2/12/00

Title Page

1. Title: On Fast Post-Processing of Global Positioning System Simulator Truth Data and Receiver Measurements and Solutions Data

2. Place and Date of Presentation: If accepted the paper will be presented at the Satellite Division of the American Institute of Navigation (ION) 13th International Technical Meeting on Global Positioning System (ION GPS 2000) in September 19-22, 2000 in Salt Lake City, Utah

3. Name of the presenter:
Semion Kizhner
National Aeronautics and Space Administration,
Goddard Space Flight Center
Electrical Systems Center
Hardware Systems Branch

6/8/00

On Fast Post-Processing of Global Positioning System Simulator Truth Data and Receiver Measurements and Solutions Data

Semion Kizhner
National Aeronautics and Space Administration

Biography

Semion Kizhner, an aerospace engineer with the National Aeronautics and Space Administration (NASA) at the Goddard Space Flight Center (GSFC), participated in the development of the Space Shuttle launched Hitchhiker carrier and payloads such as the Robot Operated Materials Processing System (ROMPS). He was responsible for establishing the Global Positioning System (GPS) test facility at GSFC and supported GPS simulations for several space projects, such as the OrbView-2 and SAC-A spacecraft. He is currently developing capabilities to access spacecrafts as nodes on the Internet and to accelerate generation of images derived from the EOS Terra spacecraft MODIS instrument and GOES-8 spacecraft data. He graduated from Johns Hopkins University with an MS degree in computer science.

Abstract

Post-processing of data, related to a GPS receiver test in a GPS simulator and test facility, is an important step towards qualifying a receiver for space flight. Although the GPS simulator provides all the parameters needed to analyze a simulation, as well as excellent analysis tools on the simulator workstation, post-processing is not a GPS simulator or receiver function alone, and it must be planned as a separate pre-flight test program requirement. A GPS simulator is a critical resource, and it is desirable to move off the pertinent test data from the simulator as soon as a test is completed. The receiver and simulator databases are used to extract the test data files for post-processing. These files are then usually moved from the simulator and receiver systems to a personal computer (PC) platform, where post-processing is done typically using PC-based commercial software languages and tools. Because of commercial software systems generality their functions are notoriously slow and more than often are the bottleneck even for short duration simulator-based tests. There is a need to do post-processing faster and within an hour after test completion, including all required operations on the simulator and receiver to prepare and move off the post-processing files. This is especially significant in order to use the previous test feedback for the next simulation setup or to run near back-to-back simulation scenarios. Solving the post-processing timing problem is critical for a pre-flight test program success. Towards this goal an approach was developed that allows to speed-up post-processing by an order of a magnitude. It is based on improving the post-processing bottleneck function algorithm using a priory information that is specific to a GPS simulation application and using only the necessary volume of truth data. The presented post-processing scheme was used in support of a few successful space flight missions carrying GPS receivers.

Introduction

For several engineering applications there is a need to perform post-processing on an experiment completion as part of test data analysis. The engineering application considered in this paper is post-processing of data, related to a GPS receiver test in a GPS simulator and test facility. Although in principle the simulator can be used to analyze the test data, the GPS receiver pre-flight test program operations require that the GPS simulator data is moved off from the simulator to a post-processing platform immediately after a test end. Moreover, before this move, some preparatory time consuming work must be done on the simulator to extract the required test parameters from the simulator database to so called truth data file(s). Each line in such a file contains a timestamp followed by a few parameters, like position or attitude at that time, and commas separate the values (*.csv files). Such file structure is compatible with most commercial data processing systems input, like Matlab (Trademark of MathWorks, Inc.) or Microsoft Excel (Trademark of Microsoft Corporation).

A few hours of simulation may result in megabyte data files. Post-processing of such large volumes of data becomes a problem.

The challenge for the GPS simulator system is to generate these files in real time or allow fast generation and extraction after a complete simulation. Also it must have the option to write to these files for a specified node on a secure Intranet.

Another challenge of a GPS receiver system is to provide more efficient tools for its archive file parameters extraction. These challenges even with the expensive GPS simulators and space flight receiver systems are still awaiting solution.

The challenge of a pre-flight test program team is to design and implement a fast post-processing scheme. Once the simulator and receiver data files are prepared and moved off to the post-processing platform, post-processing further involves certain computer operations such as loading the test data files into arrays, interpolation, computing residues and statistics, plotting and printing results. It was learned from experience that post-processing computations account for more than 90 percent of all post-processing time resources, and that computations are needed to be accomplished faster by an order of a magnitude in order to make the post-processing scheme acceptable.

