gAGE research group of Astronomy and Geomatics
Barcelona, Spain

High Accuracy Positioning using Carrier-phases with the Open Source GPSTk Software

Salazar D., Hernandez-Pajares M.,
Juan J.M. and Sanz J.
gAGE/UPC, Barcelona, Spain

Contents

- GPSTk overview
- Some current features
- GNSS Data Structures
- Some processing examples
- Conclusions and future work

GPSTk overview

gAGE research group of Astronomy and Geomatics
Barcelona, Spain

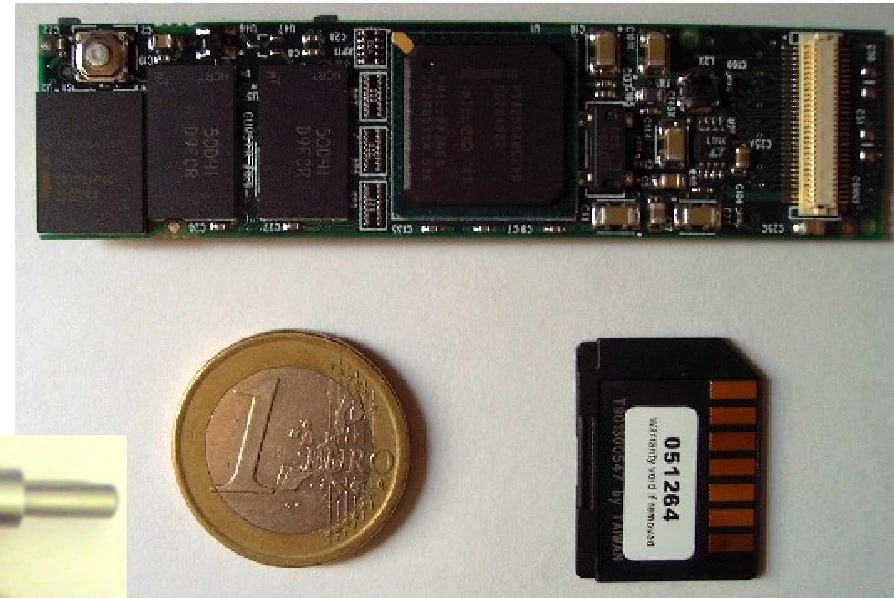
- GPSTk is:
  - A **library** to write GNSS software
  - Includes example **applications**.
- GPSTk is **Free Software** (LGPL):
  - Both non-commercial *and*
  - commercial applications.

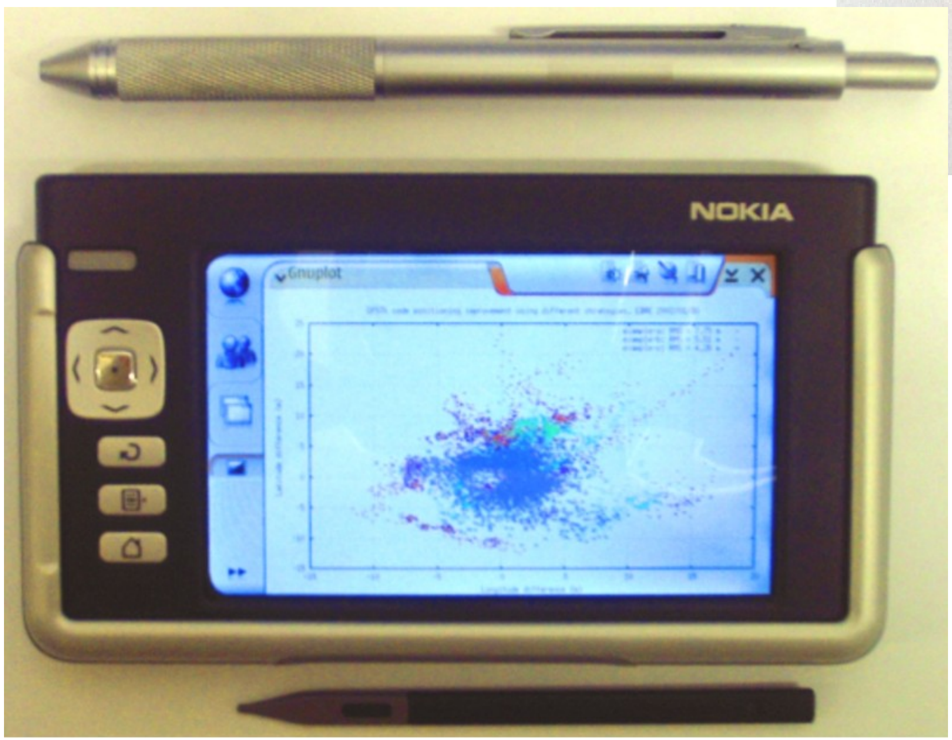
GPSTk overview

- Written in ISO-standard C++ (portable).
- Operative System portability:
  - Works in Windows, Linux, Mac OSX, AIX, Solaris, etc.
- Hardware portability:
  - Big and (very) small systems, both 32 and 64 bits.

GPSTk portability

GUMSTIX Miniature computer





NOKIA Internet Tablets

GPSTk overview

- It was initiated at the Advanced Research Laboratories at Texas University (ARL:UT)
- Now it has several official developers around the world.
- Project website:

http://www.gpstk.org

Some current features

- Time conversions.
- RINEX files reading/writing:
  - Observation
  - Ephemeris
  - Meteorological
- Ephemeris computation:
  - Broadcast
  - SP3.

Some current features

- Mathematical tools: Matrices, vectors, interpolation, numeric integration, LMS, W-LMS, Kalman filter, etc.
- Application development support:
  - Exceptions handling.
  - Command line framework.

Some current features

- Several tropospheric models: Saastamoinen, Goad-Goodman, New Brunswick, etc.
- Classes for precise modeling: Phase centers, Wind-up, gravitational delay, etc.
- Tidal models:
  - Solid tides
  - Ocean loading
  - Pole tides

GNSS Data Structures

- GNSS Data Structures (GDS) were designed to help in GNSS data processing.
- They address GNSS ***data management*** problems.

GNSS Data Structures

- Almost all GNSS data may be identified (or indexed) by:
  - **Source**: Receiver logging data.
  - **Satellite**: GPS, GALILEO, GLONASS...
  - **Epoch**: Time the data belongs to.
  - **Type**: C1, P1, L2, etc, and other types.

