# THE GPS TOOLKIT: WORLD CLASS OPEN SOURCE SOFTWARE TOOLS FOR THE GNSS RESEARCH COMMUNITY

Salazar, D., Hernandez-Pajares, M., Juan-Zornoza, M., and Sanz-Subirana, J.
Grupo de Astronomia y Geomatica (gAGE)
Universitat Politecnica de Catalunya
08034 Barcelona
Spain
Dagoberto.Jose.Salazar@upc.edu

**Keywords:** GNSS data processing software GPSTk.

**Abstract**

The "GPS Toolkit" (GPSTk) project is an advanced GNSS Open Source Software (GOSS) suite initiated by the Applied Research Laboratories of the University of Texas (ARL:UT), aiming to provide a world class GNSS library computing suite to the satellite navigation community.

The objective of this work is to show how the GPSTk suite may be used to rapidly and easily develop, implement and test advanced GNSS data processing applications in a flexible way. The main characteristics and modules of the GPSTk are also presented, as well as the current trends in its development.

Results from the GNSS data processing carried out with the examples introduced in this work are shown, as well as a brief comparison with an established GPS software package such as BRUS.

## 1 Introduction

The "GPS Toolkit" (GPSTk) project is an advanced GNSS Open Source Software (GOSS) suite initiated by the Applied Research Laboratories of the University of Texas (ARL:UT), aiming to provide a world class GNSS library computing suite to the satellite navigation community.

In this regard, one of the main goals of the GPSTk is to free the research community from implementing common GNSS algorithms, providing a publicly accessible software repository where those algorithms may be found and freely used.

The initial code of the GPSTk was released in summer 2004 and presented at the ION-GNSS-2004 [1], and its functionality has been continuously improving. A very brief list of the tools provided by the GPSTk includes: Handling of observation data and ephemeris in RINEX and SP3 formats, mathematical, statistical, Matrix and Vector algorithms, time handling and conversions, ionospheric and tropospheric models, cycle slip detection and correction, etc.

As a typical Open Source project, the GPSTk is released under the GNU LGPL license, allowing freedom to develop both commercial and non-commercial software based on it, and it is actively maintained by a dozen developers around the world using the Internet as communication medium. Development facilities are provided by the popular SourceForge open source application repository, and a website of the project may be found in http://gpstk.sourceforge.net.

The GPSTk software suite consists of a core library and extra applications. The library provides several functions that solve processing problems associated with GNSS (for instance, using RINEX files) and is the basis for more advanced applications distributed as part of the GPSTk suite.

Apart from the library, the GPSTk suite comes with a wealth of GNSS applications ready to run, explore and include in new developments. Full applications may be found in the "gpstk/dev/apps" subdirectory, and interesting and easy-to-follow examples of use are located at "gpstk/dev/examples".

A very important feature of a project of this nature is documentation. In this regard, the GPSTk is

profusely documented using the Doxygen documentation system and it is heavily based on object-oriented programming principles, ensuring a modular, extensible and maintainable code.

### *1.1 GPSTk portability*

One of the important characteristics of the GPSTk is its wide portability. The GPSTk is a highly platform-independent software code base thanks to the use of the ANSI C++ programming language.

In this regard, it is reported to run on the following operative systems:

- Unix family: Linux, Solaris, AIX, etc.
- Windows.
- Mac OS X.

Also, it may be compiled using several versions of free and commercial compilers, both in 32 bits and 64 bits PC platforms.

Besides, it is even reported to run in such disparate platforms as the Nokia 770 Internet Tablet (see Figure 1) and the Gumstix line of full function miniature computers (see Figure 2), weighting just 8 g, where the author (an active GPSTk developer) has successfully accomplished GNSS data processing using the GPSTk [2].


Figure 1: Nokia 770 Internet Tablet plotting GPS data

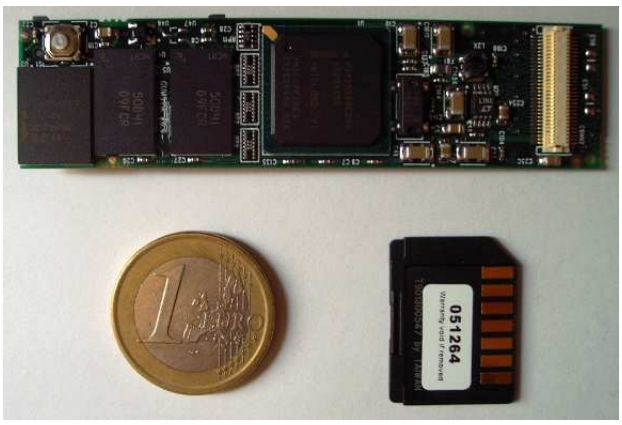


Figure 2: Gumstix Full Function Miniature Computer

## 2 GPSTk functionality

The GPSTk library provides several different modules grouping the classes by common functions. Table 1 summarizes some of the the most important and used classes. Please note that currently the data processing is limited to pseudorange processing.

## 3 Writing GPSTk-based applications

One of the great advantages of the GPSTk is its flexibility and ease of use. In this section, such flexibility will be emphasized by taking an example application that comes included in the GPSTk and adapting it, with very little effort, to do several kinds of GNSS pseudorange processing.

As previously stated, we will take an example ("example5.cpp") and will quickly modify it to get several different GNSS pseudorange processing strategies. Three new examples will then be developed:

- **example-a.cpp**: This program will process C1 pseudoranges applying an standard ionospheric model (Klobuchar model, using the ionospheric coefficients broadcasted in the GPS navigation message, as described in [3]), a simple tropospheric model (described in [4] and [5]), and the solution will be found using a standard Least-Mean-Squares algorithm.

- **example-b.cpp**: An improvement with respect to the former one, it will process C1 pseudoranges and apply Klobuchar ionospheric model, but the tropospheric model will be the one described in the RTCA/DO-229C document (called the "Minimum Operational Performance Standards" or "MOPS"). The solution will be found using the Weighted-Least-Mean-Squares algorithm also described in the aforementioned document.

