# The GPS Toolkit – Open Source GPS Software

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

**David Schlossberg,**
***School of Information Management and Systems***
***The University of California at Berkeley***

## I. BIOGRAPHY

Brian W. Tolman is a Research Scientist at Applied Research Laboratories, The University of Texas at Austin (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.

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).

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.

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..

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.

Brent Renfro is a division head within the Space and Geophysics Laboratory at 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.

David Schlossberg is a Masters student at the University of California at Berkeley, in the School of Information Management and Systems. Mr. Schlossberg is a Graduate Student Researcher at the Center for Document Research. He graduated with a dual B.S. in Computer Science and Computer Engineering from Washington University in St. Louis (2000).

## II. ABSTRACT

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 GPSTk provides a core library and collection of applications that support GPS research, analysis and development. The code is released under the terms of the Lesser GNU Public License (LGPL). LGPL grants the user a number of rights, notably the ability to choose whether to modify and redistribute the source. The GPSTk code is written in ANSI C++ that is platform independent. It has been installed and tested successfully under Linux, Solaris, and Windows.

The GPSTk supports a broad range of functionality including reading and writing observations in standard formats, such as RINEX and SP-3. Other functions provided by the GPSTk include ephemeris evaluation, position determination, receiver-autonomous integrity monitoring (RAIM), atmospheric delay modeling, and P-code generation.

The GPSTk distribution is available for download on SourceForge at http://www.gpstk.org/. The website also provides bug reporting and a feature request list. Detailed documentation can be generated from the code using the doxygen package. Documentation can also be found on the website.

The GPSTk is the base library used for GPS research and development at ARL:UT. ARL:UT has been involved with satellite navigation since Transit in the 1960's and is currently conducting research in a wide variety of GPS related fields, including ionospheric studies. ARL:UT continually improves the library and is committed to its development and maintenance.

ARL:UT's goal is to make the GPSTk a community-wide resource for all users of GPS and GNSS technology. Participation is welcome and will benefit the GPSTk in the

This paper was first presented at the ION-GNSS-2004 in Long Beach, California, U.S.A. on September 24, 2004

following ways: bug reports, new algorithms, suggestions for improvement, and contributions of additional applications.

## III. INTRODUCTION

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 GPSTk provides a core library and collection of applications that support GPS research, analysis and development. The code is released under the terms of the Lesser GNU Public License (LGPL). LGPL grants the user a number of rights, notably the ability to choose whether to modify and redistribute the source. The GPSTk code is written in ANSI C++ that is platform independent. It has been installed and tested successfully under Linux, Solaris, and Windows.

This paper presents an introduction to the GPSTk beginning with a summary of library functionality in Section IV. Similar software packages and other GPS-related work that is available elsewhere is discussed in section V. The GPSTk project itself, the GPSTk website, and licensing details are covered in Section VI. Applications that are included in the GPSTk distribution are described in detail in Section VII. The paper concludes in Section VIII with a description of how the project is expected to mature, along with an invitation to the GPS community to participate.

## IV. THE GPSTk LIBRARY

The GPSTk distribution consists of a core library, a collection of applications and a test suite. The core library consists of functionality necessary to read, process, and write GPS data. The applications, discussed in Section VII are stand-alone utilities that are valuable for examining and processing GPS data and providing more detailed programming examples. The test suite includes programs that test parts of the library. These programs provide both a validation test of the installation and an example of the use of the library in standalone programs.

### Functionality

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

- RINEX input / output,
- Ephemeris handling (broadcast/RINEX or SP3),
- Date and time conversions,
- Matrix and Vector algorithms,
- Mathematical and statistical algorithms,
- Tropospheric and ionospheric modeling,
- Positioning and RAIM, and
- Support packages (exceptions, string utilities, command line arguments).

Central to the library is the reading and writing of data files in the RINEX format. The RINEX specification version 2.1 is completely supported. Support for version 2.2 and other extensions are expected in the near future. The library also includes the capability to define custom 'extended RINEX' observation types that are useful for storing results and intermediate quantities in RINEX format files (see the Applications section below).

The library provides complete satellite ephemeris storage capabilities with input from both broadcast ephemeris (RINEX navigation files) and SP3 format files. The ephemeris storage classes encapsulate all the details of ephemeris handling. They require only input file names to load the ephemerides and implement the standard computation algorithms such as the satellite position computation algorithm defined in ICD-GPS-200 [1]. The ephemeris store objects are designed using object-oriented methods (inheritance and polymorphism) so that user can easily switch usage between broadcast and precise ephemeris.

Dates and times are fundamental to GPS and to the GPSTk. The 'DayTime' class implements many useful date and time formats into a single very powerful object. These formats include Modified Julian Date, GPS time, and calendar dates. There are also text string-based routines that allow custom formatting of time tags, as well as text string interpretation.

The GPSTk library includes an extensive matrix and vector package built entirely using templates. The package includes the usual arithmetic operators like addition, subtraction, multiplication, and division. It also includes LU decomposition, singular value decomposition, Cholesky decomposition, and inversion of matrices. There are also several mathematical algorithms in the library that include statistics (1- or 2-sample), polynomial fitting, and Lagrange interpolation.

