Project 2: GPS SATELLITE POSITIONING

Group 2: Team Members: Sana Omar, Haider Z. Ali, Sai Kalishetty

Problem Setup:

The Global Positioning System (GPS) consists of 24 satellites carrying atomic clocks, orbiting the earth at an altitude of 20,200 km. Four satellites in each of six planes, slanted at 55 degree with respect to the poles, make two revolutions per day. At any time, from any point on earth, five to eight satellites are in the direct line of sight. Each satellite has a simple mission: to transmit carefully synchronized signals from predetermined positions in space, to be picked up by GPS receivers on earth. The receivers use the information, with some mathematics (resulted in the following show equations), to determine accurate (x, y,z) coordinates of the receiver.
There is a difference in time d between the receiver and the satellite clocks, the problem of solving the distance value depends on this difference, because we approximate the location using the travel time in between both.
Also, by adding one more satellite, we solve the problem of inaccurate timing represented by wrong kilometers

1. Introduction

We need to find co-ordinates of a person standing on the circumference of the earth who is connected with three satellites revolving around the earth using a device which has a GPS connectivity.
equations used to solve this problem:

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

Activity 1

In the first activity, we use the multivariate newton method to solve the location of the receiver.
This method solves nonlinear equations, based on the Jacobian matrix and the original set of functions along with initial guess and a predetermined tolerance until a root is found which is considered the nearest plane intersection – if there is no solution or real intersection.
The approach is constructing matrices:
$Jv=-F$
J the Jacobian – it is the partial differentiation of all equations parts
V – the vector on unknown variables
F – the current guess (the function)
From the problem setup we have:

Project 2: GPS SATELLITE POSITIONING

The four satellites coordinate:
(15600,7540,20140),(18760,2750,18610),(17610,14630,13480),(19170,610,18390)

The time intervals d's consequently:
(0.07074,0.07220,0.07690,0.07242) (0.07074,0.07220,0.07690,0.07242)
Starting by initializing the satellites positions for each, constructing an F function of all r's those that represents x-A1, y-B1, etc
Then, constructing the Jacobian matrix using numpy array with applying the numpy.linalg.pinv
linalg.pinv which computes the (Moore-Penrose) pseudo-inverse of a matrix.
Why did we choose this linalg.ping?
because "The pseudo-inverse of a matrix A, denoted A+, is defined as: the matrix that solves [the least-squares problem] Ax=b - this resembles our problem here if xbar is said solution, then A+ is that matrix such that xbar =A+b

Lastly, we get the result:

	Near Earth	Far Earth
X	-41.772710	-39.747837
Y	-16.789194	-134.274144
Z	6370.059559	-9413.62455
D	-0.003202	0.185173

The supposed result of the receiver coordinates is:
(x,y,z,d)=(−41.772709,−16.789194,6370.059559,−0.003201)
Activity 2:
Using the quadratic formula to solve the same problem in the previous activity, according to the question hint, we need to reduce the 4 equations by subtracting the last three ones from the first one,this results in three equations with four unknowns
u = [x,y,z,d]T, or in the form of (matrix): Au = b

The steps (abbreviated here):
the first equation becomes:
$x^2 + y^2 + z^2 + A_1^2 + B_1^2 + C_1^2 - 2(A_1x + B_1y + C_1z) = C^2(t_1^2 + d^2 - t_1d)$
*the rest of the equations are the same with respect to A1 becoming A2.. etc.
By subtracting, we get rid of x, y, z members, and the following (x y z column wise) matrix:
$[2(A2 - A1)\ \ 2(B2 - B1)\ \ 2(C2 - C1)]$
$[2(A3 - A1)\ \ 2(B3 - B1)\ \ 2(C3 - C1)]$
$[2(A4 - A1)\ \ 2(B4 - B1)\ \ 2(C4 - C1)]$
Written as rows, having x y z as columns.
Plus d vector with constants vector - column wise.

