Network-based High Accuracy Positioning with the GPSTk

Salazar, D., Hernandez-Pajares, M., Juan-Zornoza, J.M., Sanz, J.
Group of Astronomy and Geomatics (gAGE/UPC)
Universitat Politecnica de Catalunya. Barcelona, Spain
Email: dagoberto.jose.salazar@upc.edu

***Abstract*—This paper presents an implementation of a post-process kinematic positioning technique similar to PPP (Precise Point Positioning) but based on a network of stations. This technique is independent of precise clock information because it estimates satellite clock offsets 'on-the-fly', and thence it only needs reasonable accurate orbits information (for instance, IGS precise, rapid or even predicted products) to work. Moreover, with this approach the solution rate is only limited by data rate, and not by the availability of precise satellite clock data rate as it is the case with classical kinematic PPP techniques.**

This procedure is referred to in this paper as 'Precise Orbits Positioning' (POP). Similar methods already exist in the literature, such as 'network-based clocks' and 'phase interpolation'; however, with POP method both predicted and rapid IGS orbits may also be used (not only precise products) without noticeable degradation of positioning results, which is an important advantage.

The POP procedure involves multiple stations separated hundreds of kilometers and there are a great number of unknowns of several kinds. In order to ease the implementation of such system some GPSTk-provided facilities were used, including a run-time programmable general solver. A contribution of this work is that a reference implementation is freely available at the development version of the GPSTk source code, facilitating its use and modification by other GNSS researchers.

The POP results were very similar (as expected) to the standard kinematic PPP strategy, but yielding a higher positioning rate. This higher positioning rate opens the way for post-process kinematic positioning of vehicles that usually operate very far from reference stations, such as aircraft. Also, our experiments with this network-based processing strategy show additional robustness in their results, even for receivers outside the network area with long baselines.

***Index Terms*—PPP, Kinematic positioning, POP, GPSTk.**

I. Introduction

Kinematic positioning using GPS is an important research line, and in particular airborne kinematic GPS positioning is a tough problem with an extense literature ([1], [2], and [3], to cite only a few). Several different techniques have been applied, ranging from pseudorange-based DGPS to carrier phase-based techniques such as Real Time Kinematic (RTK), network-based RTK, and Precise Point Positioning (PPP), among others.

Among the differential techniques, RTK typically yields the best accuracy (at the centimeter level, when ambiguities are fixed), but it needs reference stations near the operation area (closer than 20 km for adequate performance), while pseudorange-based DGPS operates well with reference stations more than 100 km away, at the expense of decreased accuracy (at the meter level). Network-based RTK techniques like Virtual Reference Station (VRS) fill an intermediate niche with ambiguity fixing-level accuracy at about 50 km range from nearest reference station.

On the other hand, Precise Point Positioning (PPP) [4] is a strategy that avoids the expense and logistics of ad hoc reference stations (of course, PPP needs precise ephemeris products generated by an extense network of IGS reference stations, but they are already in place). However, it has the limiting factor that solution rate is set by the availability of precise satellite clock offsets, given that precise orbits can be interpolated without losing accuracy, whereas satellite clock offsets can not.

The former has been a recurrent problem to apply PPP techniques to kinematic positioning. Nevertheless, relatively recent developments have allowed data processing centers such as CODE to generate precise satellite clock corrections with high data rates (typically 30 s, and more recently down to 5 s), using phase-consistent interpolation of precise 5-minute clock results ([5]).

However, in this paper a completely different approach will be carried out: satellite clock offsets will be estimated *on-the-fly*. This procedure is independent of precise clock corrections and, therefore, it can achieve an arbitrary positioning rate (only limited by observation data rate), opening a window of opportunity to very interesting precise positioning techniques.

In order to achieve this in an efficient and reusable way, this paper relies on the facilities provided by the GPSTk [6], and in particular on the *GNSS Data Structures* data processing paradigm explained at [7]. The use of the open source GPSTk-provided tools is an important advantage for researchers, because in this way they have a reference implementation available to test and experiment with.

II. POP description

The POP procedure starts with selecting a set of reference stations and setting one of them as the *MASTER* station. *Master*'s clock will be set as the reference for the network, so all the other clocks will be computed with respect to it. The other unknowns for the *master* station will be the zenith tropospheric delay and the ambiguities.