Key point: *Save Meta-Information:*

Save 4 indexes to identify

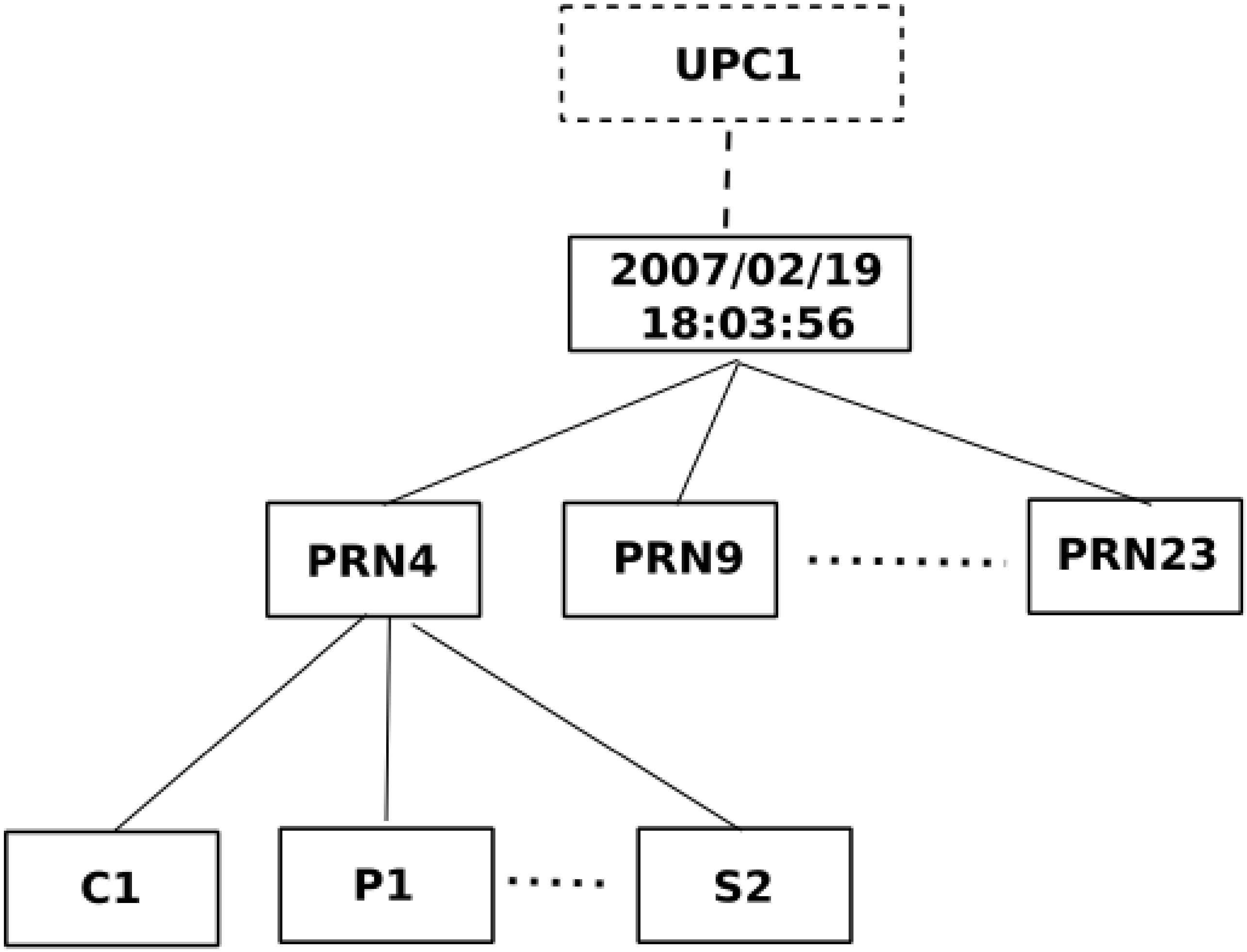
each piece of data

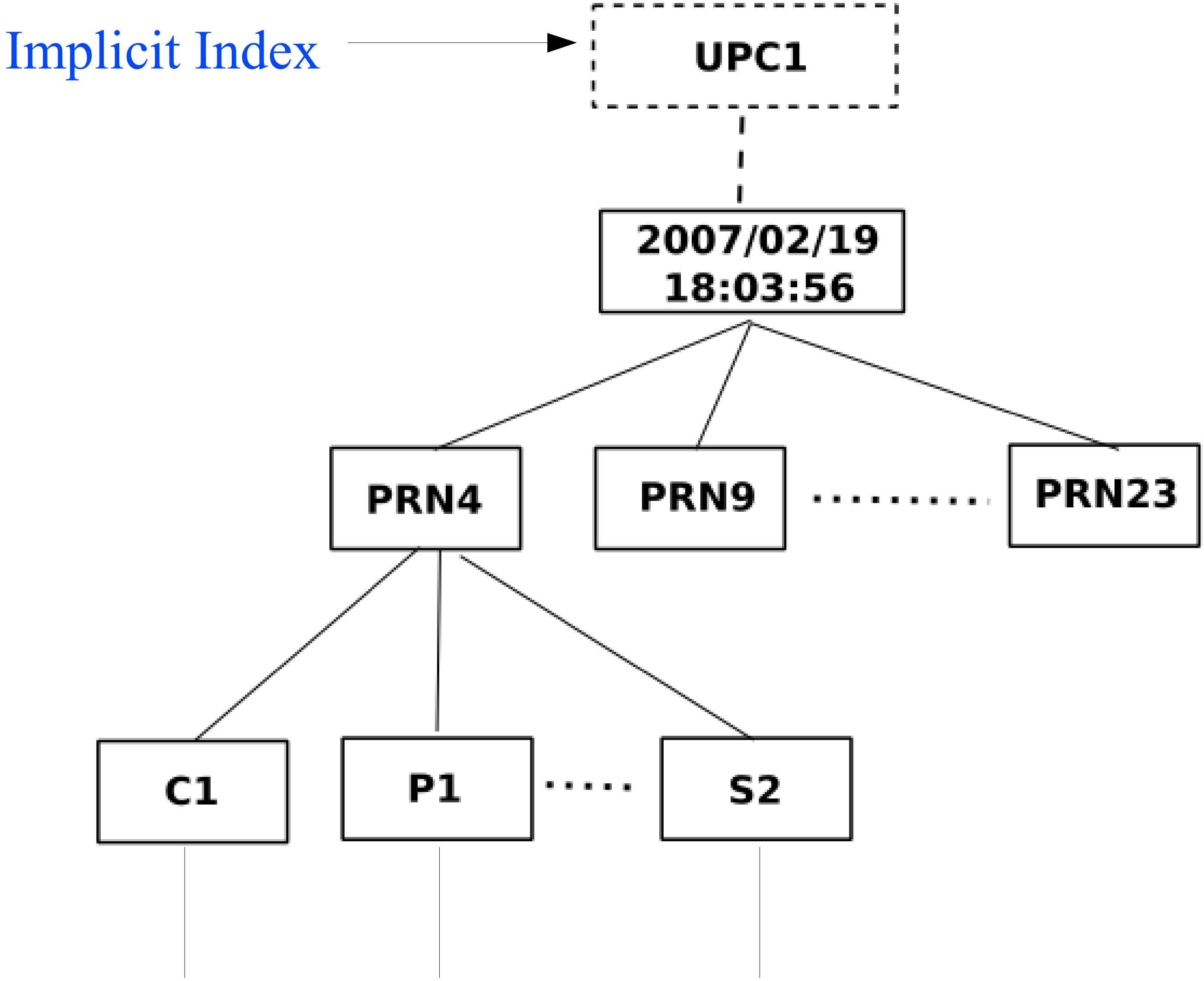
GNSS Data Structures

- **Source**: Implemented as class <u>**SourceID**</u>.
- **Satellite**: Implemented as class <u>**SatID**</u>.
  - Several systems: GPS, Galileo, Glonass...
- **Epoch**: Implemented as class <u>**DayTime**</u>.
- **Type**: Implemented as class <u>**TypeID**</u>.
  - Includes basic observations: C1, P1, etc.
  - Other “types”: Relativity, tropo, etc.
  - *You can create new types as needed!!!.*

