EENG 533: Navigation Using GPS
Project 1: Coordinate Frame Conversions

Objectives

1. Increase proficiency in Python programming and investigate the numPy, SciPy, and Matplotlib libraries
2. Demonstrate the ability to do various types of coordinate transformations
3. Understand the difference between Cartesian and surface representations of local level coordinates
4. Determine the region over which a local level frame is consistent with a surface-defined frame.

Overview

In this lab, you will convert a flight trajectory into a local level frame using the two different methods described in class. Once this is done, you will evaluate the results in order to get insight into the differences between a Cartesian representation and a surface-based representation.

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.

Local-Level Coordinates

The file `proj1_flight_trajectory.mat` has a trajectory of an aircraft flying southeast at 300 m/s at 1000 m altitude starting in Dayton.

It can be loaded with the helper code near the top of `project1_template.py`.

The contained values are all geodetic coordinates relative to the WGS-84 ellipsoid.

`map.png`

Figure 1: Map of the trajectory over the US.

For this task, you should convert the entire trajectory into local level coordinates using the two different methods that are described in the "Local Level Coordinates" video: ECEF → Local Level, and Geodetic → Local Level. Note that for the ECEF → Local Level, you will *first* need to convert the given geodetic (`lat`, `lon`, `h`) coordinates to ECEF, and then convert to a local level frame. You are given a python function to convert from geodetic (`lat`, `lon`, `h`) to ECEF (`x`, `y`, `z`): `llh2ecef.py`.

The origin of your local level frame should be the starting point of the flight, but with an altitude of zero. (The starting `lat` and `lon` are the first elements of each array.) Remember that when converting from ECEF → Local Level, the 3×3 direction cosine matrix (DCM) that you use to rotate the ECEF vector to the local level frame should only be calculated once, based on the origin coordinates, and that DCM should be used for every subsequent rotation.

Now you should have two different local level coordinates: one generated by converting to ECEF and then using the ECEF → Local Level method, and one generated by using the Geodetic → Local Level method. Both should represent east, north, and up distance from the origin in meters. Notationally, let us call these `enuE2L` (ECEF → Local Level) and `enuG2L` (Geodetic → Local Level).

Generate the following plots:

1. Horizontal plot of horizontal position (east and north) for both trajectories in the same plot (distinguish by color)
2. A plot of `enuE2L` altitude versus time and `enuG2L` altitude versus time on the same plot (distinguish by color)
3. A plot of the difference between `enuE2L` and `enuG2L` (i.e., `enuDIFF = enuE2L - enuG2L`) versus time for all three values: east, north, and up.

In the video, it was stated that the two approaches (ECEF → Local Level and Geodetic → Local Level) are equivalent as long as you are "close" to the origin of the local level frame. Let us quantify how close you need to be. Based on the plots that you have (especially the third plot), answer the following questions:

1. From plot 2, why does the height become negative for the `enuE2L` representation?
2. In what sense could you say that both heights (`enuE2L` and `enuG2L` heights) are correct in their own ways?
3. From plot 3, what is the maximum distance from the origin of the local level frame you can go in order to keep the maximum altitude difference from being more than 1 cm? (Recall that you are flying at 300 m/s, so knowing time you can also know distance flown).
4. Also, from plot 3, what is the maximum distance from the origin of the local level frame you can go in order to keep both the east and north difference from being more than 1 cm?

Deliverables

For this project, submit via Canvas the following:

- The python source code used for calculation and plotting
- The three plots specified
- The answers to the four questions

Grading

This project is graded on a 100-point scale. If your results do not seem to be correct or have anomalies, please point that out. I will deduct more points for errors that are present which you did not seem to realize were errors than for errors which are present but recognized as errors based on your comments.