Several troposphere models are provided in the library. Their design uses inheritance and polymorphism so that the user may easily interchange models or create a new one. There are currently five models implemented that include Hopfield models [2] and the New Brunswick UNB3 model [3].

Positioning and navigation is an area of the library that is expected to grow substantially in the near future. Currently, functionality is limited to autonomous pseudorange solutions implemented using either linearized least squares or the algebraic solution [4]. A receiver autonomous integrity monitoring (RAIM) algorithm is also included.

There is much more functionality in the GPSTk library that can be detailed here. There is a P-code generator that produces the bit sequence for P-code as defined by the GPS-ICD-200. An application framework is defined that allows programmers to easily define and create console applications. There are a wide variety of low-level support functionalities, such as exception handling, string utilities, formatted file handling, and command line argument processing that ease the development of console-based programs.

### Design

All of the source code for the GPSTk is highly portable ANSII C++ that makes extensive use of the standard template library (STL) and object-oriented programming. Applications of the library, with one exception, are console-based rather than graphical. These design choices make the code extremely platform-independent. The entire distribution has been installed and tested under Linux (gcc), Solaris (Forte and gcc), and Windows (Visual Studio .NET 2003 and Cygwin/gcc). Support for other development environments should be readily achievable and is part of the on-going development effort at ARL:UT.

All of the GPSTk code includes documentation designed for use by `doxygen` [ref], which is a freely available package that generates a HTML-based document set from the code itself. Like the GPSTk, `doxygen` is platform-independent. The `doxygen` documents are available on the web site or easily generated from the code.

Building the GPSTk distribution is as simple as a few commands at the console. The build is simplified by the use of `jam`, which is a freely available alternative to `make` that is very easy to use [6]. The program `jam` is a simple platform independent utility to manage a software project. It provides commands to compile, install, and test source code distributed among multiple directories.

## Design Principles

The primary design goal of the GPSTk is to provide code that is modular, understandable, extensible, and maintainable. For these reasons, the GPSTk strives to adhere to Object Oriented Analysis and Design (OOA/D) techniques. OOA/D contrasts with procedural design supported by the C, FORTRAN, and MATLAB languages. In procedural programming, a function library is provided to the user. In OO programming, a 'class library' is provided. Each class is an independent module that can be invoked by the user as an object or extended by the user in the form of a new class. Classes can build upon each other through a number of object-oriented principles, such as inheritance and encapsulation. The interested reader is referred to other references for more information about OOA/D [7] and C++ [8].

The GPSTk library relies heavily on the STL [9], which is part of the ANSI standard for C++. The STL provides OO data structures (containers). These include linked list, vectors, and standard operations such as the quicksort algorithm. The following section describes an example that includes the use of `map`, an STL class.

## RINEX Example Code

The following code segment illustrates how easy it is to use the GPSTk library to process RINEX. The example demonstrates how the expressive power of C++ is fully utilitized by the GPSTk so that the user can focus on writing application-specific code.

```
Line No.
1         RinexObsStream roffs("bahr1620.04o");
2         RinexObsData roe;
3         RinexObsData::RinexDatum dataobj;
4         RinexPrn prn(14, (RinexSystem) systemGPS);

5         using namespace std;
6         using namespace gpstk;

7         double gamma = (L1_FREQ / L2_FREQ)*
             (L1_FREQ / L2_FREQ);

8         while (roffs >> roe)
          {

9            RinexObsData::RinexPrnMap::iterator
                pointer = roe.obs.find(prn);

10           if( itr != roe.obs.end() )
             {
11              dataobj =
                   (*itr).second[RinexObsHeader::P1];

12              double P1 = dataobj.data;
13              double P2 =
                 itr->second[RinexObsHeader::P2].data;
14              double L1=
                 itr->second[RinexObsHeader::L1].data;

15              double mu = P1
                   - L1 * (C_GPS_M/L1_FREQ)
                   - 2.0*(P1-P2)/(1-gamma);

16              cout << roe.time.GPSfullweek()
                     << roe.time.GPSsow()
                     << setprecision(11)
                     << mu << endl;
             }
          }
```

Figure 1 - Example User Code

In Figure 1, a RINEX observation file is parsed. The observations are combined to isolate multipath, which is then written to standard output. This linear combination of observables is documented in a number of sources [10], [11]. In line 1, an observation stream is created that will extract observations from the named RINEX observation file. Lines 2 through 4 contain objects used to interpret the observation stream. Note that the GPS satellite associated with PRN 14 will be examined. Lines 5 and 6 demonstrate the use of C++ namespaces by the GPSTk. Namespaces are used to prevent name clashes and to organize classes by topic. All of the GPSTk code is part of the namespace `gpstk`. Line 8 shows how an observation stream can be used to extract observations and to control a loop. Lines 11 through 14 demonstrate how to traverse the data structure of observations supplied from the observation stream. While the given data structure is specific to the GPSTk, the functions are typical for use of the STL. The STL `find` function is used to search for observations for a given PRN and the bracket operator is used to pick out known observables. In line 15, the pseudorange minus phase multipath combination is formed. In line 16, the combination is written to standard output. Note, the use of the time member `roe.time` is an object of the class `DayTime`. `DayTime` is the class used by the GPSTk to compute and convert several forms of time associated with satellite navigation.

