CPLS 5100 – Project 3

Analyzing the Impact of GPS Satellite Constellations on the Newton-Raphson Algorithm Solving for User Position

John Mick

Abstract

In this research paper we will examine how changes in Global Positioning System (GPS) satellite constellations impact the number of iterations the Newton-Raphson method requires to converge to an accurate user position and clock bias. We have been asked to consider a scenario using four satellites to resolve our user position and clock bias, where the four satellites are located at the four vertices of a regular tetrahedron. By following the constraints provided by our assignment instruction, and defined later, we have found that the Newton-Raphson method converges extremely reliably and always in exactly three iterations – irregardless of the satellite constellations.

---

1. Introduction

We are interested in evaluating how varying the satellite constellation impacts the number of iterations an applied Newton-Raphson algorithm takes to converge on a user's position and clock bias as well as observe any reduction in accuracy. We anticipate that the algorithm will require more iterations, or begin to introduce errors, as the satellites are located in narrower sectors of the sky and require less iterations providing less errors when the satellites are positioned uniformly around the user's GPS receiver.

---

2. Formulation

<u>**Constants:**</u>

C: Speed of Light
299792.458 km

B: Clock Bias
1e-7 seconds

DT: Distance Tolerance
10 meters

CT: Clock Bias Tolerance
30 nanoseconds

X: Known User Position

90 degrees latitude, 0 degrees longitude, 6,700 kilometer orbital radius

XO: Initial User Position Guess

-110 kilometers from known user X
+190 kilometers from known user Y
-1000 kilometers from known user orbital radius
0 second clock bias

Satellite Positions:

We define our first satellite in the tetrahedron to be a geostationary satellite which resides at 90 degrees latitude and 0 degrees longitude. The next three satellites will have varying latitude positions with static longitudinal values at 0, 120, and 240 degrees. All of the satellites will maintain a 26,600 kilometer orbital radius from Earth.

Satellite Pseudo-range Equation:

Satellite X, Y, Z, Clock Bias Referred to as:	satX	satY	satZ	satT
User X, Y, Z, Clock Bias Referred to as:	x	y	z	t

f = (satX – x)^2 + (satY – y)^2 + (satZ – z)^2 – C^2 * (satT + t)^2

Setting Initial Satellite Clock Bias Equation:

X and satX in this equation represents a 1x3 positional vector
The X Vector contains the the known user's positional X,Y, Z
The satX Vector contains the known satellite's positional X, Y, Z
Norm function returns a scalar square root sum of the squares from the resulting vector subtraction

f = norm(X – satX) / C - B

Jacobian Matrix Partial Derivatives

All X, Y, Z Values are derived from the following, replacing ***'A'*** *with X, Y, or Z:*

J (satelliteOne, ***A****)* = - 2 * (satelliteOne**A** – userPositionGuess**A**)
J (satelliteTwo, ***A****)* = -2 * (satelliteTwo**A** – userPositionGuess**A**)
J (satelliteThree, ***A****)* = -2 * (satelliteThree**A** – userPositionGuess**A**)
J (satelliteFour, ***A****)* = -2 * (satelliteFour**A** – userPositionGuess**A**)

Time values are derived from the following:

J (satelliteOne, ***T****)* = -2 * C^2 * (satelliteOne**ClockBias** – user**ClockBiasGuess**)
J (satelliteTwo, ***T****)* = -2 * C^2 * (satelliteTwo**ClockBias** – user**ClockBiasGuess**)
J (satelliteThree, ***T****)* = -2 * C^2 * (satelliteThree**ClockBias** – user**ClockBiasGuess**)
J (satelliteFour, ***T****)* = -2 * C^2 * (satelliteFour**ClockBias** – user**ClockBiasGuess**)

Latitude and Longitude Conversation to XYZ Equations:

x = orbitalradius * cos(latitude) * cos(longitude)
y = orbitalradius * cos(latitude) * sin(longitude)
z = orbitalradius * sin(latitude)

3. Methods

To simulate a system that solves for the user's position and clock bias we begin by applying the satellite clock bias equation for all four satellites – giving us a uniform clock bias on all four satellites. Next we evaluate the satellite pseudo-range equation for all four satellites with our initial user position guess and populate the results into an x,y,z,t vector. With this resultant vector, we find the difference between it's positional and time values from our actual known position then find the norm of the difference vectors. With these two scalars we are able to check against our defined distance and clock bias tolerances. If the first guess does not meet our tolerance requirement then we enter an iterative application of the Newton-Raphson method.

