ENGO 625 – Advanced GNSS Theory and Applications (F2022)

Lab #1 – Satellite Geometry

Due Date

Monday, 20 October 2025 @ 4:30 PM via D2L dropbox. Please submit a short written report.

Objectives

1. To look at some GPS data (observations and satellite positions)
2. To become familiar with analysis based on estimated accuracy and satellite geometry.
3. To improve programming skills with regards to Geomatics Engineering (C/C++, MATLAB, or Python only, please).

Data Description

Each student will be given three data files.

- A binary file containing 1 Hz GPS observations from a static NovAtel OEMV *remote* receiver.
- A binary file containing 1 Hz GPS observations from a static *base* station. You will be given the true coordinates of both receivers
- A binary file containing the satellite coordinates and velocity components in the Earth-Centred Earth-Fixed (ECEF) frame, also at 1 Hz.

Alternatively, you may collect your own observations, or obtain any observations available online, for example from an IGS repository, but you will need to ensure that the observations include pseudorange, phase, and Doppler measurements and you'll need to convert or load them. You will also need a different source of the satellite coordinates, for example and SP3 file from the IGS.

You will be given truth coordinates for both receivers for comparison and executables to view these files. The data has already been corrected for tropospheric error, group delay (not the ionospheric error) and satellite clock offset and drift. You are expected to do the rest of the processing – e.g. additional corrections, satellites dropping in and out of view, synchronizing observations between files in this and the following labs.

Observation file – binary format, each element is 64-bit float (double)

Each second of data contains 12 channels. Each channel can hold observations for one satellite. Channels without observations will contain zeroes. The format for one channel is as follows:

<Satellite PRN>
<GPS time of observation (seconds)>
<C/A code pseudorange (m)>
<L1 carrier phase measurement (L1 cycles)>
<Doppler of L1 carrier (Hertz)>
<L2 carrier phase measurement (L2 cycles)>

<u>Satellite file</u> – binary format, each element is 64-bit float (double)

Each second of data contains 12 channels. Each channel can hold coordinates for one satellite. Channels without satellites will contain zeroes. The format for one channel is as follows:

<Satellite PRN>
<GPS time of observation (seconds)>
<X Satellite Coordinate (m)>
<Y Satellite Coordinate (m)>
<Z Satellite Coordinate (m)>
<X Satellite Velocity (m/s)>
<Y Satellite Velocity (m/s)>
<Z Satellite Velocity (m/s)>

To load these files using C/C++ you can read them into structs that contain 6 and 8 doubles respectively (yes everything is stored as a double, including the PRN number). To load them into MATLAB, you can use the `fread` function. Specifically something like :

```
fid = fopen('Satellites.sat')
SatellitePosArray = fread(fid,[8 Inf],'double')'
%reads the entire file into an 8 row, Inf column array then
%transpose the result to get a matrix of doubles where the columns correspond to each
%record
```

Tasks

1. Load the satellite coordinate file and the rover observation file
   - **a.** For the first 300 epochs, plot the coordinates of the satellites in 3D
   - **b.** Discuss: How are the satellites distributed. Is 300 epochs enough time to see the satellite paths? If not, plot some more (possibly all) epochs and discuss.
   - **c.** For the first 300 epochs, plot the pseudorange, Doppler, and L1 carrier phase observed for each of the satellites. You will need to read through the observation array you have loaded from the file row by row, because this receiver has stored the observations for each epoch in the first rows of each set of 12 rows (so the specific row that corresponds to a satellite may change from epoch to epoch).
   - **d.** Discuss: How much does the pseudorange vary from epoch to epoch. Does this match the change in the carrier phase? Does the carrier phase change match what you think it should based on the Doppler and the pseudorange?

2. Build the design matrix for each epoch and then compute the HDOP and VDOP.
   - a. Plot each DOP as a time series. If you have built the design matrix ECEF, you will need to rotate the XYZ covariance matrix to a local geodetic frame ENU to determine the H and V DOPs.
   - b. Plot a time series of the number of satellites.

3. From the tasks above, draw an intuitive general conclusion on the quality of the position estimates you should expect from GPS based on the number of the satellites in view, their spatial

distribution in the sky as well as the quality of the pseudorange. In lab 2, you will use Least squares to compute the solutions.

Tips

- The individual lab tasks are interconnected and you might find that programming separately for each task becomes very tedious. Build your code in modules that can be expanded (e.g. a design matrix building module will be used in lab 2 for least squares, etc.) so that you can re-use some parts for many different lab tasks and the next lab. If you are new to programming or are an expert in Matlab, then you may use Matlab. If you are fluent in Python, you may use Python, however I would recommend C++ as it remains an industry favorite.
- Do not design your code to work specifically with your data set (e.g. hard-coding PRNs or GPS times). You may require a different data set for Lab 2, and Lab 2 will use results from Lab 1. For those who are inexperienced with Matlab, there are many tricks that can make your plots look clearer. You can adjust the width and style of the lines, set the color, increase the font size of the labels, and so on. The "set" function in Matlab can be used to set the properties for individual plot lines or entire plot areas. The Property Inspector can be accessed as follows: right click on an object in a figure, click "Show Property Editor" and click "More Properties".
- When plotting time series, showing all six digits of the GPS time can be confusing. What you can do instead is subtract the first time value from all the times, so the time scale starts at zero. Make sure when you do this that the time value that you subtract is the same for all the time series you plot! The time axes should then be labeled sensibly, for example "Time from beginning on test, t (s)"

Report Format

Each report must be professionally written. Be concise but thorough. Explanations should accompany all plots. Assumptions must be clearly documented and justified. All source code must be submitted to the dropbox, not NOT include it in the report. Only C/C++, Matlab or Python code will be accepted.

Grading Criteria (7% of Final Grade)

Technical Accuracy (5/7)
- Correct solution/plots
- Correct analysis of results
- Correct justification and assumptions
- Insight into results
- Graphs correctly labeled (each axis must have a label that includes a description, a symbol and a unit

Source Code (1/7)
- Methodical and logical procession of algorithms
- Commented source code
- If you share software with other classmates, you must acknowledge their contribution both in the comments and in the text of the report.

Report Quality, Neatness and Readability (1/7)
- Neat, clear and effective use of scales on plots
- Logical organization of lab

References

ENGO 625 Course Notes - Advanced GNSS Theory and Applications
ENGO 421 Course Notes (might be useful for coordinate systems)
ENGO 363 Course Notes or Text (for a detailed explanation of Least Squares)