### Support for Other Languages

It is possible to provide bindings to practically any other programming language because the GPSTk is written in ANSI standard C++. The applications provided in the distribution of the GPSTk (see below) include bindings to Octave, which is an open-source alternative to MATLAB.

## V. COMPARISON TO EXISTING SOFTWARE

Working with GPS observations is a software-intensive activity. In order to transform receiver products into tangible results, software utilities must be used. Fortunately, many such software utilities have been developed by and for the GPS community but these utilities cannot address the needs of every user. A fundamental goal of the GPSTk is to provide a software library of fundamental operations so that any user can develop custom software and focus primarily on research and development rather than basics. To illustrate the scope of capabilities provided by the GPSTk, it is useful to compare it to existing GPS software projects. These projects generally fall into one or more of the following six classifications:

- Navigation tools.
- Observation extraction and editing tools
- User developed applications.
- Surveying software.
- Scientific research suites.
- Open source projects.

How the GPSTk relates to the above classifications is described in detail in the following paragraphs.

### Navigation Tools

Most GPS users only need access to positions generated within the receiver in order to navigate. For these users, numerous applicable commercial utilities and libraries exist. These utilities usually interpret NMEA 0183 messages from a GPS receiver. [12] The GPSTk, at this time, supports neither the generation nor the processing of NMEA messages.

### Observation Extraction and Editing Tools

Researchers and developers who work with GPS frequently require access to the raw observations collected by a GPS receiver. The vendors of most geodetic quality receivers provide an application to control the receiver or to record observations from it. There are utilities that convert observations from proprietary receiver formats into standard formats such as RINEX or BINEX. The best example is the `teqc` utility that is a free download from UNAVCO [13]. The `teqc` utility serves many other purposes that include enabling the user to edit, validate, and quality-check RINEX observation files. There are several GPSTk applications that support editing and checking RINEX observation files but, at this time, there is support in the GPSTk for neither receiver control nor the interpretation of receiver binary formats.

### User Developed Applications

Many analysts, researchers, and developers rely on their own code to process raw observations. Typically this end-user code is in the form of MATLAB scripts. GPS textbooks frequently provide MATLAB code for common algorithms. Examples include texts by Borre and Strang [14] [15] and Misra and Enge [16]. It is anticipated that the GPSTk will benefit this class of software users the most. If a user must implement processing software outside of the MATLAB environment, this library directly addresses that need. The GPSTk contains basic functionality required for a wide range of applications—RINEX routines, precise ephemeris routines, and routines from the ICD-GPS-200 [1]. Furthermore, application and data streaming frameworks are provided to simplify writing standalone applications. Using the GPSTk, the user can create desktop applications as well as embedded software.

### Surveying Software

The GPSTk does not at this time support precise positioning over short baselines. This is an area of growth planned for the near future.

### Scientific Research Suites

There are commercial and research packages that can be used in advanced GPS applications to support science objectives. One example is the Bernese software, now in version 5.0, that is written in FORTRAN [17]. JPL provides the Gipsy/OASIS II suite [18] that is written in C. GAMUT/GLOBK is also available as a FORTRAN package [19]. These suites provide advanced capabilities, such as solving for extremely long baselines and orbit determination. The GPSTk contrasts with these mature packages in a number of ways besides implementation language. The GPSTk to date provides primarily basic models. Advanced GPSTk applications such are available, such as the one Total Electron Content mapping applications, however such advanced models are not built into the library. Other advanced applications are under development. Development of the GPSTk along these lines is welcomed.

### Open Source Projects

Relative to the number of proprietary packages, the number of GPS related open source projects is small. The majority of open source projects provide navigation utilities: GPSDrive and GPSManager are two examples [20, 21]. The Sharc package provides extraction tools for geodetic quality receivers [22]. The OpenSourceGPS project provides real-time software for an IBM PC compatible to acquire and track GPS satellites using the Zarlink chipset [23]. The scarcity of open source packages to empower the GPS researcher is one of the prime motivations for ARL:UT to initiate the GPSTk project.

## VI. THE GPSTk PROJECT

### Background

ARL:UT has been active in satellite navigation research and application development since the time of the Transit constellation. ARL:UT has since diversified its work to include, among other areas, precise surveying, radio wave propagation, global tracking networks, receiver design, ionosphere modeling, and deployable sensors. All of these projects have included a substantial software development effort. Over the years, the work of all the GPS-related projects within ARL:UT have contributed software, experience and testing to the creation of a unified common code base for basic GPS software processing. The resulting ARL:UT software library has now been released as the open source GPS Toolkit.

