The Open Source GPS Toolkit: A Review of the First Year

Brent Renfro, R. Benjamin Harris, Dr. Brian W. Tolman,
Dr. Tom Gaussiran, Dr. David Munton,
Jon Little, Richard Mach, Scot Nelsen,
Applied Research Laboratories, University of Texas at Austin

BIOGRAPHY

Brent Renfro is a division head within the Space and Geophysics Laboratory at Applied Research Laboratory (ARL:UT). He has a B.A. in Physics from Wabash College (1979) and a M.A. in Computer Science from the University of Texas at Austin (1983). Mr. Renfro has been involved in development of systems for collecting and processing GPS data since 1979.

R. Benjamin Harris is an Engineering Scientist at ARL:UT. He graduated with a B.S. in Aerospace Engineering from the University of Texas at Austin (1994), where he is now a Ph.D. student. He obtained an M.S. in Aeronautic and Astronautics from Stanford University (2000).

Brian W. Tolman is a Research Scientist at ARL:UT, with 18 years experience in GPS-related research, data analysis, and software development. His interests and experience include many different areas, among them precise positioning, ionospheric physics, Kalman filtering, and GPS time transfer. Dr. Tolman holds a Ph.D. in theoretical physics from The University of Texas at Austin.

Thomas Gaussiran is the director of the Space and Geophysics Laboratory at ARL:UT. He received his B.S. in Physics from UT Austin (1988) and M.S. and PhD. from Rice University (1994). He has been involved in utilizing GPS as a remote sensing tool of the ionosphere since 1994.

David Munton is a Research Associate at ARL:UT. He earned a B.S. in Physics from Sonoma State University (1982), and a Ph.D. in Physics from The University of Texas at Austin (1991). His current research interests include GPS time transfer, physical geodesy, and beamforming.

Jon Little is a Senior Engineering Scientist at ARL:UT. He obtained a B.S. (1988) and a M.S. (1990) from Auburn.

Richard Mach is a project lead at ARL:UT. He has been involved with GPS applications since 1990. He currently focuses on guiding the development of a global GPS data collection and monitoring network. He holds a B.S. and M.S. in Aerospace Engineering from the University of Texas at Austin

Scotland Nelsen earned a B.S. in Electrical Engineering at the University of Texas at Austin and is an Engineering Scientist at ARL:UT. In addition, he is a triathlete and a student of German.

ABSTRACT

The open source GPS Toolkit (GPSTk) project, first presented at the ION-GNSS-2004, provides a software suite that supports general GNSS research, analysis and development. This paper will summarize improvements in the GPSTk since that first initial release. A growing community of users has adopted the GPSTk. Examples of how the GPSTk has benefited research, commercial, and military groups will be described.

The goal of the GPSTk project is to free the research community from the burden of implementing standard satellite navigation algorithms. The project provides a software suite that consists of a set of applications built upon a central library. The GPSTk distribution is available for download at http://www.gpstk.org/.

The software is organized into a core library, and a set of applications built upon the library. The GPSTk library provides functions that are common to many applications. The functions include: time conversion; matrix and

statistical analyses; reading and writing of standard formats, such as RINEX and SP3; position determination using gradient and algebraic techniques; troposphere delay models; ionosphere delay models; and P-code generation. The library functionality can be accessed by users through custom-built software applications or through the applications distributed with the GPSTk.

The GPSTk library and most of its applications are written in highly object oriented ANSI standard C++. Functionality in the library is accessed through functions associated with classes. Classes are an object oriented construct that binds data and functions into one module. One advantage of this approach is that the user can interchange equivalent classes. For example, the user can trivially switch from the use of broadcast to precise ephemerides. By providing these standard models in a production quality and modular form, the library empowers the user to concentrate on new applications rather than on base-level capabilities.

The GPSTk software suite is available to the public under the Lesser GNU Public License, or LGPL. The LGPL grants the user a number of rights; notably, the ability to choose whether to modify and redistribute the source code.

A number of new applications and library capabilities have been added the GPSTk since the ION-GNSS-2004. Many of these modifications have been introduced in order to broaden the audience that can benefit from the GPSTk. The software suite now builds under more operating systems, including AIX 5.2 and OS X. GPSTk users have contributed patches to enable support for more compilers. In order to support users of standard GNU software development tools, a makefile generation process has been added, supplementing the original build process based on the Jam utility.

In addition to making the GPSTk compatible with more compilers and operating system, the functionality of the GPSTk has also been enhanced. The latest stable version of the GPSTk, version 1.1, addresses a number of feature requests and bug reports. New RINEX applications have been added, along with new ephemeris routines to support receivers like the Novatel Superstar. In addition, a RINEX conversion utility for Novatel receivers has been added.