  It is worth noting that the MOPS algorithms are used in SBAS systems receivers such as EGNOS ones, but this example will not apply further EGNOS corrections because they are not currently implemented in GPSTk.

- **example-c.cpp**: Very similar to "example-b.cpp", but in this case the ionospheric-free combination pseudoranges (PC) are used, and therefore the ionospheric model is dropped. Tropospheric and solver algorithms remain the same (MOPS).

All these programs will read the observables and ephemeris data from corresponding RINEX files. Broadcast ephemeris will be used (although simple changes allow for using SP3 ephemeris, described in [6]) and the output will be epoch (in seconds of day) and deviations in latitude, longitude and height (in meters) from a nominal position. The full source code for these examples may be found at [7].

In order to show the flexibility and ease of use of the GPSTk, in the following lines you will find the code added to "example5.cpp" in order to allow "example-a.cpp", "example-b.cpp" and "example-c.cpp" to output the data in the format explained above:

```
// Object holding nominal position for EBRE station
Position nominalPos(4833520.2269, 41537.00768, 4147461.489);

Position diffPos;    // Object to hold difference in position

// Get the difference between epoch solution and nominal position
diffPos = solPos - nominalPos;

// Azimuth of solution with respect to nominal
double azimuth = nominalPos.azimuthGeodetic(solPos);

// Elevation of solution with respect to nominal
double elev = nominalPos.elevationGeodetic(solPos);

double magnitude = RSS(diffPos.X(), diffPos.Y(), diffPos.Z()  // Compute RSS

// Print results
```

| FUNCTION | CLASS NAME | REMARKS |
|---|---|---|
| Time handling | ANSITime | "Seconds since Unix epoch" representation |
| | CivilTime | Common year/month/day/hour/min/sec time representation |
| | DayTime | Time representation for all common formats, including GPS |
| Formatted I/O | FFStream | Formatted File Stream |
| | FFData | Formatted File Data |
| | RinexObsStream | I/O on RINEX Observation files |
| | RinexNavStream | I/O on RINEX Navigation files |
| | RinexMetStream | I/O on RINEX Meteorological files |
| | SP3Stream | I/O on SP3 files |
| Atmosphere models | IonoModel | Klobuchar ionospheric model |
| | SimpleTropModel | Simple Black tropospheric model |
| | SaasTropModel | Saastamoinen tropospheric model |
| | NBTropModel | New Brunswick tropospheric model |
| | GGTropModel | Goad and Goodman tropospheric model |
| | MOPSTropModel | Minimum Operational Performance Standards tropospheric model |
| Ephemeris | EngAlmanac | Almanac information for the GPS constellation |
| | EngEphemeris | Ephemeris information for a single satellite |
| | RinexEphemerisStore | Interface to read Rinex Navigation data |
| | SP3EphemerisStore | Interface to read SP3 Navigation data |
| Solution algorithms | Bancroft | Algebraic algorithm to get an initial guess of GPS receiver's position |
| | ModeledPR | Compute modeled pseudoranges from satellites to a given receiver |
| | PRSolution | Compute a position and time solution using RAIM concepts |
| | DOP | Encapsulates the computation of Dilution Of Precision |
| | SimpleIURAWeight | Assigns weights to satellites based on their URA Index (IURA) |
| | MOPSWeight | Assigns weights to satellites based on the Appendix J of MOPS C |
| Mathematical tools | Vector | Mathematical vector representation |
| | Matrix | Mathematical matrix representation |
| | SVD | Singular value decomposition of a matrix |
| | LUDecomp | Performs the lower/upper triangular decomposition of a matrix |
| | Cholesky | Computes the Cholesky decomposition of the given matrix |
| | PolyFit | Computes a polynomial fit |
| | RungeKutta4 | Provides a collection of integration routines |
| | Stats | Conventional statistics for one sample |
| | TwoSampleStats | Conventional statistics for two samples |
| | Expression | Solves general mathematical expressions at run time |
| | SolverLMS | Computes the Least Mean Squares Solution of a given equations set |
| | SolverWMS | Computes the Weighted Least Mean Squares Solution of given set |
| Coordinates | ECEF | Earth centered, Earth fixed geodetic coordinates in meters |
| | Position | Common 3D geographic position formats representation |
| | Triple | Three-dimensional vectors |
| | Xvt | Earth centered, Earth fixed position/velocity/clock representation |
| | GPSGeoid | Geodetic model defined in ICD-GPS-200 |
| | WGS84Geoid | Geodetic model defined in NIMA TR8350.2 |
| Miscellanea | CommandOption | Set of several command line options |
| | BasicFramework | Basic framework for programs in the GPS Toolkit |
| | Exception | Base class for all exception objects |
| | FileFilter | Sorts and filters file data |
| | FileHunter | Finds files matching specified criteria |

Table 1: Some important GPSTk classes

```
cout << rData.time.DOYsecond()  << "   ";   // Epoch in seconds of day
// Longitude change
cout << magnitude*sin(azimuth*DEG_TO_RAD)*cos(elev*DEG_TO_RAD) << "   ";
// Latitude change
cout << magnitude*cos(azimuth*DEG_TO_RAD)*cos(elev*DEG_TO_RAD) << "   ";
cout << magnitude*sin(elev*DEG_TO_RAD) << "   ";      // Altitude change
```

GPSTk's objects encapsulate much of the functionality needed to carry out common tasks in GNSS data processing. Several details are worth noting regarding the previous sample code:

- A *Position* object named "nominalPos" is declared in order to hold the nominal position, which in this case is also set simultaneously in ECEF coordinates.
- Another *Position* object named "diffPos" is also declared, in order to hold the difference between the "solution position" (solPos) and the nominal one.
- Note how the substraction ("-") operator is overloaded in the "Position" objects in order to allow them to be easily handled (diffPos = solPos - nominalPos).
- *Position* objects encapsulate several handy methods to operate on them. For instance, the relative azimuth and elevation between the nominal position and the solution position are founded using the methods azimuthGeodetic() and elevationGeodetic(), respectively.
- When printing the results, please pay attention to the method used in the *DayTime* object *rData.time* in order to easily get the proper output format: DOYsecond().