978-1-4244-8739-4/10/$26.00 ©2010 IEEE

Therefore, the corresponding equations for pseudorange and phase are:

$$PrefitPC_0^j = tmap_0^j.ztd_0 - c.dt^j \quad (1)$$

$$PrefitLC_0^j = tmap_0^j.ztd_0 + Bc_0^j - c.dt^j \quad (2)$$

Where:

- $PrefitPC_0^j$ and $PrefitLC_0^j$: These values are, respectively, the prefilter residual (observation minus modeled effects) of ionosphere-free pseudorange and phase combinations for satellite SV^j and master station 0.
- $tmap_0^j$: Tropospheric mapping function (Niell).
- ztd_0 : Zenith tropospheric path delay.
- $c.dt^j$: Relative clock delay between satellite SV^j and master station 0, in meters.
- Bc_0^j : Ionosphere-free carrier phase ambiguity.

The other "*reference*" stations will have similar equations, but adding their clock offsets (with respect to *master* clock) as an additional unknown. Hence,

$$PrefitPC_k^j = tmap_k^j.ztd_k + c.dt_k - c.dt^j \quad (3)$$

$$PrefitLC_k^j = tmap_k^j.ztd_k + Bc_k^j + c.dt_k - c.dt^j \quad (4)$$

Where $c.dt_k$ is the relative clock delay between reference station k and master (in meters).

Finally, the "*rover*" receiver will have an equation similar to the standard PPP process, but adding the estimation of satellite clock offsets:

$$\begin{aligned} PrefitPC_r^j &= \Big(\frac{x_{r0}-x^j}{\rho_{r0}^j}\Big)dx \\ &+ \Big(\frac{y_{r0}-y^j}{\rho_{r0}^j}\Big)dy \\ &+ \Big(\frac{z_{r0}-z^j}{\rho_{r0}^j}\Big)dz \\ &+ tmap_r^j.ztd_r + c.dt_r - c.dt^j \end{aligned} \quad (5)$$

$$\begin{aligned} PrefitLC_r^j &= \Big(\frac{x_{r0}-x^j}{\rho_{r0}^j}\Big)dx \\ &+ \Big(\frac{y_{r0}-y^j}{\rho_{r0}^j}\Big)dy \\ &+ \Big(\frac{z_{r0}-z^j}{\rho_{r0}^j}\Big)dz \\ &+ tmap_r^j.ztd_r + Bc_r^j \\ &+ c.dt_r - c.dt^j \end{aligned} \quad (6)$$

Where (x_0, y_0, z_0) is the a priori *rover* receiver position, (x^j, y^j, z^j) is the position of satellite SV^j, and parameters (dx, dy, dz) are the corrections to (x_0, y_0, z_0).

It can be seen that the connection between receivers is achieved by the simultaneous estimation of satellite clock offsets. In turn, this procedure is the key that allows rover precise positioning without precise satellite clock products.

Although the observations are not explicitly differentiated, the systems of Equations 1 to 6 is equivalent to a carrier phase-based differential DGPS system using the ionosphere-free combination of observations. The simultaneous estimation of the satellite clock offsets allow them to become the ligatures between the equations.

III. POP IMPLEMENTATION

Please note that implementation of an equation system for Equations 1 to 6 is a complex task. This system involves multiple stations separated hundreds of kilometers and there are a great number of unknowns of several kinds: Some unknowns are *receiver-indexed* (or *receiver-specific*, like. ztd_i, dx, dy, etc.), some are *satellite-indexed* (dt^j), and others are both *receiver-* and *satellite-indexed*, like Bc_i^j. Therefore, the number of unknowns at a given epoch has a wide variation depending on the available station data and the number of visible satellites.

One of the contributions of this work is to provide the users some tools to deal with this kind of implementation problems in an easy, efficient and reusable way. The open source GPSTk provides a class, `SolverGeneral`, to help implementing this kind of systems. The idea behind `SolverGeneral` is that equations and variables are *described* (as opposed to being hard coded in the software), indicating their stochastic models and relationships.

Then, at each epoch the `SolverGeneral` object will match the incoming data (observations and ephemeris) with the equations and variables descriptions, building the appropriate equation system for that epoch.