A test system has been added to the GPSTk. The purpose of the test system is to provide a consistent, reproducible architecture for ensuring the correctness of library functionality. The test system will run on any system that supports the programming language Perl.

By design, the GPSTk software is portable across platforms. However, applications in languages such as Perl, Python, or MATLAB cannot directly call GPSTk library functions. To address this, an interface package has been added to the GPSTk that can generate interfaces,or bindings from the GPSTk core library to a number of other languages. The bindings are created using another open source project called the Simplified Wrapper and Interface Generator (SWIG). SWIG supports the creation of bindings from C++ to over a dozen languages. In version 1.2 of the GPSTk, SWIG has been used to create bindings to the Python language.

Over the last year, a number of research groups, private companies, and individuals have adopted the GPSTk. The groups span all sectors of GNSS users, from the private sector to the research and military sectors. At Worcester Polytechnic Institute, a student group has adapted the GPSTk to provide onboard programming for their space bound formation flight experiment, the Power metallurgy And Navigation SATellite (PANSAT). In the private sector, Fusion Numerics uses the GPSTk to power its numerical ionospheric forecasting system. Finally, in the military sector, the U.S. Air Force 17th Test Squadron has adopted the GPSTk as the standard toolset for the evaluation of GPS's performance.

BACKGROUND

The Applied Research Laboratories, The University of Texas at Austin (ARL:UT) has established an open source software project called the GPS Toolkit, or the GPSTk. The goal of this project is to provide a world-class, open source computing suite to the satellite navigation community that will become the de facto standard when considering implementing new global navigation satellite system (GNSS) applications.

The code is released under the terms of the Lesser GNU Public License (LGPL), that allows the user to distribute or share software produced with the GPSTk without requiring the software to become open source. The user is free to choose whether or not to share the project's source code.

The code is object-oriented and highly modularized. The code compiles on a variety of platforms, including Windows, Linux, and UNIX. The GPSTk distribution is available for download at http://www.gpstk.org/ along with documentation and basic programming examples. The website, provided by SourceForge, facilitates communication on the project by hosting bug reporting and feature request lists. Users can also subscribe email groups related to the GPSTk.

The GPSTk distribution consists of a library, a collection of applications, and a test suite. The library consists of functionality necessary to read, process, and write GPS data. The applications are stand-alone utilities that are valuable for examining and processing GPS data. The applications also benefit the user as detailed programming examples. The test suite includes programs that test parts of the library. These programs provide a validation test of the installation and can serve as additional examples.

Library Functionality

The GPSTk library supports a wide range of functionality, including, but not limited to:

- Input and output of data in Reciver INdependent Exchange format (RINEX), [1]
- Ephemeris handling (broadcast/RINEX or SP3),
- Time handling and conversions,
- Matrix and Vector algorithms,
- Mathematical and statistical algorithms,
- Tropospheric and ionospheric modeling,
- Robust autonomous positioning,
- P-code generation,
- Runge-Kutta integration,
- Console application support, and
- Support packages (exceptions, string utilities, command line arguments).

This functionality enables any GNSS developer to quickly focus on the problem at hand without having to develop the code that performs many of the well-known functions associated with the data processing.

Application Functionality

The GPSTk applications build upon the functionality in the library to provide a number of useful utilities that also serve as programming examples. The applications cover the following areas:

- Receiver binary translation,
- RINEX utilities,
- Interactive plotting of RINEX data,
- Cycle slip detection and correction,
- Residual analysis,
- Time conversions,
- Differential baseline estimation, and
- Ionospheric modeling.

Design Philosophy

The design goals of the GPSTk library are portability, modularity, clarity, extensibility, and maintainability. These goals allow the GPSTk to maximize the audience and lifetime of the library while decreasing the costs associated with long-term maintenance. The GPSTk meets these goals by strict adherence to the ANSI C++ standard and to Object Oriented Analysis and Design (OOA/D) techniques. As an example of the portability, the GPSTk has been installed and tested successfully under AIX, Linux, Solaris, and Windows. The library and applications can build and mostly function under Macintosh OS X but some printing methods do not work.