As could be seen, in a very short code sample several very handy GPSTk characteristics were easily and effectively used in order to get the job done. There are plenty of features like these in the GPSTk.

Other small modifications remain in order to allow the original example "example-5.cpp" to fulfill the requisites stated above for "example-a.cpp":

- Declare an object "gcatTM" of class "GCATTropModel" (tropospheric modelling).
- Declare object "solver" belonging to class "SolverLMS" (standard Least-Mean-Squares algorithm).

In the other hand, the main modifications to convert "example-a.cpp" into "example-b.cpp" include:

- Tropospheric modelling will be carried out by object "mopsTM" of class "MOPSTropModel.
- "solver" object now must belong to class "SolverWMS" (Weighted-Least-Mean-Squares algorithm).
- A new object, "mopsWeights" belonging to "MOPSWeight" class, must be added to compute weights.

Some minor details remain (for example, "mopsTM" needs to be initialized and "solver" invocation must include a vector of weights supplied by "mopsWeights.weightsVector"), but it is very easy to change from a processing strategy to another thanks to the GPSTk encapsulation of important algorithms.

Finally, the change from "example-b.cpp" to "example-c.cpp" involves mainly three simple modifications:

- Object obsC1, originally belonging to class "ExtractC1", must now be declared as belonging to "ExtractPC".
- Given that the Total Group Delay is not applicable when using PC observables, the object in charge of modeling the pseudoranges ("modelPR", of class " ModeledPR") will be configured to ignore the TGD (modelPR.useTGD = false).
- All references to Klobuchar ionospheric modelling are now useless and should be deleted.

In summary, just some small changes were all what was needed in order to quickly implement several different GNSS data processing strategies, taking as starting point an example provided by the GPSTk.

The results produced by these example programs are shown in the following figures. Figure 3 plots the vertical error regarding the nominal position as a function of time for EBRE station, January 30th, 2002. Figure 4 shows the horizontal error for the same station and day, with the corresponding RMS values for each strategy. The results are consistent with what is expected from these processing strategies.

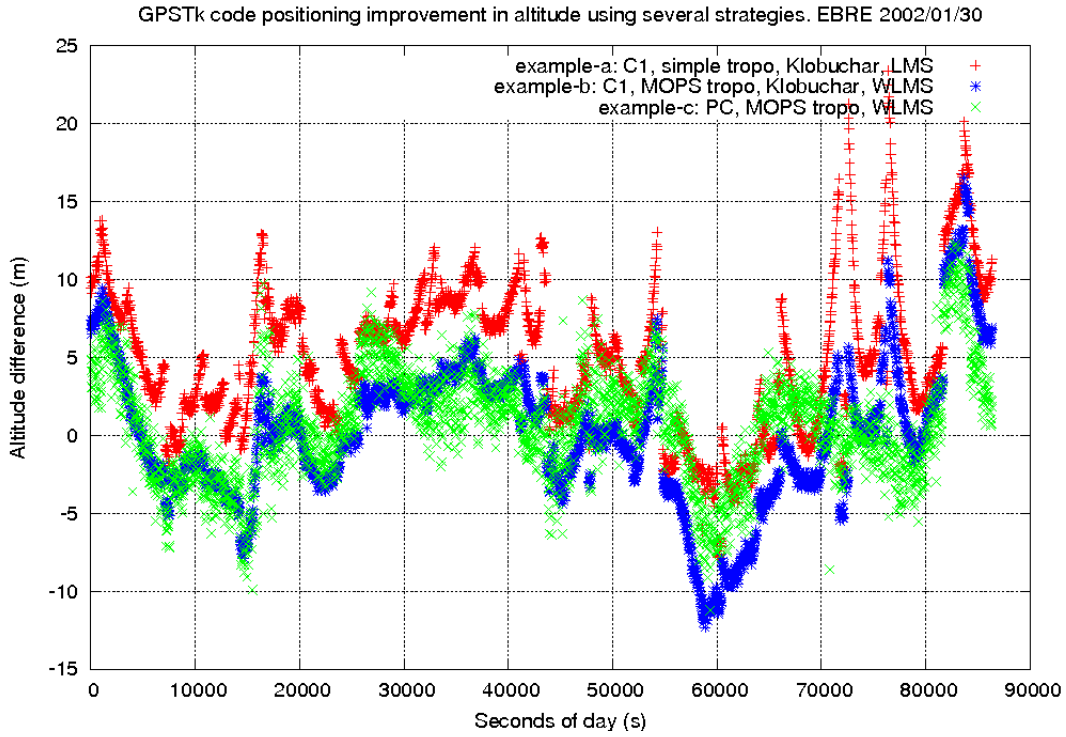


Figure 3: Vertical error for different processing strategies. EBRE 2002/01/30.

## 4 Validation with BRUS at positioning and range domains

For validation purposes, in this section the data results from one of the aforementioned example programs (example-b) will be compared with the results yielded by the BRUS software package.

The "Basic Research Utilities for SBAS" or *BRUS* [8] is a software package developed by gAGE/UPC and designed to be compliant with RTCA/MOPS Do 229A/B/C [9]. BRUS has been used since January 2002 in the ESTB Data Collection and Evaluation project of EUROCONTROL to process and analyze weekly data sets. Hence, it is a tested software suitable for comparison purposes.

BRUS in composed of three different parts: B2AConv (Binary to ASCII GPS measurements converter), BNAV (BRUS-NAVIGATOR) and BNAL (BRUS-NAVIGATION-ANALYZER). In particular, the part relevant to our comparison is the navigation module BNAV. Version 3.2.1 was used.

