gAGE research group of Astronomy and Geomatics
Barcelona, Spain

Open source Precise Point Positioning with GNSS Data Structures and the GPSTk

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

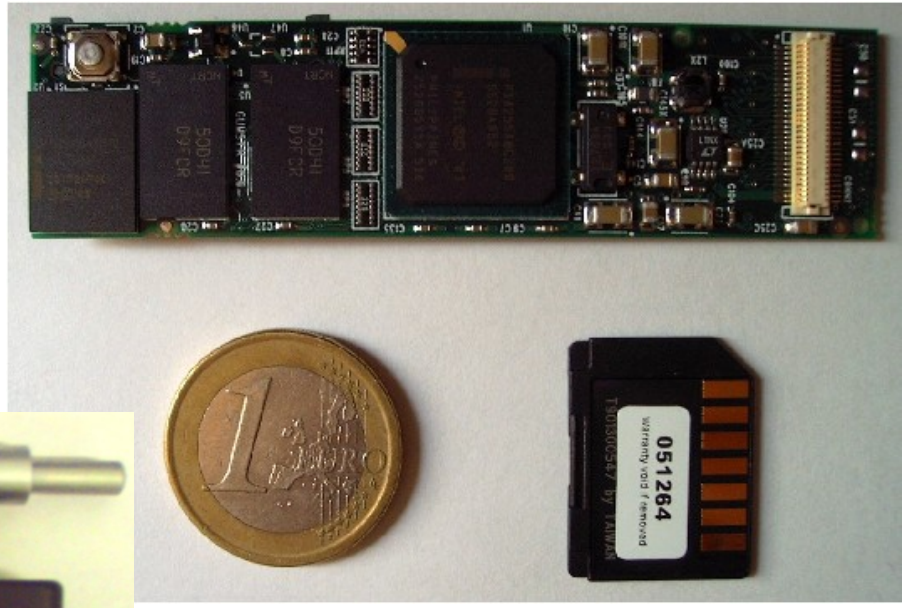
Contents

- GPSTk overview
- Current features
- GNSS Data Structures:
  - Some GDS examples
  - Implementation
  - Processing paradigm
  - Some processing examples
- Initial results for PPP
- Conclusions and future work

GPSTk overview

- Library to write GNSS data processing software, plus example applications.
- Free software (LGPL): Both commercial and non-commercial applications.
- Written in (very portable) ANSI C++.
- Initiated at ARL:UT, now it has several official developers around the world.
- Very well documented, and easy to document.
- Easy to extend and maintain.
- Website: http://www.gpstk.org

GPSTk portability



gAGE research group of Astronomy and Geomatics
Barcelona, Spain

Current features

- Time conversions
- RINEX (OBS, NAV, MET) reading/writing
- Ephemeris computation (Broadcast and SP3)
- Mathematics: Matrices, vectors, interpolation, numeric integration, LMS, W-LMS, Kalman filter.
- Code positioning, RAIM, PPP.
- Application development (exceptions, command line)
- Several models (Klobuchar, Goad-Goodman, MOPS...)
- Classes for precise modeling:
  - Receiver antenna phase centers.
  - Wind-up.
  - Solid tides, ocean tides, pole tides.
  - Neill tropospheric model.