Project History

The code that is incorporated in GPSTk has its roots in the software and expertise that ARL:UT has developed over the last two decades in support of GPS research and systems. ARL:UT recognized the value of having a common library base for facilitating the work on a new project or a new GPS research tool. Thus, ARL:UT undertook an effort to create a cross platform, production quality, C++ library with an object oriented design. After completing this effort, ARL:UT realized the value the GPSTk could bring to the greater GPS community considering the lack of open source projects of its kind. Thus, ARL:UT provides GPSTk as an open source project under the LGPL (http://www.gnu.org/licenses/lgpl.html) to offer the GPS community the same benefits that have been realized for ARL:UT projects.

The initial release of the GPSTk was performed in the summer of 2004 and has been presented in a number of forums that include the ION-GNSS-2004, Linux Journal, AGU, and BEACON. [2-5] In the year since the initial release, the project has received much positive feedback, has seen a growth in the user base, and has established an on-line community of developers.

ENHANCEMENTS TO THE GPSTk

A number of enhancements have been introduced to the GPSTk since it was first released and presented at the ION-GNSS-2004. Both the library and applications have been extended in functionality. The software suite has been ported to new development environments and a test suite has been added.

Overview of Recent Releases and Release Availability

The GPSTk was first introduced to the GNSS community at the ION-GNSS-2004. That initial presentation described version 1.0 of the GPSTk. Since that time version 1.1 has been released and is available for download through the GPSTk website. Version 1.2 is nearing completion and its release is anticipated in October 2005. The enhancements described in this paper include those associated with both releases 1.1 and 1.2.

Library Development

The library is the heart of the GPSTk. As such it undergoes frequent revision, from major design additions to minor bug fixes. It is not possible to review all library modifications in this paper. Only those with broad user impact will be described.

Processing of Incomplete Ephemeris

The EngEphemeris class is designed to hold all the quantities transmitted within a matched set of subframes 1-3. There are several means of loading data into an EngEphemeris object, however, prior to Release 1.2, all the methods required that the calling method have a complete set of information. Unfortunately, not all receivers output a complete set of data associated with subframe 1-3. In the case of the Novatel Superstar, the receiver produced words 3-10 of each subframe, but omitted the first two words, the telemetry, and handover words.

It is still possible to create an EngEphemeris object from this set of data, if an estimated time of reception is known, and the user is willing to accept a few assumptions. A new entry point was added to EngEphemeris to implement this capability.

The new entry point has the following signature

```
bool EngEphemeris::addIncompleteSF1Thru3(
               const long sf1[8],
               const long sf2[8],
               const long sf3[8],
               const long sf1TransmitSOW,
               const long gpsWeek,
               const short PRN,
               const short track);
```

where

sf1	words 3-10 of navigation subframe1 stored in the 30 least-significant bits of each array index.
sf2	words 3-10 of navigation subframe 2
sf3	words 3-10 of navigation subframe 3
sf1TransmitSOW	time in GPS seconds of week corresponding to leading edge of first bit of subframe 1
gpsWeek	full GPS week number.
PRN	PRN ID of source satellite.
track	tracker number (typically receiver channel number)

The method addIncompleteSF1Thru3 returns true if successful.

The critical time missing from the telemetry and the handover words are the transmission times. EngEphemeris::addIncompleteSF1Thru3 resolves this by allowing the calling method to specify the time. While it is not important that the time be completely accurate, there are some ground rules from how the transmit time is derived. For example, the parameters sf1TransmitSOW and gpsWeek should be consistent and should represent the time when the data in sf1, sf2, and sf3 were received. As a general rule, the transmit time should be somewhere in the range of 0-2 hours before the epoch time of the ephemeris.

It is also important to note that EngEphemeris objects built via this method will not contain valid TLM message data, A-S flags, and "Alert" bits. In addition, methods EngEphemeris::getTLMMesage() and EngEphemeris::getASAlert() should not be used. The data for these fields are also contained in the missing words and no attempt is made to fill these variables appropriately.

SP3-c support

Precise ephemerides are recorded in the SP3 format from a variety of academic and government agencies. The largest academic consortium, the International GPS Service (IGS), has developed an improved version of the format called SP3-c. [5] The new format provides these enhancements over the original:

1. Satellite identifiers can represent satellites in any GNSS constellation such as GLONASS or Transit. This feature was present in the dialect format called SP3-b.
2. Each estimate of satellite position and velocity can be flagged as predicted to support real time ephemeris processing.
3. Each estimate of position and velocity can be associated with an error covariance.
4. Satellite events such as clock swaps can be indicated.