This paper describes a solution to the post-processing computational problem and comprises post-processing overview (1.0), interpolation algorithms and time complexity overview (2.0), GPS engineering application post-processing problem specifics (3.0), problem formalization (4.0), direct lookup interpolation algorithm applicability (5.0), solution algorithm (6.0), and description of the timing results (7.0). The algorithm rationale for GPS position data post-processing, using the minimal and sufficient amount L of truth data, is provided in (8.0), followed by conclusions and references.

1.0 Post-Processing Overview

This paper addresses the computational problem on the precedent of interpolation computing. Interpolation is related to tests where input (reference) values for some time points are well known, while output may be measured at a slightly different set of time points. It is then necessary to perform interpolation within a subset of reference values for the measurement time points if we want to compare measurement results to input stimulus at approximately the same instant of time.

Traditionally for small data volumes the standard mathematical tables supplemented interpolation needs. In research and development some cases of experiments and tests will result in medium size data volumes. In such cases post-processing using a computer and general-purpose functions provided by commercial software is sufficient. However, general-purpose functions are notoriously slow. Applications involving signal data processing, like GPS simulations and imaging satellite operational telemetry processing, produce large volumes of data. For example, a few hours of GPS simulation may result in a 250 Megabyte volume of data and post-processing time complexity then becomes a problem. This problem must be addressed within a GPS receiver pre-flight test program in order for the program to become viable.

For example, consider the case of the pre-flight testing program of the Tensor (Trademark of Loral) GPS receiver for the SSTI/Lewis mission. The GPS receiver under test was a 1 Hz receiver producing once per second position and velocity solutions in the Earth Centered Earth Fixed (ECEF) coordinate frame

{t, x, y, z, dotx, doty, dotz}.

The receiver also produced 1 Hz attitude azimuth, elevation, height and attitude rate solutions in the spacecraft centered local geographic frame

{t, az, el, h, dotaz dotel, doth}.

In pre-flight tests, a GPS attitude simulator STR2760 (Trademark of Global Simulation Systems Inc.) was used to simulate the spacecraft's on-orbit dynamics and the GPS signals expected to be observed by the receiver's antennae in orbit. The simulator stored corresponding truth data position and attitude parameters in its database. The receiver telemetry was logged to a single archive file on the receiver control PC. After a test the simulator truth database parameters and their timestamps were extracted and written to twelve files. The receiver archive (log) file was used to extract the receiver solution parameters to 12 corresponding output files. An hour was required to generate the data files on the simulator and receiver from corresponding databases and then copy them to the post-processing platform. The post-processing program dealt with these 24 files. It was soon discovered that the post-processing of data for a shift long test was slow and that it required more than 8 hours to complete the post-processing for a 6-hour simulation test. This quickly became a significant problem in completing the test program on schedule. The performance problem was traced to the interpolation software algorithm and an effort to improve it was undertaken. The new algorithm described in this paper reduced the post-processing time from 8 hours to 20 minutes and solved the problem.

The ultimate approach that may be taken is to implement the post-processing bottleneck functions' algorithms in hardware. This may turn out to be difficult to do when the software code already exists and is large or not well documented. Another difficulty will rise when the software programming language is not conducive for hardware implementation. In any case, pre-hardware

implementation on a Field Programmable Gate Array (FPGA) platform or Digital Signal Processing (DSP) board should be preceded, in our view, by the software algorithm and code analysis with the goal of their optimization first. This paper presents such analysis and new algorithm design for a chosen bottleneck function of interpolation. This paper also demonstrates that a fast software linear interpolation algorithm is applicable to a large class of engineering applications. This becomes possible by using prior knowledge about applications, and by trivializing the problem as far as possible which simplifies problem solving. Such a class includes applications that are using a GPS simulator and test facility. A GPS simulation carries important a priory information that allows to optimize the post-processing bottleneck computations. The following describes how this information was used to optimize the post-processing bottleneck interpolation function.

2.0 Interpolation Algorithms and Time Complexity Overview

2.1 General Interpolation Algorithm Overview

There are two parts in any interpolation algorithm:

Part 1) Search for a particular time point t, at which a measurement was made for the subset of a few reference values around time t that will be used in the interpolation at time t. For linear interpolation a subset of two values is needed, and for a bilinear interpolation more than two points are required.

Part 2) Compute the interpolate reference value at time t using the subset of values found above in Part 1.

