Project 2: The GPS Problem

Math 5400 /AIM 5003 Numerical Methods

Yeshiva University: Katz School of Science and Health

Radek Holik, Pinxue Lin, Rivka Shapiro

Fall 2022

Introduction

The Global Positioning System (GPS) is an invaluable tool in the modern world, but it is also an excellent case from which to study the properties of nonlinear systems and their solvers. The GPS receives the location from multiple satellites, which all orbit Earth. It then identifies the time delay of the transmission, allowing it to adjust the coordinates to real-time. Finally, the GPS calculates the intersection of the spherical distances of all the satellites, which is the location of the GPS receiver. To compensate for inaccuracy inherent in many Earth-bound clocks, an extra satellite is added, purely to denote the difference in counting between the satellite and earth-bound clocks. Finding the intersection of spheres requires complex calculations and the solving of nonlinear equations. The goal of this project is to apply our knowledge of solving nonlinear systems to understand these calculations, and to assess the capacity for error through multiple avenues of analysis.

Activity 1

In order to experiment and analyze the GPS problem, it is necessary to better understand the equations which derive position from satellite data. For every satellite, *i*, the equation:

$$r_i(x, y, z, d) = \sqrt{(x - A_i)^2 + (y - B_i)^2 + (z - C_i)^2} - c(t_i - d) = 0$$

describes the position of the receiver on earth as the x, y, and z solutions to the function r_i, in which the values A_i, B_i, and C_i are the coordinates of the satellite. The value t_i is the time it takes the transmission of the data to travel from the satellite to the receiver, and *d* is the time-keeping difference between the orbital and grounded clocks. With multiple satellites, it is possible to produce and solve the system of equations:

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

In this first activity, we apply a multivariate Newton solving method to find the position of the receiver based on given satellite data.

The multivariate Newton Method is iterative, just like the univariate Newton Method. The key difference is that instead of using the derivative of a univariate function, a Jacobian matrix is used for the system of equations. The Jacobian is then applied to the linear equation $J(x_k)\vec{s} = F(x_k)$, which is then solved. Using this form circumvents the need to invert the Jacobian matrix, a computationally burdensome process. The algorithm alters the value of *x* before each new iteration to be $x_{k+1} = x_k - s$. The multivariate newton algorithm can be stopped using a set tolerance or maximum iteration number, as well as a combination of the two.

In order to execute this and the subsequent activities, we coded the project in python. Notably, we developed a class of an object called GPS. Included in the class are many methods, which we will use to implement the mathematical analysis of the GPS problem. We produced methods which will find the Jacobian matrix, and carry out the multivariate newton algorithm. To test the efficacy of the Multivariate Newton Method, we will use the given data:

$$A_1 = 15600, B_1 = 7540, C_1 = 20140, t_1 = 0.07074$$
$$A_2 = 18760, B_2 = 2750, C_2 = 18610, t_2 = 0.07220$$
$$A_3 = 17610, B_3 = 14630, C_3 = 13480, t_3 = 0.07690$$
$$A_4 = 19170, B_4 = 610, C_4 = 18390, t_4 = 0.07242$$

For a starting point, we will use the recommended vector: (0, 0, 6370, 0). Our multivariate Newton solver was set with a maximum iteration number of 30, and a tolerance of 10^{-16}, with instruction to stop when either has been met. With these stopping conditions, our solver produced a solution in six iterations, in a time of 4.986 milliseconds. The solution found was:

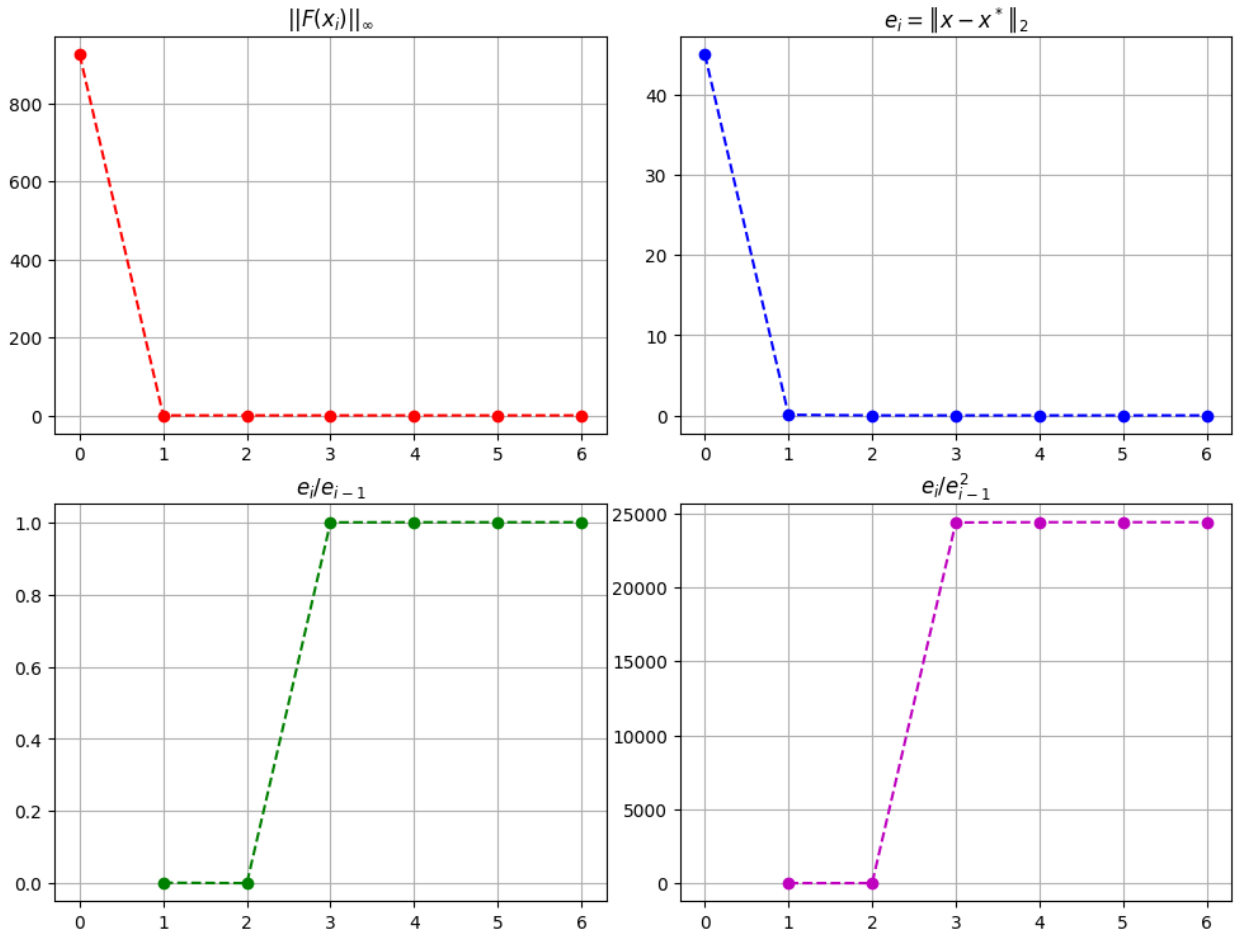
$$x = -41.7727096, y = -16.7891941, z = 6370.05956, d = -0.00320156583$$

This solution is exactly equal to the true solution provided as a check, demonstrating the accuracy of the solver.

We demonstrated the quadratic convergence of the multivariate Newton method through calculating the error between each step, and applying the equation:

$$M = \lim_{i \to \infty} \frac{e_{i+1}}{e_i^2} < \infty$$

The convergence rate is then approximately Me_i^2, a quadratic. We also plotted graphs of errors for the solution, to demonstrate the efficiency of our solver.