Implementation starts with declaration and initialization of the `Variable` objects to be used, as well as their associated stochastic models:

In the following code, lines #1 to #3 set the stochastic models to be used. Line #4 declares a `Variable` called `dLat`, of `TypeID` "`dLat`", with a white noise stochastic model (kinematic positioning). The first "`true`" parameter indicates that this Variable is "*source-indexed*" (i.e., it is a distinct variable for each `SourceID`, i.e., receiver), and the following "`false`" parameter tells that it is *not* "*satellite-indexed*", meaning that the same variable will be used for all visible satellites. The final numeric value (100.0) sets the initial sigma. Variables `dLon` and `dH` (lines #5 and #6) follow the same pattern.

```
1   WhiteNoiseModel coordinatesModel( 100.0 );
2   TropoRandomWalkModel tropoModel;
3   PhaseAmbiguityModel ambiModel;

4   Variable dLat( TypeID::dLat, &coordinatesModel,
                   true, false, 100.0 );
5   Variable dLon( TypeID::dLon, &coordinatesModel,
                   true, false, 100.0 );
6   Variable dH( TypeID::dH, &coordinatesModel,
                 true, false, 100.0 );

7   Variable cdt( TypeID::cdt );
          // Force coefficient (1.0)
       cdt.setDefaultForced(true);

8   Variable tropo( TypeID::wetMap, &tropModel,
                    true, false, 10.0 );

9   Variable ambi( TypeID::BLC, &ambiModel,
                   true, true );
          // Force coefficient (1.0)
       ambi.setDefaultForced(true);

10  Variable satClock( TypeID::dtSat,
                       false, true );
          // Set coefficient
       satClock.setDefaultCoefficient(-1.0);
          // Force coefficient
       satClock.setDefaultForced(true);

11  Variable prefitPC( TypeID::prefitC );
12  Variable prefitLC( TypeID::prefitL );
```

Line #7 declares `cdt`, the `Variable` representing receiver clock offsets. The defaults are used (white noise model, source-indexed, not satellite indexed, big preset sigma), and it is forced to always use the value "1.0" as coefficient (by default, coefficients are looked for inside the GDS).

Declaration of variables `tropo`, `ambi` (ambiguities), and `satClock` (SV clock offsets) are similar, with the exception that ambiguities are *source-* and *satellite-indexed*, whereas satellite clocks are only *satellite-indexed*. The last couple of lines (#11, #12) declare default, dummy "variables" representing the independent terms of equations, `prefitPC` and `prefitLC`.

Again, it is important to emphasize that in the former procedure the variables *characteristics* were *described*, instead of *declaring* a variable for each possible receiver-satellite combination.

Once the `Variables` are properly declared and initialized, it is the turn of describing the `Equation` objects. First, let's declare the equations for *master* station:

```
1   Equation equPCMaster( prefitPC );

2   equPCMaster.addVariable( tropo );
3   equPCMaster.addVariable( satClock );
4   equPCMaster.header.equationSource=master;

5   Equation equLCMaster( prefitLC );

6   equLCMaster.addVariable( tropo );
7   equLCMaster.addVariable( satClock );
8   equLCMaster.addVariable( ambi );
9   equLCMaster.header.equationSource=master;

10  equLCMaster.setWeight( 10000.0 );
```

Line #1 declares the `Equation` object for pseudorange, setting the independent term type. Then, lines #2 and #3 add the variables to the equation and finally line #4 sets what receiver (data source) this equation applies to: `master` is an object of class `SourceID` holding the information corresponding to the *master* station.

Declaration of the equation for carrier phase is very similar, except for line #8, that adds an additional variable (`ambi`), and line #10 that sets the *relative weight* of this equation: the carrier phase sigma is 100 times smaller, so the associated weight is 100*100 times larger.

Equations for reference stations and rover receiver are declared in the same way. However, it must be noted that reference stations form a `SourceID` set, instead of a single station, so they need an additional treatment. Thus `equPCRef` and `equLCRef` are the equations for the reference stations' pseudorange and carrier phase, respectively:

```
1   equPCRef.header.equationSource =
                        Variable::someSources;

2   equLCRef.header.equationSource =
                        Variable::someSources;

3   for( std::set<SourceID>::const_iterator
         itSet = refStationSet.begin();
         itSet != refStationSet.end();
         ++itSet )
4   {
5      equPCRef.addSource2Set( (*itSet) );
6      equLCRef.addSource2Set( (*itSet) );
7   }
```

The special `SourceID` object called "`Variable::someSources`" indicates that equations `equPCRef` and `equLCRef` will apply to more than one data source. Thus, it is necessary to add those data sources to each equation's internal set. The "`for`" loop spanning from line #3 to line #7 achieves this in a general, reusable way.