GNSS Data Structures

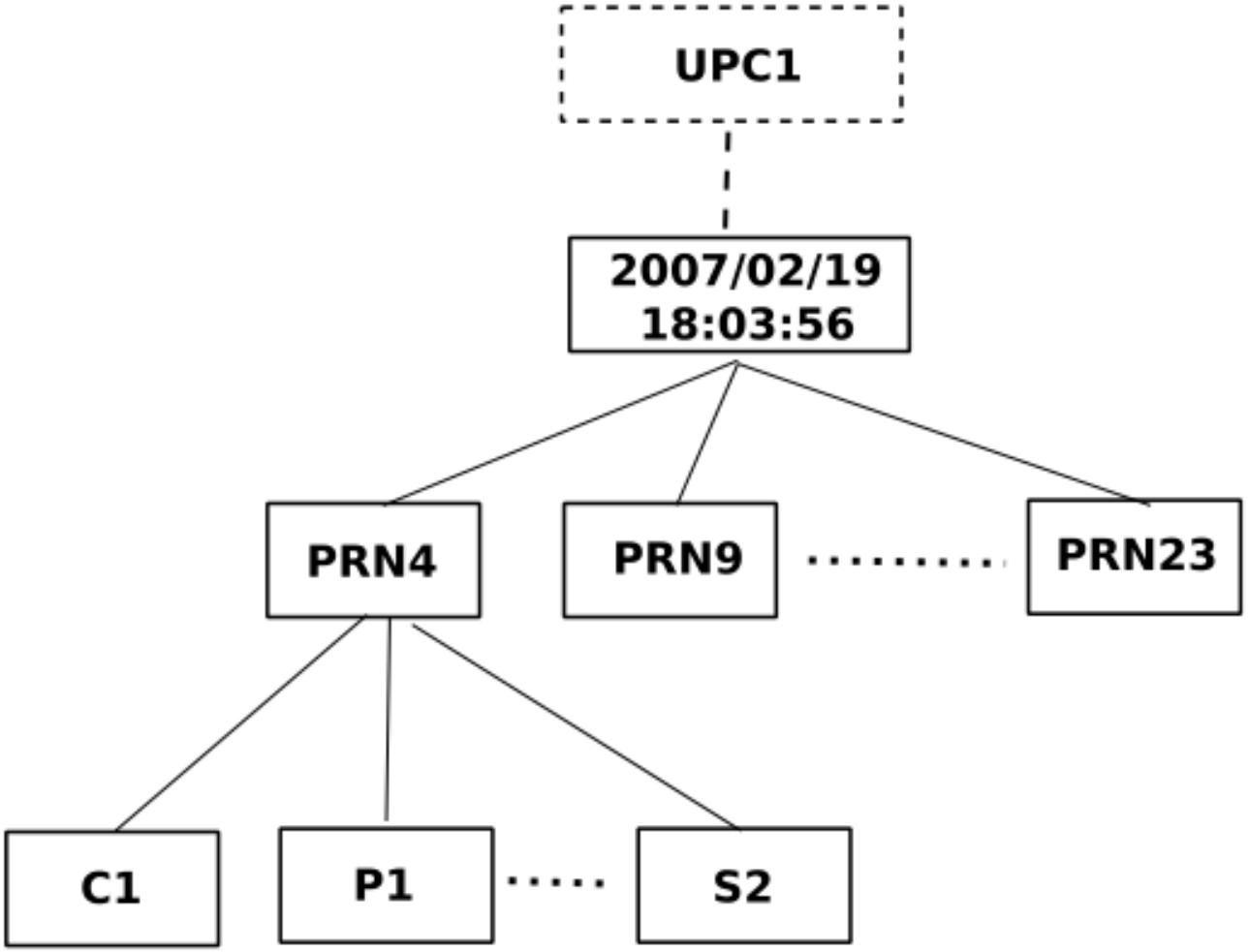
- GNSS Data Structures (GDS) address common processing problems.
- Processing adds extra data, and these data must be matched with the corresponding observations.
- Sometimes, this implies data intersection and/or union taking into account source, time, satellite ID, data type, etc.
- Also, there may be synchronization issues among different data streams.
- GNSS+INS hybrid systems bring further complexity.