### License

The source code for the GPSTk is licensed under the Lesser GNU Public License, or LGPL, which grants any user a number of rights and responsibilities. Under terms of this license, users are allowed to redistribute the source code and to modify it. However, unlike the related GNU Public License, or GPL, users may choose not to redistribute code. Notably, commercial applications may use the library and the resulting products may be sold for profit. The details of the LGPL are included in the GPSTk distribution.

### Documentation

The GPSTk is available for download at *http://www.gpstk.org/*. The project web presence is hosted by SourceForge [24], a popular site for open source projects. SourceForge provides a number of services for the project, including bug tracking, mail lists, file downloads, and a project-defined web site. In the project defined website, the library is fully documented. In addition, there is a “Getting Started” tutorial aimed at familiarizing new users to the GPSTk. An article describing the GPSTk has been published in the September 2004 issue of Linux Journal. [25] The full text of the article will be available on the GPSTk website in October, 2004. Another publication is anticipated in the GPS Toolbox column of GPS World [26].

## VII. THE GPSTk APPLICATIONS

The GPSTk distribution includes several complete application programs. When the GPSTk tarred, zipped file is decompressed, the applications are found in the `gpstk/apps` subdirectory. These programs make use of the library to read, process, write, and display GPS data. The programs are, with one exception, console-based and read and write RINEX format files. These programs serve both as useful utilities and as detailed complete examples of programming using the GPSTk library.

These applications include utilities to summarize and edit RINEX files, to generate residuals and corrections from raw data, to find and estimate cycleslips in the carrier phase, and to fit a simple model to the ionosphere. The single non-console application is a menu-driven graphical interface to these RINEX utility programs that allows the user to see and interact with 2-D plots of RINEX data, including the residuals and corrections generated from the raw data. Finally, another application provides bindings that allow the functionality of the GPSTk to be used interactively in the computer algebra system called Octave.

### RINEX Utilities

The GPSTk distribution includes, under the `apps/Rinextools` directory, four programs useful for handling RINEX data. Program These programs are `RinSum`, `RinexDump`, `EditRinex`, and `ResCor`.

`RinSum` produces a summary of the content of a RINEX observation file. The summary includes the header information and a table of the number and time limits of observations versus satellite.

Program `RinexDump` simply dumps a selected portion of the data in a RINEX observation file to a flat output file, with the data arranged in columns. This program is very useful in providing data to other programs. An example would be a spreadsheet or a plotting program such as `gnuplot` [28], in a simple format.

Program `EditRinex` will read a RINEX observation file, edit its content, and write out the result to another RINEX file. Editing can include specifying each of the header records, adding comments, deleting individual satellites or observation types, creating new observation types, time-windowing, writing out to multiple files, biasing data, setting the data, LLI, or SSI flags, and more.

The design of the editor program itself is very simple. Basically, it is just a shell that reads the command line including the input file name and passes that information to an object called a `RINEXEditor`. The `RINEXEditor` does all the work.

`ResCor` reads a RINEX observation file (and possibly ephemeris data and a receiver position) and computes any of several residuals or corrections from the data. `ResCor` can generate ionospheric TEC, elevation and azimuth data, geometry-free phase, and wide-lane bias.

`ResCor` defines an object that inherits the ‘RINEX Editor’ class and adds the functionality of computing the values of selected quantities and storing them in newly created observation types before writing to the output file. Thus, `ResCor` is automatically capable of doing everything `EditRinex` can do; via inheritance, the `ResCor` processor *is* a `RINEXEditor`, but with additional capabilities.

`ResCor` writes its results to a file in RINEX format, making use of the GPSTk ‘extended RINEX’ functionality, in which the ‘observation type’ of the output may be a custom-defined quantity, rather than just a standard RINEX observation type. (For examples of `ResCor` in operation, as well as `RinSum` and `RinexDump`, see the `RinexPlot` application below.)

ARL:UT routinely uses a script that runs `RinSum`, the GPSTk discontinuity corrector (see the next application) and `ResCor` to clean, quality check and smooth large amount of raw RINEX data, and then compute the ionospheric TEC for use in studies of the ionosphere (also see the ionospheric model application below).

### Cycleslips

The GPSTk includes an implementation of a phase discontinuity detection and correction algorithm based on the work of Blewitt (1990) [28]. This algorithm uses the geometry-free and wide lane linear combinations of raw dual frequency pseudorange and carrier phase data to find cycleslips and estimate their magnitude. The GPSTk algorithm is implemented as a single function that can be called from within the user's program. It defines, and takes as input, a 'satellite pass' object, which the caller has filled with dual frequency range and phase data. The algorithm detects and estimates, and also optionally repairs, cycleslips in the L1 and L2 phase; there is also the option of smoothing the range with the phase. The output of the algorithm is either the corrected phase data, which can be written to a new RINEX file, or a set of editing commands that can be passed to the `EditRinex` program (see the RINEX Utilities application), or both.