Activity 2

In this activity, we will try to solve the system of satellite equations quadratically. This first requires some algebraic manipulations. We begin by squaring the original system to achieve:

$$(x - A_1)^2 + (y - B_1)^2 + (z - C_1)^2 - [c(t_1 - d)]^2 = 0 \quad (1)$$

$$(x - A_2)^2 + (y - B_2)^2 + (z - C_2)^2 - [c(t_2 - d)]^2 = 0 \quad (2)$$

$$(x - A_3)^2 + (y - B_3)^2 + (z - C_3)^2 - [c(t_3 - d)]^2 = 0 \quad (3)$$

$$(x - A_4)^2 + (y - B_4)^2 + (z - C_4)^2 - [c(t_4 - d)]^2 = 0 \quad (4)$$

Then, the equations are rearranged to place the term containing *c*, the speed of light, on the right-hand side.

$$x^2 + y^2 + z^2 + A_1^2 + B_1^2 + C_1^2 - 2(A_1x + B_1y + C_1z) = c^2(t_1^2 - 2t_1d + d^2) \quad (1)$$

$$x^2 + y^2 + z^2 + A_2^2 + B_2^2 + C_2^2 - 2(A_2x + B_2y + C_2z) = c^2(t_2^2 - 2t_2d + d^2) \quad (2)$$

$$x^2 + y^2 + z^2 + A_3^2 + B_3^2 + C_3^2 - 2(A_3x + B_3y + C_3z) = c^2(t_3^2 - 2t_3d + d^2) \quad (3)$$

$$x^2 + y^2 + z^2 + A_4^2 + B_4^2 + C_4^2 - 2(A_4x + B_4y + C_4z) = c^2(t_4^2 - 2t_4d + d^2) \quad (4)$$

Then, subtract the equations from one another to compile all terms into the resulting equation:

$$A_1^2 - A_4^2 + B_1^2 - B_4^2 + C_1^2 - C_4^2 + 2x(A_4 - A_1) + 2y(B_4 - B_1) + 2z(C_4 - C_1) = c^2(t_1^2 - t_4^2 + 2d(t_4 - t_1))$$

The equation can also be put into a matrix form:

$$\begin{bmatrix} 2(A_2 - A_1) & 2(B_2 - B_1) & 2(C_2 - C_1) & 2c^2(t_1 - t_2) \\ 2(A_3 - A_1) & 2(B_3 - B_1) & 2(C_3 - C_1) & 2c^2(t_1 - t_3) \\ 2(A_4 - A_1) & 2(B_4 - B_1) & 2(C_4 - C_1) & 2c^2(t_1 - t_4) \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ d \end{bmatrix} = \begin{bmatrix} A_2^2 - A_1^2 + B_2^2 - B_1^2 + C_2^2 - C_1^2 + c^2(t_1^2 - t_2^2) \\ A_3^2 - A_1^2 + B_3^2 - B_1^2 + C_3^2 - C_1^2 + c^2(t_1^2 - t_3^2) \\ A_4^2 - A_1^2 + B_4^2 - B_1^2 + C_4^2 - C_1^2 + c^2(t_1^2 - t_4^2) \end{bmatrix}$$

The matrix can be converted to row-reduced echelon form:

$$r = \left[\begin{array}{cccc|c} 1 & 0 & 0 & r_{14} & r_{15} \\ 0 & 1 & 0 & r_{24} & r_{25} \\ 0 & 0 & 1 & r_{34} & r_{35} \end{array}\right]$$

The RREF is used to find *x, y*, and *z* in terms of *d*, and write the previous equation in terms of *d*.

$$x = -r_{14}d + r_{15}, y = -r_{24}d + r_{25}, z = -r_{34}d + r_{35}$$

$$(\mathrm{r}_{14}^2 + r_{24}^2 + r_{34}^2 - c^2)d^2 +$$
$$+2(\text{- } \mathrm{r}_{14}r_{15} - r_{24}r_{25} - r_{34}r_{35} + A_1r_{14} + B_1r_{24} + C_1r_{34} + c^2t_1)d +$$
$$+(\mathrm{r}_{15}^2 + A_1^2 - 2r_{15}A_1 + r_{25}^2 + B_1^2 - 2r_{25}B_1 + r_{35}^2 + C_1^2 - 2r_{35}C_1 - c^2t_1^2) = 0$$

This equation takes a quadratic form, and the multi-term coefficients of *d* can be assigned to *a, b, c*. One can solve for *d* using the quadratic formula, and then find the coordinates by plugging the value of *d* into the equations for *x, y,* and *z*.