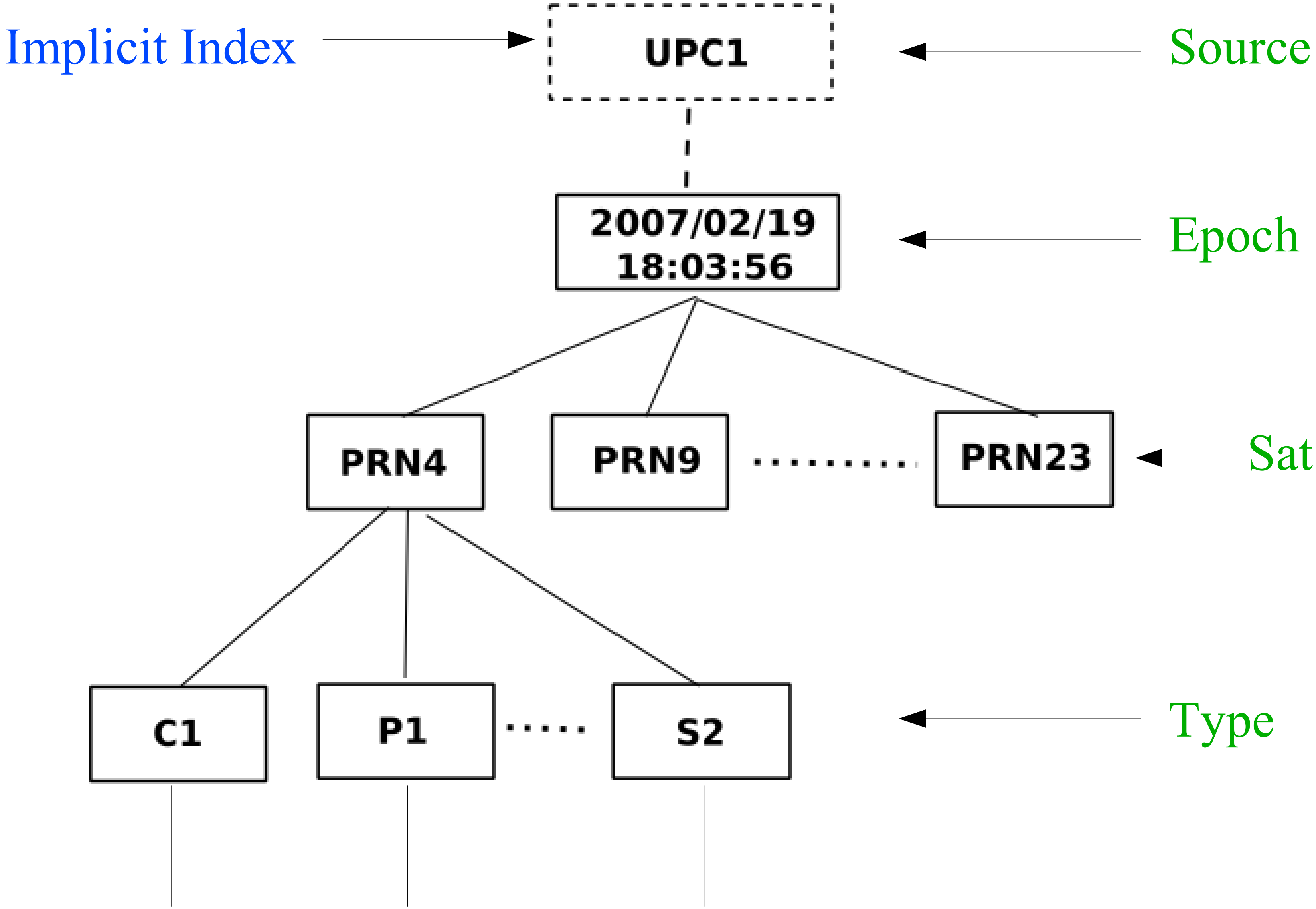
Some conceptual GDS examples

Example: Rinex observation data for a given epoch



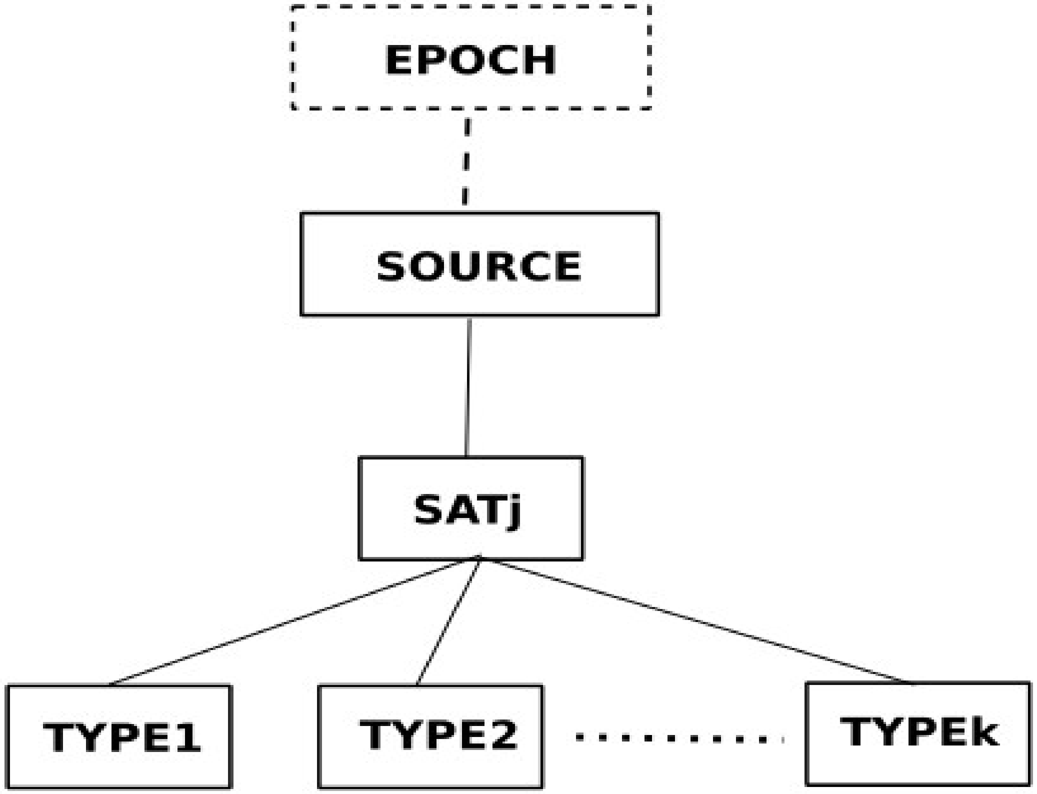


Real numerical values are down here...