The iterative method begins by calculating the 4x4 Jacobian Matrix providing us with partial derivative positional x, y, z values and clock bias values for each satellite as it relates to our guessed user position and clock bias. We then find the inverse of our Jacobian Matrix and multiply it by the vector returned by the previously mentioned satellite pseudo-range equations (the system of equations evaluated at our guess). Finally we subtract this new vector from our current guess and re-evaluate our tolerances. If the tolerances are still not met, then the iterative method repeats itself; otherwise we have found an acceptable user position and clock bias and we step out of our Newton-Raphson algorithm.

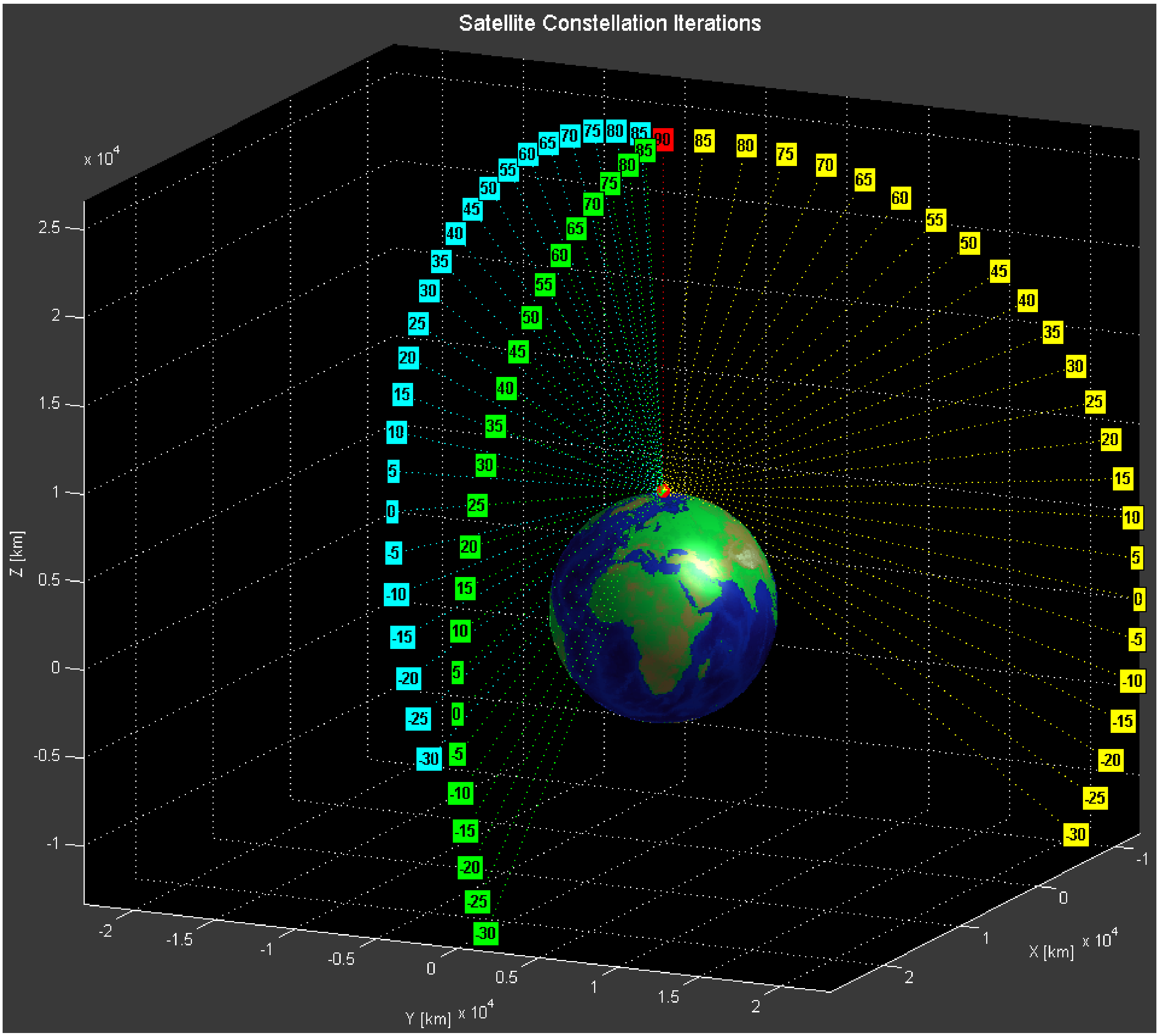
---

4. Integrity

The integrity of this model is subject to several assumptions and simplifications that make it difficult for us to conclude that a real GPS system would behave in the way we find. By assuming that our satellite constellation always forms a tetrahedron, with a one satellite directly overhead of our user position, we create an unrealistically ideal scenario which would not likely occur frequently (if ever) in reality. By setting our satellite's orbital radius to a constant value we have impacted the realism of our simulation as no satellite currently can truly maintains a perfectly constant orbital radius over the Earth. Our X, Y, Z coordinate system has been meant to mimic the ECEF coordinate system, but our conversation from latitude and longitude to XYZ does not take the elliptical shape of the Earth into consideration. We also do not consider which satellites would be visible to a user from Earth, allowing for satellite's on sides of the Earth opposite of the userto be used in the solution. There are also many other factors which could impact a user's ability to receive accurate pseudo-range measurements from a satellite; such as the Earth's local ionospheric and tropospheric activity, signal multipath, and a variety of local interferences which could corrupt satellite communication.

