Global Positioning System Project

Luke Johnston

Biola University

Numerical Analysis

October 4, 2022

Abstract

This paper demonstrates the application and results of the GPS project for Math 321 - Numerical Analysis. This project applies the multivariable case of Newton's method to find the solutions to the nonlinear equations needed for finding gps coordinates from synchronized satellite locations.

1 Introduction

A key procedure in Numerical Analysis is finding the roots of systems of equations. One method for this purpose is called Newton's method or Newton-Raphson iteration. Newton's method can also be used on a system of nonlinear equations.

A system of N nonlinear equations with the unknowns x_i can be written using vector notation:

$$\mathbf{F(X) = 0}$$

With the column vectors defined as

$$\mathbf{F} = [f_1, f_2, f_3, \ldots, f_n]$$
$$\mathbf{X} = [x_1, x_2, x_3, \ldots, x_n]$$

Newton's method for nonlinear systems is

$$\mathbf{X^{(k+1)} = X^{(k)} - [F'(X^{(k)})]^{-1}F(X^{(k)})}$$

Where $\mathbf{F'(X^{(k)})}$ is the Jacobian Matrix. It is the partial derivatives of **F** evaluated at $\mathbf{X^{(k)}}$. We may assume that the Jacobian Matrix is nonsingular, so its inverse exists. Solving for H:

$$\mathbf{H \approx -[F'(X^{(k)})]^{-1}F(X^{(k)})}$$

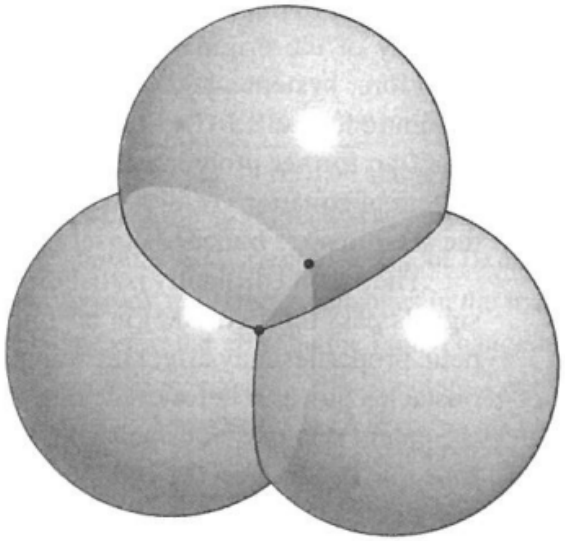
Letting $\mathbf{X^{(k+1)} = X^{(k)} + H}$, will give us a better approximation. This is done recursively to find the approximate solution.

(Cheney & Kincaid, p.132-133)

2 GPS Application

2.1 Theory

Satellites transmit carefully synchronized signals to GPS receivers here on Earth. The receiver uses the information transmitted, with some clever math, to determine the exact coordinates of the receiver. The receiver only needs three synchronized satellite signals to determine the intersection of three spheres. Only one of these intersection points is physically possible to be the location of the receiver, so we can ignore the other point. This would be all we need, except the clocks found on most receivers are not accurate enough to rely on for these precise measurements. So we must include a forth satellite signal in order to find the difference between the synchronized time of the satellites and the receiver.



(Sauer, 239)

We will use Newton's multivariable method to solve these four equations required for the GPS system. We are looking to find the values of (x, y, z) and d.

$$r_1(x, y, z, d) = \sqrt{(x - A_1)^2 + (y - B_1)^2 + (z - C_1)^2} - c(t_1 - d) = 0$$
$$r_2(x, y, z, d) = \sqrt{(x - A_2)^2 + (y - B_2)^2 + (z - C_2)^2} - c(t_2 - d) = 0$$
$$r_3(x, y, z, d) = \sqrt{(x - A_3)^2 + (y - B_3)^2 + (z - C_3)^2} - c(t_3 - d) = 0$$
$$r_4(x, y, z, d) = \sqrt{(x - A_4)^2 + (y - B_4)^2 + (z - C_4)^2} - c(t_4 - d) = 0$$

(Sauer, 239)

The values of A_i, B_i, C_i represent the locations of the 4 satellites, and t_i represents the time required for the satellite to communicate with the receiver. The values of x, y, z represent the location of the GPS receiver and d represents the time difference between the satellites and the receiver. c is a constant for the speed of light, which we will approximate to be 299792.458 km/s. The unknown values can be found by using Newton's multivariable method.

2.2 GPS Activity #1

(Sauer, p.240)

Find the receiver position (x, y, z) and the time correction (d) for the simultaneous satellite positions (15600, 7540, 20140), (18760, 2750, 18610), (17610, 14630, 13480), (19170, 610, 18390) in km, and the measured time intervals are 0.07074, 0.07220, 0.07242, 0.07242 sec. Set the initial vector to be $(x_0, y_0, z_0, d_0) = (0, 0, 6370, 0)$.

We must develop a matlab program to solve the gps systems of equations, using Newton's multivariable method. Before we can get to working with the satellite data we must understand how Newton's method works. So, we created a Matlab function called newtonM.m, which contains the code required for the Multivariable Newton Method. The newtonM function was tested with multiple different inputs and all of the outputs were correct. The next step required for our gps program is to use the Multivariable Newton Method to find the convergence values associated with the values given from 4 different satellites. These convergence values will give us the gps coordinates (x, y, z) and time correction (d) from the numbers given by the satellites. We modified the Newton method program, so it contains the given equations needed to find the location of the receiver. The new program is called gpsNewton.m, and it will return the location and time correction value.

