NOTE 235 – Ionospheric Corrections in AIPS++

Oleg Smirnov

20 June 2000

Contents

1 Overview

1.1 Predicting and observing the ionosphere

The workhorse of the ionospheric calibration system is the *Ionosphere* class. The basic function of the Ionosphere class is to predict the electron density distribution (aka ED profile) and rotation measure (RM) along specified *slants*. A *slant* is defined as a time, position and line of sight (T, Lon, Lat, Az, El).

The Ionosphere class relies on two other class hierarchies, inherited from two abstract base classes. Descendants of *IonosphModel* provide apriori ionospheric models, such as PIM. Descendants of *IonosphData* handle observations of the ionosphere, such as GPS or ionosonde data. Ionosphere uses IonosphModel to build up apriori estimates of the ED distribution, then applies secondary corrections to achive the best fit with IonosphData.

Currently implemented is an *IonosphModelPIM* class, encapsulating *PIM* (the Parametrised Ionospheric Model developed by the USAF). An *IonosphDataGPS* class, for secondary corrections via GPS observations, is currently under development. Observations of GPS sattelites provide measurements of TEC (the total, or integral electron content) along lines of sight. Since at least six sattelites are passing across the sky at any given point in time, a GPS receiver can produce a wealth of TEC measurements, which can significantly improve any models of the ionosphere. Note that the GPS receiver does not need to be co-located with an inteferometer – just near enough, so that the GPS sattelites are observed through "interesting" parts of the ionosphere at least some of the time.

PIM itself is a monolithic edifice of FORTRAN code. Additional FORTRAN routines have been added by Bob Campbell, to facilitate access to the model using slants. IonosphModelPIM is basically a C++ wrapper around all this FORTRAN code. None of the FORTRAN was modified. Thus, we can reasonably expect future versions of PIM to plug into AIPS++ without too much trouble.

1.2 Calibration

The *FVisJones* class (inherited from TimeVarVisJones) implements the F-term of the Measurement Equation. This facilitates correction for Faraday Rotation in the visibilities plane (thus assuming a spatially-constant ionosphere over the whole beam). FVisJones extracts the epochs, source and station coordinates from the Measurement Set, passes them to Ionosphere to obtain ED profiles and derive rotation measures, and constructs a rotation matrix for each frequency channel.

Note that ionospheric prediction is thus completely decoupled from the calibration. FVisJones knows nothing of how the ionosphere is predicted. The Ionosphere classes can be revised and developed independently of FVisJones. Finally, an FSkyJones, for corrections in the image plane, may be developed independently of Ionosphere.

2 Global data used in ionospheric calibration

Operationally, the Ionosphere classes require a large number of external data sets from a variety of sources. These data sets must be somehow integrated into AIPS++ in a way transparent to the end-user.

2.1 Data used by PIM

The PIM FORTRAN code reads an impressive amount of external data files. Since this code is plugged into AIPS++ "as is", these files must be available at run-time, in their "native" (i.e., PIM-readable) format.

2.1.1 PIM databases

This is a large set of data files used by the PIM model. This data set is a total black box. It should be considered an integral part of the PIM software, since the databases are never updated in between PIM releases.

PIM databases are distributed by the USAF along with the PIM software. The distribution format is ASCII files; a set of utilities is provided for converting the ASCII files into native binary format which is required by PIM at run-time. Total size is roughly 30 Mb.

2.1.2 Geophysical data

Daily solar flux ($F_{10.7}$, K_p, A_p) and interplanetary magnetic field (IMF) observations. PIM itself expects this data to be in the form of ASCII tables. Data size is negligible.

Daily solar flux data is available by anonymous ftp from NOAA, with a 1–2 month lag. More recent data is available in "preliminary" format, as well as near-future "forecast" data. PIM cannot work with preliminary/forecast data (yet?).

Daily IMF observations are available via the Web from NASA/GSFC. Time lag is about 3 months. An important open issue is that IMF observations contain gaps. For some days, there is no IMF data at all, so some fallback strategy must be devised. Note that the only IMF datum of interest to PIM is the *sign* of the B_z component (which ultimately determines whether the IMF force lines are easily coupled with the EMF force lines).

2.2 Data used by Ionosphere

Since the rotation measure is determined by both electron density and the magnetic field strength, Ionosphere needs to know the Earth magnetic field. The current version relies on a FORTRAN routine within PIM, which in turn uses the IGRF model. Because IGRF is already incorporated into the AIPS++ Measures system (see, e.g., MVEarthMagnetic, EarthMagneticMachine), I plan to implement a C++ version of this routine in the near future.