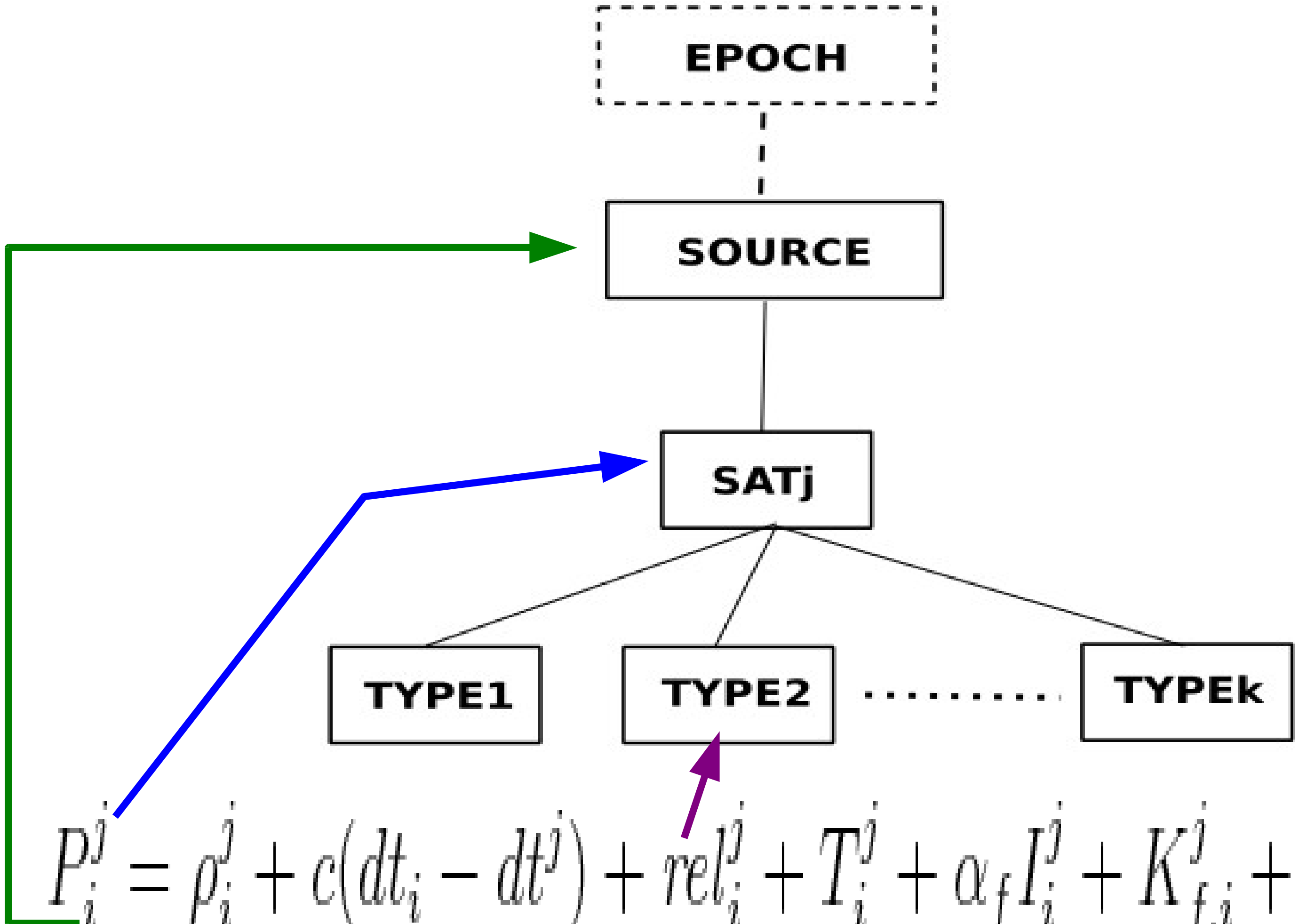


Real numerical values are down here...

Example: Pseudorange model equation



$$P_i^j = \rho_i^j + c\left(dt_i - dt^j\right) + rel_i^j + T_i^j + \alpha_f I_i^j + K_{f,i}^j + M_{P,i}^j + \varepsilon_{P,i}^j$$



$$P_i^j = \rho_i^j + c\left(dt_i - dt^j\right) + rel_i^j + T_i^j + \alpha_f I_i^j + K_{f,i}^j + M_{P,i}^j + \varepsilon_{P,i}^j$$

Epoch is implicit: Equation values are valid for a *given epoch*

GNSS Data Structures

- Almost all GNSS data may be identified (or indexed) by:
  - **Source**: Receiver logging data.
  - **Satellite**: GPS, GALILEO, GLONASS...
  - **Epoch**: Time the data belongs to.
  - **Type**: C1, P1, L2, etc, and other types.

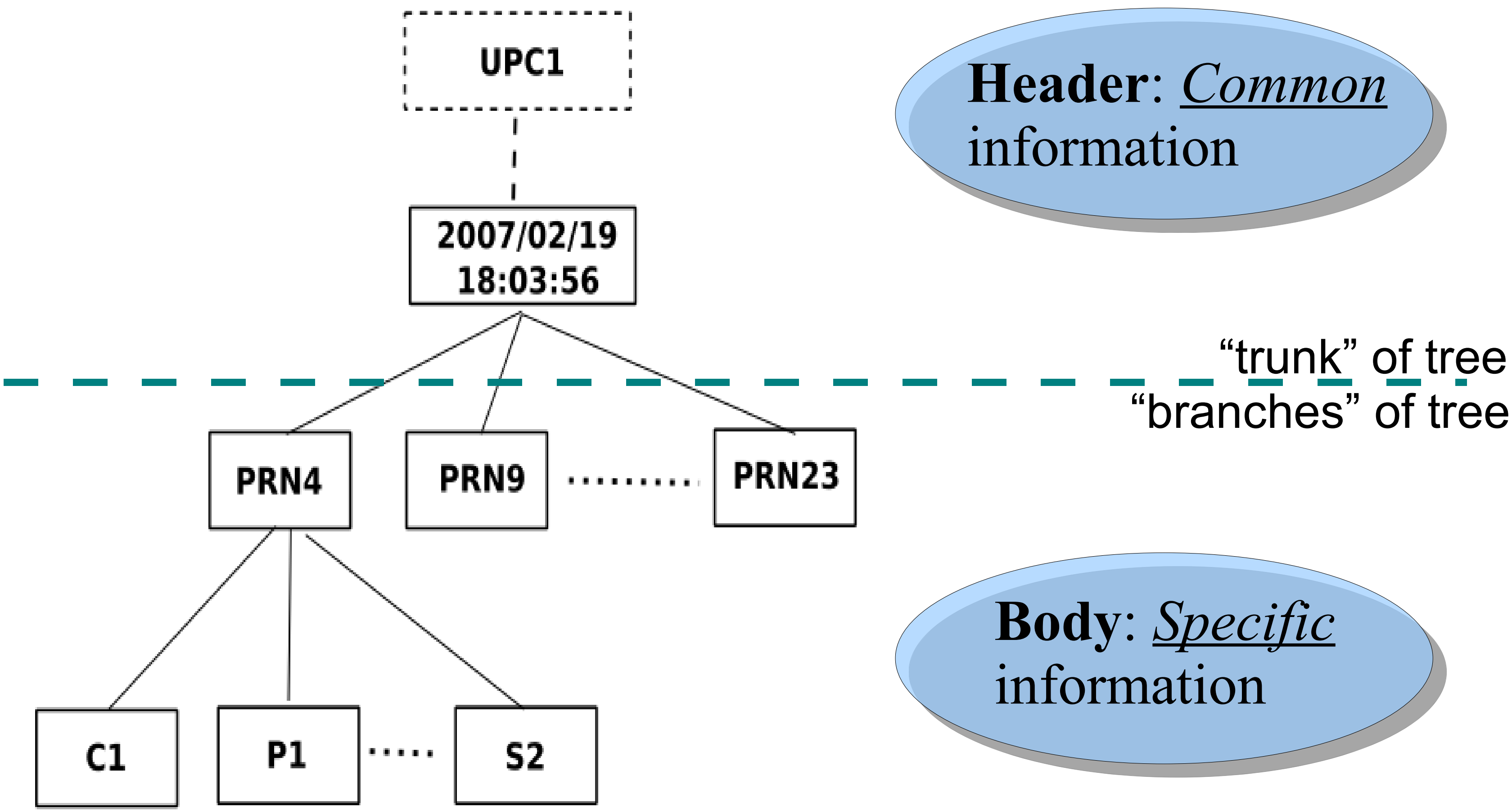
Key point: *Save Meta-Information:*

Save 4 indexes to identify

each piece of data

Implementation

- GDS have two parts:



gAGE research group of Astronomy and Geomatics
Barcelona, Spain

Implementation

- GDS provide several methods to ease handling. For instance:

Keep only C1 observable in GDS:

gpsData.keepOnlyTypeID(TypeID::C1)

Implementation

- GDS provide several methods to ease handling. For instance:

Keep only C1 observable in GDS:

```
gpsData.keepOnlyTypeID(TypeID::C1)
```

GDS

Method

Type to keep

Implementation

- GDS provide several methods to ease handling. For instance:

Access GDS in a matrix-like fashion:

gpsData.body(TypeID::LI)(sat14)

Implementation

- GDS provide several methods to ease handling. For instance:

Access GDS in a matrix-like fashion:

gpsData.body(TypeID::LI)(sat14)

GDS "Branches" Type Satellite

Processing paradigm

- Operator "**>>**" is *overloaded*:

  We *redefine it* in C++ to make data "*flow*" from one processing step to the next

Processing paradigm

Example:

Declare object "rinexFile" to take data out of Rinex observation file:

```
RinexObsStream rinexFile("ebre0300.02o");
```

Processing paradigm

Example:

Declare object "rinexFile" to take data out of Rinex observation file:

RinexObsStream rinexFile("ebre0300.02o");

Class to handle Rinex Obs files — Object name — Rinex file name

Processing paradigm

Example:

*Declare object "gpsData". This is a **GDS**:*

gnssRinex gpsData;

Processing paradigm

Example:

Declare object "gpsData". This is a ***GDS****:*

gnssRinex gpsData;

GDS Class   Object name

Processing paradigm

Example:

Combine everything and take ***one epoch*** *of data* ***out*** *of Rinex file* ***into*** *GDS:*

```
RinexObsStream rinexFile("ebre0300.02o");
gnssRinex gpsData;

rinexFile  >>  gpsData;
```

Processing paradigm

Example:

*Combine everything and take **one epoch** of data **out** of Rinex file **into** GDS:*

```
RinexObsStream rinexFile("ebre0300.02o");
gnssRinex gpsData;

rinexFile  >>  gpsData;
```

Processing line

Processing paradigm

Example:

*Combine everything and take **one epoch** of data **out** of Rinex file **into** GDS:*

```
RinexObsStream rinexFile("ebre0300.02o");
gnssRinex gpsData;

rinexFile >> gpsData;
```

Processing line

"Flow" operator

Processing paradigm

- **Main Idea**: GNSS data processing becomes like an "*assembly line*".
  - The GDS "*flows*" from one "*workstation*" to the next, like a car in factory.
  - And like the car, the GDS *changes* along the processing line.