There is a slight catch to solving quadratically; the formula produces two solutions. However, it is easy to identify the desired solution, as the second will often be outside or inside the dimensions of earth, or very large *z* (high above/below the surface).

Putting the steps outlined above into a method for our GPS class, the solution was found to be $x = -41.7727096, y = -16.7891941, z = 6370.05956, d = -0.0032016583$. It took 2.9 milliseconds to compute, faster than the multivariate newton method.

The solution through the quadratic solver is better than the Multivariate Newton Method for a few reasons. First, it requires a derivative that can be computationally expensive. Secondly, it is an iterative method that uses an initial guess of the solution. If this guess is "enough far" from the true solution then convergence is not guaranteed for this method. In contrast, the method based on the Quadratic formula does not have these drawbacks. For these advantages, it is better to use the method based on the Quadratic formula.

Activity 4

In this activity we will calculate new spherical coordinates for the satellites, and compute a true solution to this new system. Then, we will investigate the errors in our multivariate Newton solver by applying a miniscule time difference and solving the system again. These results will be used to calculate the forward error, backward error, the error magnification factor, maximum forward error, and the condition number of the problem.

The new satellite coordinates are calculated based on the following:

$$\text{for i} = 1, 2, 3.... : (\phi_i, \theta_i) = \left(i\frac{\pi}{8}, (i-1)\frac{\pi}{2}\right)$$

The new spherical coordinates are then implemented to the form taken by the satellite equations detailed in Activity 1 through the following conversion:

$$\text{A}_i = \rho cos(\phi_i)cos(\theta_i), B_i = \rho cos(\phi_i)sin(\theta_i), C_i = \rho sin(\phi_i)$$

This ultimately results in the following input coordinates for the four satellites:

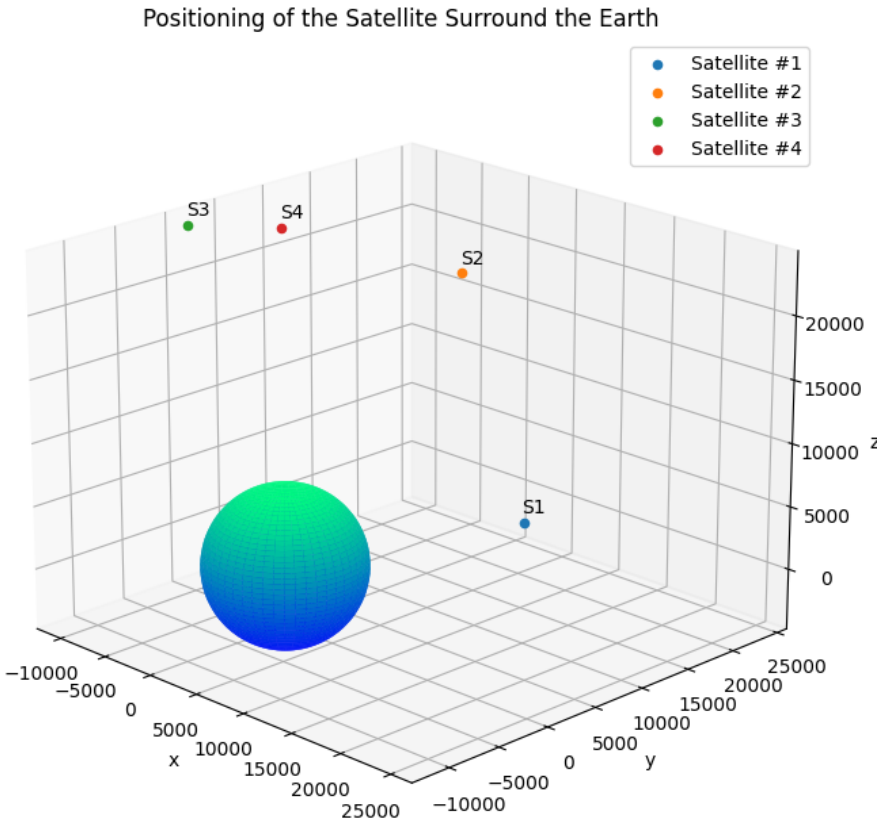
$$A_1 = 24547.4792, B_1 = 0, C_1 = 10167.8988, d_1 = 0.0829557883 \quad (1)$$