GNSS Data Structures

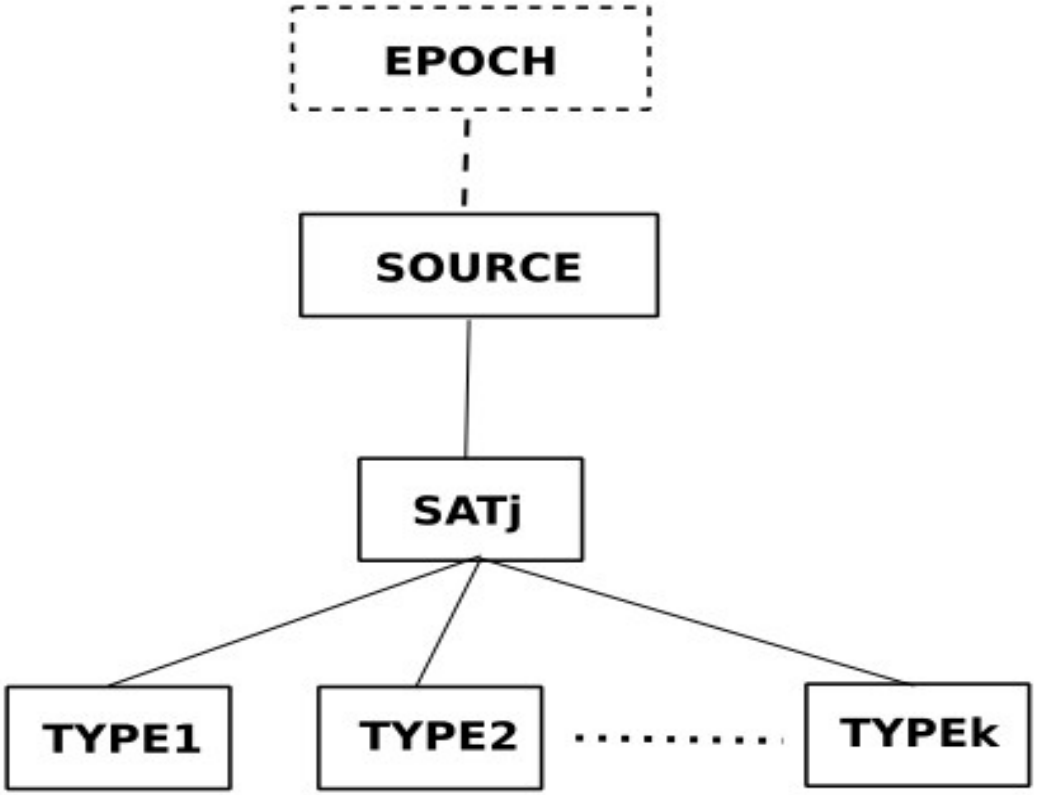
- Some sort of hierarchy of data structures was needed to address data management issues.
- These data structures should be as *light* as possible, yet *flexible* and *easy* to use.
- Reference implementation was developed using facilities provided by the C++ STL.
- The authors have been extending this concept from code-based to phase-based data processing. First results in this work.

- ***Key point:*** Saving Meta-Information along data.