- For the developer, the GDS is like a "white box" holding all the information needed, and properly indexed.

Processing *code-based* GPS data with the GPSTk and GDS:

Some examples

C1 code observable + least-mean squares solver: Plain standard GPS processing

C1 + LMS

- Get data out of Rinex file, ***one epoch at a time***
  - Get data into GDS (*gpsData*)
  - Do it until Rinex file ends.

C1 + LMS

- Get data out of Rinex file, ***one epoch at a time***
  - Get data into GDS (*gpsData*)
  - Do it until Rinex file ends.

```
while ( rinexFile >> gpsData )
{

}
```

C1 + LMS

- Get data into modeling object *gpsModel*
  - *GpsModel* was previously initialized:
    - Ephemeris, tropospheric model, Klobuchar ionospheric model, etc.
  - Initialization phase is not shown here.

```
while ( rinexFile >> gpsData )
{

}
```

C1 + LMS

- Get data into modeling object *gpsModel*
  - *GpsModel* was previously initialized:
    - Ephemeris, tropospheric model, Klobuchar ionospheric model, etc.
  - Initialization phase is not shown here.

```
while ( rinexFile >> gpsData )
{
   gpsData >> gpsModel
}
```

C1 + LMS

- Resulting GDS with original data and modeled data gets into solver object *lmsSolver*
  - *lmsSolver* is from *SolverLMS* class.
  - *lmsSolver* was previously initialized.
  - Initialization phase is not shown here.

```
while ( rinexFile >> gpsData )
{
   gpsData >> gpsModel
}
```

C1 + LMS

gAGE research group of Astronomy and Geomatics
Barcelona, Spain

- Resulting GDS with original data and modeled data gets into solver object *lmsSolver*
  - *lmsSolver* is from *SolverLMS* class.
  - *lmsSolver* was previously initialized.
  - Initialization phase is not shown here.

```
while ( rinexFile >> gpsData )
{
   gpsData >> gpsModel >> lmsSolver;
}
```

C1 + LMS

- Resulting GDS with original data and modeled data gets into solver object *lmsSolver*
  - *lmsSolver* is from *SolverLMS* class.
  - *lmsSolver* was previously initialized.
  - Initialization phase is not shown here.

```
while ( rinexFile >> gpsData )
{
    gpsData >> gpsModel >> lmsSolver;
}
```

← Processing line

C1 + LMS

- **All GNSS data processing is done!!!**
  - Print results to the screen.
  - C++ *cout* object is used to print to screen.
  - Solution is inside *lmsSolver*: Ask for type.

```
while ( rinexFile >> gpsData )
{
   gpsData >> gpsModel >> lmsSolver;




}
```

C1 + LMS

- **All GNSS data processing is done!!!**
  - Print results to the screen.
  - C++ *cout* object is used to print to screen.
  - Solution is inside *lmsSolver*: Ask for type.

```
while ( rinexFile >> gpsData )
{
   gpsData >> gpsModel >> lmsSolver;

   cout << lmsSolver.getSolution(TypeID::dx) << " ";
   cout << lmsSolver.getSolution(TypeID::dy) << " ";
   cout << lmsSolver.getSolution(TypeID::dz) << endl;
}
```

C1 + LMS + ENU

- To get solution in a East-North-Up frame:
  - Insert a *XYZ2NEU* object (*baseChange*).
  - Use a *SolverLMS* object properly tuned.
  - Solution is in *enuSolver*: Ask for right type.

```
while ( rinexFile >> gpsData )
{
   gpsData >> gpsModel >> baseChange >> enuSolver;

   cout << enuSolver.getSolution(TypeID::dLat) << " ";
   cout << enuSolver.getSolution(TypeID::dLon) << " ";
   cout << enuSolver.getSolution(TypeID::dUp) << endl;
}
```

PC (ionosphere free) code + smoothing with LC + weighted least-mean squares solver + East-North-Up (ENU) system

Smoothed-PC + WMS + ENU

- First, get combinations we will need:
  - PC and LC (ionosphere-free).
  - LI (ionospheric combination).
  - MW (Melbourne-Wübbena)
- Look for corresponding classes in GPSTk API

```
while ( rinexFile >> gpsData )
{
   gpsData >>
}
```

Smoothed-PC + WMS + ENU

- First, get combinations we will need:
  - PC and LC (ionosphere-free).
  - LI (ionospheric combination).
  - MW (Melbourne-Wübbena)
- Look for corresponding classes in GPSTk API

```
while ( rinexFile >> gpsData )
{
   gpsData >> getPC >> getLC >> getLI >> getMW
}
```

Smoothed-PC + WMS + ENU

- Then, detect and mark cycle-slips:
  - Two different and complementary methods are used.
- Everything is in a single processing line.

```
while ( rinexFile >> gpsData )
{
   gpsData >> getPC >> getLC >> getLI >> getMW



}
```


gAGE research group of Astronomy and Geomatics
Barcelona, Spain

Smoothed-PC + WMS + ENU

- Then, detect and mark cycle-slips:
  - Two different and complementary methods are used.
- Everything is in a single processing line.

```
while ( rinexFile >> gpsData )
{
   gpsData >> getPC >> getLC >> getLI >> getMW
         >> markCSLI >> markCSMW


}
```

Smoothed-PC + WMS + ENU

- Smooth code with phase

```
while ( rinexFile >> gpsData )
{
    gpsData >> getPC >> getLC >> getLI >> getMW
            >> markCSLI >> markCSMW


}
```

Smoothed-PC + WMS + ENU

- Smooth code with phase

```
while ( rinexFile >> gpsData )
{
    gpsData >> getPC >> getLC >> getLI >> getMW
        >> markCSLI >> markCSMW  >> smoothPC


}
```

Smoothed-PC + WMS + ENU

- Smooth code with phase
- Apply model

```
while ( rinexFile >> gpsData )
{
   gpsData >> getPC >> getLC >> getLI >> getMW
         >> markCSLI >> markCSMW  >> smoothPC
         >> gpsModel

}
```

Smoothed-PC + WMS + ENU

- Smooth code with phase
- Apply model
- Compute weights

```
while ( rinexFile >> gpsData )
{
    gpsData >> getPC >> getLC >> getLI >> getMW
            >> markCSLI >> markCSMW  >> smoothPC
            >> gpsModel >> getWeights

}
```

Smoothed-PC + WMS + ENU

- Smooth code with phase
- Apply model
- Compute weights
- Change base and solve with a WMS solver

```
while ( rinexFile >> gpsData )
{
    gpsData >> getPC >> getLC >> getLI >> getMW
            >> markCSLI >> markCSMW  >> smoothPC
            >> gpsModel >> getWeights >> baseChange
            >> wmsSolver;
}
```

Smoothed-PC + WMS + ENU

- **All GNSS data processing is done!!!**
  - Print results to the screen.
  - Solution is inside *wmsSolver*: Ask for type.
- Everything is in a single processing line.

```
while ( rinexFile >> gpsData )
{
    gpsData >> getPC >> getLC >> getLI >> getMW
            >> markCSLI >> markCSMW  >> smoothPC
            >> gpsModel >> getWeights >> baseChange
            >> wmsSolver;
}
```

Differential GPS + C1 code observable + weighted least-mean squares + East-North-Up (ENU) system

