix=filetype` | specifies list of filetypes seperated by commas (default is `--prefix=podTec,ionTec`) |
| `--db` | specifies that the database should be populated (default is `--db`) |
| `--bydump` | specifies that files should be updated based on dump `yr/doy` or … |
| `--byday` | specifies that files should be updated based on `arcyr/arcdoy` (default is `--bydump`) |
| `--pub=pubdir` | specifies where to find the `podTec/ionTec` files (default is `--pub=/pub`) |

## 4.4 Input data:

- `podTec_IIII.YYYY.DDD.HH.MM.UUUU.GGG.TT_SSSS.VVVV_nc` (low rate data)
  `ionTec_IIII.YYYY.DDD.HH.MM.UUUU.GGG.TT_SSSS.VVVV_nc` (medium rate data)

  NetCDF level 1b file including total electron content (TEC) data between LEO and GPS satellites. The `updateTEC.pl` script reads:

  - Filename information (via `PubFile.pm`; cf. section 4.1)
    - File type (`podTec` or `ionTec`)
    - `IIII`: CDAAC LEO name
    - `YYYY`: year
    - `DDD`: day of year
    - `HH`: hour
    - `MM`: minute
    - `GGG`: PRN number
    - `TT`: antenna ID
  - `TEC`: Total Electron Content along LEO-GPS link (TECU)
  - `elevation`: Elevation angle of LEO-GPS link (deg)
  - Global attributes
    - `dump_id`: ID of the dump where the arc ends
    - `leodcb_flag`: Flag indicating if arc is LEO DCB calibrated
    - `leodcb`: LEO DCB value (TECU; -999 if not calibrated)
    - `gpsdcb_flag`: Flag indicating if arc is GPS DCB calibrated
    - `gpsdcb`: GPS DCB value (TECU; -999 if not calibrated)

- `brnFil/MISSION.NN.STA`

  ASCII 'config' file including information about Bernese satellite and antenna names. The `AtecTools.pm` module (`readStation`; cf. section 4.1) reads:

  - CDAAC LEO name
  - Bernese LEO ID and antenna ID

- `leoDcb_YYYY.DDD_SSSS.VVVV_txt`

  ASCII level 1b file including LEO DCB values for all antennas. The `updateTEC.pl` script reads:

  - Filename information (via `PubFile.pm`; cf. section 4.1)
    - `YYYY`: year

    * `DDD`: day of year

  The `AtecTools.pm` module (`readDcb`; cf. section 4.1) reads:

  - Bernese LEO ID and antenna ID
  - LEO DCB value (ns)
  - RMS of LEO DCB value (ns)

- `codDcb_YYYY.DDD_txt`

  ASCII level 0 file including GPS DCB values for all PRNs. The `updateTEC.pl` script reads:

  - Filename information (via `PubFile.pm`; cf. section 4.1)
    * `YYYY`: year
    * `DDD`: day of year

  The `AtecTools.pm` module (`readDcb`; cf. section 4.1) reads:

  - PRN
  - GPS DCB value (ns)
  - RMS of GPS DCB value (ns)

## 4.5 Output data:

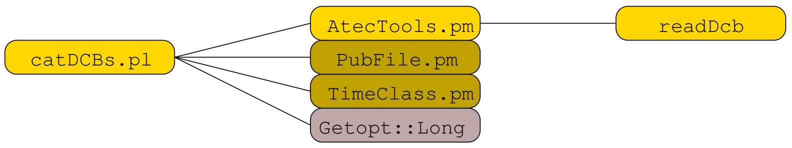
- `podTec_IIII.YYYY.DDD.HH.MM.UUUU.GGG.TT_SSSS.VVVV_nc` (low rate data)
  `ionTec_IIII.YYYY.DDD.HH.MM.UUUU.GGG.TT_SSSS.VVVV_nc` (medium rate data)

  NetCDF level 1b file including total electron content (TEC) data between LEO and GPS satellites. The `updateTEC.pl` script (via `PDL::NetCDF`; cf. section 4.1) updates:

  - `TEC`: Total Electron Content along LEO-GPS link (TECU)
  - Global attributes
    * `creation_time`: Time stamp indicating when file was created
    * `leodcb_flag`: Flag indicating if arc is LEO DCB calibrated
    * `leodcb_age`: Age in days of LEO DCB value (-999 if not calibrated)
    * `leodcb`: LEO DCB value (TECU; -999 if not calibrated)
    * `leodcb_rms`: RMS of LEO DCB value (TECU; -999 if not calibrated)
    * `gpsdcb_flag`: Flag indicating if arc is GPS DCB calibrated
    * `gpsdcb_age`: Age in days of GPS DCB value (-999 if not calibrated)
    * `gpsdcb`: GPS DCB value (TECU; -999 if not calibrated)
    * `gpsdcb_rms`: RMS of GPS DCB value (TECU; -999 if not calibrated)
    * `tecmin`: Minimum TEC in arc (TECU)
    * `tecmax`: Maximum TEC in arc (TECU)
    * `tecsinmax`: TEC times the sine of elevation at maximum elevation (TECU)

## 5 catDCBs.pl

### 5.1 Dependency chart:



The yellow boxes indicate Perl script/module and subroutines within the `atec` module. The brown boxes indicate other Perl modules, not within the `atec` module. Some of these may have underlying dependencies not included in the diagram. The `AtecTools.pm` module also depends on the brown modules. The grey box indicates open-source code, not within the `atec` module. It may have underlying dependencies not included in the diagram.

### 5.2 Short description:

The `catDCBs.pl` script concatenates a series of `leoDcb` or `corDcb/codDcb` files given a date range. Header info is disregarded. This is useful if one wants to plot a time series of the LEO DCBs. The output files are named according to the Bernese `PRN / STATION NAME`s (e.g., `G.C001.L21.POD1_txt`) and is put in the current directory.

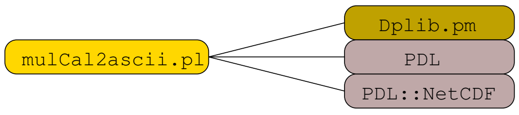
### 5.3 Usage and command-line options:

```
catDCBs.pl YYYY.DDD[-[YYYY.]DDD] MISSION [--prefix=filetype] [--pub=pubdir]
```

| | |
|---|---|
| `YYYY.DDD-YYYY.DDD` | is the time span of the time series |
| `--prefix=filetype` | specifes the filetype `leoDcb`, `corDcb`, or `codDcb`, but only one of them (default is `--prefix=leoDcb`) |
| `--pub=pubdir` | specifies where to find the `leoDcb` or `corDcb/codDcb` files (default is `--pub=/pub`) |

## 6 mulCal2ascii.pl

### 6.1 Dependency chart:



The yellow box indicates a Perl script within the `atec` module. The brown box indicates a Perl module, not within the `atec` module. It may have underlying dependencies not included in the diagram. The grey boxes indicate open-source code, not within the `atec` module. Some of these may have underlying dependencies not included in the diagram.

## 6.2 Short description:

The `mulCal2ascii.pl` script Converts a netCDF `mulCal` file to a space-delimited ASCII file that gnuplot likes for 3D plotting. The output file is put in the current directory with the extention `_txt` instead of `_nc`.

## 6.3 Usage:

```
mulCal2ascii.pl mulCal_YYYY.DDD.LLL.TT_nc
```

## References

Blewitt, G., 1990: An automatic editing algorithm for GPS data. *Geophys. Res. Lett.*, **17**, 199–202.

Kuipers, J. B., 2002: *Quaternions and Rotation Sequences.* Princeton University Press, Princeton.