Given that BRUS/BNAV is able to process all the EGNOS messages, and it does it in many different configurations, and in the other hand the GPSTk does not have yet implemented the modules regarding EGNOS messages, then BNAL was configured to ignore EGNOS messages and only implement the MOPS standards regarding modeling.

### *4.1 Validation at positioning domain*

In order to compare the results, a first approach has been to compute the vertical and horizontal positioning errors for two different fixed GPS stations at different times:

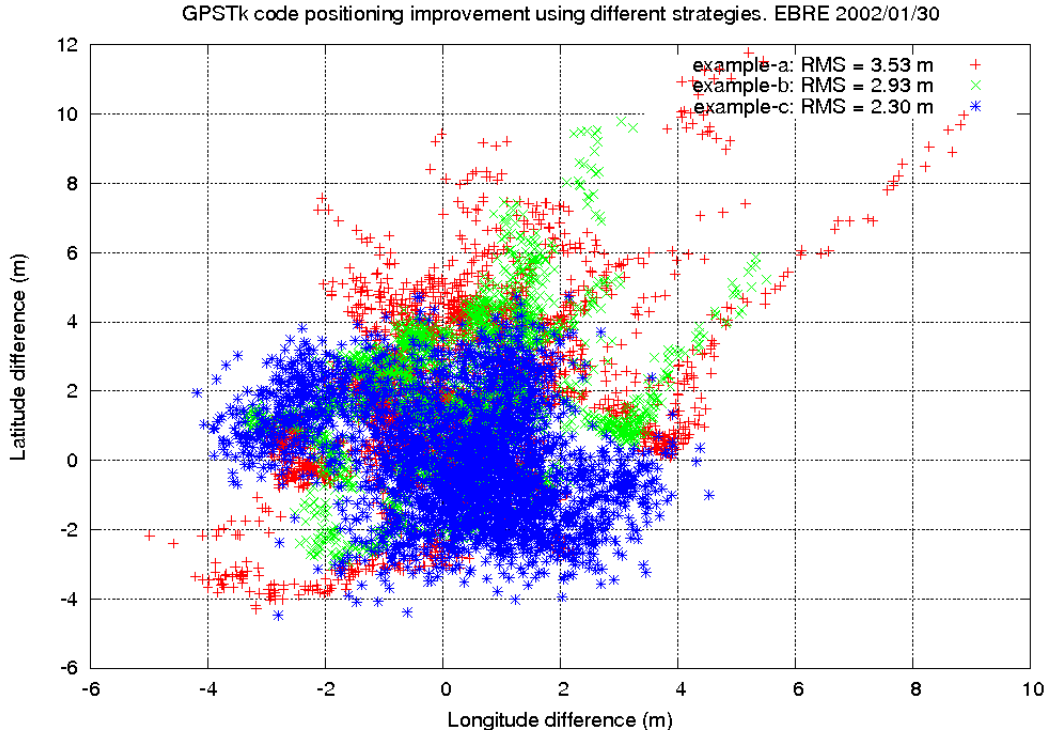


Figure 4: Horizontal error for different processing strategies. EBRE 2002/01/30.

- Station EBRE (middle latitude) at 2002/01/30.
- Station COCO (ecuatorial latitude) at 2000/07/26.

In Figure 5 can be seen how the GPSTk-based program match very well with BRUS, typically better than several centimeters, for the vertical error at EBRE. The main differences, specially at the end of the data set, are due to satellites being dropped by "example-b" before being dropped by BRUS.

On the other hand, Figure 6 shows a similar agreement in horizontal error at EBRE. The same can be seen for vertical (Figure 7) and horizontal (Figure 8) errors for COCO station.

### *4.2 Validation at range domain*

Another approach to compare the results has been to plot the modeling of some important parameters for a given station, time span and satellite. In this case the modeling data for ionosphere effects and Prefit Residuals were generated for EBRE station and satellite PRN #14 at 2002/01/30. Also, the differences between GPSTk and BRUS/BNAV modeling in both cases (ionosphere and prefits) are also showed.

As can be seen in the plots in Figure 9 and Figure 10, the modeling is very close (within 1 mm) in both software suites for the slant Klobuchar ionospheric modeling.

The comparison of Prefit Residuals is important because they combine the information of all modeling algorithms. In this case, Figure 11 and Figure 12 show that the agreement is remarkably good: Differences between Prefits Residuals are within 1 mm, confirming that the applied algorithms are equivalent.

## 5 Current trends of development

Several lines of work are being currently pursued by the GPSTk developers around the world. Support for new formats such as RINEX 3.0 and BINEX is under way, as well as a rewrite of the support for time records and conversions [10].