For example, in an engineering experiment a linear interpolation may involve a function of one variable y(t), where t is time. Namely, given a vector of known y: y0, y2...yN-1 at increasing times t0, t1,...,tN-1, linearly interpolate y(t) for time t at which an output measurement was obtained. The yj are also called input reference values or just reference values. The times tj and their corresponding yj may be stored in two one-dimensional matrices of size N. Pairs (tj, yj) of input reference times tj and their corresponding yj, where j=0,1,...,N-1, may also be considered as points in a plain rectangular coordinate frame (t, y(t)) as shown to the right in Figure 1. Figure 3 in Section 8.0 depicts a two-dimensional interpolation case for a spacecraft orbit. First, for each measurement time t in a linear interpolation we search for a subset pair of input reference times, such that t is inside this pair. This search corresponds to Part 1 of the interpolation algorithm, as described above. There are different ways to implement a search, but they are usually non-trivial. Then, the found reference points are used to compute the interpolate value.

2.2 One-Dimensional Linear Interpolation Overview

First, consider for this example of linear interpolation that the pair is found and let it be [t1, t2] such that t1 <= t <= t2. The corresponding subset of given reference values comprises y1 = y(t1) and y2 = y(t2). They determine two points P1 and P2 with coordinates (t1, y1) and (t2, y2) on the plane (Figure 1). Second, we compute the interpolate value y = y(t) such that point P(t, y(t)) is situated on the straight line connecting the two points P1 and P2. This computation corresponds to Part 2 of the interpolation algorithm and is quite simple. Indeed, it is known from plain analytic geometry [4] that the equation of a straight line, given by two points (t1, y1) and (t2, y2), can be expressed as

$$(t - t1)/(t2 - t1) = (y - y1)/(y2 - y1) \qquad (1)$$

By multiplying both sides of the equation (1) by (y2 - y1) we obtain

$$y - y1 = (t - t1)*(y2-y1)/(t2 - t1)$$

Whilst the solution to the linear interpolation for measurement argument time t is

$$y = (((y2 - y1)/(t2 - t1)) * (t - t1)) + y1 \qquad (2)$$

It can be seen from (2) that computation of **y** requires 6 operations for a single point interpolation. **y** can also be expressed in terms of a sum of scaled values y1 and y2

$$y = ((t - t1)/(t2 - t1))*y2 - ((t - t1)/(t2 - t1))*y1 + y1 =$$
$$((t - t1)/(t2 - t1))*y2 +(-((t - t1)/(t2 - t1))) + 1)*y1 =$$
$$((t - t1)/(t2 - t1))*y2 + ((t2 - t)/(t2 - t1))*y1$$

or

$$y = ((t - t1)/(t2 - t1))*y2 + ((t2 - t)/(t2 - t1))*y1 \qquad (3)$$

This distance-weighted form of interpolate **y** requires 9 operations.

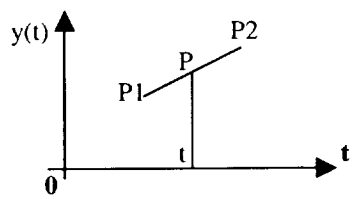

Figure 1. 1-Dimensional Linear Interpolation

2.3 Two-Dimensional Linear Interpolation Overview

A function of two variables f(x, y), often encountered in satellite image processing, may be considered as a real function of a complex variable f(z), where z = x + iy and i is the square root of −1. There are different algorithms for interpolation at a measurement point (x, y). For example, the known reference arguments (xk, yj) may be stored as complex numbers (xk + iyj) in a two-dimensional array [zkj]. The corresponding known real function reference values fkj may be stored in a two-dimensional array [fkj]. First, the search finds, for example, four points of the matrix [zkj] that are closest to measurement argument point (x,y). Let this subset be described by the reference argument matrix indices {(0, 0), (0, 1), (1, 0), (1, 1)}. Then the "center" point is assigned the value of the four points' average, namely value fc, where

$$fc = \tfrac{1}{4}(f00 + f01 + f10 + f11)$$

Point (x, y) belongs to one of the four triangles comprised by two of the four vertices and the center point. Let this triangle be comprised by {(0, 0), (0, 1), (cx, cy)}. The plane through the corresponding three points {(0, 0, f00), (0, 1, f01), (cx, cy, fc)} is used for the interpolation. The interpolate is found as the point on the plane that corresponds to argument (x, y). It is known from solid analytic geometry [4] that a plane passing through three points

{(x0, y0, z0), (x1, y1, z1), (x2, y2, z2)}

is given by the equation

$$\begin{vmatrix} \mathbf{x}-x0 & \mathbf{y}-y0 & \mathbf{z}-z0 \\ x1-x0 & y1-y0 & z1-z0 \\ x2-x0 & y2-y0 & z2-z0 \end{vmatrix} = 0 \qquad (4)$$

Interpolation value **z** for argument (**x**,**y**) is then found from this equation. As demonstrated by this example for a two dimensional case, both Parts 1 and Part 2 of the bilinear interpolation algorithm are contributing to algorithm time complexity and the selection of an interpolation algorithm for a specific application becomes even more significant compared to one dimensional case.

