EENG 533: Navigation Using GPS

Project 6: Calculation of Position from GPS Measurements

Theory

Pseudoranges and Space Vehicle Positions The true distance, ρ, to a GPS satellite (also known as a space vehicle) is the speed of light, c, times the difference of the true receive time, t_{rx}, and the transmit time, t_{sv} (the space vehicle's time):

$$\rho = c \cdot (t_{rx} - t_{sv}) .$$

Unfortunately, both the space vehicle's time and the receiver's time have clock errors (δt_{sv} and δt_{rx}). What we actually know are the approximate times:

$$\hat{t}_{sv} = t_{sv} + \delta t_{sv}$$
$$\hat{t}_{rx} = t_{rx} + \delta t_{rx}.$$

So, we do not get the true range from our measurements. Rather, our GPS receiver automatically calculates for us an apparent pseudorange, $\hat{\rho}$, based on the receiver's approximate clock time, $\hat{t}_{rx}$, and the space vehicle's approximate clock time, $\hat{t}_{sv}$, indicated in the GPS signal. This pseudorange contains both the space vehicle's clock error, δt_{sv}, as well as the receiver's clock error, δt_{rx}:

$$\hat{\rho} = c \cdot \left(\hat{t}_{rx} - \hat{t}_{sv}\right)$$
$$\rightarrow \hat{\rho} = c \cdot (t_{rx} + \delta t_{rx}) - c \cdot (t_{sv} + \delta t_{sv})$$

such that the relationship to the true range is

$$\rho = \hat{\rho} + c\, \delta t_{sv} - c\, \delta t_{rx}.$$

We can use the equation we learned on week 3 to estimate the space vehicle's clock error:

$$\delta t_{sv} = a_{f_0} + a_{f_1} (t_{sv} - t_{0_c}) + a_{f_2} (t_{sv} - t_{0_c})^2 + \delta t_{rel}.$$

However, this is a function of the space vehicle's time, t_{sv}, which we do not have yet. So, we will solve this in a somwhat interative manner. The first thing we need to calculate is the approximate space vehicle time:

$$\hat{t}_{sv} = \hat{t}_{rx} - \frac{\hat{\rho}}{c}.$$

Hint: this should be the first step in your `calc_rx_pos` function! Then, we will use this value as an approximation to t_{sv} in the week 3 equation and we will disregard the relativistic correction term, δt_{rel}, because we do not need this much accuracy yet:

$$\delta t'_{sv} = a_{f_0} + a_{f_1} \left(\hat{t}_{sv} - t_{0_c}\right) + a_{f_2} \left(\hat{t}_{sv} - t_{0_c}\right)^2 .$$

(The coefficients a_{f_0}, a_{f_1}, a_{f_2}, and t_{0_c} come from the ephemeris data for each given space vehicle.) Now we can get a better approximation of the true space vehicle time:

$$t'_{sv} = \hat{t}_{sv} - \delta t'_{sv}.$$

The `calc_sv_pos` function will calculate more carefully the space vehicle's clock error using the above time (t'_{sv}) and the receiver's position. But, since we do not yet know the receiver's position, we will use the `pos_rx` value sent to your `calc_rx_pos` function as an input. This will be the output of your function from the previous time step (epoch). The `project6_template.py` script will start this whole process with the center of the Earth as the least biased guess:

```
pos_rx = np.array([0.0, 0.0, 0.0])
```

So, we just call the `calc_sv_pos` function (only once per observation time epoch) as follows:

```
[pos_sv, dt_sv] = calc_sv_pos(ephem, t_sv_prime, pos_rx);
```

where `pos_rx` is the receiver's position, `t_sv_prime` is the t'_{sv} from above, `pos_sv` is the space vehicle's position, $[x_{sv}, y_{sv}, z_{sv}]$, and `dt_sv` is the space vehicle's clock error, δt_{sv}. The `calc_sv_pos` function uses the receiver's position to adjust the space vehicle's position for the rotation of the Earth during the propagation of the signal through space. The poor guess of the receiver's position (the center of the Earth) will cause some error for the first epoch. But, after the first epoch, the receiver's position will be better known and little error will remain.