C1 + DGPS + WMS + ENU

- First, process reference station data:
  - *synchro* is a *Synchronize* class object.
  - It is in charge of synchronizing reference and rover data streams.
  - Initialization is not shown here. Consult API.

```
while ( rinexFile >> gpsData )
{
   gpsDataRef  >> synchro >> refModel;

}
```

C1 + DGPS + WMS + ENU

- Then, indicate the reference data:
  - *delta* is a *DeltaOp* class object.
  - It computes single diffences using common satellites.
  - You must tell it which is the reference data.

```
while ( rinexFile >> gpsData )
{
   gpsDataRef  >> synchro >> refModel;

   delta.setRefData( gpsDataRef.body );

}
```

C1 + DGPS + WMS + ENU

- Finally, process rover receiver data:
  - You must insert delta object into processing line to compute single differences.

```
while ( rinexFile >> gpsData )
{
   gpsDataRef  >> synchro >> refModel;

   delta.setRefData( gpsDataRef.body );

   gpsData >> gpsModel >> getWeights >> delta
         >> baseChange >> wmsSolver;
}
```

C1 + DGPS + WMS + ENU

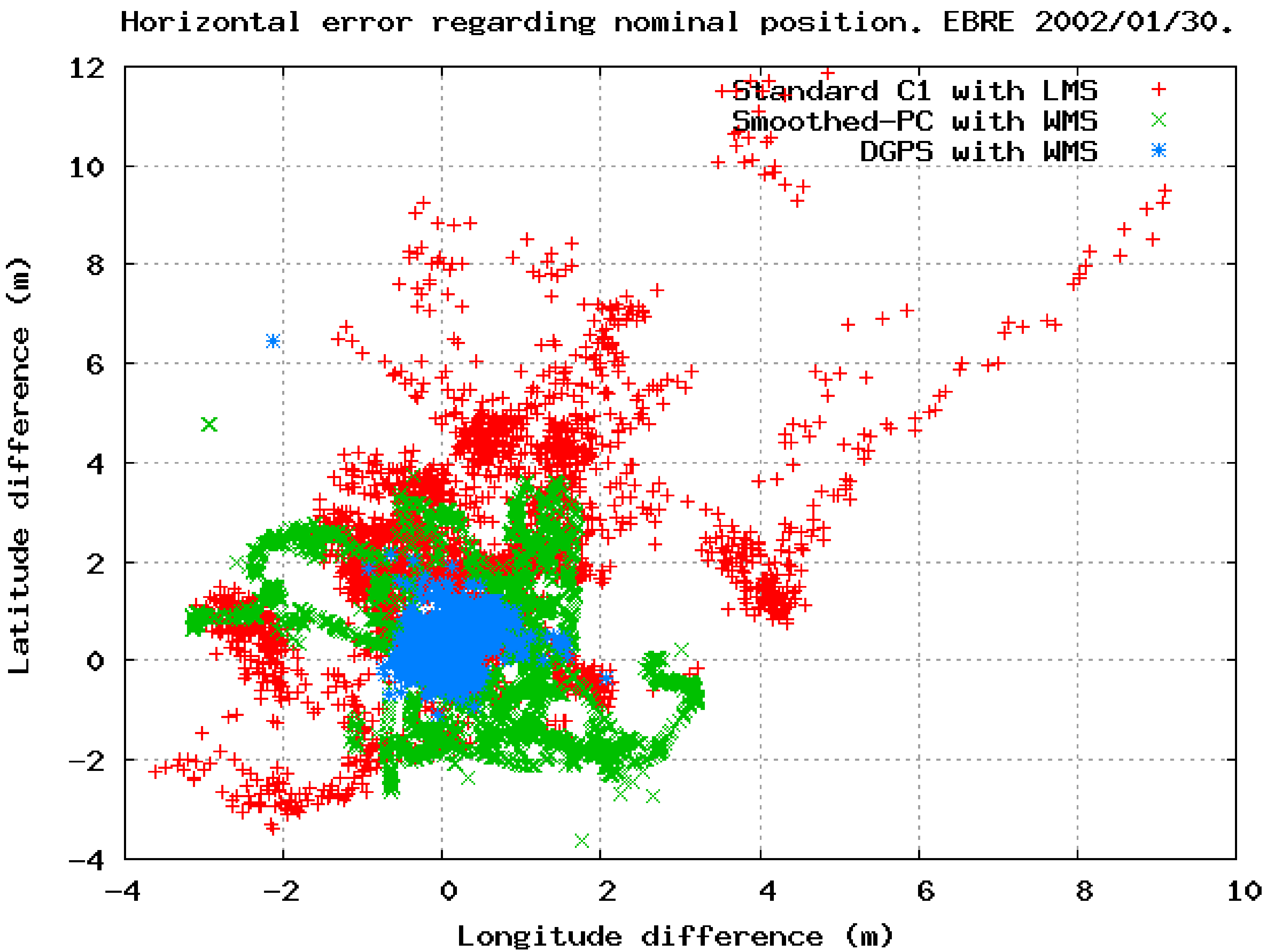
- **All GNSS data processing is done!!!**
  - Print results to the screen.
  - Solution is inside *wmsSolver*: Ask for type.

```
while ( rinexFile >> gpsData )
{
    gpsDataRef  >> synchro >> refModel;

    delta.setRefData( gpsDataRef.body );

    gpsData >> gpsModel >> getWeights >> delta
            >> baseChange >> wmsSolver;
}
```

Code-based results (as expected)



Processing <u>phase-*based*</u> GPS data with the GPSTk and GDS:

Some examples of *Precise Point Positioning (PPP)*

PPP (static, forward mode)

```
correctObservables.setExtraBiases(tides);
```

Set “correcting” object with tide information.
It also includes RX antenna phase center and eccentricity

PPP (static, forward mode)

```
correctObservables.setExtraBiases(tides);

gpsData >> basicGPSModel
```

Compute basic components of model

PPP (static, forward mode)

```
correctObservables.setExtraBiases(tides);

gpsData >> basicGPSModel
        >> removeEclipsedSatellites
```

Remove satellites in eclipse

PPP (static, forward mode)

```
correctObservables.setExtraBiases(tides);

gpsData >> basicGPSModel
        >> removeEclipsedSatellites
        >> gravitationalDelay
```

Compute gravitational delay

PPP (static, forward mode)

```
correctObservables.setExtraBiases(tides);

gpsData >> basicGPSModel
        >> removeEclipsedSatellites
        >> gravitationalDelay
        >> computeSatellitePhaseCenter
```

Compute the effect of satellite phase centers

PPP (static, forward mode)

```
correctObservables.setExtraBiases(tides);

gpsData >> basicGPSModel
        >> removeEclipsedSatellites
        >> gravitationalDelay
        >> computeSatellitePhaseCenter
        >> correctObservables
```

Correct observables from tides, RX phase center, etc

PPP (static, forward mode)

```
correctObservables.setExtraBiases(tides);

gpsData >> basicGPSModel
        >> removeEclipsedSatellites
        >> gravitationalDelay
        >> computeSatellitePhaseCenter
        >> correctObservables
        >> windUp
```

Compute wind-up effect

PPP (static, forward mode)

```
correctObservables.setExtraBiases(tides);

gpsData >> basicGPSModel
        >> removeEclipsedSatellites
        >> gravitationalDelay
        >> computeSatellitePhaseCenter
        >> correctObservables
        >> windUp
        >> computeTropo
```

Compute tropospheric effect (Niell model)

PPP (static, forward mode)

```
correctObservables.setExtraBiases(tides);

gpsData >> basicGPSModel
        >> removeEclipsedSatellites
        >> gravitationalDelay
        >> computeSatellitePhaseCenter
        >> correctObservables
        >> windUp
        >> computeTropo
        >> computeLinearCombinations
```

Compute common linear combinations: PC, LC, LI, ...