Finally, once all the `Equation` objects, and their corresponding `Variables`, have been described, they are added to an `EquationSystem`, which in turn feeds a `SolverGeneral` object:

```
1   EquationSystem equSystem;

2   equSystem.addEquation( equPCRover );
3   equSystem.addEquation( equLCRover );
4   equSystem.addEquation( equPCRef );
5   equSystem.addEquation( equLCRef );
6   equSystem.addEquation( equPCMaster );
7   equSystem.addEquation( equLCMaster );

8   SolverGeneral solver( equSystem );
```

From now on, object `solver` is an Extended Kalman Filter (EKF) configured to solve the defined equation system (`equSystem`), building its internal matrices and vectors automatically according to the incoming data. It just needs to be fed with the appropriate GDS.

A reference implementation of the POP algorithm is freely available as open source software in the development version of the GPSTk, at the `examples` directory. Please refer to the GPSTk website (http://www.gpstk.org) for details about downloading and installing the development version.

IV. POP DATA PROCESSING

The approach to this multi-station problem is to pre-process all the stations, one by one, applying the standard modeling of PPP processing (see [4] for the general model and [7] for a reference implementation), but without applying the solver object.

The results from this preprocessing are stored in an appropriate multi-epoch, multi-station GNSS data structure that automatically takes care of all indexing (structure `gnssDataMap` is used for this). Then, an epoch-worth of data is extracted each time from the `gnssDataMap` GDS and fed to solver, and the results are printed.

For this experiment, 5 IGS stations were used: `ACOR`, `MADR`, `SCOA`, `SFER` and `TLSE`, forming a network across Iberian Peninsula spanning 1023 km (`SFER-TLSE`). Station `ACOR` was set as the "*master*", while `MADR` was the "*rover*", 392 km away from nearest reference station (`SCOA`). This network comprises more than 580,000 km^2 and can be seen in Figure 1.

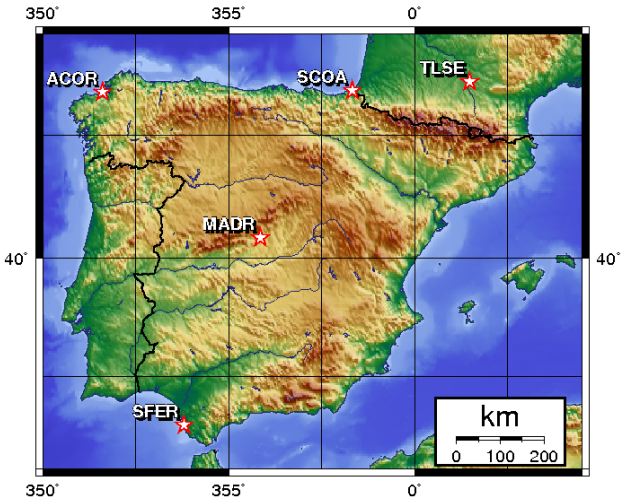

Fig. 1. POP network.

Standard IGS products (precise orbits and satellite clocks) with a 900 s data rate were used, but the data was processed at 30 s, the rate given by the RINEX observation files. Note again that in this case the IGS satellite clocks were not interpolated, but ignored: The SV clocks used for this POP positioning were estimated on-the-fly.

Figure 2 shows the good results from this approach, presenting both the 3D-error in position (with respect to the known IGS of POP, and the 3D-error for the standard kinematic PPP processing.

The results are very similar, as was expected: a 3D-RMS of 0.046 m for the kinematic PPP case versus a 3D-RMS of 0.049 m for the POP case (from 2 h onwards), but POP yields a higher positioning rate.

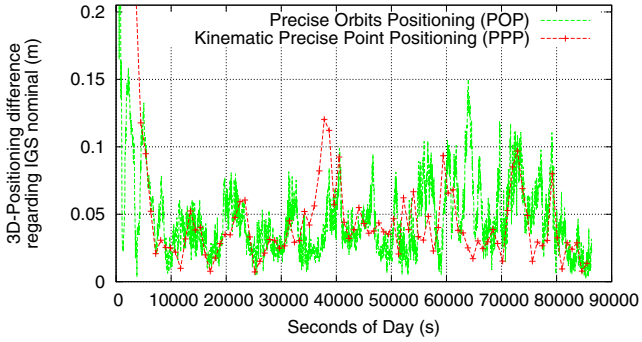

Fig. 2. POP versus kinematic PPP processing. MADR 2008/05/27.

As previously said, although the observations are not explicitly differentiated, the POP procedure is equivalent to a carrier phase-based differential system using the ionosphere-free combination of observations. However, it is a network-base processing and this provides additional robustness to the results, even when using long baselines and for receivers outside the network area.