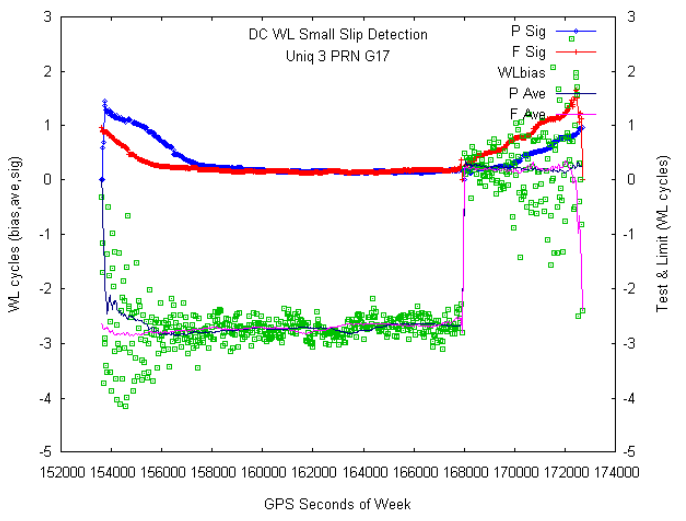

Figure 2 - Wide lane bias data (green) and statistics generated from it, in a case where a cycle slip was detected.

The directory `apps/cycleslips` contains both the GPSTk discontinuity corrector (DC) and a driver program, `DiscFix`, which is used to process RINEX observation files using the DC. `DiscFix` is a simple program that reads options from the command line, reads the input file(s) into a series of 'satellite pass' objects, and then calls the DC for each complete satellite pass. `DiscFix` will write the corrected data to a RINEX output file, or the editing commands in the output may then be used as input to `EditRinex`, which will read the original input file, correct the cycleslips and write the result to a new RINEX file.

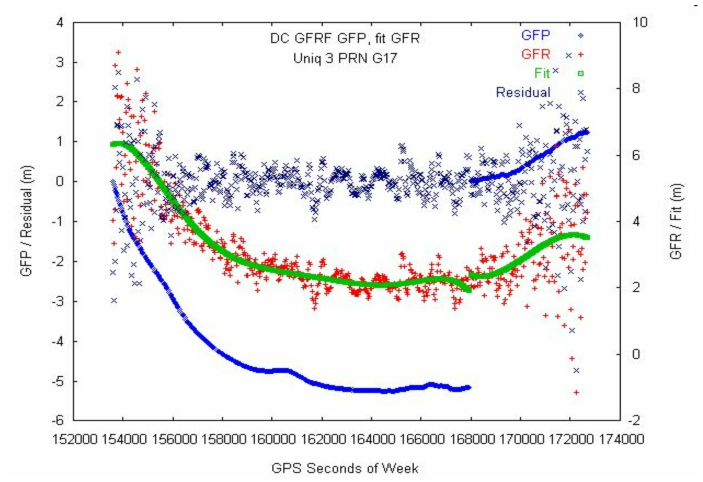

Figure 3 - The geometry-free range (red) and phase (blue), with polynomial fits (green) to the range, and the residuals of fit (dark blue).

Also included in `apps/cycleslips` is an `examples` subdirectory, which contains two complete examples of the operation of `DiscFix`, including scripts, input files, data files, and plots of the output. `DiscFix` can output a large amount of intermediate information during a run, in a form such that it can easily be plotted. Figure 2 shows the wide-lane bias (green) for a pass in which a cycleslip is detected. The slip is estimated using statistics in the past and future of the slip (running averages and standard deviations are also plotted).

Figure 3 shows the geometry-free range (GFR) and phase (GFP) for the same slip. A polynomial (green) is fit to the GFR data (red) in each continuous segment; residuals of this fit are shown in dark blue. The GFP (blue) is compared to the polynomial fit to detect a slip, and then new polynomials are fit to the phase data on either side of the slip, and these are used to estimate the slip magnitude.

**RinexPlot**

`RinexPlot`, found in the `apps/RinexPlot` directory, is a menu-driven graphical interface to the RINEX utility programs `RinSum`, `RinexDump` and `ResCor` that are described above. It makes use of `RinexDump` to put the data into a flat file, and then reads it and displays a 2D scatter plot of selected data. It will plot whatever data is found in the RINEX input file, and can display the output of `RinSum` in a window. It also allows the user to create new observation types using `ResCor`, and then display them. Using the mouse, the user can see the coordinates (both screen and data) of any point, and can select a rectangle on the plotting surface and then zoom in on it. The program's configuration at any time may be stored in a file, which can be re-loaded at another time, or used on the command line to reproduce the plot at a later time.

