MA 325 Module 2
Moody T. Chu
Quentin Young

Project – Modeling GPS with Nonlinear Least Squares

This module explored the applications of optimization & error reduction techniques to global positioning systems. The task of locating a GPS receiver at an unknown location in 3-space was assigned as the module project.

For the first part, the locations of four satellites and corresponding measurements of signal latency to a receiver at an unknown point are given. There is an equation that models the location of the receiver, called the navigation equation, which is as follows:

$$r_i(x, y, z, d) = \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2} - c(t_i - d)$$

The *i*th set of terms in the equation correspond to the location (x_i, y_i, z_i) of a satellite and its measurement of the time t_i taken for an EM signal traveling at c = speed of light to reach the receiver. The point (x, y, z) is the location of the receiver (which we wish to solve for) and d is the cumulative clock latency of satellite and receiver (assumed not to vary across satellites). The functional value r_i is the difference between the measured distance and the actual distance between satellite and receiver. The ideal value for this quantity is 0, indicating no error in measurement.

Given (x_i, y_i, z_i) and t_i for four satellites, a system of 4 nonlinear equations in 4 unknowns is easily formed. Since numbers in these equations are not particularly amenable to hand calculation, Matlab is utilized to solve the system numerically via *fsolve()* (code listings 1 and 2). Matlab is also used to plot a sphere representing Earth, the locations of the satellites, and the calculated location of the receiver.

Given satellite positions and time latencies, the following system is formed:

$$0 = \sqrt{(x - 15600)^2 + (y - 7540)^2 + (z - 20140)^2} - c(0.07074 - d)$$

$$0 = \sqrt{(x - 18760)^2 + (y - 2750)^2 + (z - 18610)^2} - c(0.07220 - d)$$

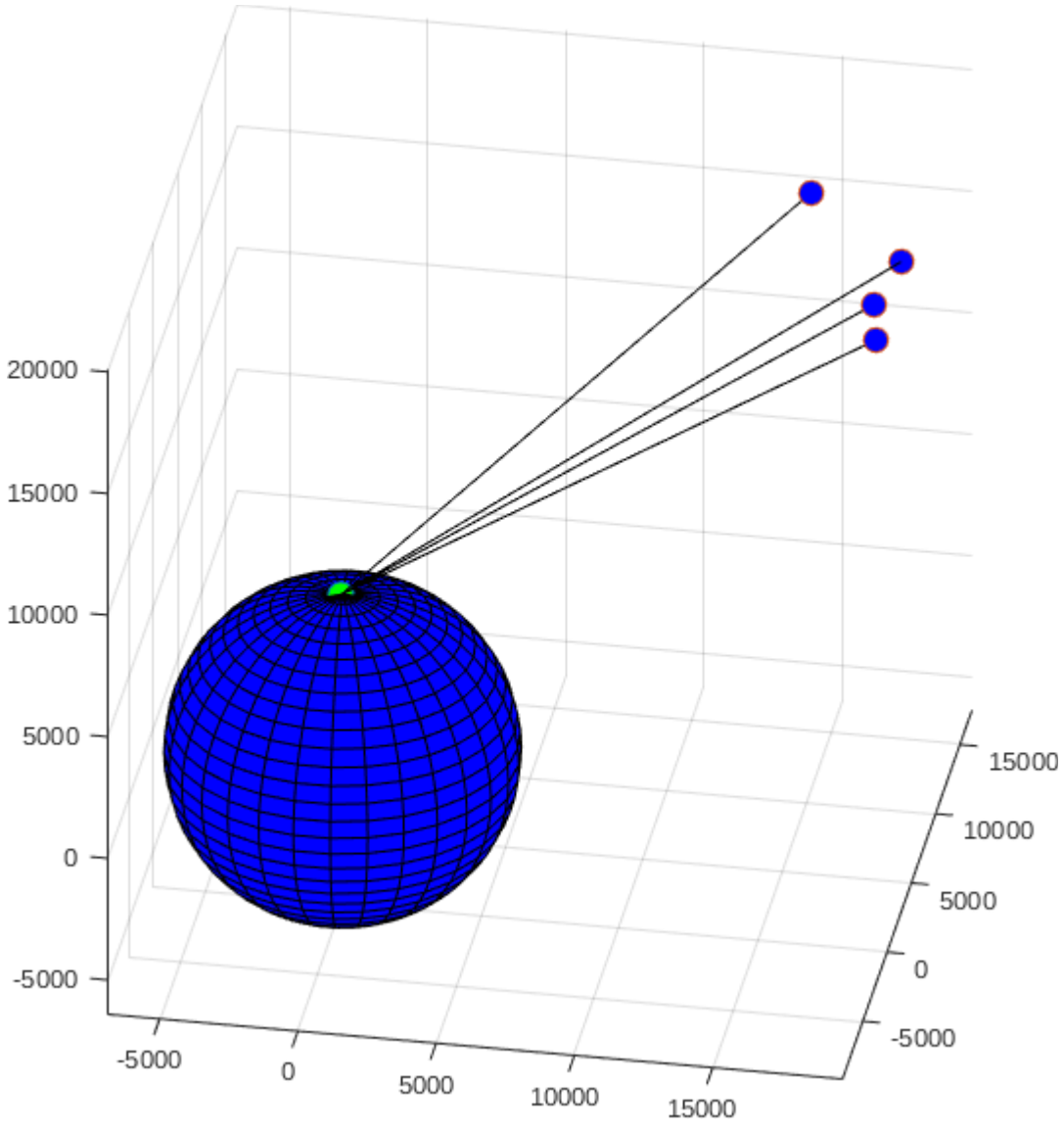
$$0 = \sqrt{(x - 17610)^2 + (y - 14630)^2 + (z - 13480)^2} - c(0.07690 - d)$$