2.3 Additional data required for GPS-based corrections

RINEX (Receiver Independent Exchange Format). Data from GPS receiver(s), containing observations of GPS sattelites for a given period. RINEX is basically an ASCII table. The data is downloaded from the receiver directly at the observing station. Alternatively, RINEX data from receivers participating in the IGS network is available by ftp from IGS data centers around the world, most notably NASA/GSFC and IGN (France). A day's worth of RINEX data (typically, at 30-sec sampling) is about 2 Mb in ASCII form. Time lag is anywhere from a few days to a couple of weeks.

Ephemeris data for the GPS sattelites. Per-day predicted and observed ephemeris tables are available via the Web from the JPL, in a standard ASCII format called IGS-SP3. There is no time lag (since even up-to-date predicted ephemeris provide sufficient accuracy for our needs). 24 hours' worth of ephemeris, typically at 15-min sampling, is about 160 Kb.

TGD The individual satellites' clock offsets ("group delays"). These vary in time (albeit slowly) and must be updated somehow; I have not yet found an adequate source for automatic updates. The data seems to be available for the asking at various data centers, but, unlike the other data sets mentioned here, not made publicly accessible in a standard format. Data size is negligible.

3 Where the data goes

PIMDB. Since PIMDB won't change without a corresponding PIM release, it belongs with the code, and ought to be distributed along with AIPS++ itself. The ASCII-to-binary utilities can be run once (i.e., at installation time) to generate the required binary tables.

Geophysical data, TGD clearly fits into the Global Data Proposal. Let the Consortium worry about keeping it up-to-date; the end-user will get it all centrally, in one piece. Consequently, the data will be distributed as AIPS++ tables; utilities for exporting these tables into PIM-friendly ASCII files will have to be provided (and at some point, it is quite feasible to get rid of these intermediate files altogether.)

RINEX data clearly belongs in a sub-table of the Measurement Set, since it relevant to a particular observation.

Ephemeris. I haven't come to a definite conclusion regarding Ephemeris data. On one hand, it is clearly global like geophysical data, on the other, keeping all of it around, with every AIPS++ install, does tend to get pointlessly expensive – a full year would run to about 60 Mb. It may be worthwhile to keep ephemeris only for the relevant dates, in a sub-table of the Measurement Set. E.g., Consortium sites could maintain a full ephemeris archive, and end-users would only download the relevant chunks as needed.

4 Table formats

Note: So far, I've only written up RINEX and Ephemeris tables. They are the only ones that involve the Measurement Set. Table formats for geophysical data are, for the most part, quite trivial.

The strategy here is to stay as close as possible, content-wise, to the "raw" data formats.

4.1 The RINEX table

The RINEX table is supposed to store RINEX data relevant to the MS. This data is not necessarily homogenous, as it can come in several RINEX files and perhaps from several GPS receivers (with different sampling rates, etc.). I propose to store the data in "chunks". One chunk is a block of samples from a single receiver, contiguous in time. A single RINEX file corresponds to a chunk; several RINEX files from the same receiver for adjacent time periods can be (but do not have to be) merged into one chunk.

Each row of a RINEX table will correspond to a chunk; attaching another RINEX file to the MS adds one row to this table.

RINEX table keywords: none so far.

RINEX table columns:

HEADER (String) The full header of the original RINEX file.

RCV_POSITION (MPosition) The position of the GPS receiver (ITRF frame).

SAMPLING_INTERVAL (Float) The sampling interval, in seconds.

NUM_EPOCHS (Int) Number of samples in this chunk.

EPOCH_BEG, EPOCH_END (MEpoch) First/last epoch in chunk.

SAT_COUNTS (Vector<uInt>) Per-sattelite sample counts.

DATA (subtable) Actual delay data for this chunk. Each row of the subtable corresponds to a single RINEX record. The columns are:

  EPOCH (MEpoch) time of observation;

  SVN (Int) satellite ID;

  D (Double$\times$5) L1,L2,P1,P2,CA delay values;

  SSI (uChar$\times$5) Signal strength indicators;

  LLI (uChar$\times$5) Loss-of-lock indicators.

4.2 The EPHEMERIS table

The EPHEMERIS table holds GPS sattelite ephemeris. Since this table is going to be either stored in the MS (as a small sampling), or distributed incrementally, it seems reasonable to hold one day's data for all sattelites in a single row of the table. This makes it easier to separate the data into daily chunks when necessary. This also simplifies mapping of the EPHEMERIS table to original IGS-SP3 files (which are distributed on a 1-day basis).

EPHEMERIS table keywords: none so far.

EPHEMERIS table columns:

HEADER (String) Full header of original IGS-SP3 file.

MJD (Int) MJD for which ephemeris are given.

INTERVAL (Float) sampling time, in seconds.

SVN (Vector<Int>) vector of satellite IDs.

ORBIT_ERROR (Vector<Int>) orbital error indicators, per each SVN.

MJD_FRAC (Vector<Double>) epoch of each sample (fractional day).

X, Y, Z (Matrix<Double>) actual ephemeris. The rows of each matrix correspond to samples, the columns to sattellites.