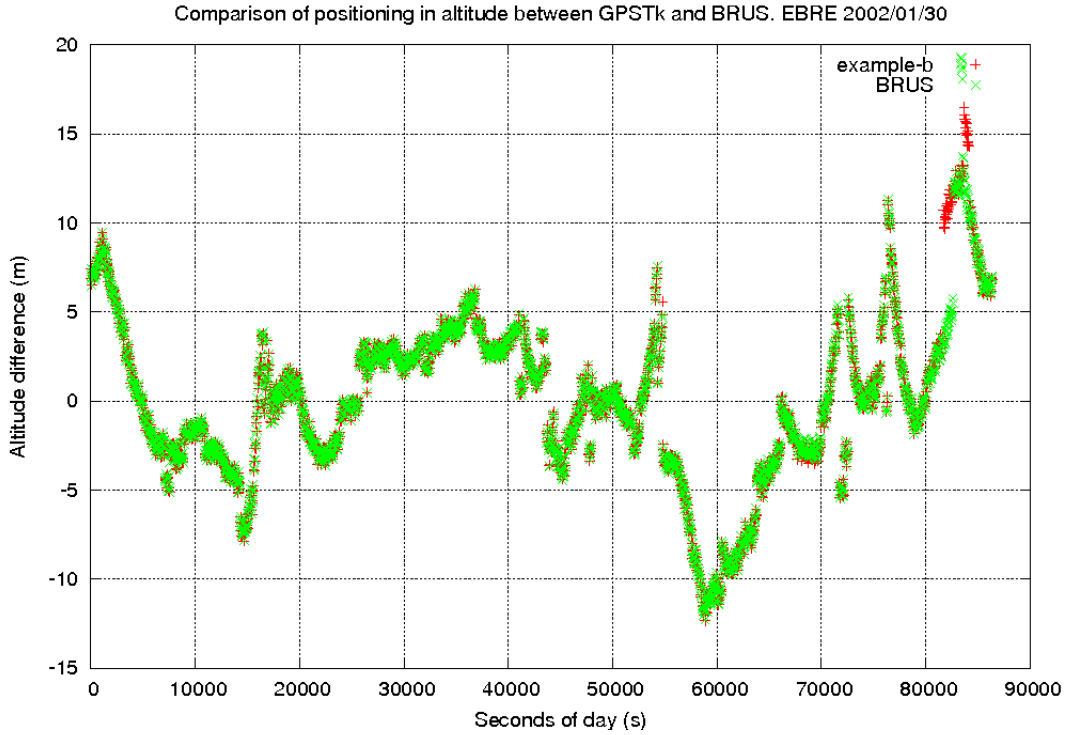


Figure 5: Comparison of vertical error between example-b and BRUS. EBRE 2002/01/30.

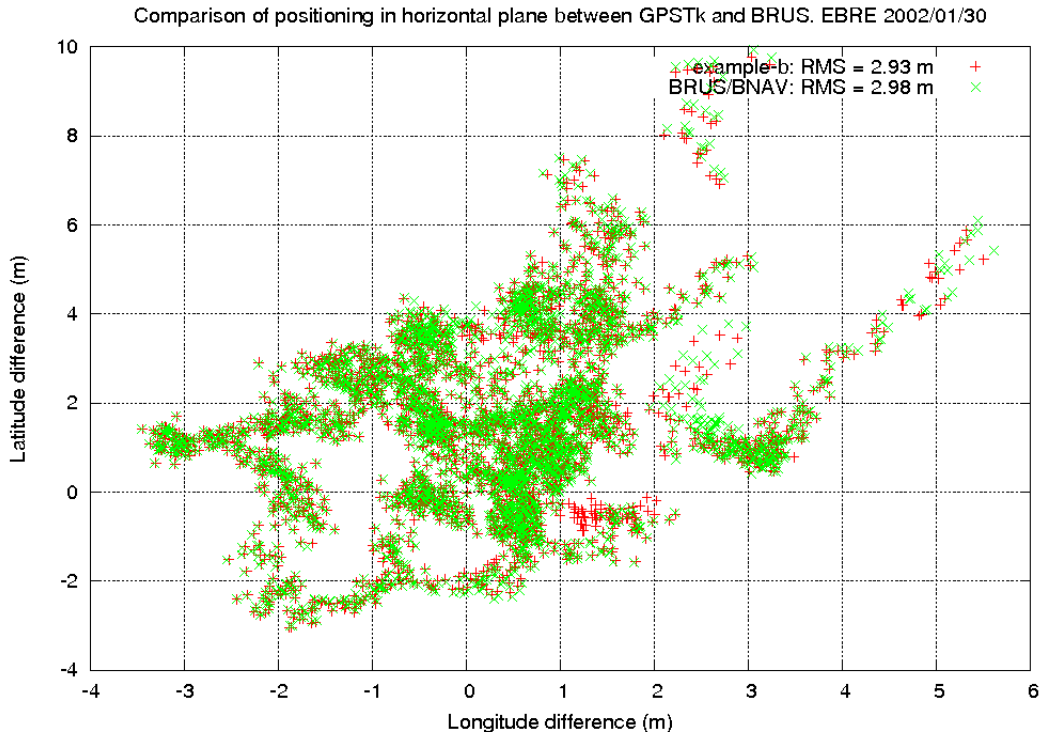


Figure 6: Comparison of horizontal error between example-b and BRUS. EBRE 2002/01/30.

