CSCI–6280 Mobile Robotics, Spring 2005
Assignment 1: out Thursday January 27, due February 17

GPS simulation

For this assignment, you and a partner will write two programs: a "satellite" program that simulates the transmissions of GPS satellites to a GPS receiver, and a "receiver" program that calculates the receiver's position from the satellite transmissions. The first program entails simply doing some coordinate transformations and calculating satellite positions. The second program involves implementing the Newton-Raphson method for finding the roots of a set of nonlinear equations.

Another part of this assignment is to evaluate the error of the position calculated by the receiver. We will be making some simplifying assumptions, so there will be only three types error: satellite clock synchronization, ephemeris (satellite position), and "other errors."

I will ask you to turn in your code and a written report containing some documentation and your results. Details are provided in this handout and will be clarified as necessary.

Implementation notes

You may implement this assignment in any programming language you choose, so long as I am able to (compile and) run your code. I don't do Windows, so if you do your work on Windows, make sure it is compatible with language implementations on UNIX (SunOS) or Linux. In order to simplify cross-platform compatibilty, all required I/O will be through files.

You are to implement the Newton-Raphson method yourselves, however I encourage you to look for and use (and properly cite) code for other parts of this assignment. For example, should should find code for solving a system of linear equations instead of writing it yourself. Matrix representations and manipulation code would be another thing to look for.

You and your partner will, of course, collaborate. I encourage you to share information with other teams (e.g., about finding a linear equation solver), but there should be no detailed discussion of implementation or results with other teams. General discussions about the project (including general discussion about implementation or results) is fine. Needless to say, no team may share any code that they have written with another team.

Please ask me if there is any need for clarification on the above.

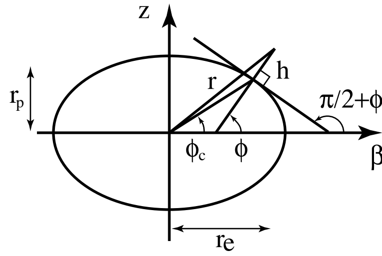
1 Background

1.1 Coordinate frame

The coordinate frame we will use is an Earth Centered Earth Fixed (ECEF) cartesian coordinate frame. The z axis is coincident with the axis of rotation of the Earth. The x axis is set so that the xz plane passes through the prime meridian, i.e., longitude 0. This coordinate frame is right handed, and the origin of the frame is at the center of the Earth.

GPS computes latitude, longitude, and altitude based on the WGS 84 (1984 World Geodetic System) "reference ellipsoid." This is a somewhat idealized model of the Earth's shape; because of this, GPS altitudes are not necessarily correct with respect to the "geoid."

For our purposes, the main implication is the difference between geocentric latitude (ϕ_c in the figure below) and geodetic latitude (ϕ).



WGS-84 ellipsoid parameters		
r_e	Equatorial radius	6378137 m
r_p	Polar radius	6356752.3142 m
ϵ	First eccentricity	0.0818191908426

The eccentricity is defined as:

$$\epsilon = \sqrt{1 - \frac{r_p^2}{r_e^2}}$$

To convert from geodetic latitude ϕ, altitude h, and longitude λ on the WGS-84 ellipsoid, to coordinates in the ECEF frame, you can use the following equations:

$$x = \left(\frac{r_e}{\sqrt{1 - \epsilon^2 \sin^2 \phi}} + h \right) \cos\phi \cos\lambda \tag{1}$$

$$y = \left(\frac{r_e}{\sqrt{1 - \epsilon^2 \sin^2 \phi}} + h \right) \cos\phi \sin\lambda \tag{2}$$

$$z = \left(\frac{r_e(1 - \epsilon^2)}{\sqrt{1 - \epsilon^2 \sin^2 \phi}} + h \right) \sin\phi \tag{3}$$

I will leave the conversion from the cartesian coordinates to latitude, longitude, and altitude as an exercise.

1.2 Satellites

Since we are operating in the ECEF frame, the orbital planes of the satellites will appear to rotate relative to the Earth. We will compute the satellite position by the equation:

$$\vec{x} = \text{Rot}(\hat{z}, -\frac{2\pi t}{p_e})(r_e + a) \left[\vec{u} \cos(\frac{2\pi t}{p_s} + \theta) + \sqrt{1 - e^2}\vec{v} \sin(\frac{2\pi t}{p_s} + \theta) \right] \tag{4}$$

where $\text{Rot}(\cdot, \cdot)$ produces a left operator that rotates a vector, and the constants and variables are defined as follows:

t	Time (seconds)
p_e	Period of Earth's rotation (seconds)
p_s	Period of satellite's orbit (seconds)
a	(Equatorial) altitude of satellite's orbit (m)
e	Eccentricity of satellite's orbit
θ	Phase of the satellite (rad)
$\vec{u}, \vec{v}$	Vectors defining an orbital plane
r_e	Equatorial radius of the Earth (m)

The parameters that are not part of the WGS-84 model (or the problem input) will be specified in a parameter file. Also note that we will require a satellite to be at least 15 degrees above the horizon in order to view/use it.