$$0 = \sqrt{(x - 19710)^2 + (y - 610)^2 + (z - 18390)^2} - c(0.07242 - d)$$

Using fsolve, the solution obtained is

$$(x, y, z) = (-41.8, -16.8, 6370.1)\ km$$
$$d = 0.00\ s$$

These coordinates are relative to the center of the earth. Crosschecking these values with the radius of Earth = 6371 *km*, this solution is consistent. Graphing the satellites, Earth (centered at $(0, 0, 0)$ with radius 6371), and the receiver, the following figure is obtained:



Lines from each satellite to the receiver have been added for clarity.

In the second part, the locations and time measurements of two additional satellites are added to the data set. With this new data set, a system of 6 equations in 4 unknowns is formed. However, this leads to an overdetermined system with no singular solution. The next best solution is to optimize such that the sum of the squares of the residuals is minimized. This technique is known as the *method of least squares*. This was done in Matlab using *lsqnonlin()* (code listings 1 and 3).

The system formed is:

$$0=\sqrt{(x-15600)^2+(y-7540)^2+(z-20140)^2}-c(0.07074-d)$$

$$0=\sqrt{(x-18760)^2+(y-2750)^2+(z-18610)^2}-c(0.07220-d)$$

$$0=\sqrt{(x-17610)^2+(y-14630)^2+(z-13480)^2}-c(0.07690-d)$$

$$0=\sqrt{(x-19710)^2+(y-610)^2+(z-18390)^2}-c(0.07242-d)$$

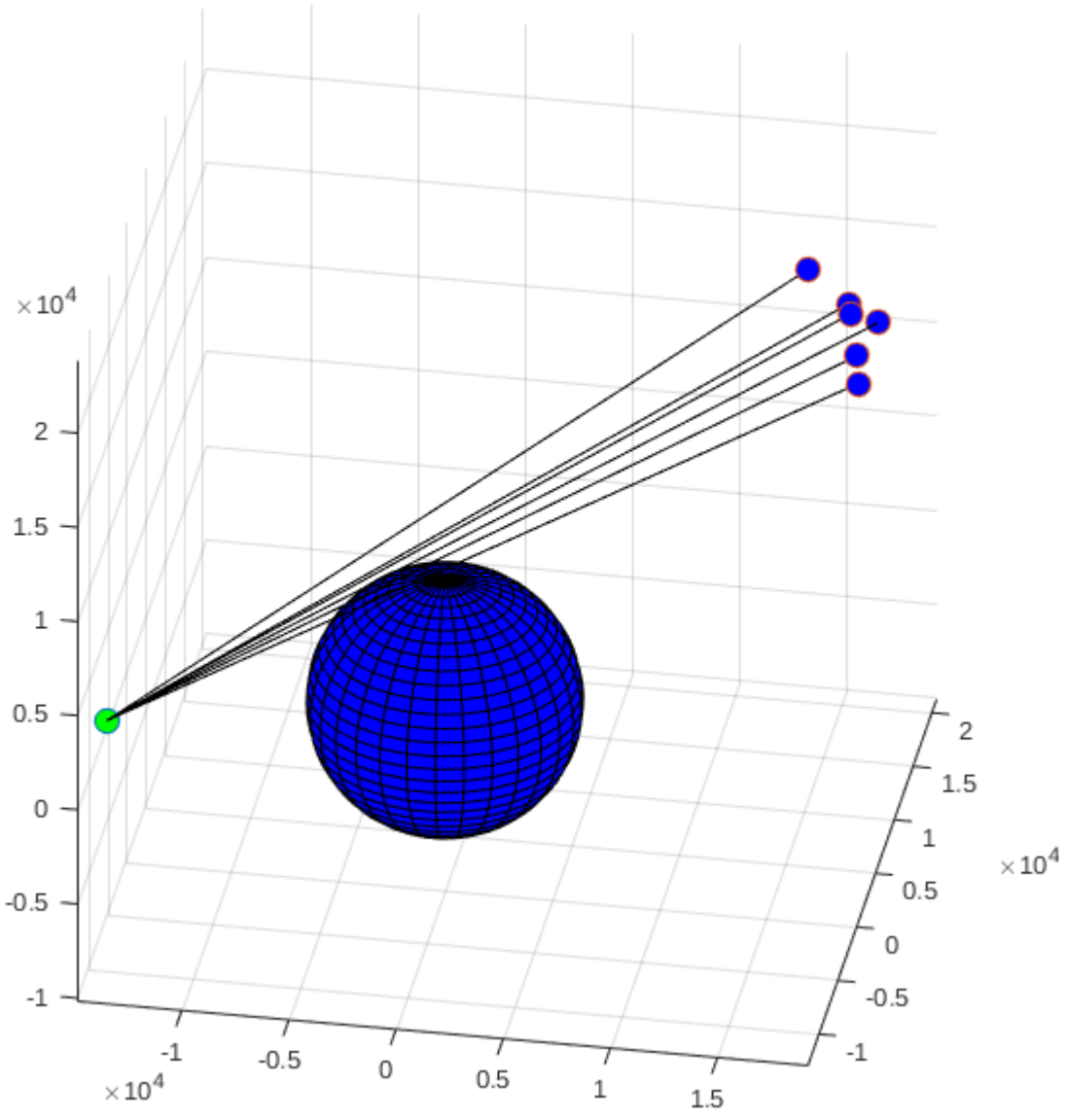
$$0=\sqrt{(x-17800)^2+(y-6400)^2+(z-18660)^2}-c(0.07320-d)$$

$$0=\sqrt{(x-17500)^2+(y-7590)^2+(z-18490)^2}-c(0.07350-d)$$

Using nonlinear least square optimization, the solution obtained is

(x, y, z) = (-14822, -5497, 557) km
d = 0.00 s

Graphing this result:



This clearly isn't correct! The receiver appears to be floating in space, well away from the Earth. Assuming that the receiver is actually on Earth, it is worth exploring what could have happened to yield such a solution.