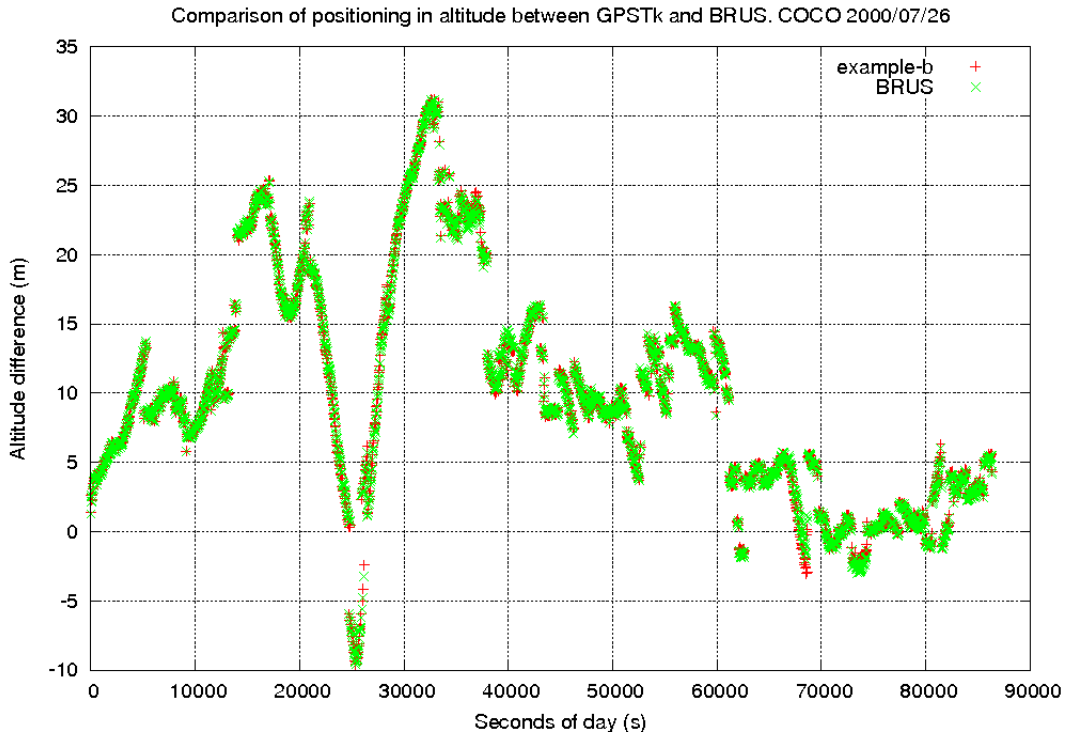


Figure 7: Comparison of vertical error between example-b and BRUS. COCO 2000/07/26.

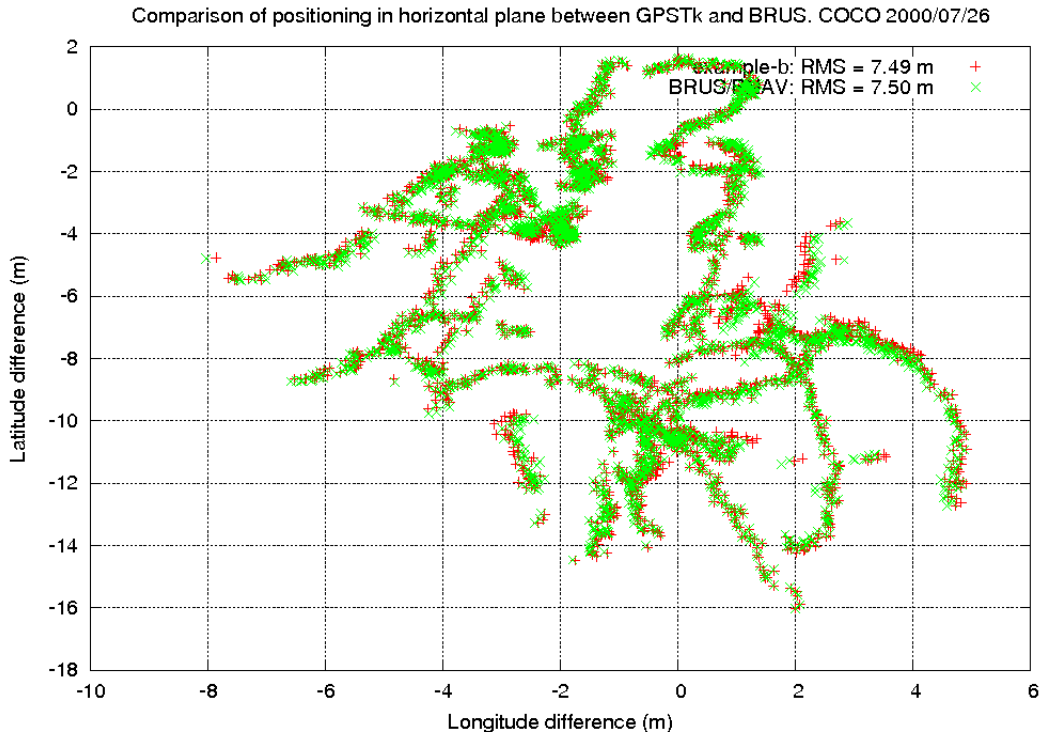


Figure 8: Comparison of horizontal error between example-b and BRUS. COCO 2000/07/26.

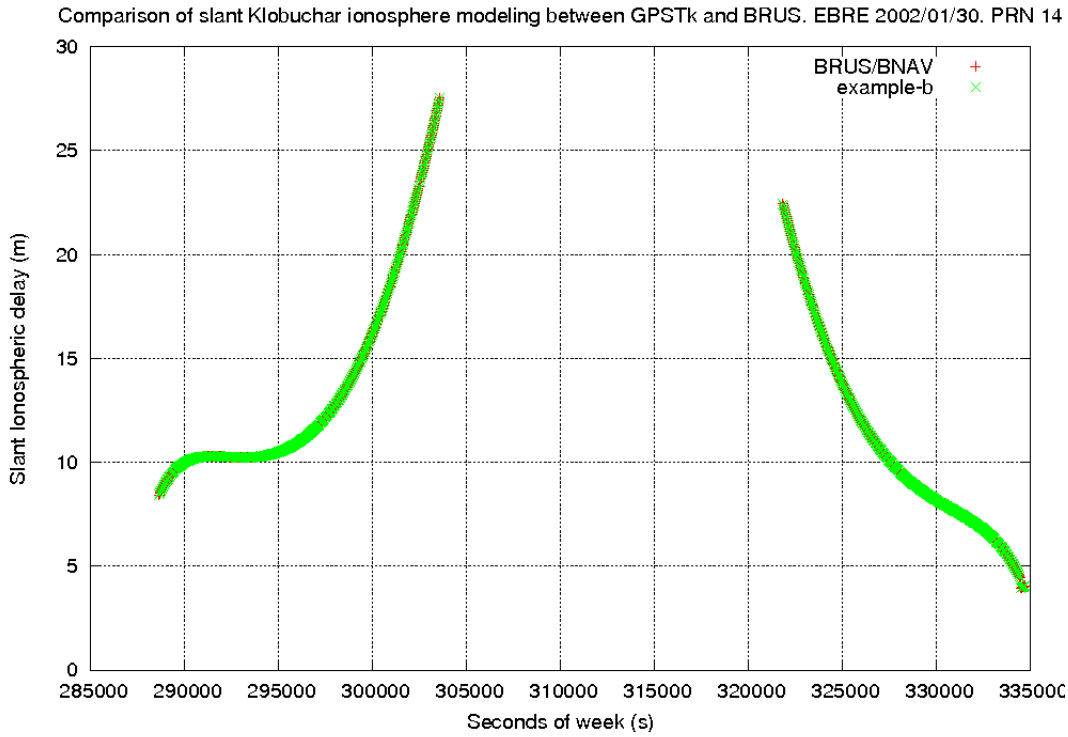


Figure 9: Comparison of slant Klobuchar ionospheric modeling between example-b and BRUS. EBRE 2002/01/30. PRN #14.