Project 2: GPS SATELLITE POSITIONING

$$2C^2(t_1 - t_2) \quad A_1^2 - A_2^2 + B_1^2 - B_2^2 + C_1^2 - C_2^2 + C^2t_2^2 - C^2t_1^2$$
$$2C^2(t_1 - t_3) \quad A_1^2 - A_3^2 + B_1^2 - B_3^2 + C_1^2 - C_3^2 + C^2t_3^2 - C^2t_1^2$$
$$2C^2(t_1 - t_4) \quad A_1^2 - A_4^2 + B_1^2 - B_4^2 + C_1^2 - C_4^2 + C^2t_4^2 - C^2t_1^2$$

Activity 2

Transforming the equation and calculating the determinants:

$$x\vec{u}_x + y\vec{u}_y + z\vec{y}_z + d\vec{u}_d + \vec{w} = 0$$
$$0 = det[\vec{u}_y|\vec{u}_z|x\vec{u}_x + y\vec{u}_y + z\vec{u}_z + d\vec{u}_d + \vec{w}]$$
$$0 = det[\vec{u}_y|\vec{u}_z|x\vec{u}_x] + det[\vec{u}_y|\vec{u}_z|y\vec{u}_y] + det[\vec{u}_y|\vec{u}_z|z\vec{u}_z] + det[\vec{u}_y|\vec{u}_z|d\vec{u}_d] + det[\vec{u}_y|\vec{u}_z|\vec{w}]$$

The quadratic formula relies on the numpy.linalg.det() Method in Python, to calculate determinants of x,y,z and get each point of A,B,C in these directions, then getting d as well.l

Applying our quadratic function, we get

X	X	Y	Z	Distance
Near Earth	-41.772710	-16.789194	6370.059559	-0.003202
Far Earth	-39.747837	-134.274144	-9413.624554	0.185173

How many solutions are there to system (4.37)?
How did you choose the physically correct one among them?

This depends on the fact if this solution has intersection with all polynomial functions or not, so we have four functions, nonlinear with a degree > 1, the solution could be a plane intersecting with some of them, all of them, none of them or the nearest point of intersection.

Activity 4

Equation sets in coordinates (ρ,φi, θi):
Ai = ρ cosφi cos θi
Bi = ρ cosφi sinθi
Ci = ρ sinφi,

This activity differs in the fact that the equations are in spherical from,

Project 2: GPS SATELLITE POSITIONING

we have four A's,B's,C's as well, 4 satellites in the upper hemisphere, and wanting to calculate the r's (residuals) and d travel time with initial set of t x = 0, y = 0,z = 6370,d = 0.0001, and conditions: ρ = 26570 km is fixed, while 0 ≤ φi ≤ π/2 and 0 ≤ θi ≤ 2π for i = 1,...,4 are chosen arbitrarily.

We will define an error magnification factor specially tailored to the situation. The atomic clocks aboard the satellites are correct up to about 10 nanoseconds, or 10−8 second. Therefore, it is important to study the effect of changes in the transmission time of this magnitude. Let the backward, or input error be the input change in meters. At the speed of light, Δti = 10−8 second corresponds to 10−8c ≈ 3 meters. Let the forward, or output error be the change in position ||(Δx,Δy,Δz)||∞, caused by such a change in ti , also in meters. Then we can define the dimensionless

error magnification factor = ||(Δx,Δy,Δz)||∞ / c||(Δt1,...,Δtm)||∞

and the condition number of the problem to be the maximum error magnification factor for all small Δti (say, 10−8 or less).

Change each ti defined in the foregoing by Δti = +10−8 or −10−8, not all the same. Denote the new solution of the equations (4.37) by $(\bar{x},\bar{y},\bar{z},\bar{d})$, and compute the difference in position ||(Δx,Δy,Δz)||∞ and the error magnification factor. Try different variations of the Δti 's. What is the maximum position error found, in meters? Estimate the condition number of the problem, on the basis of the error magnification factors you have computed.