$$A_2 = 1.15042262 * 10^{-12}, B_2 = 18787.8272, C_2 = 18787.8272, d_2 = 0.0752211879 \quad (2)$$

$$A_3 = -10167.8988, B_3 = 1.24520847 * 10^{-12}, C_3 = 24547.4792, d_3 = 0.0695748363 \quad (3)$$

$$A_4 = -2.98864631 * 10^{-28}, B_4 = -1.62694327 * 10^{-12}, C_4 = 26570.0000, d_4 = 0.0674799472 \quad (4)$$

We used these coordinates to plot the four satellites in relation to their positions above the Earth



As the reference solution we use the suggested one.

$$[0,\ 0,\ 6370,\ 0.0001]$$

Forward error can be described as the change in position, $\|(\Delta x, \Delta y, \Delta z)\|_\infty$, while backward error relates to the change in distance, or $c \cdot \|(\Delta t_1, \ldots, \Delta t_m)\|_\infty$. Forward error can also be thought of as the error in the solution, while backward error is related to the error in the data itself. We will calculate both types of error for this system using four cases of time differences, where $\epsilon = 10^{-8}$.

Case	Time Variation	Forward Error	Backward Error	EMF
1	+ϵ, +ϵ, +ϵ, -ϵ	0.01641452326111903	0.0029979245799999998	5.475295599704189
2	+ϵ, +ϵ, -ϵ, -ϵ	0.005427501490885334	0.0029979245799999998	1.8104196239937878
3	+ϵ, -ϵ, +ϵ, -ϵ	0.016414523678577098	0.0029979245799999998	5.475295738953212
4	+ϵ, -ϵ, -ϵ, -ϵ	0.005427501438134641	0.0029979245799999998	1.8104196063980507

Based on this data, the maximum position error is 16.414523678577098 meters, and the maximum EMF is 5.475295738953212. The condition number is simply the maximum EMF, which from this data is 5.475295738953212. The backwards error is the same for all four cases, as the absolute value of the change remained the same throughout, as we variated only in the signs of ϵ.

Activity 5

In this activity, we will repeat the analysis conducted in Activity 4, but with satellites which are more tightly grouped. We can then observe the effect of the distance between the satellites on the results of the solver. With the same ρ as in activity 4, 26570.0, the equation for generating new, closely bounded, spherical coordinates is:

$$(\phi_i, \theta_i) = \left(\frac{\pi}{2} + (i-1)\frac{5}{100}\frac{\pi}{2}, (i-1)\frac{5}{100}2\pi\right)$$

For the four satellites, this equation produced cartesian input coordinates of:

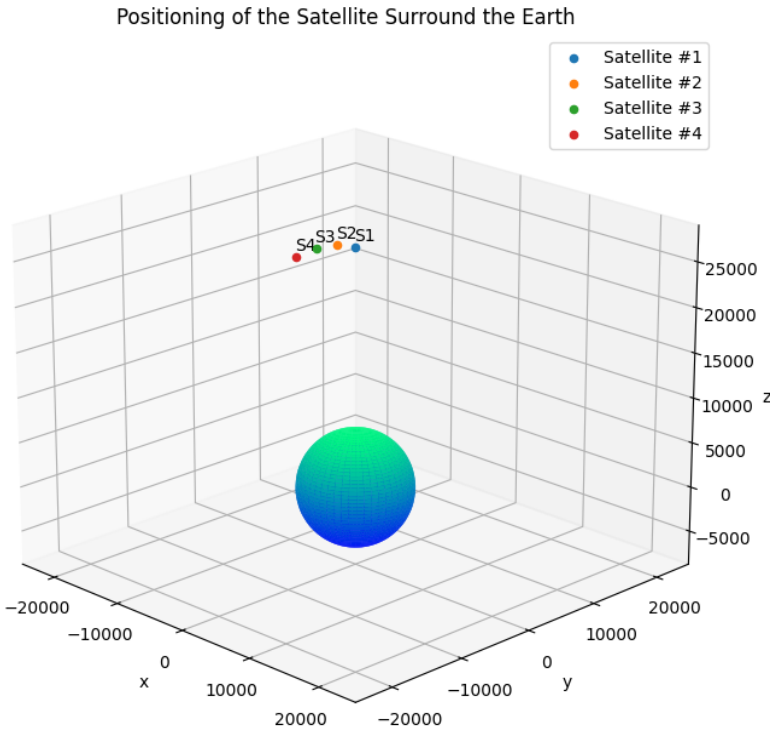
$$A_1 = 1.6269432726672588 * 10^{-12}, B_1 = 0.0, C_1 = 26570.000000000000, d_1 = 0.067479947230026718 \quad (1)$$