The output of the gpsNewton function returns the correct values of (x, y, z) = (-41.77271, -16.78919, 6370.0596), and d = -3.201566 x 10^{-3} sec.

2.3 Mission Impossible: Fallout

Secret agent Ethan Hunt needs our help to find the location of the fourth plutonium core! Benji and Luther have found the following four simultaneous satellite locations with the time correction. This is what we will use to find the exact coordinates of the plutonium.

	Sat 1	**Sat 2**	**Sat 3**	**Sat 4**
x	9261	12820	-10317	530
y	1214	-9599	9597	-2397
z	24874	21201	22526	1120
t	0.07407	0.07055	0.07789	0.07100

Sending this satellite data into the gpsNewton.m function from the previous activity, with the initial vector set to (x_0, y_o, z_0, t_0) = (0, 0, 15000, 0). The function returns the values: (x, y, z, d) = (-4869.1, -9260.8, 17178, 0.010028). Converting to latitude and longitude: The location of the fourth plutonium is latitude: 58.655 , longitude: -117.73.

The plutonium is in Alberta, Canada!

2.4 Buried treasure

Lara Croft needs our help to find the location of buried treasure. She intercepted the simultaneous satellite locations sent to the pirate king's receiver while he buried the treasure. Find the x, y, z coordinates and d for the given simultaneous satellite locations. Then find the latitude and longitude of the buried treasure.

	Sat 1	**Sat 2**	**Sat 3**	**Sat 4**
x	4376	15846	23689	740
y	890	12948	9783	8437
z	9376	19375	13693	1367
t	0.06501	0.07167	0.07234	0.07077

Solution:
(x, y, z) = (11926, 8809, 7600.9)
d = 0.028034
latitude: 27.141
longitude: 36.45

The buried Treasure is in Saudi Arabia!

3 Appendix: Matlab Scripts

3.1 newtonM.m:

```
function [x] = newtonM(X)

x = X.';

for k=1:10
   F = [(x(1) + x(2) + x(3) -3);
        ((x(1))^2 + (x(2))^2 + (x(3))^2 -5);
        (exp(x(1)) + (x(1)*x(2)) - (x(1)*x(3)) -1)];

   J = [ 1 1 1;
        (2*x(1)) (2*x(2)) (2*x(3));
        (exp(x(1)) + x(2) - x(3)) x(1) (-x(1))];

   H = J\F;
   x = x-H;

end
end
```

3.1.1 testNewton.m:

```
X = [1 0 1];
newX = newtonM(X);
disp(newX)
```

3.2 gpsNewton.m:

```
function [v] = gpsNewton(sat1, sat2, sat3, sat4, x0, y0, z0, d0)

cc=299792.458;    %speed of light
A = [sat1(1), sat2(1), sat3(1), sat4(1)];
B = [sat1(2), sat2(2), sat3(2), sat4(2)];
C = [sat1(3), sat2(3), sat3(3), sat4(3)];
t = [sat1(4), sat2(4), sat3(4), sat4(4)];

v = [x0, y0, z0, d0];

for k=1:4
   x = v(1);
   y = v(2);
   z = v(3);
   d = v(4);
```

```
   F = [((x - A(1)).^2 + (y - B(1)).^2 + (z - C(1)).^2 - (cc*(t(1) - d)).^2);
   ((x - A(2)).^2 + (y - B(2)).^2 + (z - C(2)).^2 - (cc*(t(2) - d)).^2);
   ((x - A(3)).^2 + (y - B(3)).^2 + (z - C(3)).^2 - (cc*(t(3) - d)).^2);
   ((x - A(4)).^2 + (y - B(4)).^2 + (z - C(4)).^2 - (cc*(t(4) - d)).^2); ];

   J = [(2*(x-A(1))) (2*(y-B(1))) (2*(z-C(1))) (2*cc*(cc*t(1)-cc*d));
       (2*(x-A(2))) (2*(y-B(2))) (2*(z-C(2))) (2*cc*(cc*t(2)-cc*d));
       (2*(x-A(3))) (2*(y-B(3))) (2*(z-C(3))) (2*cc*(cc*t(3)-cc*d));
       (2*(x-A(4))) (2*(y-B(4))) (2*(z-C(4))) (2*cc*(cc*t(4)-cc*d)); ];

   H = J\F;

   v(1) = v(1) - H(1);
   v(2) = v(2) - H(2);
   v(3) = v(3) - H(3);
   v(4) = v(4) - H(4);

end
end
```

3.2.1 testGPS.m:

```
format shortG;

%satellite locations and time corrections
sat1 = [9261 1214 24874 0.07407];
sat2 = [12820 -9599 21201 0.07055];
sat3 = [-10317 9597 22526 0.07789];
sat4 = [530 -2397 1120 0.07100];

%initial values
x0=0;
y0=0;
z0=15000;
d0=0;

v = gpsNewton(sat1, sat2, sat3, sat4, x0, y0, z0, d0);
disp(v)

%find the latitude and longitude
r = sqrt((v(1))^2+(v(2))^2+(v(3))^2);
long = rad2deg(atan2(v(2),v(1)));
lat = rad2deg(pi/2-acos(v(3)/r));
disp([lat, long]);
```

References

Cheney, W., Kincaid, D., (2013). Numerical Mathematics and Computing (7th ed.). Cengage Learning.

Sauer, T., (2006). Numerical Analysis. Pearson.