Some GDS examples



Some GDS examples



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

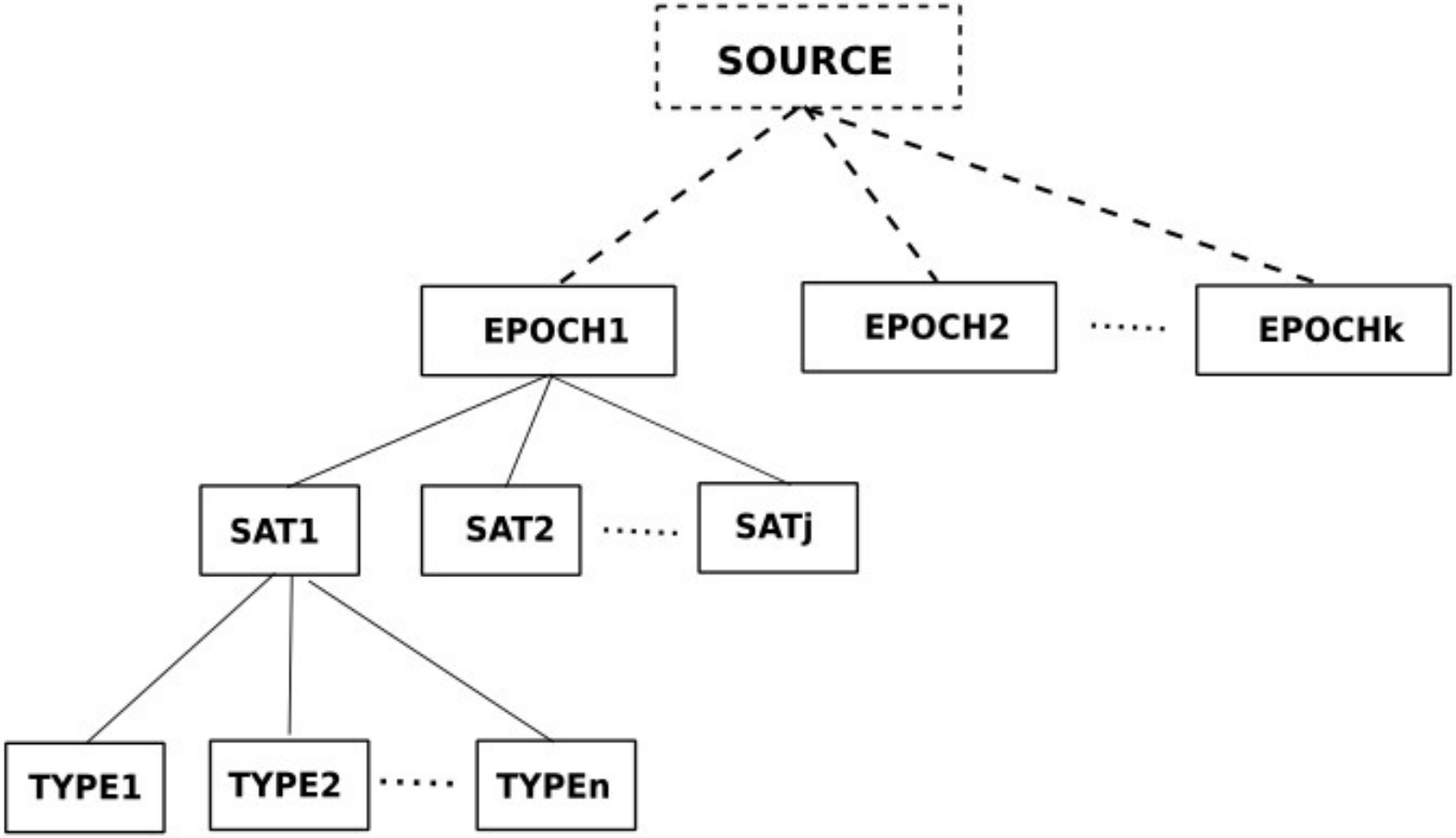
Some GDS examples

$$\begin{bmatrix} prefit^1 \\ prefit^2 \\ \vdots \\ prefit^n \end{bmatrix} = \begin{pmatrix} \frac{x_0-x^1}{\rho_0^1} & \frac{y_0-y^1}{\rho_0^1} & \frac{z_0-z^1}{\rho_0^1} & 1 \\ \frac{x_0-x^2}{\rho_0^2} & \frac{y_0-y^2}{\rho_0^2} & \frac{z_0-z^2}{\rho_0^2} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ \frac{x_0-x^n}{\rho_0^n} & \frac{y_0-y^n}{\rho_0^n} & \frac{z_0-z^n}{\rho_0^n} & 1 \end{pmatrix} \begin{bmatrix} dx \\ dy \\ dz \\ cdt \end{bmatrix}$$