If we are using a single frequency (which we are) we need to remove the group delay correction:

$$\delta t_{sv} = \delta t_{sv} - T_{GD}.$$

(This group delay value T_{GD} comes from the ephemeris data for each given space vehicle.) The improved estimate of the space vehicle's clock error, δt_{sv}, (`dt_sv` above) can be used to get a better estimate of the range, ρ':

$$\rho' = \hat{\rho} + c\ \delta t_{sv}.$$

This is still a pseudorange which has in it the receiver's clock error. But, this pseudorange is good enough because in the next step we will calculate δt_{rx} and the receiver's location using two things: the calculated space vehicle's position, `pos_sv`, and the pseudorange, ρ'. So, the results of this first part of your function should be a vector of pseudoranges, $\underline{\rho}'$, and a matrix of positions, $[\underline{x}_{sv},\ \underline{y}_{sv},\ \underline{z}_{sv}]$. (We notate vectors with underlines.)

Receiver Position and Clock Error If we have a set of pseudoranges, $\underline{\rho}'$, with the δt_{sv} errors removed and a set of corresponding space vehicle positions, $[\underline{x}_{sv},\ \underline{y}_{sv},\ \underline{z}_{sv}]$, we can iteratively calculate the position of the receiver and the receiver's clock error which would be required for all those pseudoranges to make sense. We would use a guess for the receiver's

position and clock error to calculate what the pseudoranges to the space vehicles would be, compare those pseudoranges to the $\underline{\rho}'$ values we already calculated, and adjust our receiver's position and clock error accordingly. We do this until we begin to get consistent results. This process is the Taylor-series approximation that was explained in the lecture and is the same concept as the Newton-Raphson technique.

First, we need to define a state vector. It is the concatenation of two of the function inputs: the receiver's position and the receiver's clock error, with the speed of light factored in:

$$\underline{x} = [x_{rx} \quad y_{rx} \; z_{rx} \quad c\,\delta t_{rx}]^\top.$$

There are two ways to get the pseudorange to a space vehicle. The first is time-based and is what we did in the previous step to get the ρ' values. The second is position-based and requires the positions of the space vehicles, the position of the receiver, and the receiver's clock error:

$$\underline{\tilde{\rho}} = \underline{r} + x_4 \tag{1}$$

$$\underline{r} = \sqrt{(\underline{x}_{sv} - x_1)^2 + \left(\underline{y}_{sv} - x_2\right)^2 + (\underline{z}_{sv} - x_3)^2},$$

where x_j is the jth element of $\underline{x}$. Now, we can get the errors in our distance-based pseudoranges:

$$\delta\underline{\rho} = \underline{\rho}' - \underline{\tilde{\rho}}.$$

We need a way to relate these errors in pseudoranges to errors in the $\underline{x}$ vector. This relationship is the Jacobian matrix of (1) with respect to the states of the state vector $\underline{x}$:

$$\boldsymbol{H} = \begin{bmatrix} -\dfrac{\underline{x}_{sv} - x_1}{\underline{r}} & -\dfrac{\underline{y}_{sv} - x_2}{\underline{r}} & -\dfrac{\underline{z}_{sv} - x_3}{\underline{r}} & \underline{1} \end{bmatrix}.$$

The Jacobian relates the error in pseudoranges to the error in the state vector, $\underline{x}$, this way:

$$\delta\underline{\rho} = \boldsymbol{H}\delta\underline{x}.$$

So, to solve for the error in the state vector, we would use the pseudoinverse of $\boldsymbol{H}$:

$$\delta\underline{x} = (\boldsymbol{H}^\top\boldsymbol{H})^{-1}\boldsymbol{H}^\top\delta\underline{\rho}.$$

Be sure to use matrix multiplication operations (e.g. `np.matmul()`, or the `@` symbol). Then, we can adjust our state vector with its error:

$$\underline{x} = \underline{x} + \delta\underline{x}.$$

We repeat this loop, starting with equation (1), while the `norm` of $\delta\underline{x}$ is greater than 10. When we are done with the loop, our receiver's position and clock error will be

$$x_{rx} = x_1 \qquad y_{rx} = x_2 \qquad z_{rx} = x_3 \qquad \delta t_{rx} = \frac{x_4}{c}$$

and the residuals will be

$$\underline{v} = \delta\underline{\rho} - \boldsymbol{H}\delta\underline{x}.$$

Objectives

- Develop an algorithm to perform single-point, single-frequency positioning using pseudorange measurements
- Understand some of the errors in your solution by comparing to a known location

Collaboration

This is an individual laboratory. You may discuss this lab with other students. However, all source code that you generate and use and anything you turn in must be your own.

Task A: Write your `calc_rx_pos` function

In this lab, you are given an hour of GPS RINEX observation data (`*.YYo` file) and the corresponding RINEX navigation (ephemeris) data (`*.YYn` file). Be sure to utilize the provided `helper.py` and `ephemeris.py` files as they have changed slightly from previous projects. You will also need to utilize the previously developed `calc_sv_pos` function.