Within the GPSTk, no existing interfaces will be modified to support SP3-c. Additional interfaces and data structures will be added. Existing GPSTk applications that use the SP3 classes—namely SP3EphemerisStore and SP3Data—will be able to process SP3-c formatted files transparently, with no modifications. Existing applications need to merely link to the new version 1.2 of the GPSTk library. (Note: the design modifications to support SP3-c are still under review and are subject to change prior to the release of version 1.2)

In particular, a new version of the method TabularEphemerisStore::getPrnXvt will be added to support specifying which system to associate with the specified satellite number. In object oriented terms, the additional interface variation is referred to as an overloaded method. SP3EphemerisStore inherits from TabularEphemerisStore so that the modifications will be available to SP3EphemerisStore as well.

The other features associated with SP3-c will be implemented as modifications to the class SP3Data. For each epoch, the additional covariance and satellite event flags will be stored if available. An additional method will be added to SP3Data called getCovarianceMatrix that will transform the covariance information into a familiar matrix of floating point numbers.

Additional Applications

Several applications have been added to the GPSTk. Many of these are standalone or differential positioning utilities. One of them supports the translation of receiver-specific binary messages to the RINEX format.

Position Coordinate System Conversion

The poscvt utility is a command line utility used to convert between coordinate systems. These systems include Earth-centered, Earth-fixed (ECEF) Cartesian, geodetic, and geocentric. Here is an example run of poscvt, converting ECEF Cartesian coordinates to all the other types.

```
bash:>  poscvt  --ecef="-740289.807  -5457071.744
3207245.649"
ECEF (x,y,z) in meters -740289.807 -5457071.744
3207245.649
Geodetic in deg, deg, m 30.38366468 262.27458770
217.6950
Geocentric   in   deg,   deg,   m   30.21602829
262.27458770 6372918.1440
Spherical   in   deg,   deg,   m     59.78397171
262.27458770 6372918.1440
```

Autonomous Position Solution

The rinexpvt application is a utility that generates user positions from pseudoranges recorded in the RINEX format. The application can be used to track the location of a receiver. If applied to a static reference station, the results can be used to investigate the quality of that station's pseudorange observations.

One user position, or PVT, is generated per epoch of observation. The rinexpvt application forms positions based on geometric solutions with no smoothing applied to the pseudoranges, nor filtering of the solutions. A number of error models are applied to the pseudoranges before generating the position calculation, such as atmospheric delay. The user can select an elevation mask by which range observations are edited.

The user must provide an observation and an ephemeris file. Broadcast ephemerides represented in the RINEX navigation file format can be used. If broadcast ephemerides are provided, only healthy satellites, as defined by the ephemeris, are used in the solution. For higher resolution results the user may provide precise ephemerides in the SP3 format.

The user may wish to transform the results of the position calculation to static local reference frame, sometimes referred to as a topocentric frame. The new coordinates, still Cartesian, refer to the cardinal directions: East, North, and Up. The topentric transformation can be performed by rinexpvt.

Figure 1 is based on positions calculated by rinexpvt, expressed in a topocentric frame. The observations were taken by a geodetic-quality GPS receiver at a static location. Precise ephemerides were used. The plot itself was generated using the free tool gnuplot.

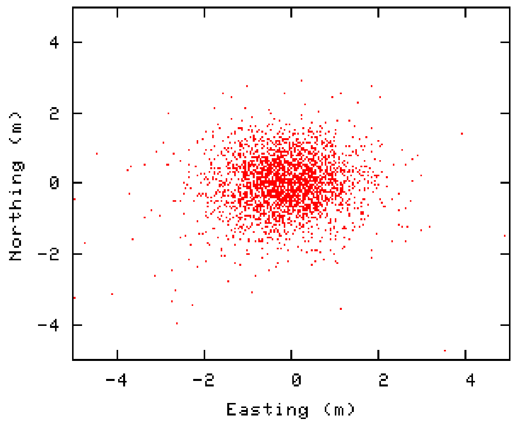

Figure 1: rinexpvt output using topocentric transformation

Baseline Estimation

The vecsol application computes a vector baseline given two RINEX observation files and one ephemeris file. This application was developed and contributed to the GPSTk by Dr. Martin Vermeer, a professor of geodesy at the Helsinki University of Technology.

The baseline solution uses dual-frequency carrier phases. A double difference algorithm is applied with properly computed weights (elevation sine weighting) and correlations. The program iterates to convergence and attempts to resolve ambiguities to integer values if close enough. Crude outlier rejection is provided based on a triple-difference test. Ephemeris used are either broadcast or precise. Alternatively, also P-code processing is provided. The solution is computed using the ionosphere-free linear combination. The ionospheric model included in broadcast ephemeris may be used. A standard tropospheric correction is applied, or tropospheric parameters (zenith delays) may be estimated.