In another simplified example of a bilinear interpolation, four neighboring points may be used that are forming a 1x1 square. The interpolate is a sum of weighted distances where each addend is the area of one of the four rectangles made by the argument point vertical and horizontal lines inside the grid square and multiplied by the value of the diagonally opposite point. This involves near 24 computational operations in the interpolation algorithm Part 2 for a single point interpolation. In case when truth data arguments are given by a well defined and even spaced (in both dimensions) rectangular grid, the problem formulation is trivialized. It is directly solved by an algorithm of time complexity O(n) for both the one-dimensional and two-dimensional case. However in general, such mitigating grid constraints are not evident, even if they exist. It is the challenge of the post-processing design to configure the engineering problem at hand in a way that a direct solution of time complexity O(n) becomes possible.

2.4 On One-Dimensional Linear Interpolation Algorithm Time Complexity

Because computational Part 2 of the linear interpolation is simple, the complexity of the interpolation algorithm is attributed chiefly to the complexity of searching for the subset of reference points as described in Part 1. General-purpose interpolation functions, like the Matlab function interp1, also easily solve the computational Part 2 of the interpolation. However, because of such functions natural generality, their time complexity is attributed to the complexity of Part 1 or that of a binary search and is of order $O(n\log_2(N))$ [1]. In other words, if we have N reference points and n measurement points, the computational time complexity of a linear interpolation algorithm is $C*n*\log_2(N)$, where C is some large constant. This complexity is typical for an algorithm that is using a binary search to find the set of reference points. In takes a binary search in worst-case $\log_2(N)$ operations to find the pair of points needed for the interpolation computing for a time instance t, and it becomes a formidable problem to process data from even a medium duration experiment. However, it is known that the lower performance boundary for linear interpolation is O(n). It can be achieved when instead of a binary search for the set in Part 1, a direct table lookup is possible. The latter improves interp1 performance by an order of a magnitude for a long-duration experiment. The development of the algorithm for a GPS engineering application, that assures O(n) time performance, follows. This algorithm is using the GPS application specifics as a priori information which allows to optimize the post-processing bottleneck interpolation function.

3.0 GPS Engineering Application Post-Processing Problem Specifics

Consider the class of engineering applications such as the post-processing of data obtained from a terrestrial laboratory experiment involving a GPS simulator and a GPS space receiver. The GPS simulator generates a user specified spacecraft orbit and GPS radio frequency (RF) signals as would be observed by a GPS receiver on board the spacecraft. A sketch of such an orbit is depicted in Section 8, Figure 3. In such a test configuration, the GPS

receiver solution Y_r components (RF measurement based solutions) and the simulator input reference data (also called truth data, that is the digital data source of RF signals to the receiver) must be compared for identical timestamps. The processing that is needed to evaluate the receiver pre-flight performance in simulated orbit is time consuming and presents an operational problem. However, for a GPS simulation application, the problem carries a few specifics that allow the process speedup. These are described in the remainder of this section primarily as timing specifics and a GPS simulation operational configuration.

3.1 GPS simulation specifics

The GPS simulator script extracts truth data for post-processing at times T_s = t0, t1,...,tN-1 while the receiver produces solutions Y_r at time points $T_r > 0$, that are generally different from T_s. In order to compare the receiver solutions with simulator truth data, a linear interpolation Z_r of the truth data for receiver solution time points T_r must be determined.

It is specific to this GPS application that GPS simulators use an operator specified constant interval

$$\Delta = T_{s+1} - T_s$$

for time points T_s = 0, Δ, 2Δ...L/Δ (for an L second duration simulation we have N time points, where N = L/Δ + 1). Once this is observed, the attempt to implement a time complexity O(n) interpolation algorithm becomes plausible. Nominally Δ is specified as 1 second, but can also be chosen as 0.1 seconds in an attempt to get better interpolation results for a receiver that produces solutions at rates higher than 1 Hz. Evidently the selection of Δ=0.1 will generate 10 times more data compared to Δ=1.

We will demonstrate that for a GPS simulation, which provides some a priori information, the lower performance boundary O(n) for post-processing linear interpolation can be achieved. This is mainly because the GPS application generated truth time is of an atomic time standard quality that permits interpolation direct lookup. In addition we have proved that for post-processing of GPS positional data a minimal size N=L+1 seconds (Δ=1) of simulation truth data is sufficient in linear interpolation for any k $\geq$ 1, k Hz receiver in a near circular low earth orbit simulation. Following is the more detailed description of the GPS simulation timing specifics and problem formalization, as well as Figure 2, that represents the block-diagram of a GPS simulation test configuration and related data flow.