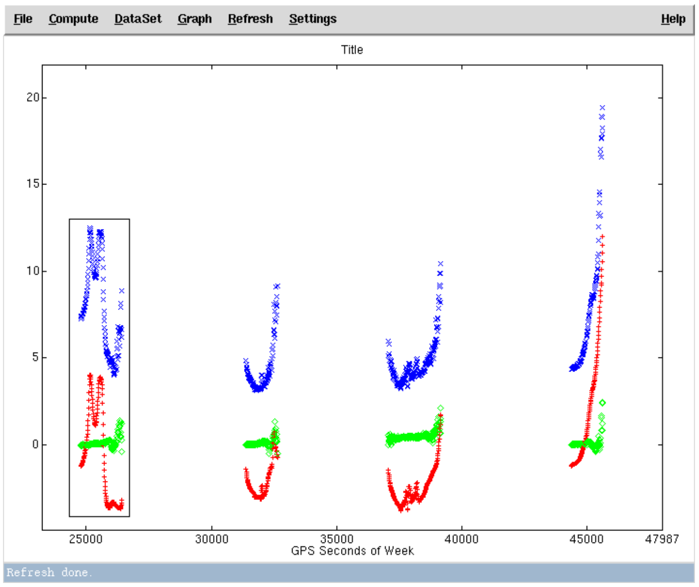

Figure 4 - RinexPlot output (under Linux) with zoom rectangle shown.

Figure 4 and Figure 5 present an example of RinexPlot; in this case the input file is a RINEX observation file containing L1, L2, P1 and P2 data that was collected on a low-Earth-orbiting satellite, receiving a GPS signal from a satellite that was below its horizon. Figure 4 shows the default plot of the geometry-free range and phase (both are measures of ionospheric delay) as well as the wide-lane bias, for several satellite passes. In this figure the user has selected a rectangle for zooming; the result of the zoom is shown in Figure 5.

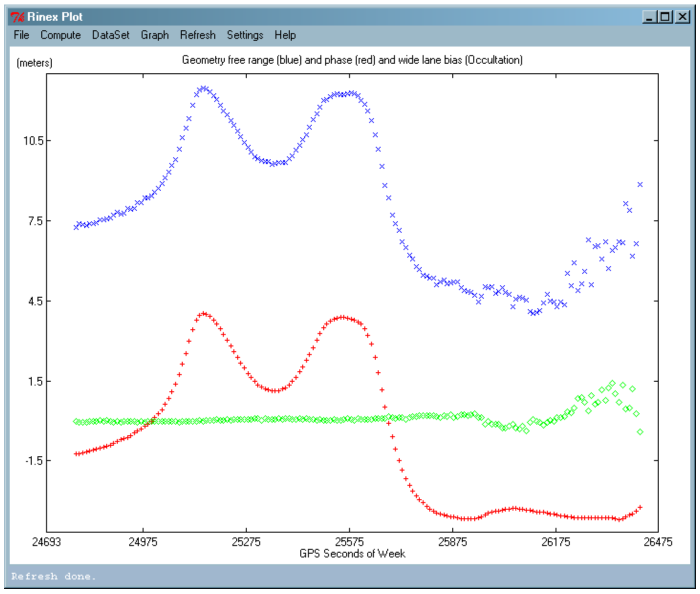

Figure 5 - `RinexPlot` output (under Windows), zoomed, with titles added.

`RinexPlot` is written in Perl/Tk, which is remarkably efficient and portable; it runs on all the platforms of the GPSTk. Under Microsoft Windows, Perl and the Tk module may be obtained for free research use from ActiveState. [29] (e.g, in the ActiveState distribution at http://www.activestate.com/Products/ActivePerl/.)

## Residual Analysis

The `reszilla` utility is found in the `apps/reszilla` subdirectory; it is essentially a differencing engine. For a single receiver, it can generate the difference between the expected and measured pseudoranges, known as the range residual or observed range deviation (ORD). The expected ranges are computed using the known station position and either broadcast or precise ephemerides. The differences are summarized statistically but also can be output by observation into a file that is parseable by numerical analysis programs. The `reszilla` utility can also form the double difference for two receivers in a zero baseline. Similar statistical and raw output can be generated for the double difference case.

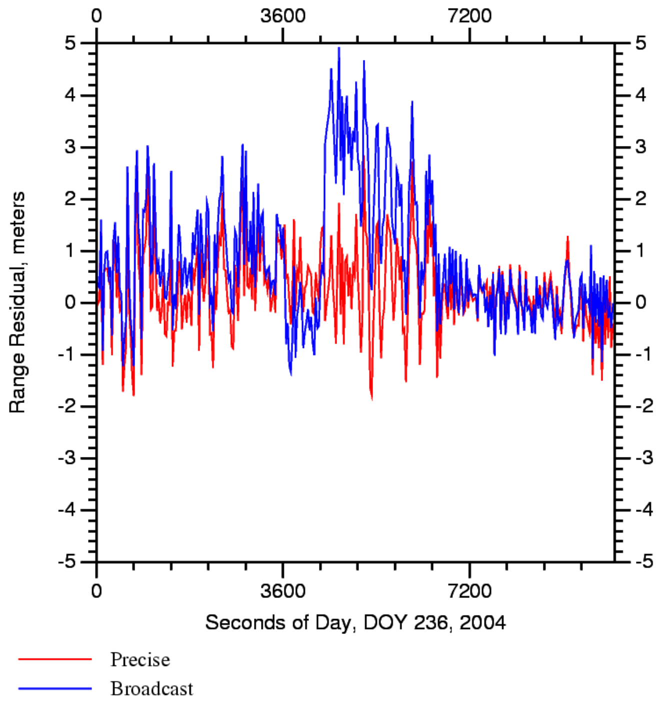

Figure 6 - Range residuals for PRN 3 from day 236, 2004.