Receiver Binary Conversion

Program novaRinex converts raw Novatel binary data files, created by OEM2 and OEM4 receivers, into RINEX observation and navigation files. It is based on the classes NovatelStream and NovatelData. NovatelStream is a stream class that encapsulates reading and validating the binary stream. NovatelData is a data class that encapsulates storing the binary data records, parsing them, and creating RINEX objects. This code was based on the Novatel manuals "GPSCard Command Descriptions" (Rev 3) (for OEM2) and "OEM4 Family of Receivers Users Manual Volume 2" (OM-20000047 Rev 12), obtained from the Novatel web site www.novatel.com.

Class NovatelStream inherits from FFBinaryStream; therefore, all the I/O machinery therein becomes available to the Novatel data conversion process. The input stream operator (<<), through the functionality of FFBinaryStream, calls the NovatelData::reallyGetRecord(FFStream&) routine that searches for sync bytes, reads the binary header and record size, and stores the full binary record. It also computes a checksum to validate the record. No attempt has been made in this code to provide output operators for Novatel binary data.

The class NovatelData facilitate the processing of multiple records. It encapsulates a Novetel binary data object that contains the raw (unparsed) Novatel data records. The NovatelData module includes cast operators that parse the binary data record and create RINEX objects (RinexObsData and RinexNavData). The class includes an enum of records types that include the main data records for OEM2 and OEM4 receivers. The design of the code will allow support for other record types to be added easily. Specifically, record types RANGE, RANGECMP, RAWEPH, REPB, and RGEC are implemented (RGEB, POSB, and RCSB are included but not yet fully implemented). The class includes the following boolean methods.

- isValid() - if true, the record was read correctly and is valid,
- isData() & isHeader() - always true & false,
- isNav() & isObs() - refers to RINEX data, and
- isOEM2() and isOEM4() - referring to Novatel data types.

The Novatel converter is simple because the Novatel format almost matches the RINEX format one-to-one. This is true for both observation and navigation message records. The only exception to this is the case of OEM2 records, in which the navigation records contain only the 10-bit GPS week number. (This is a common problem in receiver binary data because the broadcast navigation message contains only a 10-bit week number.) Therefore one of the tasks novaRinex must handle is to obtain a time that will serve to resolve the ambiguity in the week number. NovatelData allows the ambiguity to be resolved directly by input of a (full) time tag. The novaRinex program accepts direct user input of this timetag. If a timetag is not provided to it, NovatelData waits until the first observation record has been read and its timetag (that is unambiguous) is used. If navigation data precedes any observations in the file, NovatelData uses the system time as a last resort to disambiguate the GPS week. Note that there is the potential here for some navigation data to be given the wrong time tag. It may be that the Novatel receiver software makes certain that at least one observation record is written before any navigation data,but this could not be verified from the Novatel manuals.

It is hoped that this converter code will prompt others to write similar utilities for other receiver types, perhaps following the pattern established here. Of course, other designs ideas are welcome, however we feel strongly that any classes submitted to GPSTK for this purpose should be inherited from FFBinaryStream. An Ashtech converter along these lines is currently under development at ARL:UT for the GPSTK.

Test Framework

New submissions to the toolkit are tested by their authors prior to submission and by the Librarian prior to incorporation into the library. However, even if both the authors and the Librarian are conscientious in testing new submissions, there remains the question of how to conduct regression testing on the contents of the toolkit when a new release is being prepared.

An initial implementation of a test infrastructure has been developed in the past summer to partially address this question. The test infrastructure supports highly automated execution of command-line test programs that write to standard output (stdout) and/or to one or more ASCII output files. The assumption is that at some point (preferably during initial implementation) a test program will be written that produces known outputs. These known outputs will be captured (along with any required input files) and stored in the configuration management system. The input truth, the output truth, and the test program will all be retained as part of the submission process (or added later in the case of some of the functionality already present in the toolkit). Once this is done, it is practical to execute any of the test programs using the known inputs and then verify that the outputs match the stored "output truth" files.

Of course, each test program will have a unique command line sequence and the number of input and output files will vary between test programs. To allow a generalized process, a format was developed for an ASCII test configuration file. For each test execution, the configuration file records

- the command line for a test including the input files and arguments,
- output file pairs, and
- source of output truth for comparison.

The format allows substitutions for the execution directory, the directory from which input files will be read, and the directory to which the results will be written.