3.2 GPS Simulator timing specifics

It is assumed that a common time format, namely that of the GPS time, is used by the GPS simulator and receiver under test, and that for post-processing purposes the time is further transformed to the number of seconds into a simulation. Timing issues in GPS applications are subtle. On the GPS simulator the test scenario start time in GPS time format may be chosen as test time seconds count 0. This allows continuing counting of seconds into test, including a GPS week update case. This time count parameter is stored in the simulator database and can be requested as the post-processing truth files data timestamp. On the GPS receiver side, the solution time is also available in GPS time format and can be converted to seconds-into-the-test format by subtracting from it the simulator test scenario start time. The post-processing addressed in this paper deals with a complete test, starting at seconds count 0. The matters become more complex when a test scenario needs to be restarted at some time offset ts into the scenario in order to investigate a receiver problem that is partially attributed to orbit and GPS constellation state at time ts. This can be handled by requesting from the simulator database the spacecraft state vector at time (T0 + ts), where T0 is the nominal test scenario start time, and restarting the scenario with a new start time specified as (T0+ts) and using the state vector at (T0+ts) for spacecraft orbit initialization.

3.3 GPS receiver timing specifics

On the receiver side, the time complications are usually attributed to the receiver performance issues. It is often the case that a time based device performance problem manifests itself as "a starry sky" phenomenon. For example, space GPS receivers produce a one pulse per second analog signal (1 PPS) when a position solution (fix) is attained by the receiver. This pulse is aligned with the Greenwich Mean Time (GMT) second boundary (GPS time is also aligned with GMT) within one microsecond and this pulse may be used for spacecraft clock synchronization to GMT time. However, if a position fix is missed because of a performance glitch, even in a terrestrial test using GPS sky signals, the 1 PPS output is not assured. If the edge of the 1 PPS output from a receiver is compared to an atomic time standard corresponding edge that was independently aligned on a GMT second boundary, a spike point (star) will occasionally appear on the time difference graph. Under-performance occurrences within a receiver in an orbit simulation are more prominent and may result in a graph resembling "a starry sky" or, in other words, "no fix no time tics". In this paper we are more concerned with a receiver and its control computer over-performance, resulting in receiver timestamp duplicates. The duplicates must be removed in the post-processing initial step, mainly for computation of meaningful statistics. The assumption is that the GPS simulator data timestamps and receiver timestamps that are used as inputs to the post-

processing algorithm are each an increasing time series without duplicates, and all receiver timestamps are within the simulation time interval. Once the timing issues are resolved, the subject post-processing consists of basic operations: load input data files into arrays, interpolation, compute the truth and solution data differences and statistics, plot and print the plots. The interpolation in this basic post-processing is the only bottleneck computational operation. The solution of this computational problem is described in the rest of the paper.

3.4 GPS Application Hardware Configuration Specifics

The setup for a test within the GPS simulation and test facility may vary from mission to mission. However, for a given mission, once the basic setup has been decided upon, the configuration is frozen and any changes to it are strictly controlled. Below is the picture of a typical GPS simulation test program configuration. The truth data resides on the simulator workstation disk system, and the receiver telemetry is logged on the receiver control PC. The receiver solution parameters and corresponding timestamps may be passed in real-time to the simulator workstation for recording in the simulator database. The GPS simulation system configuration carries the specifics of a closed loop system and this can be used in post-processing optimization, namely for post-processing files generation in real-time, and fast transfer to the post-processing platform using an Ethernet based secure local area network.