$$\begin{bmatrix} \Delta prefit^1 \\ \vdots \\ \Delta prefit^n \end{bmatrix} = \begin{pmatrix} \Delta\frac{x_0-x^1}{\rho_0^j} & \Delta\frac{y_0-y^1}{\rho_0^j} & \Delta\frac{z_0-z^1}{\rho_0^j} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ \Delta\frac{x_0-x^n}{\rho_0^j} & \Delta\frac{y_0-y^n}{\rho_0^j} & \Delta\frac{z_0-z^n}{\rho_0^j} & 1 \end{pmatrix} \begin{bmatrix} dx \\ dy \\ dz \\ \Delta(cdt) \end{bmatrix}$$

gAGE research group of Astronomy and Geomatics
Barcelona, Spain

Some GDS examples



Implementation

- GDS have two main parts:
  - Header: The common information
  - Body: The specific information
- Several methods are provided to ease handling. For instance:

```
gRin.keepOnlyTypeID(TypeID::C1)
gRin.removeSatID(sat21)
gRin.body(TypeID::LI)(sat89)
```

- Operator “>>” is overloaded to make the data “flow” from one processing step to the next:

```
gRin >> modeler >> solver
```

Processing paradigm

- GNSS data processing becomes like an “assembly line”. The GDS “flow” from one “workstation” to the next.
- The GDS is treated like a “white box” holding all the information needed. It shrinks or expand automatically as needed; all data is properly indexed.
- The other “processing” objects (“ComputeLC”, for instance, computes de ionosphere-free combination) act like “workstations” in the assembly line:
  - They are pre-configured (sensible defaults)
  - They may be reconfigured

Example: C1 + LMS

```
while(rin >> gRin)
{
    gRin >> obsFilter >> model >> solver;
 }
```

rin = *RinexObsStream*: Reads RINEX files

gRin = *gnssRinex*: GNSS Data Structure

obsFilter = *SimpleFilter*: Filters out satellites with observations grossly out of bounds

model = *ModelObs*: Computes modeled (corrected) observations from satellites to a mobile receiver

solver = *SolverLMS*: Computes the Least Mean Squares Solution

Example: Smoothed-PC + WMS

```
while (rin >> gRin)
{
   gRin >> getPC >> getLC >> getLI >> getMW
        >> markCSLI >> markCSMW >> smoothPC
        >> pcFilter >> modelPC >> weightsMOPS >>
        >> baseChange >> solverWMSNEU;

}
```

getPC, getLC, etc = Classes to compute observable comb.

markCSLI, markCSMW = Cycle-slips detectors

smoothPC = *PCSmoother*: Smoothes PC with LC

weightsMOPS = *ComputeMOPSWeight*: Relative weights

baseChange = *XYZ2NEU*: Changes reference frame

solverWMSNEU = *SolverWMS*: WMS in North-East-Up

gAGE research group of Astronomy and Geomatics
Barcelona, Spain

Example: Simple PPP

```
while (rin >> gRin)
{
   gRin >> basicMod  >> correctObj >> windup
        >> computeTropo >> linearComb >> markCSLI
        >> markCSMW >> pcFilter >> mopsWeights
        >> baseChange >> pppSolver;
 }
```