The `project6_template.py` script will load the observation and ephemeris data. It will iterate through each of the epochs of time in the observation data, parsing out the PRNs and $\hat{\rho}$ values (L1 C/A-code pseudoranges) from the observation data for the given epoch. The script will call your function. Your function will then calculate the receiver's position and clock error using single-point, single-frequency positioning. It should have the following interface:

```
[pos_rx, dt_rx, v] = calc_rx_pos(prn_array, rho_hat_array,
    t_rx_hat, ephem_list, pos_rx, dt_rx)
```

where

Variable	Size	Description
`prn_array`	N	PRN values
`rho_hat_array`	N	pseudoranges, $\hat{\rho}$
`t_rx_hat`	1	receiver time, $\hat{t}_{rx}$
`ephem_list`	N	list of ephemeris objects
`pos_rx`	3	receiver position, $[x_{rx},\ y_{rx},\ z_{rx}]$
`dt_rx`	1	receiver clock error, δt_{rx}
`v`	N	residuals, $\underline{v}$

In the above table, N is the number of space vehicles available during that epoch to help locate the receiver. Your code should follow these steps in order to calculate the position and clock error of each space vehicle:

1. Get $\hat{t}_{sv}$
2. Get the ephemeris data for that space vehicle, estimate the clock error (without the relativistic correction term), and remove the clock error from $\hat{t}_{sv}$
3. Call `calc_sv_pos` to get the space vehicle's position and more accurate clock error
4. For single-frequency positioning, remove the group delay
5. Remove the effect of the space vehicle's clock error from its pseudorange

For each space vehicle identified in the call to your function, you should now have a pseudorange with $c\ \delta t_{sv}$ removed and a position for the space vehicle in ECEF coordinates. With the set of corrected pseudoranges and space vehicle positions, follow the steps outlined in the **Theory** section to get the receiver's position and single clock error.

The `project6_template.py` script will continue from there to store the values returned by your function. It will print the true receiver position and your calculated receiver position for the second time epoch. The true position (ECEF coordinates in meters and obtained from the online OPUS tool) is

```
True Position =
[  497796.51 -4884306.58  4058066.62]
```

There will be a discrepancy; that is expected. To verify that your function is working correctly, make sure your are getting exactly the following for the **second epoch**:

```
pos_rx_array[1, :] =
[  497794.82 -4884316.34  4058076.96]

v_array[1, :] =
[-1.3    nan  5.42   nan   nan   nan -4.03 -1.22   nan   nan   nan
      nan
 -0.66  1.04   nan   nan  1.24   nan -0.56   nan -2.54 -0.18   nan
       nan
   nan   nan   nan  2.88   nan -0.08   nan   nan]
```

The second time epoch is chosen to avoid the problems associated with initializing at the center of the earth for the first epoch.

Finally, the `project6_template.py` script will convert your calculated receiver position to an East, North, Up (ENU) local level frame. It uses the true position of the receiver as the origin. So, these are position errors now.

Task B: Evaluate accuracy of your algorithm

Towards the bottom of the `project6_template.py` script, add plotting code and comments. Generate the following plots:

1. A plot of the PRNs of the satellites used in the solution vs. time (sec of day). (Hint: plot PRNs vs. time using dots)
2. ENU errors vs. time. Plot all three as separate lines on the same axis.
3. A horizontal map of the errors (East on x axis and North on y axis). This map should be scaled such that 1 m of error in the east direction is the same visible length as 1 m of error in the North direction.
4. A plot of the receiver clock error vs. time
5. A plot of the range residuals as a function of time. (Hint: plot as individual points to avoid connecting lines)

Make sure that all plots are correctly labeled. That means give (1) a description of the value and (2) units (use 'ND' for non-dimensional). As an example, a time axis would be "Time [s]". Answer questions such as, but not limited to the following:

- What are the errors like?
- Is any axis worse than the others?
- How quickly do the errors change?
- Are there any anomalies (like jumps)?
- What would cause them?

Deliverables

There are two files that you should upload to Canvas for your turn-in:

- Your `calc_rx_pos.py` function file
- Your modified `project6_template_LASTNAME.py` script with code for plots and comments answering the questions

Grading

You will be graded for coming up with the correct results (90%) and your analysis (10%). "Correct results" includes the correct generation of the requested plots (not just that your `calc_rx_pos` function works).