PPP (static, forward mode)

```
correctObservables.setExtraBiases(tides);

gpsData >> basicGPSModel
        >> removeEclipsedSatellites
        >> gravitationalDelay
        >> computeSatellitePhaseCenter
        >> correctObservables
        >> windUp
        >> computeTropo
        >> computeLinearCombinations
        >> markCSLI
        >> markCSMW
```

Marck cycle slips: Two complementary algorithms

PPP (static, forward mode)

```
correctObservables.setExtraBiases(tides);

gpsData >> basicGPSModel
        >> removeEclipsedSatellites
        >> computeSatellitePhaseCenter
        >> correctObservables
        >> windUp
        >> computeTropo
        >> computeLinearCombinations
        >> markCSLI
        >> markCSMW
        >> computePrefitResiduals
```

Compute prefit residuals

PPP (static, forward mode)

```
correctObservables.setExtraBiases(tides);

gpsData >> basicGPSModel
        >> removeEclipsedSatellites
        >> computeSatellitePhaseCenter
        >> correctObservables
        >> windUp
        >> computeTropo
        >> computeLinearCombinations
        >> markCSLI
        >> markCSMW
        >> computePrefitResiduals
        >> decimateData
```

Decimate data if epoch is not a multiple of 900 seconds

PPP (static, forward mode)

```
correctObservables.setExtraBiases(tides);

gpsData >> basicGPSModel
        >> removeEclipsedSatellites
        >> computeSatellitePhaseCenter
        >> correctObservables
        >> windUp
        >> computeTropo
        >> computeLinearCombinations
        >> markCSLI
        >> markCSMW
        >> computePrefitResiduals
        >> decimateData
        >> baseChange
```

Change from ECEF to ENU reference frame

PPP (static, forward mode)

gAGE research group of Astronomy and Geomatics
Barcelona, Spain

```
correctObservables.setExtraBiases(tides);

gpsData >> basicGPSModel
        >> removeEclipsedSatellites
        >> computeSatellitePhaseCenter
        >> correctObservables
        >> windUp
        >> computeTropo
        >> computeLinearCombinations
        >> markCSLI
        >> markCSMW
        >> computePrefitResiduals
        >> decimateData
        >> baseChange
        >> pppSolver;
```

Solve equations with a properly adjusted Kalman filter

PPP (static, forward mode)

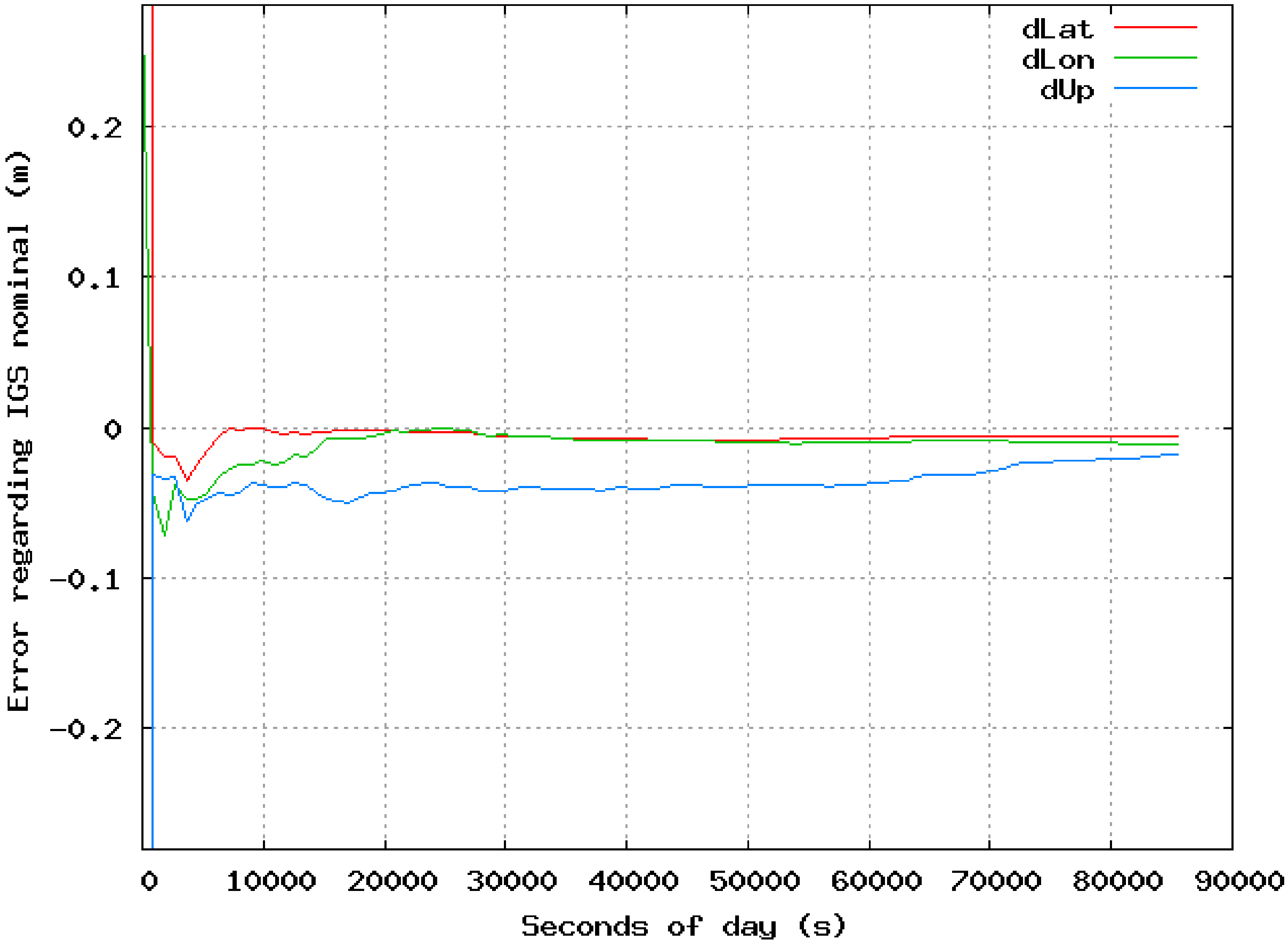
```
correctObservables.setExtraBiases(tides);

gpsData >> basicGPSModel
        >> removeEclipsedSatellites
        >> gravitationalDelay
        >> computeSatellitePhaseCenter
        >> correctObservables
```

All this processing is repeated for each epoch

```
        >> computeLinearCombinations
        >> markCSLI
        >> markCSMW
        >> computePrefitResiduals
        >> decimateData
        >> baseChange
        >> pppSolver;
```

PPP: ONSA 2005/08/12, static, forward



PPP (kinematic, forward mode)