The `reszilla` utility can be used to investigate a number of phenomena. In double difference mode, the results are a function of only receiver noise and hardware biases. When used to compute the range deviation, the results are also due to ephemeris error. The plot in Figure 6 demonstrates how the `reszilla` utility can be used to compute ephemeris error. In this plot, the range residuals associated with broadcast and precise ephemeris are compared for PRN 3, on day of year 236, 2004.

## Modeling the Ionosphere

The final example of GPSTk applications involves processing GPS data for remote sensing of the ionosphere. GPS receivers can measure the ionospheric total electron content (TEC) via the propagation delay introduced into the GPS signal by free electrons along the signal path. The TEC is in fact proportional to the dual frequency correction for the ionospheric delay. This measurement is biased, however, because of hardware delays, both within the receiver and the satellite. These biases can be estimated, however, using TEC data at nighttime, when the ionospheric TEC is nearly zero.

The `apps/ionosphere` directory of the GPSTk distribution contains programs used to estimate both the TEC measurement biases and a simple model for the ionospheric TEC using GPS data. The first step is preprocessing of raw RINEX data using `ResCor` to produce RINEX format files that contain the ionospheric delay, the latitude and longitude of the 'pierce point', which is where the signal passes through the ionosphere, and the elevation and azimuth of the satellite. Then program `IonoBias` simultaneously reads these preprocessed files, making use of a solar ephemeris to limit consideration to nighttime hours. (`IonoBias` was originally developed by Dr. David Coco at ARL:UT.) It then applies all the TEC data to a least squares fit of both a simple model of the (nighttime) ionospheric TEC and all the 'satellite + receiver' biases.

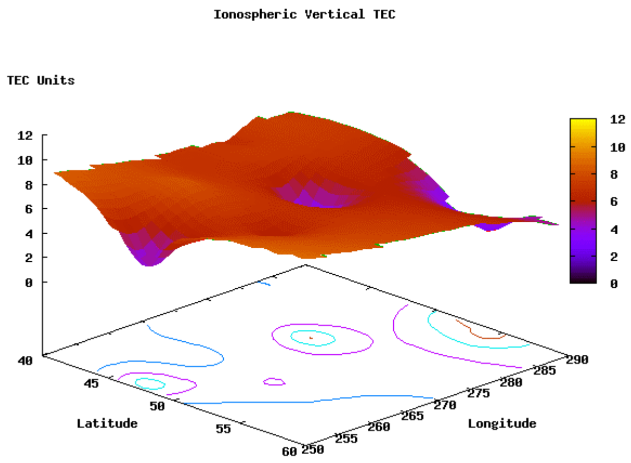

Figure 7 - A typical daytime ionospheric model.

Program `TECMaps` reads the same preprocessed RINEX files that `IonoBias` does, as well as the estimated 'satellite + receiver' biases that were output by `IonoBias`, and at each epoch uses the delay data to estimate the ionospheric TEC on each point of a horizontal grid. The result is output to a file at each epoch, in a form that can be displayed as a surface plot either in MATLAB or by `gnuplot`. Figure 7 and Figure 8 present the resulting map of ionospheric TEC for two different epochs, one near midday and one at dawn.

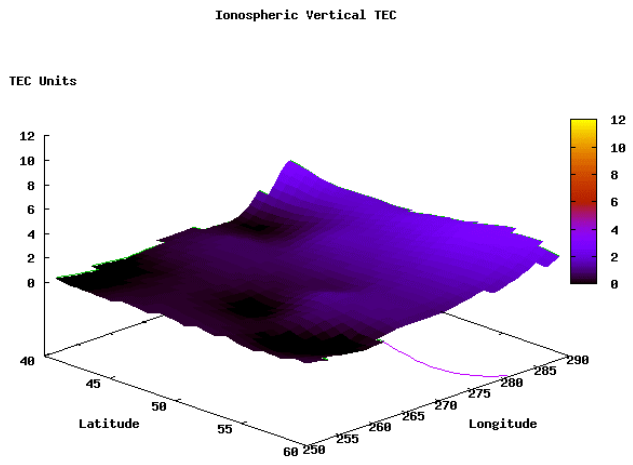

Figure 8 - The ionospheric model at dawn; note the electron density increasing west to east with the sunrise.

This computation used RINEX data downloaded from the IGS for 283 sites on day 210 of 2004. The plots were generated by `gnuplot`. Units on the vertical axis are TEC units, equal to $10^{16}$ electrons/m$^2$ along the signal path, or 0.542 ns of delay at L1. On the horizontal axes are latitude (degrees, N) and longitude (degrees, E).