A Perl script processes the test configuration file. During initialization, the script determines the directory substitutions that will be used during this execution and examines the control script (provided by an input argument) to verify the contents. The script will then proceed to execute each command found in the test configuration file. As each command is processed, the script generates a log file that records the success or failure of each command based on return code (0 means the test passed, any other value indicates failure) and whether each of the pairs of output files matched without differences. In addition to the command-by-command pass/fail status in the log file, the script produces a count of the failures and a summary list of the test programs that did not succeed and provides this via stdout and the log file.

At this time a small percentage of the functionality within the toolkit is being tested by an initial implementation of the test script and the test configuration file. The goal is to expand the functionality under test with each release.

The test code is not built as part of the default build process for the distribution. However, the test code, the test script, and the test configuration file are included in the distribution for those who may have need or interest (see gpstk/dev/tests). For example, these tools might be useful in testing a port of the toolkit on a new platform. In the near future, further documentation on the test infrastructure will be added to the website in an attempt to encourage submitters to consider developing and providing suitable tests concurrently with submission of code for the toolkit.

Language Bindings

C++ is an excellent language to support the GPSTk's goal to provide a cross-platform, open source software suite; however, it is only one of many programming languages used by GPS analysts and developers. While it is possible to write C++ wrappers that interface the GPSTk to any language, to maintain such a code base would become prohibitive as the number of languages grows.

One open source project, the Software Wrapper Interface Generator (SWIG), allows the user to develop one interface that generates bindings for several languages. SWIG supports the creation of interfaces from C++ to number of other languages: Python, Perl, Tcl, and Java are among those supported.

In version 1.1 of the GPSTk, a few bindings were provided as a proof-of-concept in Tcl and Python. In version 1.2, the Python bindings have been developed extensively. A majority of the library classes have been wrapped.

In theory, because SWIG specifies a generic interface, these bindings can be built for other languages such as Java or Perl. To date, the bindings have only been tested in Python. Investigation into the support of other languages are left to future versions of the GPSTk.

Platform Support

The GPSTk can compile and run under a number of new environments. This includes compatibility with new development tools. Also, the number of operating system to which the GPSTk has been ported has increased.

Compilers and compilation tools are essential to building the GPSTk and to accessing its library. Keeping the code compatible with new compiler versions is essential to its future. Users have contributed patches to support newer compilers, the GNU compiler gcc version 3.4, and Sun ONE Studio 8.

The standard build tool for the GPSTk is jam. However, the jam system provided with the GPSTk supports a limited set of operating system and compiler combinations. To assist users who wish to port the GPSTk to unsupported environments, a new set of build scripts based on the make utility have been added. The new build scripts are standard for the open source community. An example command sequence to build and install the GPSTk on a GNU system might look like the following.

```
cd gpstk
./configure
make
make install
```

The new build system has enabled the GPSTk to build and run under Mac OS X. While most of the GPSTk operates successfully in that environment, not all functions work. It appears that methods that access advanced formatting control characters will hang. It is hoped that interested parties will contribute patches that enable the Mac OS X port to fully function in future releases of the GPSTk.

Online Presence

As with other open source projects, the Internet plays a central role in the development of the GPSTk. As the user base of the GPSTk has grown, so has its online presence.

Much of the content has changed to directly support user requests. The example user code distributed with the source is explained line by line. Journal articles describing the toolkit have been posted. More detailed instructions to assist users in building the code have been added.

In order to track the usage patterns of the GPSTk, all downloads are now stored in the SourceForge file distribution service. This service distributes the files to a number of severs across the globe, to reduce network traffic and to decrease download times. Through this service it is estimated that the GPSTk has been downloaded approximately 5,000 times in the last 18 months as of September 2005.

SourceForge also hosts lists that support the GPSTk. These lists provide a conduit for announcements; as well as, support for developers. List users generously advise one another regarding detailed programming questions; as well as, general GPS processing issues. Because the lists are archived to the web, these exchanges can also benefit future users of the GPSTk.

GPSTk CUSTOMERS

To date all documentation describing how the GPSTk supports GNSS research and development has been highly general. The purpose of this section is to provide the reader with specific projects that have benefited from the GPSTk. These concrete examples may help readers determine that the GPSTk is relevant to their needs.

Worcester Polytechnic Institute

Under the leadership of Dr. William Michalson, a student group at Worcester Polytechnic Institute (WPI) is using the GPSTk to develop a GPS based orbit and attitude determination navigation system. The system is specifically designed for the WPI Power metallurgy And Navigation SATellite (PANSAT); however, it is general enough to be used for other satellite systems.

Previously flown nanosatellites have used GPS antenna baselines in excess of 0.5 meters. Our design uses a sub 0.5 meter antenna baseline using three coplanar non-linear GPS antennas. Kalman filtering and quaternion based least squares filtering is then used by the orbit and attitude determination algorithms, respectively, to achieve the desired accuracies.