Since the first four satellites haven't moved and their time measurements haven't changed, we can restrict our attention to the two new satellites that have been added. One possible source of error is the information on the location of the satellites. In order for the location of an object on Earth to be accurately measured, the true location of each satellite must be known as well. Satellites report data on their position periodically, but not constantly, so it's possible that small errors in ephemeris data can accumulate. This should not be a significant source of error, though.

Another possibility is that time measurements have been skewed due to signal interference with the receiver. Typically ground receivers observe not just the signals from each satellite in its cluster, but also reflected signals produced on the way to the ground. These signals can be significantly delayed and, if they are strong enough, skew the signal latency measurements. This type of error is called a “multipath” error, referring to the measurement signal becoming fragmented & reflected along multiple varying-length paths on its way to the receiver.

It is also possible that clock delay is present. Satellites require very accurate atomic clocks in order to obtain precise measurements. While a multi-million dollar satellite may have very accurate clocks, typical receivers do not, and this introduces a clock latency. In our model this was accounted for by d. Since *d* resolved to 0.00, this is likely not a source of error.

Finally, on its way the receiver, the signal must pass through Earth's ionosphere, a gaseous layer of charged particles in the upper atmosphere. Since the signal is an electromagnetic wave, it interacts with charged particles and may be distorted on the way. Distortion may be mathematically modeled, but in order to do so, a variety of other parameters must be known. For instance, the entry angle of the signal into the atmosphere must be known in order to calculate the distance the signal must travel through the ionosphere and subsequently obtain an estimate for expected distortion / skewing.