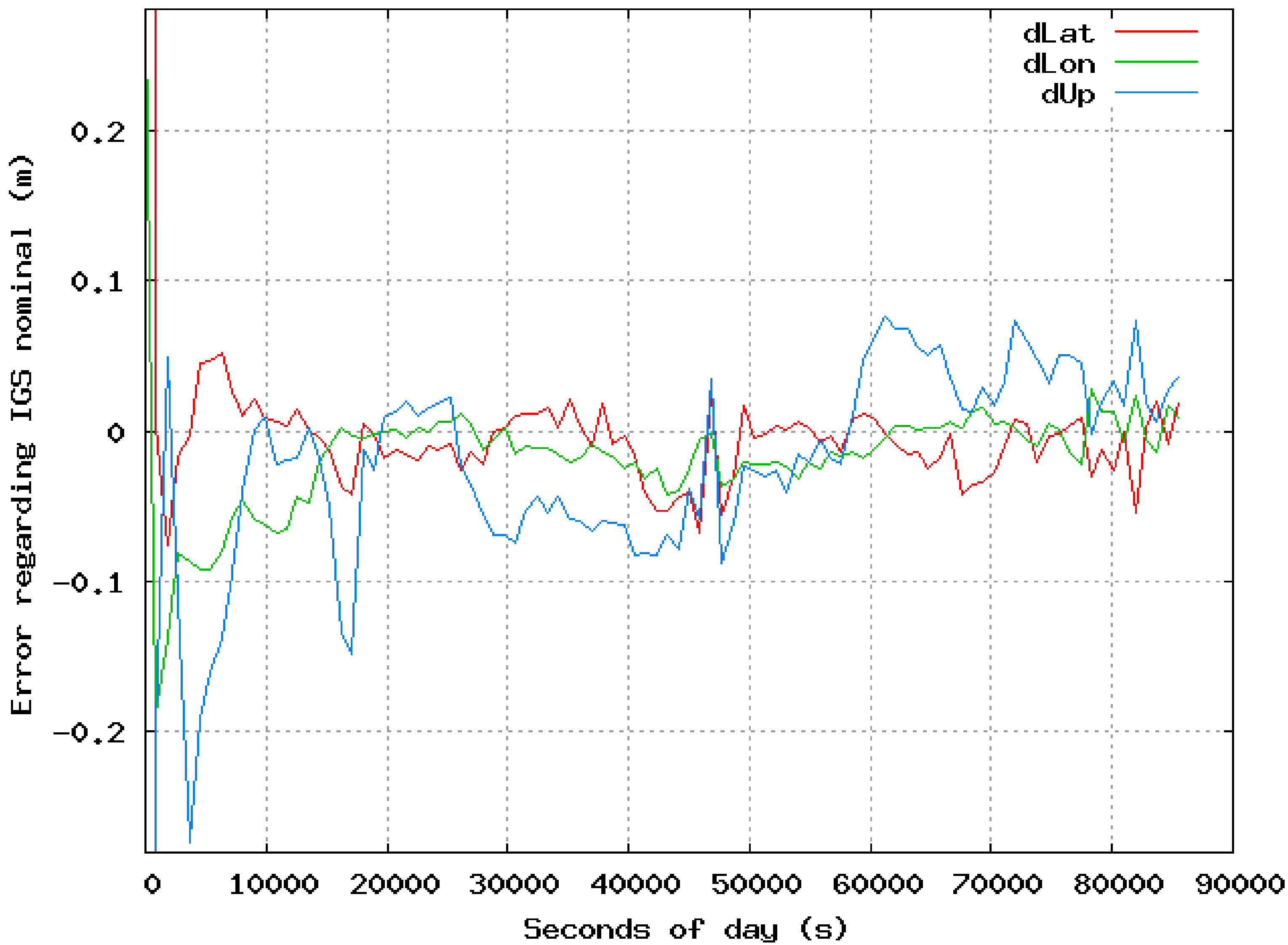
```
WhiteNoiseModel  newCoordinatesModel(100.0);

pppSolver.setCoordinatesModel(&newCoordinatesModel);

correctObservables.setExtraBiases(tides);

gpsData >> basicGPSModel
        >> removeEclipsedSatellites
        ...
        ...
        >> pppSolver;
```

PPP:ONSA 2005/08/12, kinematic, forward



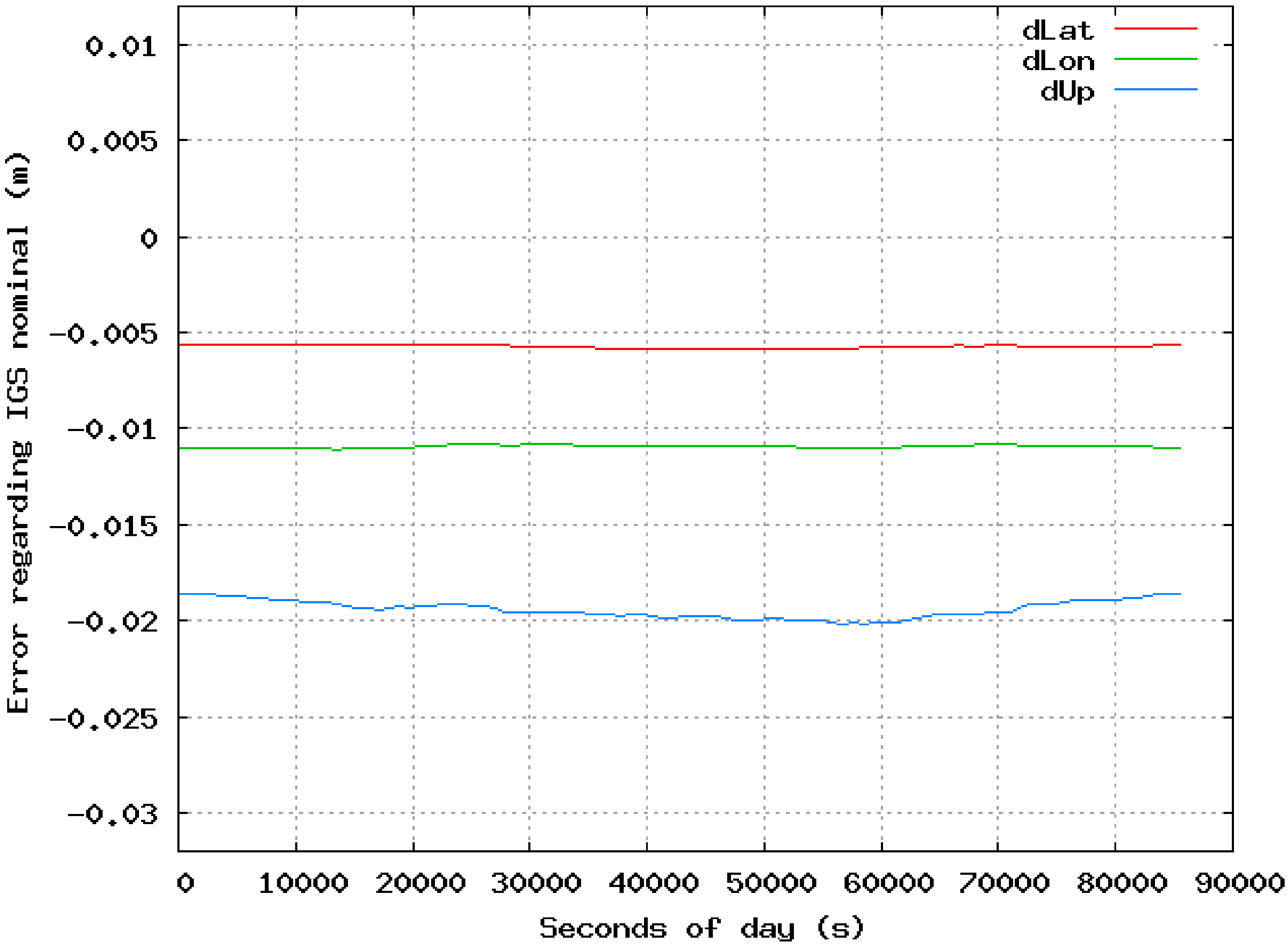
gAGE research group of Astronomy and Geomatics
Barcelona, Spain

PPP (static, forward-backward)

```
SolverPPPFB  fbpppSolver.ReProcess(4);
   ...
   correctObservables.setExtraBiases(tides);

   gpsData >> basicGPSModel
              >> removeEclipsedSatellites
              ...
              ...
              >> fbpppSolver;
   ...
while ( fbpppSolver.LastProcess(gpsData) )
{
   cout << fbpppSolver.getSolution(TypeID::dLat) << " ";
   cout << fbpppSolver.getSolution(TypeID::dLon) << " ";
   cout << fbpppSolver.getSolution(TypeID::dH) << " ";
}
```

PPP: ONSA 2005/08/12, static, fw/back



Conclusions and future work

- GPSTk is already a solid base to work upon, saving tedious work to the researcher.
- GDS are providing a powerful (yet flexible and easy to use) processing framework.
- PPP results using GDS are in agreement with other GNSS processing software.
- There are several areas for improvement:
  - RINEX version 3 handling.
  - IONEX files processing.
  - Robust outlier detection classes.
  - More sophisticated models.
  - Other GNSS processing strategies (RTK).

gAGE research group of Astronomy and Geomatics
Barcelona, Spain

Thanks for your attention and time!

Dagoberto.Jose.Salazar@upc.edu