Take, for instance, the network shown in Figure 1 but with `TLSE` station as "*rover*" and station `MADR` as just another reference station. In this case, `TLSE` will be outside the network area and 257 km away from nearest reference station (`SCOA`).

The 3D-position error from this new processing is shown in Figure 3, and it can be seen that in this case the POP solution behaves better between epochs 35000 s and 50000 s, when some problem is affecting the kinematic PPP solution (at this epoch, TLSE receiver suffered from the sudden lost and posterior gain of 2 satellites). 3D-RMS values (from 2 h onwards) are 0.069 m for PPP and 0.044 m for POP.

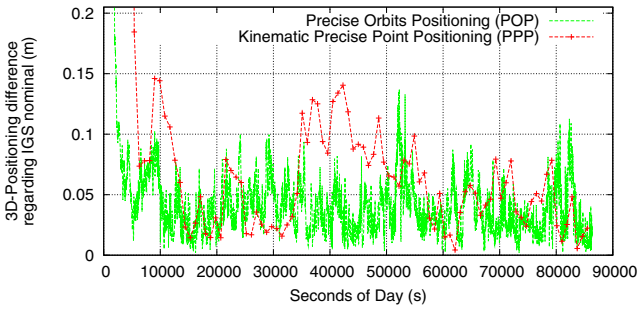

Fig. 3. POP versus kinematic PPP processing. TLSE 2008/05/27.

Also, distance from "*rover*" to nearest reference station does not seem to be a critical factor. If station `SCOA` is taken out from Figure 1 leaving a 4 station network (including

"*rover*") with TLSE still as "*rover*" and station MADR as nearest reference station (588 km away), the results are not significantly degraded as Figure 4 shows: In this case, the POP 3D-RMS values (from epoch 7200 s on) barely increases from 0.044 m to 0.049 m.

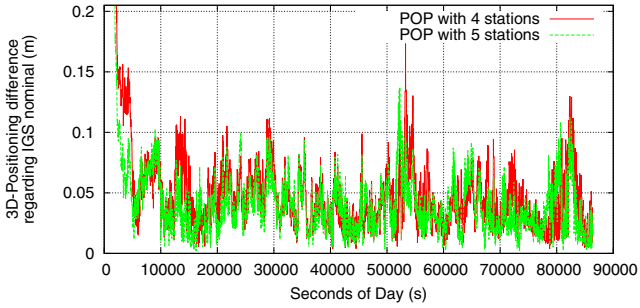

Fig. 4. POP results for 4 and 5-stations networks. TLSE 2008/05/27.

The important aspect here is that the satellites being used should be in view from as many network stations as possible, because that will provide better on-the-fly estimations of the satellite clock offsets. When using the POP strategy with only two stations (MADR as "*master*" and TLSE station as "*rover*"), the data processing effectively becomes the aforementioned carrier phase-based DGPS with a 588 km-long baseline. With such a long baseline the results will degrade, given that the estimations of satellite clocks will not be as accurate, and there will be satellites that are not common for both stations.

Figure 5 illustrates this case. The POP 3D-RMS values (from 2 h on) for the 2 station processing raises to 0.061 m (compared with 0.049 m of the 4 station case).

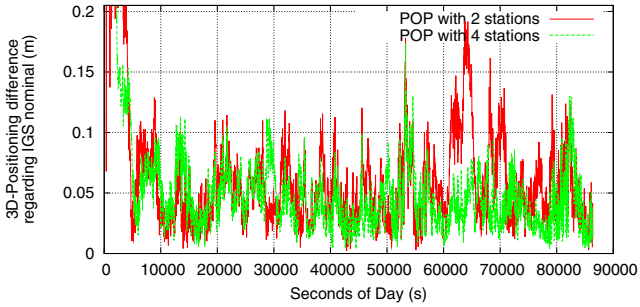

Fig. 5. POP results for 2 and 4-stations networks. TLSE 2008/05/27.

Regarding convergence time, Figure 6 plots the resulting 3D-RMS of error as function of the epoch since it is computed and the data processing strategy used. The values shown correspond to starting computing the 3D-RMS of error from 1800 s, 3600 s, 5400 s and 7200 s (i.e., 30 min, 1 h, 1 h:30 min and 2 h).