PANSAT is designed to fly in low earth orbit. The orbit determination system is designed to output position X, Y, and Z in ECEF coordinates at a rate of 1 Hz with an accuracy of +/- 10 meters for position, velocity +/- 10 meters/second, and acceleration within +/- 2 meters/second^2. The attitude determination will output pitch, yaw, and roll at 10 Hz. The group anticipates obtaining 0.1 degrees of accuracy from the attitude determination system.

The team is using the GPSTk library to support real-time, onboard control of the PANSAT system. In order to host the GPSTk functionality, on this nanosatellite, a number of challenges had to be overcome. First, the students had to port the toolkit to a new environment, RTLinux, a form of Linux that can be run on embedded processors and that can support real time operating constraints. Their second challenge involved processing incomplete ephemeris frames collected by the embedded Novatel Superstar receiver. With the assistance from ARL personnel, new functionality was added to the GPSTk library to support this and other similar receivers. These modifications are described in more detail in the Library Development section of this paper.

USAF 17th Test Squadron

The GPSTk is being used by the US Air Force's 17th Test Squadron to evaluate the effectiveness of the Legacy Accuracy Improvement Initiative (L-AII). The goal of L-AII is to increase the effectiveness of GPS to all users, without an upgrade to user equipment. [10] The primary element of the L-AII is the addition of National Geospatial-Intelligence Agency monitor station data to the existing Air Force monitor station data set. The additional observations will not only increase the number of observations used in creating the broadcast ephemeris, but it will also improve the geometry associated with those observations.

The GPSTk assists in the evaluation by producing position residuals for stations at known locations. These stations are IGS reference station, part of the Jet Propulsion Laboratory (JPL) fiducial network. By examining the position solution using the broadcast ephemeris over time for these sites, the impact of L-AII will be measured. This study is examining the position residuals for two 30-day periods; pre L-AII and post L-AII.

Fusion Numerics

Fusion Numerics uses GPSTK in its operational ionospheric modeling and forecasting system. The system, developed under a series of contracts with the US Air Force, is capable of generating near real-time three-dimensional ionospheric electron densities and corresponding GPS propagation delays. It adapts data assimilation technologies developed and routinely used for operational numerical weather forecasting, in order to nowcast and forecast ionospheric conditions. It consists of two parts: a first-principles numerical model of the ionosphere, and a data assimilation component. The GPSTk is used to process reference station observations to support the data assimilation module.

The core ionospheric model solves plasma dynamics and composition equations governing the evolution of density, velocity, and temperature for seven ion species on a fixed global three-dimensional grid. It uses a realistic model of the Earth's magnetic field and solar indices obtained in real-time from the NOAA Space Environment Center (SEC). While the core model is capable of delivering realistic results, its accuracy can be significantly improved by employing a special set of numerical techniques known as data assimilation. In the process of data assimilation, the core ionospheric model is continuously fed observational data from a network of reference GPS ground stations, via a very large scale Kalman filter. This improves both the nowcast and the forecast. The system employs the sequential approach and the approximate Kalman filtering algorithm outlined in Khattatov et al. (2000) [8,9]. Public access to the system is provided at http://www.fusionnumerics.com/ionosphere or http://gpscorrections.com.

Ionospheric GPS delays from the system are currently used for differential GPS positioning. Preliminary results indicate that relative positioning accuracy of approximately 10 cm is achieved and precise point positioning of less than 1 m is feasible with inexpensive off the shelf receivers.

FUTURE OF THE GPSTk

A number of fundamental enhancements are underway for versions of the GPSTk beyond 1.2. A new design is under development that will change how time is stored and converted within the library. New contributions are scheduled to support new file formats.

A new class to support time records and conversion called TimeTag is being introduced. This class may replace DayTime in the long term. Preliminary versions of TimeTag and its associated subclasses will be released with version 1.2.

Projects with ARL:UT that use the GPSTk will use it to process formats new to the GNSS community: RINEX version 3.0 and BINEX. The support for RINEX 3.0 will follow the standard being formed by Werner Gurtner.

In contrast the BINEX support will provide fundamental processing functions only. BINEX stands for BINary EXchange and is a binary format GPS, GLONASS, and SBAS data. It is intended to eventually replace a number of file formats including RINEX, SINEX, IONEX, SP3, and others. One project is currently developing software that will utilize private BINEX messages to transport data between two systems. The software provides for reading and writing BINEX files or streams; as well as, parsing the messages. The architecture established with this code will provide the infrastructure to build full BINEX support into the GPSTk. The plan is to donate this code to the GPSTk at the end of 2005.