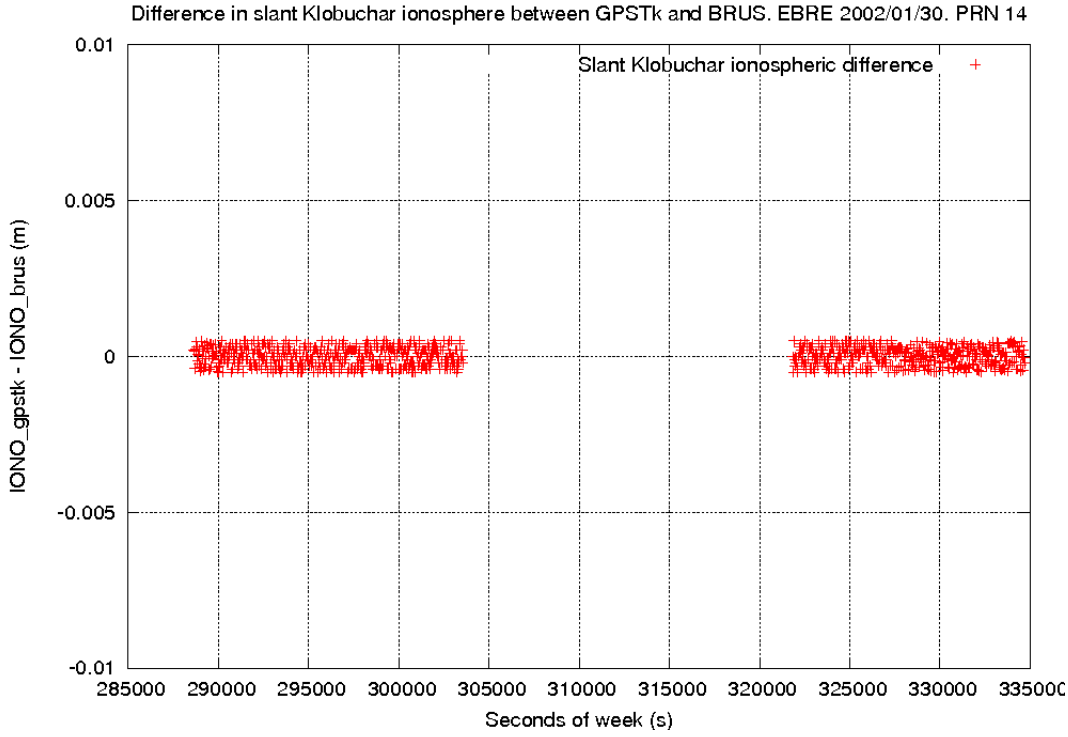


Figure 10: Difference in slant Klobuchar ionospheric modeling between example-b and BRUS. EBRE 2002/01/30. PRN #14.

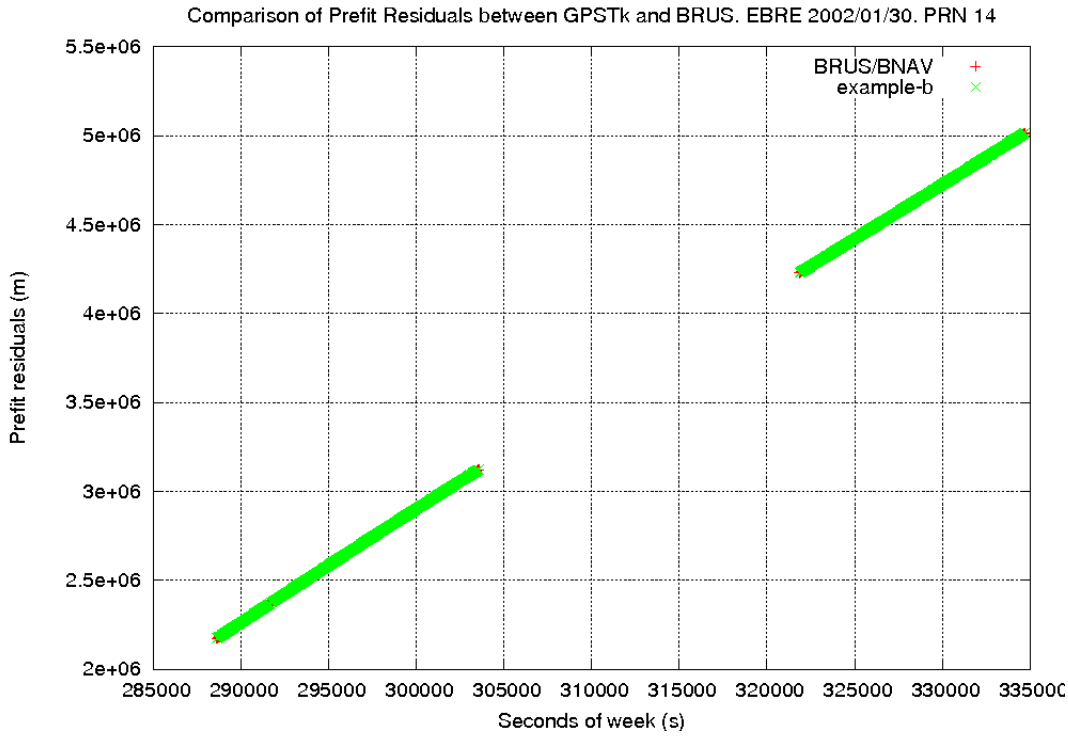


Figure 11: Comparison of Prefit Residuals modeling between example-b and BRUS. EBRE 2002/01/30. PRN #14.