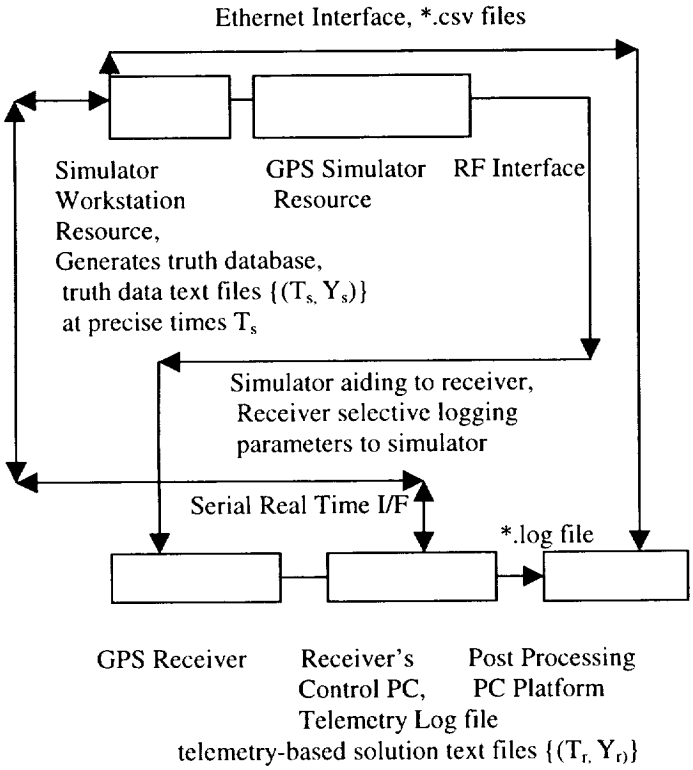

Figure 2. GPS Simulator/Receiver Test Setup

4.0 Problem Formalization

Let Y be a function of real variable t, Y=Y(t), and Y_s be a vector of Y(t) function values at argument t values given by vector T_s or $Y_s = Y(T_s)$ for all s. Both vectors (one dimension matrices) T_s and Y_s are of size N or s=0,1,2,...,N-1. The problem is to find a fast linear interpolation of Y(t) within matrices T_s, Y_s at argument points given by a matrix T_r of size n or r=0,1,2,...,n-1. Matrices T_s and T_r are a priori known to be each an increasing monotonic time series and such that $\min(T_s) < T_r < \max(T_s)$ for all r. The Matlab (Trademark of Matlab) Version 4.2c.1 function interp1(T_s, Y_s, T_r) easily solves the interpolation Part 2 of the problem and returns the solution as a size n matrix Z_r = interp1(T_s, Y_s, T_r). However, because of its natural generality, interp1 Part 1 computational time complexity is $O(n\log_2(N))$ and it takes hours to process not very large experimental test data files. This is the problem and the solution it to construct a linear search algorithm that allows direct lookup in the search for the interpolation set of points. Although N is not a computational performance factor in a direct lookup algorithm, any reduction in size N of data, sufficient to accomplish interpolation within required accuracy, is also of significant value. It allows running longer duration tests by using less memory and disc space on a system where disc space is a critical resource. It also requires less time in test data file migration off the simulator to a post-processing PC platform.

5.0 Direct Lookup Interpolation Algorithm Applicability

Following is the description of the a priori information used to prove the direct lookup interpolation algorithm applicability for the class of GPS simulation applications.

5.1 Simulator timestamps as lookup indices

It is a priori known from the GPS application that in an L second duration GPS simulation T_s is a consecutive count $T_s = 0, \Delta, 2\Delta...L/\Delta$ of seconds into the GPS simulation starting at 0. Although this is a dynamic application, the time is of an atomic time standard quality and is precise. This is one of the main reasons that make the direct lookup possible in this dynamic application because these timestamps may serve as direct lookup indices.

5.2 1 Hz receiver timestamps as lookup indices

It is also a priori known that the elements T_r are accurate times at which the receiver found solutions. These times

can be expressed as a fractional decimal number K.mmm where K is seconds and mmm are milliseconds relative to the GPS simulation start time. Indeed, K.mmm can be obtained by subtracting the receiver GPS time from the simulator start GPS time in seconds. This allows using the receiver timestamp's non-zero integer part K as a direct lookup index into the simulation truth data matrix Y_s. It is known that $T_r(1)$ is greater than 0 because it takes some time for a receiver to find the first solution. It is known that $T_r(n)$ is smaller than $T_s(N)$ because the receiver test is operationally stopped before the simulator test scenario end. This ensures that K is always within the T_s bounds and is always a legitimate direct lookup index.

5.3 10 Hz receiver timestamps as indices

For a 10 Hz receiver (a high performance receiver provides 10 solutions in a second) the matrix T_r must be transformed into a new matrix $10*T_r$ that is then used in the algorithm instead of T_r. This expands the algorithm applicability for high performance GPS receivers.

5.4 Minimum and sufficient volume of simulation data

It is proved below that for a near circular low earth orbit the largest positional error of the linear interpolation within a second of simulation time points pair $[T_s, T_{s+1}]$, where $T_s < T_r < T_{s+1}$, is far smaller than the required receiver accuracy for position solutions $Y_r(T_r)$. This allows for a k Hz receiver with $k \geq 1$ to restrict the necessary size of the position truth data matrix Y_s to the minimum, namely to the use of data for time counts of seconds $T_s = 0, 1, 2...(N-1)$. As a consequence 10 times less disc space is required for storage of truth position data files on the simulator, faster downloading of the files from the simulator to a post-processing PC platform is possible, and less computer memory is needed to store the data matrices, compared with unnecessary truth position data generation at $\Delta = 0.1$ seconds resolution in a post-processing script.