The future of the GPSTk extends beyond the previously described library enhancements. User feedback and contributions have already shaped the GPSTk in unanticipated and productive directions. Motivated users have proposed future contributions that range from detailed signal processing algorithms to a full manual written for Chinese readers. The interested reader is encouraged to periodically check the GPSTk website for the latest updates.

ACKNOWLEDGEMENTS

The authors would like to thank the Applied Research Laboratories at the University of Texas at Austin (ARL:UT) for funding this project.

There are a number of ARL:UT personnel not listed as authors who have contributed to the GPSTk effort. They are: John Knutson, Tony Hughes, Kevin Kraatz, Bruce Haufler, Shawn Wolff, Liam Campbell, Jon Wyant, J. Clark Hughes, Shawn Furgason, Tina Hatla, Margaret Evans, Ryan Mire, and Rob Chang. These staff and students have performed beyond the call of duty to not only serve their projects but also the general GPS community via the GPSTk.

Details regarding customer stories were supplied voluntarily. The authors would like to thank customers who shared the details of their projects: Dr. William Michalson, JP Salmon and the rest of the WPI PANSAT team; Bart Ewers of the USAF 17th Test Squadron; and Dr. Boris Khattatov of Fusion Numerics. These uses have demonstrated that those who participate in open source get the most from it.

The following users have contributed to the GPSTk by contributing important feature requests, bug reports, fixes or new code: Dr. Martin Vermeer (Helsinki University of Technology); Kevin Miller and Dr. Angelyn Moore (JPL); Bryce Deary (ARINC); Luciano Mendoza (La Plata Astronomic Observatory, Argentina); Robert Bruton, Kenneth Rogers and John Pekar (U.S. Naval Surface Warfare Center); Krystof Kamieniecki (kkGPS open source project).

REFERENCES

1. Gurtner, W., “RINEX: The Receiver-Independent Exchange Format,” *GPS World*, Volume 5, Number 7, pp. 48-52. Also available from: ftp://igscb.jpl.nasa.gov/igscb/data/format/rinex210.txt

2. David Munton, Brian Tolman, R. Benjamin Harris, Aaron Kerkhoff, Tom Gaussiran, Gary Bust, Scot Nelsen, “GPSTk: An Open Source Toolkit for Working With GPS Data.” Poster presentation at the Fall Meeting of the American Geophysical Union, San Francisco, California, 2004.

3. Tom Gaussiran, Brian Tolman, Ben Harris, “An Open Source Toolkit for GPS Processing, Total Electron Content Effects, Measurements and Modeling.” *Proceedings of the International Beacon Satellite Symposium 2004*. Trieste, Italy. October, 2004.

4. Brian Tolman, R. Benjamin Harris. “The GPS Toolkit.” *Linux Journal*. September 2004, pp. 72-76.

5. R. Benjamin Harris, Brian Tolman, Tom Gaussiran, David Munton, Jon Little, Richard Mach, Scot Nelsen, Brent Renfro, ARL:UT; David Schlossberg, University of California Berkeley. “The GPS Toolkit -- Open Source GPS Software.”

Proceedings of the 16th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS 2004). Long Beach, California. September 2004.

6. SP3 format specification. Available at http://igscb.jpl.nasa.gov/igscb/data/format/sp3_docu.txt.

7. Hilla, S., "The Extended Standard Product 3 Orbit Format (SP3-c)". Available at http://igscb.jpl.nasa.gov/igscb/data/foramt/sp3c.txt

8. Khattatov, B., M. Murphy, M. Gnedin, B. Cruickshank, J. Adams, V. Yudin, T. Fuller-Rowell, "Ionospheric Corrections from a Prototype Operational Assimilation and Forecast System", *Proceedings of IEEE Position, Location, and Navigation Symposium (PLANS)*, Monterrey, CA, April 26-29, 2004a.

9. Khattatov, B., M. Murphy, M. Gnedin, B. Cruickshank, J. Boisvert, J. Sheffel, V. Jayaraman, "An Ionospheric Forecasting System", *Proceedings of the 16th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS 2004)*. Long Beach, California, September 2004.

10. Stephen Malys, Margaret Larezos, Steven Gottschalk, Shawn Mobbs, Bryant Winn, William Feess, Michael Menn, Everett Swift, Michael Merrigan, William Mathon. "The GPS Accuracy Improvement Initiative," *Proceedings of the 10th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GPS 97).* Kansas City, Kansas, September 1997.