### Octave Bindings

The benefits of the GPSTk can be extended beyond the users of C++. It is possible for GPSTk algorithms to be used from MATLAB, Java, perl, C, FORTRAN or any other programming language. In order to provide access to the GPSTk, it is necessary to write new functions that wrap GPSTk C++ calls. These new functions are often referred to as bindings. An example of bindings to the Octave language is provided with the GPSTk distribution, in the apps/octave directory. Octave is an open source computer algebra system, quite similar to MATLAB. [30] The Octave function `ReadRinexObsFileFast.oct` provides a concrete example of one of the benefits of binding. This function can read an RINEX observation file, at an observation sample rate of 30 seconds, and convert it into a matrix in less that 2 seconds on one of the authors' PC, an Athlon 1800 with 512 Mb RAM. A similar function written in purely Octave takes over half a minute to execute on that same PC.

## VIII. THE FUTURE OF THE GPSTk

The growth of the GPSTk will depend strongly on user contributions. The contributions can take a variety of forms, from bug reports to new and modified source code. Users within ARL:UT have already identified a number of near term goals. These include platform support for other development environments such as Mac OS X and AIX. Other near term goals include the addition of differential GPS, ambiguity resolution, and software receiver capabilities.

Growth of the GPSTk will also be driven by shifts in the GNSS community. We anticipate supporting RINEX version 2.2

soon, and any further developments of the RINEX standard. In the near term, the first satellite to provide dual frequency pseudoranges to civilians is scheduled for launch in 2005. Furthermore, the European community is creating Galileo, which will provide a public, regulated service that is compatible with GPS, essentially augmenting the current constellation with a new one. In the long term GPS will have new signals in the L5 and M code. The GPSTk, with its emphasis on fundamental observations, can provide the basis to explore and exploit these changes.

It is our hope that university students, laboratory researchers, system engineers and software developers will contribute to, as well as benefit from, the GPS Toolkit. We have already seen many benefits to using this code within our lab, and believe that the GPS community as a whole will see a similar benefit.

## IX. REFERENCES

[1] ICD-GPS-200. http://www.navcen.uscg.gov/pubs/gps/icd200/

[2] Hopfield, H. S. “Tropospheric Effect on Electromagnetically Measured Range: Prediction from Surface Weather Data.” Applied Physics Laboratory, Johns Hopkins University. Baltimore, MD, July 1970.

[3] Collins, J. P. and Langley, R. B. “A Tropospheric Delay Model for the User of the Wide Area Augmentation System.”, Geodesy and Geomatics Engineering, University of New Brunswick, Technical Report No. 187, September 1997.

[4] Bancroft, S. “An Algebraic Solution of the GPS Equations.” IEEE Transactions on Aerospace and Electronic Systems. 21.7, (1985).

[5] http://www.doxygen.org/

[6] http://www.perforce.com/jam/jam.html

[7] http://www.objectfaq.com/oofaq2/

[8] Stroustrup, B., The C++ Programming Language, Addison-Wesley Professional, 3rd edition, 1997.

[9] http://www.sgi.com/tech/stl/

[10] Evans, Alan. “Comparison of GPS Pseudorange and Biased Doppler Range Measurements to Demonstrate Signal Multipath Effects.” *4th International Geodetics Symposium on Satellite Positioning*. Austin, TX. pp. 573-588

[11] Harris, R. Benjamin. “Evaluation, Refinement and Fusion of Software-Based Pseudorange Multipath Mitigation Techniques.” *ION-GPS-2002*. Portland, OR.

[12] NMEA 0183 Interface Standard, http://www.nmea.org/

[13] http://www.unavco.org/facility/software/teqc/teqc.html

[14] Borre, K. and Strang, G. Linear Algebra, Geodesy and GPS. Wellesly, MA: Wellesly-Cambridge Press, 1997.

[15] Borre, K., "The Easy Suite - Matlab code for the GPS Newcomer," GPS Solutions 7.1 (2003): 47-51.

[16] Misra, Pratap and Enge, Per. Global Positioning System: Signals, Measurements and Performance. Lincoln, MA: Ganga-Jumana Press, 2001.

[17] http://www.bernese.unibe.ch/index.html

[18] http://gipsy.jpl.nasa.gov/orms/rtg/index.html

[19] http://bowie.mit.edu/%7Esimon/gtgk/

[20] GPS Drive project home page, http://gpsdrive.kraftvoll.at/

[21] GPS Manager project home page, http://www.ncc.up.pt/gpsman/

[22] Sharc Project home page, http://sharc.sourceforge.net/

[23] OpenSourceGPS home page, http://home.earthlink.net/~cwkelley/

[24] SourceForge main page, http://www.sourceforge.net/

[25] Tolman, B. W. and Harris, R. B., “The GPS Toolkit,” Linux Journal, September 2004, p. 72. (http://www.linuxjournal.com).

[26] http://www.ngs.noaa.gov/gps-toolbox/index.html

[27] http://www.gnuplot.info/

[28] Blewitt, G., "An Automatic Editing Algorithm for GPS Data." Geophysical Research Letters 17. 3 (1990):199-202.

[29] ActiveState perl distribution, http://www.activestate.com/Products/ActivePerl/.

[30] Octave project home page, http://www.octave.org/