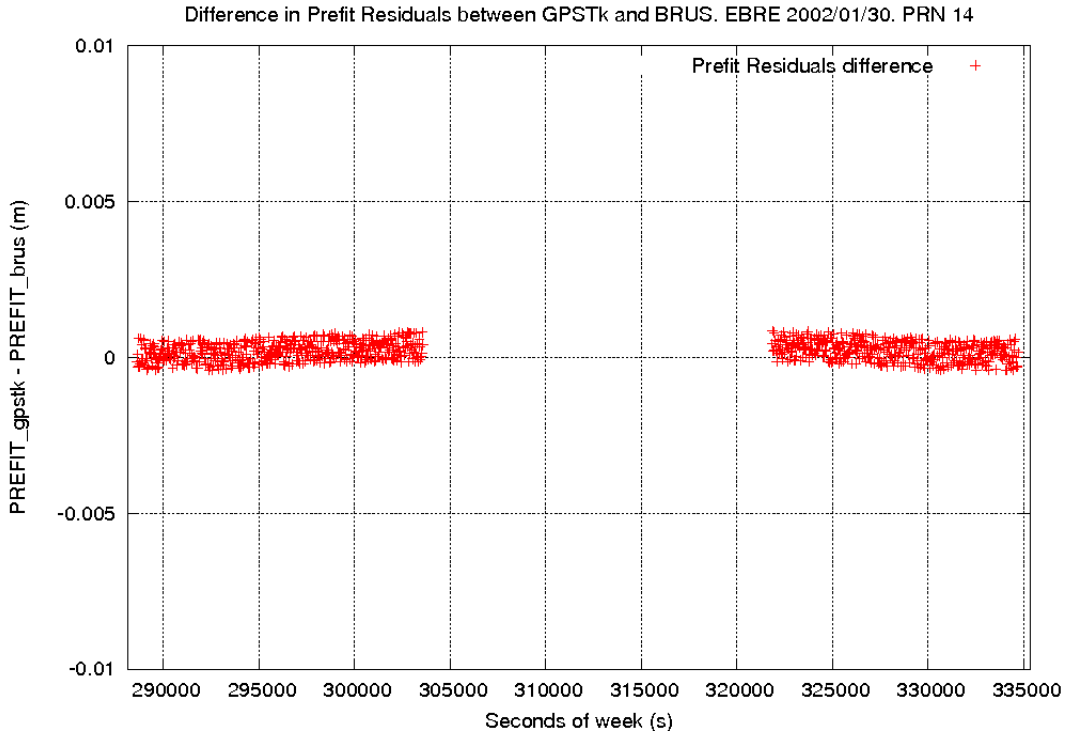


Figure 12: Difference in Prefit Residuals between example-b and BRUS. EBRE 2002/01/30. PRN #14.

Besides, a unified way to handle GNSS processing data is currently under discussion in the developers mail list. Also, work (by the first author of this manuscript) to add support for phase processing and associated models (aiming to achieve Precise Point Positioning) is expected to start at January 2007.

## 6 Conclusions

In this work, the main characteristics and modules of the GPSTk have been presented, as well as the current lines of work on its development.

Also, several examples of GNSS data processing were easily and quickly set up taking as starting point an example code provided by the GPSTk itself. This has showed how to implement and test GNSS data processing applications in a very flexible way.

Finally, the validation with BRUS/BNAV has also been presented, showing an excellent agreement both in the positioning domain (vertical and horizontal components of error for a couple stations at different latitudes and epochs) and in the modeling data. This confirms the viability of the GPSTk as a source code base for developing reliable GNSS data processing software.

## References


[1] B. Tolman et al., 2004. The GPS Toolkit – Open Source GPS Software. Proceedings 17th International Technical Meeting of the Satellite Division of the ION (ION GNSS 2004). Long Beach, California.

[2] D. Salazar, M. Hernandez-Pajares, J.M. Juan and J. Sanz. 2006 Rapid Open Source GPS software development for modern embedded systems: Using the GPSTk with the Gumstix. 3rd. ESA Workshop on Satellite Navigation User Equipment Technologies NAVITEC '2006. Noordwijk, The Netherlands.

[3] ARINC Research Corp. 2000 Navstar GPS Space Segment / Navigation User Interfaces (ICD-GPS-200).

[4] H. D. Black, A. Eisner. 1984. Correcting Satellite Doppler Data for Tropospheric Effects. Journal of Geophysical Research. Vol 89.

[5] M. Hernandez-Pajares, J.M. Juan and J. Sanz. 2001 GPS Data Processing: Code and Phase. Algorithms, Techniques and Recipes. gAGE/UPC. Barcelona. Spain. 2001. Available at website: http://gage14.upc.es/BOOK/

[6] S. Hilla. 2006 The Extended Standard Product 3 Orbit Format (SP3-c). http://igscb.jpl.nasa.gov/igscb/data/format/sp3c.txt

[7] D. Salazar. Examples for the GPSTk and the Gumstix. Website: http://nacc.upc.es/gpstk/gumstix/

[8] M. Hernandez-Pajares, J.M. Juan, J. Sanz, X. Prats and J. Baeta. 2003 Basic Research Utilities for SBAS (BRUS). 5th Geomatics Week. Barcelona. Spain.

[9] RTCA Minimal Operational Performance Standards for GPS/WAAS Airborne Equipment. Doc. No. Do 229A. June 1998.

[10] B. Renfro at al. 2005 The Open Source GPS Toolkit: A Review of the First Year. Proceedings of the 18th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS 2005). Long Beach, California.