Hardware/Software Codesign for Wireless Systems (E168b)

Harris

Lab 6: Positioning

In this lab, you will determine position from the navigation data. More specifically, you will complete a least squares algorithm that finds the best position given the pseudo ranges of the satellites and the satellite positions. You will be editing code in the function LeastSquarePos_class.m.

You will find all of the necessary files on the network drive. The position.mat file includes the positions of the satellites and the pseudo ranges. There are also two sub-functions. e_r_corr.m corrects for the earth's rotation, and cart2geo.m converts from Cartesian coordinates to latitude, longitude, and altitude.

Recall that the positioning algorithm knows the positions of M satellites

$$S = \begin{bmatrix} X_1 & Y_1 & Z_1 \\ X_2 & Y_2 & Z_2 \\ & & \\ & & \\ X_M & Y_M & Z_M \end{bmatrix}$$ and the pseudoranges to those satellites,

obs = (obs_1, obs_2, …, obs_M). S contains a correction for the rotation of the earth during the travel time from the satellite and the receiver so that we can compute in an earth-centered, earth-fixed coordinate system. The psueoranges have units of distance, but might all be off by some constant amount because of uncertainty in the receiver clock.

The algorithm solves for the user position in four-dimensional space pos = (x, y, z, ct) = (pos_1, pos_2, pos_3, pos_4). The distances from the user to the satellite are nonlinear functions of the user position. We solve these equations iteratively by linearinzing the distance equation around a guess of the user's position and repeatedly solving the linearized equation to refine the guess better. The initial guess is at the center of the earth: pos = (pos_1, pos_2, pos_3, pos_4) = (0, 0, 0, 0). At each step, we compute a correction to the position, add that to the current position, then repeat based on the improved position guess. The correction is called x = (x_1, x_2, x_3, x_4).

On the jth iteration, we solve Ax = b, where A is the linearized set of equations, x is the 4-dimensional position correction term, and b is the vector of M errors in distance given the current guess in the position. We solve for x, then compute an updated position of pos = pos + x. In our equations,

$b_i = obs_i - ct - \sqrt{(X_i - pos_1)^2 + (Y_i - pos_2)^2 + (Z_i - pos_3)^2}$. The full set of equations is:

$$\underbrace{\begin{bmatrix} -\frac{X_1 - pos_1}{obs_1} & -\frac{Y_1 - pos_2}{obs_1} & -\frac{Z_1 - pos_3}{obs_1} & 1 \\ -\frac{X_2 - pos_1}{obs_2} & -\frac{Y_2 - pos_2}{obs_2} & -\frac{Z_2 - pos_3}{obs_2} & 1 \\ & \cdots & & \\ & \cdots & & \\ -\frac{X_M - pos_1}{obs_M} & -\frac{Y_M - pos_2}{obs_M} & -\frac{Z_M - pos_3}{obs_M} & 1 \end{bmatrix}}_{A} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \cdots \\ b_{M-1} \\ b_M \end{bmatrix}$$

Open the leastSquarePos_class file and make sure it makes sense to you. It is missing code in two places: one to set up the A matrix, and the other to solve the least squares. Complete the code.

To test your code, load the position.mat file to load the satellite positions and pseudoranges that have been already computed given the ephemeris you deciphered in the previous lab. Run [lat, long, height] = leastSquarePos_class(satpos, obs). You should get a result giving the position converted to latitude (phi), longitude (lambda), and height (h). If you have Google Earth, you can plug in the latitude and longitude to find where the data was recorded. Note that there is some error in the position, leading to a negative height (the actual site is close to sea level).

What to Turn in

1) The latitude and longitude coordinates where the data was recorded.
2) Your Matlab code
3) How long did you spend on this lab?