2 Programs

2.1 Satellite program

Your “satellite” program should read a file named `satinput.dat` that contains a position (in latitude-longitude coordinates) and a time (in seconds). For example:

```
 42 43 50.0
-73 40 55.0
68.58
600.0
```

This corresponds to 42° 43′ 50.0″ north latitude, 73° 40′ 55.0″ west longitude, an altitude of 68.58 meters (225 feet), and (assuming $t = 0$ is midnight) 10:00 in the morning. (This location corresponds approximately to the Amos Eaton / Sage area of campus.) The time specified here will be the time at the receiver (not of a satellite) when the satellites transmit their message.

Your program will also need to read a file containing parameters for the satellites. This file will be available on the course web page.

Your program should write a file called `gpsxmit.dat` that contains the information from all satellites visible at least 15 degrees above the horizon. Note that there may be more than four such satellites. This file should contain one line for each visible satellite of the form:

```
<i> <X> <Y> <Z> <t_s> <t_r>
```

where i is the satellite number (an integer), (X, Y, Z) are the satellite coordinates in the ECEF frame (real numbers, in meters), and t_r is the satellite's time (a real number, in seconds) when the satellite transmitted the message, and t_r is the time on the receiver's clock when the message was received (a real number, in seconds).

The satellite time t_s should be computed by adding a random offset to the receiver's time specified in the input file. This offset should be the same for all satellites; however, later in this assignment you will be adding error to the satellite time for each satellite.

The time that the receiver gets the message t_r should be the starting time at the receiver (given in the input file) plus the time for the message to reach the receiver: the distance from the satellite to the receiver divided by the speed of light (2.99792458×10^8 m/s).

2.2 Receiver program

Your receiver program should read the `gpsxmit.dat` file, calculate the receiver's position, and write it to a file named `gpsloc.dat`. The format of this file should be:

```
<x> <y> <z>
<lat-deg> <lat-min> <lat-sec>
<long-deg> <long-min> <long-sec>
<altitude>
<time>
```

where (x, y, z) are the ECEF coordinates in meters. Note that the number of seconds should be a real number.

Recall, that you need to solve a system of nonlinear equations, each of the form:

$$\tilde{\rho} = \sqrt{(X - x)^2 + (Y - y)^2 + (Z - z)^2} + c\Delta t_r \tag{5}$$

where $\tilde{\rho}$ is the pseudorange, c is the speed of light, and Δt_r is the receiver's clock offset. This system of equations can be written:

$$\vec{\tilde{\rho}} = \vec{F}(x, y, z, c\Delta t_r) = \vec{F}(\vec{x}) \tag{6}$$

Note that there may be more than four equations here.

You will use the Newton-Raphson method to iteratively solve for $\vec{x}$. You will need to give it an initial guess $\vec{x}_0$, and then you can iteratively compute estimates of the solution with:

$$\vec{x}_{i+1} = \vec{x}_i + J^{-1}(F(\vec{x}) - \tilde{\vec{\rho}}) \tag{7}$$

3 Error characterization

Modify your programs to add error to the satellite's transmissions. There are three types of error you should add:

- satellite clock error — this reflects the fact that the satellite clocks are not perfectly synchronized. This error is added to t_s.
- ephemeris (satellite position) error — this reflects the fact that the satellite will not know its position exactly. This error is added to (X, Y, Z)
- other error — here, we will collect atmostpheric error, multipath error, etc. The effect of all these errors is to increase the amount of time that the signal takes to reach the receiver. This error is added to t_r.

You will probably find it convenient to put error parameters into a configuration file so you don't have to recompile your code.

Characterize the error of the position estimate at the receiver. Perhaps the easiest way to do this is to run a large number of trials and compute statistics on the resulting positions.

Note that the errors in different directions will not be independent, though you may simply compute overall statistics on each axis separately. You may find it useful to transform your position estimates into an "east-north-up" coordinate frame. Also keep in mind that the geometry of the satellites relative to the receiver also affects the error (and the geometry of the satellites will be different at different times).

4 Writeup

In addition to turning in your code, you will turn in a 5–10 page writeup that covers the following points:

- Provide instructions for me to compile and run your code.
- Discuss the convergence of the Newton-Raphson method. Be sure to include how your initial position estimate affects convergence. Did the method ever "blow up" or converge to an incorrect solution? How did you address these problems?
- Discuss what kind (and how much) error you added to the satellte messages. How did you characterize the resulting error in position estimated by the receiver? Give an analysis of the receiver positioning error, including statistics/data from your tests.