$$A_2 = -1982.6277401445122, B_2 = -644.19480307068841, C_2 = 26488.093557289209, d_2 = 0.067566048197813727 \quad (2)$$

$$A_3 = -3362.6497990234034, B_3 = -2443.1080857791821, C_3 = 26242.879209612809, d_3 = 0.067823165827943163 \quad (3)$$

$$A_4 = -3645.8222673459527, B_4 = -5018.0438540261921, C_4 = 25835.868784966267, d_4 = 0.068247792155330125 \quad (4)$$

We then plotted the new set of satellites using these coordinates. The plot helps to illustrate the difference between the sets of satellite positions used.



As the reference solution we use the suggested one.

$$[0,\ 0,\ 6370,\ 0.0001]$$

Using this solution, we then calculated forward and backward error using time variance with an $\epsilon = 10^{-8}$ as previously. The results are pictured in the table below.

Case	Time Variation	Forward Error	Backward Error	EMF
1	+ϵ, +ϵ, +ϵ, -ϵ	1.9073021811436774	0.0029979245799999998	636.2075263226527
2	+ϵ, +ϵ, -ϵ, -ϵ	3.4719285808905624	0.0029979245799999998	1158.1107156773646
3	+ϵ, -ϵ, +ϵ, -ϵ	17522.92530370137	0.0029979245799999998	5845018.724153952
4	+ϵ, -ϵ, -ϵ, -ϵ	2.145353514901217	0.0029979245799999998	715.6129040781997

It is evident that the forward error, and therefore the EMF and maximum position error has greatly increased. This shows that closely bounded satellites are much less accurate than satellites which are farther apart. Thus, close satellites can be considered ill-conditioned. The increase in error reduces the accuracy with which we can calculate the position of the receiver. The maximum position error is 17522925.30370137 meters, and the maximum EMF is 5845018.724153952.

Activity 6

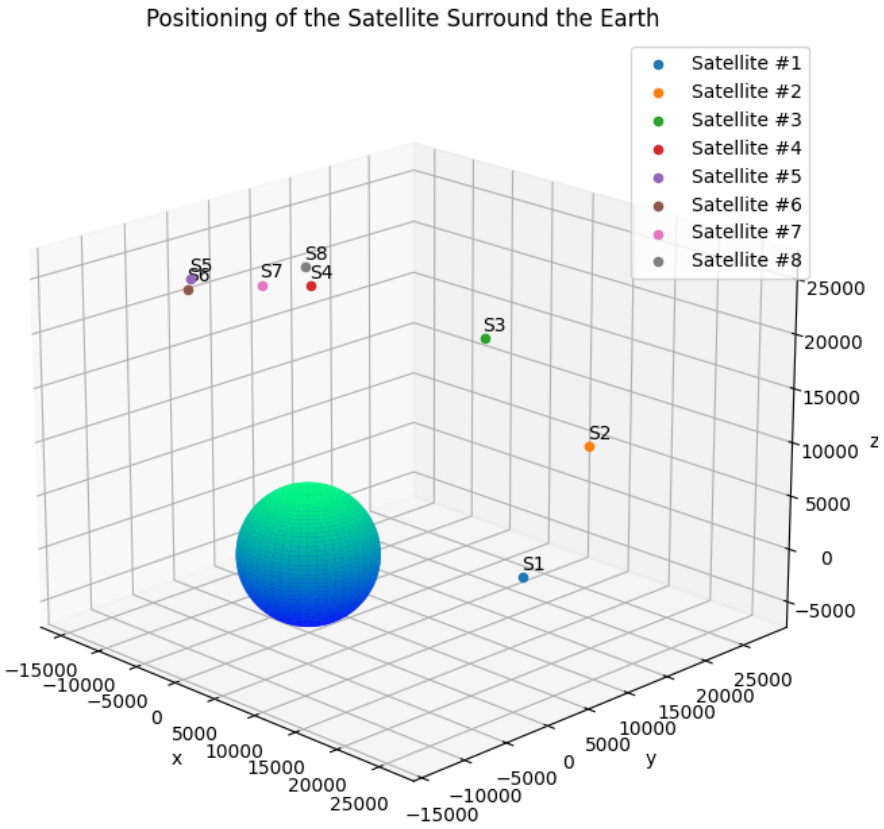
In this activity we will further investigate the effects of the number of satellites and their arrangement on the accuracy of the results. We will return to a more spread out configuration, but this time with a total of eight satellites. The spherical coordinates of the satellites is dictated by the equation:

$$\text{for i} = 1,2\ldots 8\text{:}\ (\phi_i, \theta_i) = \left(i\frac{\pi}{16}, (i-1)\frac{\pi}{4}\right)$$

The satellite parameters can be written in the following matrix form:

$$\begin{bmatrix} 26059.464900313833 & 0 & 5183.5498559685275 & 0.087115062957788855 \\ 17357.688988382663 & 17357.688988382663 & 10167.898797940436 & 0.082955788323008886 \\ 1.3527538921628798 * 10^{-12} & 22092.147598878626 & 14761.501091330831 & 0.078928491708490159 \\ -13284.999999999998 & 13285.000000000002 & 18787.827176126568 & 0.075221187899139244 \\ -14761.501091330832 & 1.8077625062108464 * 10^{-12} & 22092.147598878626 & 0.072036083490136474 \\ -7189.7901904422297 & -7189.7901904422279 & 24547.479178824888 & 0.069574836349750174 \\ -9.5220266089988761 * 10^{-13} & -5183.5498559685302 & 26059.464900313833 & 0.068014847159579603 \\ 1.1504226207088527 * 10^{-12} & -1.1504226207088531 * 10^{-12} & 26570.0 & 0.067479947230026718 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ d \end{pmatrix}$$

The eight-satellite system looks like:



We can then use a Gauss-Newton solver to find a least squares solution. Least squares is necessary in this case because our system is inconsistent, with four unknowns and eight equations. The Gauss-Newton solver takes the gradient of the system as

$\nabla E(x) = \nabla\left(\frac{1}{2}r^T r\right) = Dr^T r = 0$, where *Dr* is the Jacobian matrix of the system of equations. Multivariate Newton can then be applied to solve the equation $Dr^T r = 0$, treating $Dr^T r$ as *F(x)*. However, Multivariate Newton requires the Jacobian of *F(x)*, which will be cumbersome to compute as *F(x)* itself contains a Jacobian. Thus, we simplify the calculation by ignoring second order (Hessian) terms and writing $J_F = Dr^T Dr$. Then, instead of solving with the normal equations, QR factorization is used to achieve the least squares solution.

We ran the Gauss-Newton solver with the given true solution from activity 1, [0,0,6370,0] , as the initial vector. It took the solver just one iteration and 2 milliseconds to produce a solution within the 10^{-16} tolerance. The solution found was:

$x = -6.9164305 * 10^{-15}$, $y = -2.367908 * 10^{-15}$, $z = 6370$, $d = 1.0000000000000358 * 10^{-4}$

Universally, the Gauss-Newton Method is only locally convergent if the initial vector is that which is "close enough" to the true solution. It is hard to make a clear statement about what "close enough" means. Therefore a good initial vector is "close enough" to the true solution.