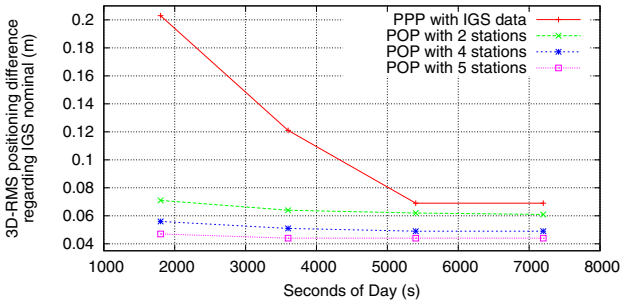

Fig. 6. Convergence time. TLSE 2008/05/27.

Convergence accelerates as the number of station increases, but up to some point, and the same can be said about the improvements in the 3D-RMS error figure, suggesting that the improvements achieved by having more observations available to estimate satellite clock offsets reach a limit shortly after 5 or 6 stations (for a network of this size). The results for a standard PPP processing with IGS products are shown for reference purposes.

V. Conclusion

In this paper a complex kinematic PPP-like processing based on a network of stations has been easily implemented taking advantage of the *GNSS Data Structures* provided by the GPSTk, as well as its "general solver" object. This procedure was named *POP* because it is independent of precise clock information and only needs precise orbits to work.

This procedure involved multiple stations separated hundreds of kilometers and there are a great number of unknowns of several kinds: Some unknowns are *receiver-indexed* (or *receiver-specific*, like. ztd_i, dx, dy, etc.), some are *satellite-indexed* (dt^j), and others are both *receiver-* and *satellite-indexed*, like Bc_i^j. Therefore, the number of unknowns at a given epoch has a wide variation depending on the available station data and the number of visible satellites. The GPSTk-provided class `SolverGeneral` helps implement this kind of systems, *describing* (rather than hard coding the procedure in software), the equations, variables, and their associated stochastic models and relationships. Moreover, a reference implementation is freely available at the development version of the GPSTk source code, facilitating its use and modification by other GNSS researchers.

The results from this approach were very similar (as expected) to the standard kinematic PPP strategy, but yielding a higher positioning rate. This higher positioning rate opens the way for post-process kinematic positioning of vehicles that usually operate far from reference stations, such as aircraft. Also, the network-based processing of POP seems to provide additional robustness to the results, even for receivers outside the network area. The distance from "*rover*" to nearest reference station does not seem to be a very critical factor, because in our test cases the results are not significantly degraded when this distance nearly doubled.

REFERENCES

[1] N. Castleden, G. R. Hu, D. A. Abbey, D. Weihing, O. Ovstedal, C. J. Earls, and W. E. Featherstone, "First results from Virtual Reference Station (VRS) and Precise Point Positioning (PPP) GPS research at the Western Australian Centre for Geodesy," *Journal of Global Positioning Systems*, vol. 3, no. 1-2, pp. 79–84, 2004.

[2] M. M. R. Mostafa, "Precise Airborne GPS Positioning Alternatives for the Aerial Mapping Practice," in *Proceedings of FIG Working Week 2005 and GSDI-8*, Cairo, Egypt, April 2005.

[3] X. Zhang and R. Forsberg, "Assessment of long-range kinematic GPS positioning errors by comparison with airborne laser altimetry and satellite altimetry," *Journal of Geodesy*, vol. 81, no. 3, pp. 210–211, 2007.

[4] J. Kouba and P. Heroux, "Precise Point Positioning Using IGS Orbit and Clock Products," *GPS Solutions*, vol. 5, no. 2, pp. 12–28, 2001.

[5] U. Hugentobler, M. Meindl, G. Beutler, H. Bock, R. Dach, A. Jaggi, C. Urschl, L. Mervart, M. Rothacher, S. Schaer, E. Brockmann, D. Ineichen, A. Wiget, U. Wild, G. Weber, H. Habrich, and C. Boucher, "CODE IGS Analysis Center Technical Report 2003/2004," in *Gowey K., Neilan R., Moore A. (eds). IGS 2004 technical reports. IGS Central Bureau.*, Jet Propulsion Laboratory, Pasadena, California, USA, 2006.

[6] R. B. Harris, B. Tolman, T. Gaussiran, D. Munton, J. Little, R. Mach, S. Nelsen, and B. Renfro, "The GPS Toolkit: Open Source GPS Software," in *Proceedings of the 16th International Technical Meeting of the Satellite Division of the Institute of Navigation*, Long Beach, California, September 2004.

[7] D. Salazar, M. Hernandez-Pajares, J. Juan, and J. Sanz, "GNSS data management and processing with the GPSTk," *GPS Solutions*, 2009.