basicMod = *BasicModel*: Basic parts of GNSS model

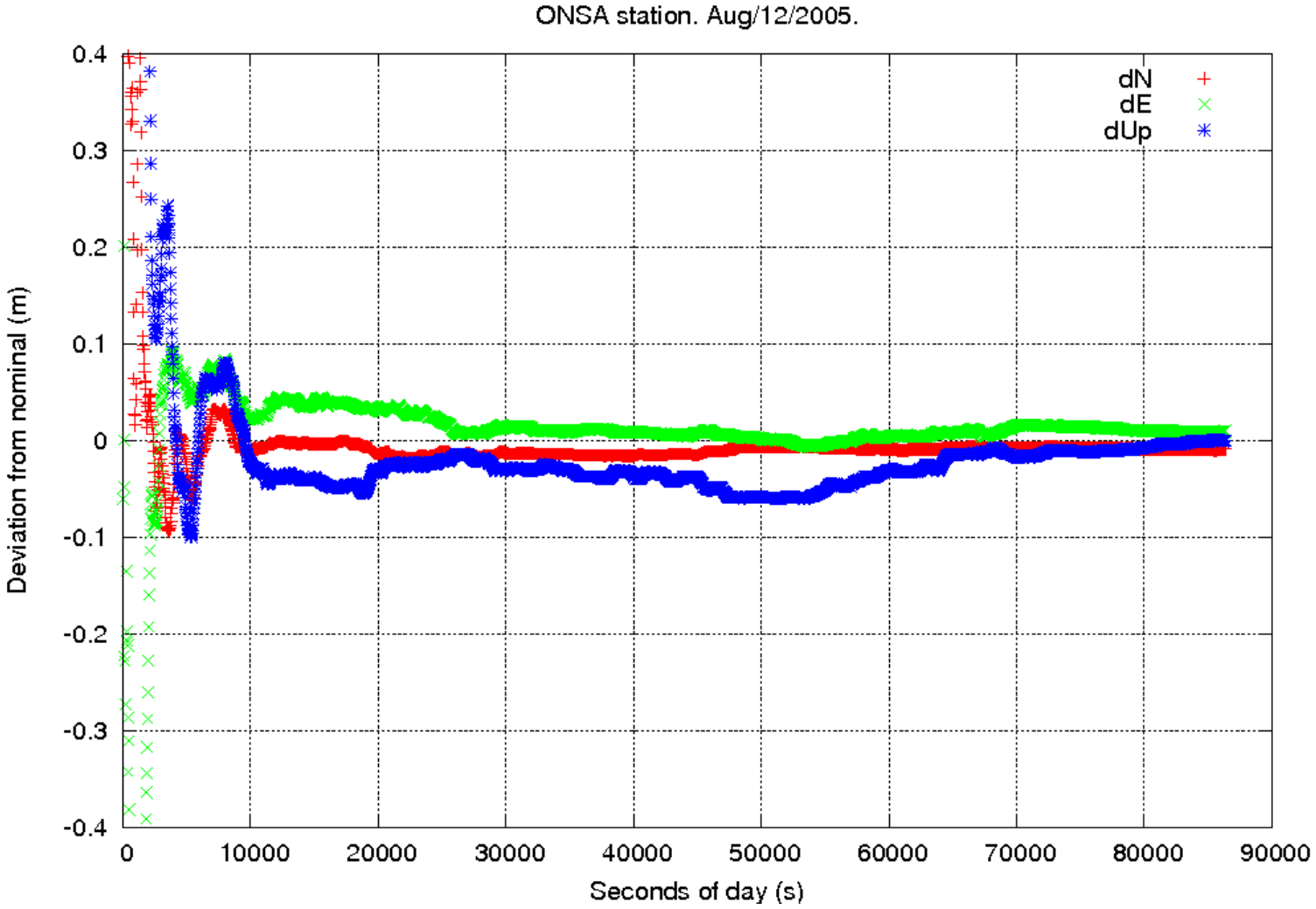
correctObj = *CorrectObservables*: tides, phase centers, etc.

windup = *ComputeWindUp*: Wind-up effect

computeTropo = *ComputeTropModel*: Neill trop. model

pppSolver = *SolverPPP*: Implements PPP Kalman filter

Initial results for PPP



Module of final error vector: 0.013 m

gAGE research group of Astronomy and Geomatics
Barcelona, Spain

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 very flexible and easy to use, processing framework.
- Initial PPP results using GDS are promising.
- There are several areas for improvement:
  - Adapting forward-backward Kalman filter.
  - PPP for networks of stations.
  - Robust outlier detection classes.
  - More sophisticated tide models.
  - Other GNSS processing strategies (RTK)

Thanks for your attention and time!