Sources used:
http://www.tech.mtu.edu/courses/su4100/Reference_Material/GPS_Erro.pdf
http://www.trimble.com/gps_tutorial/howgps-error2.aspx

Code Listing 1 – project.m

```
% Part 1 -- square system
% ============================

% numerically solve system for x, y, z, and d
fun = @gps_resid;
guess = [0, 0, 0, 0];
result = fsolve(fun,guess)

% Graphing
% ----------------------------
figure
hold on
grid on
view(10, 30)
axis vis3d

% satellites
satx = [15600; 18760; 17610; 19170];
saty = [7540;  2750;  14630; 610  ];
satz = [20140; 18610; 13480; 18390];

% receiver
recx = result(1);
recy = result(2);
recz = result(3);

% sphere with radius of earth in km
r = 6371;
[X,Y,Z] = sphere(30);
X = X*r;
Y = Y*r;
Z = Z*r;

% plot earth, receiver, satellites, & lines from sats to receiver
surf(X, Y, Z, 'FaceColor', 'blue')
scatter3(recx, recy, recz, 100, 'MarkerFaceColor', 'green')
scatter3(satx, saty, satz, 90, 'MarkerFaceColor', 'blue')
for n = 1:4
    line([recx, satx(n)], [recy, saty(n)], [recz, satz(n)], 'Color', 'black')
end


% Part 2 -- least squares optimization
% ====================================

% optimize overdetermined system for least squared residuals
fun = @gps_resid_overdetermined;
guess = [0, 0, 0, 0];
result = lsqnonlin(fun,guess)

% satellites
satx = [15600, 18760, 17610, 19170, 17800, 17500];
saty = [7540, 2750, 14630, 610, 6400, 7590];
satz = [20140, 18610, 13480, 18390, 18660, 18490];

% receiver
recx = result(1);
recy = result(2);
recz = result(3);
receiver = [recx, recy, recz];

% sphere with radius of earth in km
```

```
r = 6371;
[X,Y,Z] = sphere(30);
X = X*r;
Y = Y*r;
Z = Z*r;

% Graphing
% ------------------------------
figure
hold on
grid on
view(10, 30)
axis vis3d

% plot earth, receiver, satellites, & lines from sats to receiver
surf(X, Y, Z, 'FaceColor', 'blue')
scatter3(recx, recy, recz, 90, 'MarkerFaceColor', 'green')
scatter3(satx, saty, satz, 90, 'MarkerFaceColor', 'blue')
for n = 1:6
    line([recx, satx(n)], [recy, saty(n)], [recz, satz(n)], 'Color', 'black')
end
```

Code Listing 2 – gps_resid.m

```
function F = gps_resid(a)
    % speed of light km/s
    c = 299792;

    F(1) = sqrt( (a(1)-15600)^2 + (a(2)-7540 )^2 + (a(3)-20140)^2 ) - c*(0.07074 - a(4));
    F(2) = sqrt( (a(1)-18760)^2 + (a(2)-2750 )^2 + (a(3)-18610)^2 ) - c*(0.07220 - a(4));
    F(3) = sqrt( (a(1)-17610)^2 + (a(2)-14630)^2 + (a(3)-13480)^2 ) - c*(0.07690 - a(4));
    F(4) = sqrt( (a(1)-19170)^2 + (a(2)-610  )^2 + (a(3)-18390)^2 ) - c*(0.07242 – a(4));
```

Code Listing 3 – gp_resid_overdetermined.m

```
function F = gps_resid_overdetermined(a)
    % speed of light km/s
    c = 299792;
    satx = [15600, 18760, 17610, 19170, 17800, 17500];
    saty = [7540, 2750, 14630, 610, 6400, 7590];
    satz = [20140, 18610, 13480, 18390, 18660, 18490];
    time = [0.07074, 0.07220, 0.07690, 0.07242, 0.07320, 0.07350];

    F = sqrt( (a(1)-satx).^2 + (a(2)-saty).^2 + (a(3)-satz).^2 ) - c*(time - a(4));

    % Creates a vector with the following entries:

    % F(1) = sqrt( (a(1)-15600)^2 + (a(2)-7540 )^2 + (a(3)-20140)^2 ) - c*(0.07074 - a(4));
    % F(2) = sqrt( (a(1)-18760)^2 + (a(2)-2750 )^2 + (a(3)-18610)^2 ) - c*(0.07220 - a(4));
    % F(3) = sqrt( (a(1)-17610)^2 + (a(2)-14630)^2 + (a(3)-13480)^2 ) - c*(0.07690 - a(4));
    % F(4) = sqrt( (a(1)-19170)^2 + (a(2)-610  )^2 + (a(3)-18390)^2 ) - c*(0.07242 - a(4));
    % F(5) = sqrt( (a(1)-17800)^2 + (a(2)-6400 )^2 + (a(3)-18660)^2 ) - c*(0.07320 - a(4));
    % F(6) = sqrt( (a(1)-17500)^2 + (a(2)-7590 )^2 + (a(3)-18490)^2 ) - c*(0.07350 - a(4));
```