We must check whether the solution found is a minimum, maximum, or saddle point. Usually, one would conduct a second-derivative test to make this identification. However, since we are working with a multivariable system, taking a second derivative would be complex and hugely costly. Therefore, we will approximate, and say that a minimum is found when F(x_s, y_s, z_s, d_s) is close or equal to 0.

To find the error in the Gauss-Newton solver, we executed the same test of variance as in activities 4 & 5. The results are as follows:

Case	Time Variation	Forward Error	Backward Error	EMF
1	+ϵ, +ϵ, +ϵ, -ϵ, +ϵ, +ϵ, +ϵ, -ϵ	0.004226754634146346	0.0029979245799999998	1.409893585163622
2	+ϵ, +ϵ, -ϵ, -ϵ, +ϵ, +ϵ, -ϵ, -ϵ	0.009461211387133517	0.0029979245799999998	3.155920415827645
3	+ϵ, -ϵ, +ϵ, -ϵ, +ϵ, -ϵ, +ϵ, -ϵ	0.004214662751110154	0.0029979245799999998	1.405860167139413
4	+ϵ, -ϵ, -ϵ, -ϵ, +ϵ, -ϵ, -ϵ, -ϵ	0.009449118881093455	0.0029979245799999998	3.151886789991713

From this data, we can understand the maximum position error to be 9.461211387133517 meters, the maximum error magnification factor is about 3.155920415827645, and the condition number is approximately the same as emf.

We repeated this experiment with different variations of time, to see if we could produce consistent results, or if the low error was related to the specific variations shown. Indeed, the least-squares solutions produced lower error across the board.

Case	Time Variation	Forward Error	Backward Error	EMF
1	-ϵ, -ϵ, -ϵ, -ϵ, -ϵ, -ϵ, -ϵ, +ϵ	0.006531146203087701	0.0029979245799999998	2.1785558738397954
2	+ϵ, +ϵ, +ϵ, +ϵ, -ϵ, -ϵ, -ϵ, -ϵ	0.005286281373628299	0.0029979245799999998	1.763313663357168
3	-ϵ, -ϵ, -ϵ, -ϵ, +ϵ, +ϵ, +ϵ, +ϵ	0.005286283267196268	0.0029979245799999998	1.7633142949834544
4	-ϵ, +ϵ, -ϵ, +ϵ, -ϵ, +ϵ, -ϵ, +ϵ	0.004214663842503796	0.0029979245799999998	1.4058605311891457

The maximum position error for this experiment is 6.531146203087701 meters, and the condition number is approximately 2.1785558738397954. This is certainly an improvement from previous experiments, and looks very promising. However, we cannot fully prove that this method actually decreases error without far more thorough experimentation with the time variance.

Conclusion

The Multivariate Newton Method has a few drawbacks compared to the method based on the Quadratic formula. First, it requires a derivative that can be computationally expensive. Secondly, it is an iterative method that uses an initial guess of the solution. If this guess is "enough far" from the true solution then convergence is not guaranteed for this method. In contrast, the method based on the Quadratic formula does not have these drawbacks due to this method deals with several algebraic operations and one matrix operation. Finally, calculate the roots of the quadratic equation. For these advantages, it is better to use the method based on the Quadratic formula.

If we look back and assess the deployment of the four unbunched and bunched satellites, we find that the unbunched satellites provide higher accuracy. Specifically, the four unbunched satellites have a maximum positional error of 16.41 meters and the condition number of approximately 14.57. In contrast, the four bunched satellites have the maximum position error of 17522925.3 meters and the condition number of approximately 5845018.72. From these numbers, we can see the "immense" difference between these two cases. When adding another four satellites (eight unbunched satellites together), we find that they have the maximum positional error of 9.46 meters and the condition number of approximately 3.16. When using other combinations of Δt, even the maximum positional error was reduced to the value of 6.53 meters and the maximum emf to the value of 2.17. Using eight satellites increased the accuracy by approximately twice compared to using four satellites. If we decide that accuracy is our criterion, and higher accuracy is always better then more satellites increase accuracy. It is clear that eight unbunched satellites give us the best accuracy over four unbunched or bunched satellites.

Resources

Numerical Analysis by Timothy Sauer, (2nd Edition, p. 238-242)