6.0 Solution Algorithm

6.1 Direct Lookup Linear Interpolation Algorithm

```
n = size(Tr, 2)
% returns number of columns in Tr,
% while size(Tr) gives #rows and #col
for J= 1:1:n
  K = fix(Tr(J)); %returns integer part of Tr > 0
  Zr(J) = (Ys(K+1) - Ys(K)) * (Tr(J) - K) + Ys(K);
% Tr=K.mmm for K seconds and mmm milliseconds
End
```

6.2 Algorithm Discussion

The apparent triviality of the algorithm is deceptive and the proof of its applicability is more subtle than trivial as can be seen from preceding Section 5.0. Note that symbol % in the algorithm denotes a comment. In the above algorithm the receiver T_r timestamp's integer part K is used trice as the index for a direct lookup of the truth data matrix Y_s. The algorithm expression

$$(Y_s(K+1) - Y_s(K)) * (T_r(J) - K)$$

describes the truth data difference (for a time interval Δ) divided by 1000 milliseconds and multiplied by mmm milliseconds. Matrix Ts is not even used in the new algorithm because of a priori knowledge of the GPS simulation integrity. The better than an order of magnitude performance improvement is achieved in this algorithm for any of the 12 data type files considered in the GPS application. These data types were described above in Section 1 and in [3]. This algorithm is imbedded in the larger GPS post-processing application algorithm, that generally comprises the steps described in the following section.

6.3 GPS Simulation Data Post-Processing Script

6.3.1 Start the simulation task for duration L seconds with logging option on both the GPS simulator and receiver.

6.3.2 At time L – 300 seconds stop the receiver and save the receiver telemetry log file.

6.3.3 At time L the simulator automatically completes the simulation and saves the simulator truth database single file.

6.3.4 Run a post-processing script on the simulator to extract the required twelve parameters of the truth data with their timestamps to twelve comma-separated files at specified resolution Δ. This script is a small text file created by operator in the GPS simulator output control language, where operator specifies time resolution Δ, desired for analysis time interval start and stop times, parameter names and corresponding output file names. As a result the truth data text files are produced from the simulator database file, each file containing pairs of a timestamp and truth parameter value within the specified time interval. These text files are compatible with major commercial software systems like Matlab. Move off the files to the post-processing platform. The simulator challenge is to further improve this process as outlined in the Introduction.

6.3.5 Move off the receiver log file to the post-processing platform and run a vendor provided receiver utility to extract from the receiver telemetry log file the corresponding telemetry parameters or solution data, for example $\{T_r, X_r\}$ to a comma-separated text file, where Tr is time of the solution and X_r is the corresponding solution position or velocity component X in ECEF coordinate frame. The receiver challenge is to further improve this process as outlined in the Introduction.

6.3.6 On the post-processing platform run a script (for example a Matlab script) to transform T_r from $\{T_r, X_r\}$ into form K.mmm by subtracting from it the GPS simulator scenario start time, where K is seconds and mmm are the milliseconds into the simulation test. For a k Hz receiver, where k > 1, the resulting K.mmm must be multiplied by Δ=10. This operation is fully described in Section 5.3.

6.3.7 Run the post-processing script that invokes the direct lookup interpolation algorithm described above in 6.1 for each pair of truth and measurement data files. This script, when running on the post-processing platform, will load the simulator truth and receiver solution data files into arrays, perform the solution data fast interpolation, compute the truth and solution data differences (residues) and statistics, plot the results and print the plots and the statistics.

6.3.8 Use the truth and the interpolate data plots and the statistics to complete the receiver under test performance analysis.

All above steps except step 6.3.7 interpolation are of complexity C + O(n), where C is some time value required to read in the files and print the plots. Achieving complexity O(n) in the interpolation step 6.3.7 makes the entire post-processing application algorithm one with complexity O(n) and solves this GPS application class time performance problem.

7.0 Description of Timing Results

Table 1 is summarizing the timing results from tests performed using a commercial interpolation function and the proposed algorithm.

