EENG 533: Navigation Using GPS

Project 2: Calculating GPS Satellite Position

Objectives

1. Become more familiar with Python programming and explore the georinex RINEX file processing library.
2. Develop a routine to determine satellite position, which will be used throughout the rest of the course.
3. Develop a better understanding of coordinate frames and timing issues.

Overview

You will be calculating satellite positions using broadcast ephemeris data from RINEX (Receiver Independent Exchange Format) file throughout this course, and the purpose of this lab is to develop a tool, a Python function, for doing this. You will write a Python function which can be called to calculate the coordinates of the satellite for use in positioning applications. It will also calculate the satellite clock error, which will be needed in future labs. The equations needed are found on pp. 95-96, and 102-103 of document IS-GPS-200L.

Collaboration

This is an individual lab. You are allowed to discuss any aspect of the lab with other students, and you may look at each other's source code for debugging purposes. However, your programming must be your own; you may not copy or transcribe someone else's program, in part or in whole.

Satellite Position Function Description

Your assignment is to write the following MATLAB function

```
import math
import numpy as np


def calc_sv_pos(eph, tow, rcvr_ecef):
    """
    Satellite position and satellite clock correction calculation based
        on IS-GPS-200L (Table 20-IV and pp. 95-96).
    The satellite position is for a particular GPS time, expressed in the
        ECEF coordinate frame valid at the time the signal was received
        by a GPS receiver.
```

```
    :param eph:               ephemeris object (from georinex) for a
        single satellite and time epoch
    :param tow:               time of transmission [GPS week sec]
    :param rcvr_ecef:         1x3 numpy array of ECEF receiver position [x
        y z in meters]

    :return sv_ecef:          1x3 numpy array of ECEF satellite position
        [x y z in meters] at time of transmission, expressed in ECEF
        frame at the time of reception
    :return sv_clock_error:   satellite clock error [sec] at time of
        transmission
    """
    # <<<your code here>>>

     return sv_ecef, sv_clock_error
```

The algorithms for calculating the ECEF satellite position are provided in IS-GPS-200L (Table 20-IV), which can be obtained from this link. In addition, you should reference the class slide handouts to get additional information that you will need to complete this lab. Note that the coordinates calculated by the equations in IS-GPS-200L (x_k, y_k, and z_k) are the ECEF coordinates of the satellite at the time of transmission (shown as x_t, y_t, and z_t in the slide titled "Coordinate Frame Rotation" in the fourth video). You will need to perform the rotation to express them as the ECEF coordinates at the time of reception (x_r, y_r, and z_r).

The algorithms for calculating the SV clock correction (Δt_{sv}) are found in the IS-GPS-200L section entitled "User Algorithm for SV Clock Correction" (pp. 95-96). Note that you need the eccentric anomaly (an intermediate calculation when calculating the position) in order to do this. This SV clock correction represents a significant error that needs to be removed if anything useful is done with GPS measurements.

Obtaining the Ephemeris Data

The raw ephemeris data is in a provided text file called `ohdt3490.20n`.

The georinex library and functions in the provided `helper.py` file will perform the work needed to isolate the correct ephemeris record from the RINEX nav file. The provided `project2_template.py` file contains an example for a specific PRN and transmission time.

The georinex library needs to be added to your Anaconda environment. Unfortunately, it is not contained in the Anaconda repository so you will need to choose the "CMD.exe Prompt" (or similar depending on your operating system) application and type `pip install georinex` at the prompt. It may take a few minutes to install. Typing `pip list` will show all the installed libraries in your environment and georinex should now be listed.

The `ephem` object created near the end of `project2_template.py` from the class definition in `ephemeris.py` will contain the ephemeris values for the chosen satellite and for the time

closest to the `tow` specified. When implementing the equations in your function to calculate satellite position and clock error, the ephemeris values can be accessed as `eph.M_0`, `eph.e`, etc. The full listing of parameters is shown in `ephemeris.py`.

Checking Your Work

Below is a sample run from a satellite positioning algorithm that you can use to check your work.

```
Filename       : ohdt3490.20n
PRN            : G03
Transmit Time  : 80000.000
Rcvr Position  : -1485881.487 -5152018.353 3444641.847 (meters)
SV Position    : -12334966.483 -22955725.027 -5479289.332 (meters)
SV Clock Error : -0.0000290720 (seconds)
```

Hint: Here is a sample command that would generate some of the above printout. You should not have this in your function, but you could use something like it in a test program which would call your function.

```
print("SV Position    : %.3f %.3f %.3f (meters)" % (sv_ecef[0],
    sv_ecef[1], sv_ecef[2]))
```

Deliverables

For this project, please submit your function as a `.py` file in Canvas. Please name it

```
calc_sv_pos_<lastname>.py
```

where `<lastname>` is your last name in lower case. If John Smith were turning in the lab, the filename would be `calc_sv_pos_smith.py`.

Note that I only want your function so that I can call it myself, not a program that will generate an output similar to that shown in the "Checking Your Work" section above. In fact, your function should not print out anything at all: it should just return the output vectors.

Grading

You will be graded primarily upon accuracy. If you get the correct positions for satellites and times that I pick when I run your function, then you will get full credit for the lab. I will try several different PRNs at different times. If you do not get the correct answers, then

I will look at your source code and make a judgment on your grade based on what I see. Feel free to include any comments that you would like in Canvas. Also, make sure that your function works with a 1-by-3 `rcvr_ecef` vector and returns a 1-by-3 `sv_ecef` vector. Points will be deducted if either of these is wrong. Also, make sure that your function can handle end of week rollover (as described in the IS-GPS-200L) for both the satellite position and clock error calculations. Failure to do so will result in points being deducted for each case in which the error is made (position and clock error).