The approach:

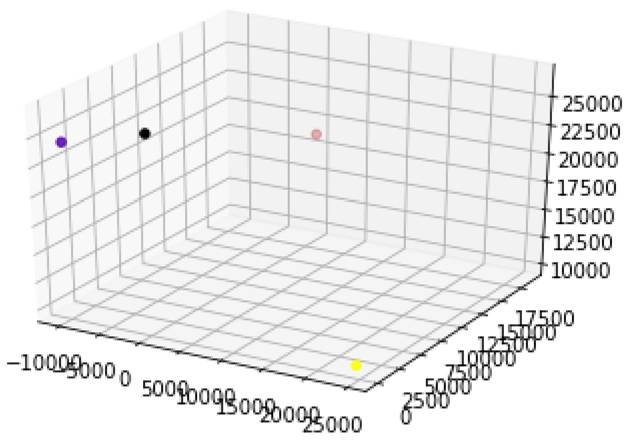
First in this activity we must consider the spherical coordinates and the error.,
We follow the same approach of the second activity; initializing the satellites, for their phi and theta values, from i to 4. Calculating the determinants of A, B, and C w.r.t x, y, z.
then we construct a function TimeEMF with parameters delta_t, phi, and theta
that returns the values of errors and Error Magnification Factor. TimeEMF(10**-8, np.pi/8, np.pi/2)

Maximum error and the Error magnification factor is:
(0.0014547869604939478, 5.475299060948242)

The max error is around 5.5 which is considered high, we think that adding more satellites will reduce the error, as we know that 4 ones are the minimum.

we also get the values of x, y and z: x_sol_neg, y_sol_neg , z_sol_neg, dis_neg
(327.0763212698555, -277.11016473634334, 15134.450670495777, 8770.520666773837)
As a final step we plot the satellites:

Project 2: GPS SATELLITE POSITIONING



Activity 5

Case 1: When the satellites are somewhat clustered.

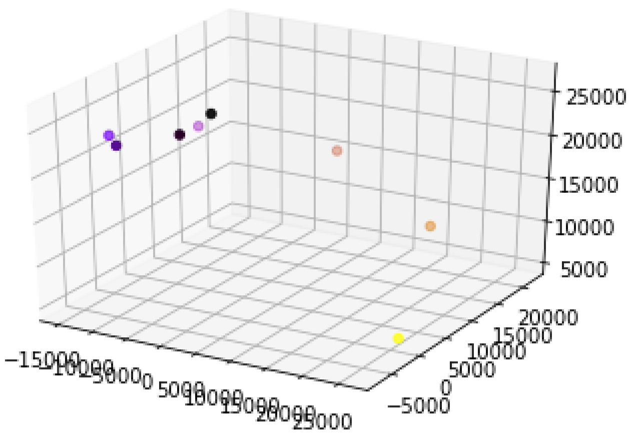
Delta 0:   Max error 1.000444171950221e-11, and EMF is inf
Delta 10^-8: Max error 0.2149067650352663, and EMF is 71.68518063095047

Case 2: When the satellites are tightly clustered.
Delta 0: Max error 4.089997673872858e-09, and EMF is inf
Delta 10^8: Max error 7.527553158578485, and EMF is 2510.921458397224

Activity 6

We tried using the quadratic function for some but we were getting some errors.

Maximum error and EMF
(6382.163630643575, 212.8860636862177)



Project 2: GPS SATELLITE POSITIONING

Figure: 8 satellites.

References

1.Solve Linear Equations using Determinants method:
https://byjus.com/jee/system-of-linear-equations-using-determinants/

2. Multivariate Newton's method:
https://drlvk.github.io/nm/section-multivariate-newton.html

3. Goerge Mason: https://mason.gmu.edu/~acurran3/RealityCheck4/RC_4.html

4. https://www.codespeedy.com/calculate-derivative-functions-in-python/

5.
https://stackoverflow.com/questions/61116868/python-numpy-linalg-linalgerror-last-2-dime
nsions-of-the-array-must-be-square

6. https://stackoverflow.com/questions/49553006/compute-the-jacobian-matrix-in-python