N	n	interp1 (seconds)	new algorithm (seconds)	speed up factor
05000	02500	3*60	60	03
10000	05000	5*60	75	04
20000	10000	18*60	67	16
40000	21600	85*60	85	60

Table 1. Algorithms Timing Results

N = 2*n was chosen to account for a twice better performance of a 200 MHz PC that was used in original utilization of interp1 function. The timing results of Table 1 were obtained on a 100MHz PC for a 1 Hz receiver and truth data set. The interp1 and this new algorithm interpolation results were compared point-by-point and proved to be identical for a representative test case. Also note that in the above tests for N = 21600 the $\log_2(21600) = 14.382$, and thus performance of interp1 is of order larger than 14*C* n.

8.0 Minimum and Sufficient Volume of Position Reference Data

We are going to prove that for post-processing of position data it is sufficient to use a matrix Ys of minimum size L for time points $T_s = 0,1\ldots(L-1)$. L is the GPS simulation duration in seconds and is the minimum data volume that allows achieving the required processing accuracy for position data. This also supports the approach of using just linear interpolation in simulation data post-processing of position and attitude data and data rates.

Let γ be the spacecraft orbit motion angle rate as shown in the following Figure 3. The spacecraft orbit period P for an orbit with a semi-major axis R, as described in [2], is

$$\mathbf{P} = 0.000\ 0165\ 87 * R^{-3/2} \text{ min, where R is in km} \quad (5)$$

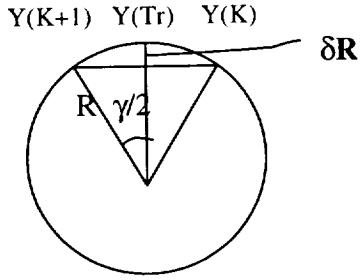

Figure 3. 2-Dimensional Interpolation Sketch

It can then be observed that

$$\gamma/2 = 360/(P*60) \text{ degrees/second} \quad (6)$$

It can also be shown using circle trigonometry that for the radial position difference δR between truth and a linear interpolation data points, the maximum deviation within a

one second interval (namely at a second interval middle point), is

$$\delta \mathbf{R} = R * (1 - \cos(\gamma/2)) \qquad (7)$$

For example, for R = 7200 km, we estimate correspondently from (1), (2) and (3) that

$$\mathbf{P} = 101.3366 \text{ min}$$
$$\gamma/2 = 0.029 \text{ deg/sec}$$

$$\delta \mathbf{R} = 0.98 \text{ m} \qquad (8)$$

Since

$$\delta \mathbf{R}^2 = (\delta x^2 + \delta y^2 + \delta z^2)$$

it can be observed that

$$\max(|\delta x|, |\delta y|, |\delta z|) \leq \delta \mathbf{R} \qquad (9)$$

In other words, magnitudes $|\delta x|$, $|\delta y|$, $|\delta z|$ of coordinate component deviations are less or equal than $|\delta \mathbf{R}|$ obtained in (8), that in turn is better than the required commercial receiver performance for position accuracy of a few meters. This proves that the GPS simulator truth position data components (x, y, z) in ECEF frame, generated at a 1 second resolution, are sufficient to perform linear interpolation for a k Hz GPS receiver, where $k \geq 1$ or that the minimum and sufficient volume of reference position data is equal to L, the GPS simulation duration in seconds.

Conclusions

In a GPS simulation the computational time of post-processing is a critical resource and is of important considerations. It has been demonstrated that in the GPS simulation data post-processing domain, the computational performance of the considered algorithm is better than that of general-purpose algorithms by an order of a magnitude. This algorithm solved the post-processing problem at the GSFC GPS test facility. It has also been demonstrated that for a GPS simulation of a near circular low earth orbit, and for any high performance receiver, the sufficient size of truth position data files required for post-processing is also the optimal size L, where L is the duration of the GPS simulation in seconds. This further reduced the time required for post-processing. The algorithm was used in support of pre-flight qualifications of GPS receivers for many missions including SAC-A [3], SeaStar, AMSAT and others. It was also implemented in programming systems, other than Matlab, for example, in C++. Future work of interest may be in determining the minimum and sufficient volume of truth data required for post-processing of position rates and attitude data, as well as in hardware implementation of portions of the post-processing task using DSP.

References

[1]. A. K. Dewdney
The Turing Omnibus
Computer Science Press, 1989

[2]. Wiley J. Larson and James R. Wertz (editors)
Space Mission Analysis and Design
Second Edition,
Microcosm, Inc and Kluwer Academic Publishers Inc., 1995

[3]. S. Kizhner, E. Davis, R. Alonso
Pre-Flight Testing of Spaceborne GPS Receivers Using a GPS Constellation Simulator,
Proceedings of ION GPS '99, 14-17 September 1999, Nashville, TN pages 2313-2323

[4]. CRC Standard Mathematical Tables, 26th Edition, Editor W. H. Beyer, CRC Press, 1981