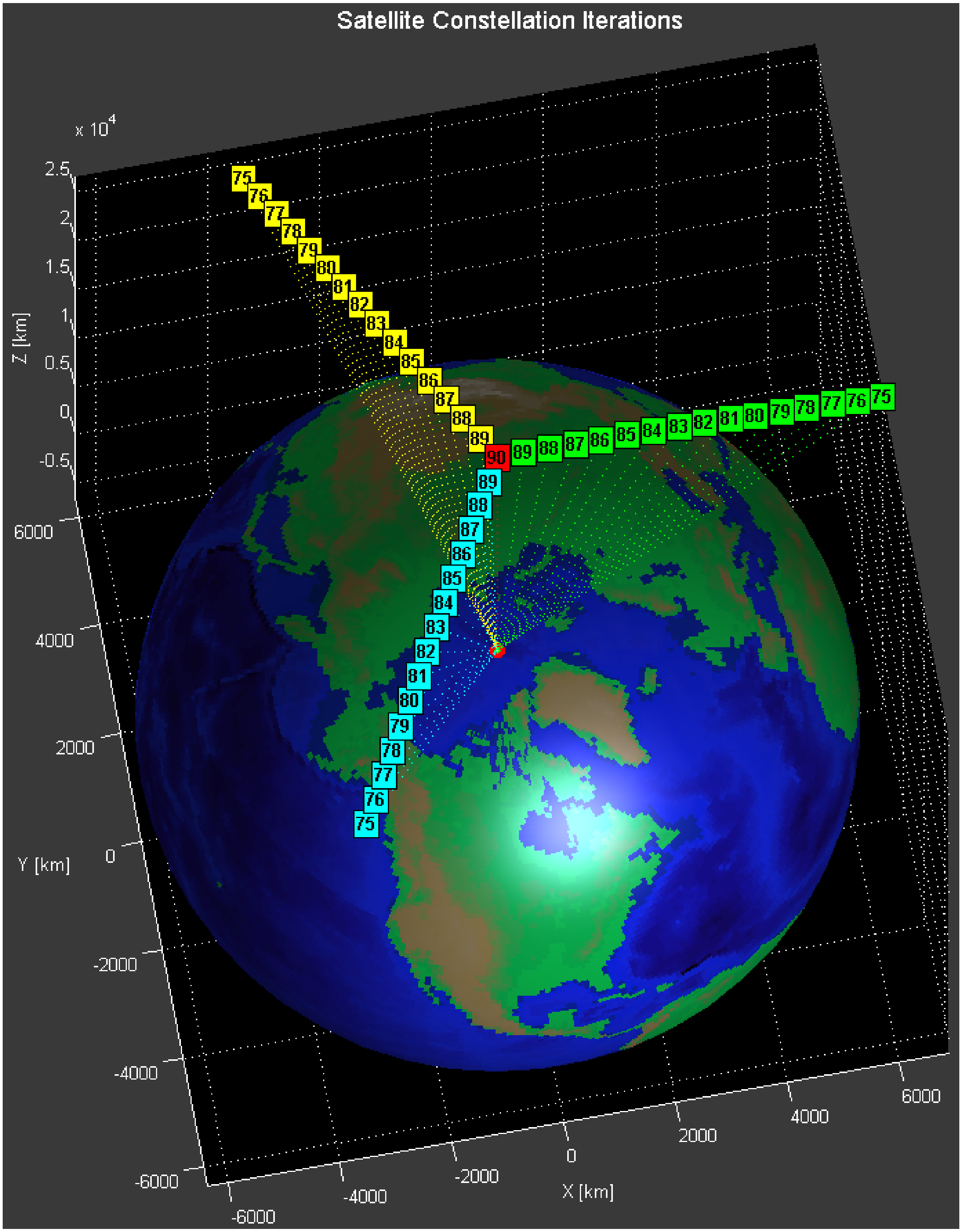
5. Computational Experiments (Model runs)

We begin by modeling the satellite constellation's in a tetrahedron iteratively incrementing our latitude values for 3 of our satellites by 5 degrees at a time. For each iteration we apply our numerical method and observe how many iterations are required for Newton-Raphson to converge to our user position, located at the red sphere at the north pole. The plot shows all of the iterations evaluated, each unique color represents a unique satellite. The satellite's latitudinal value is also placed in each satellite marker.

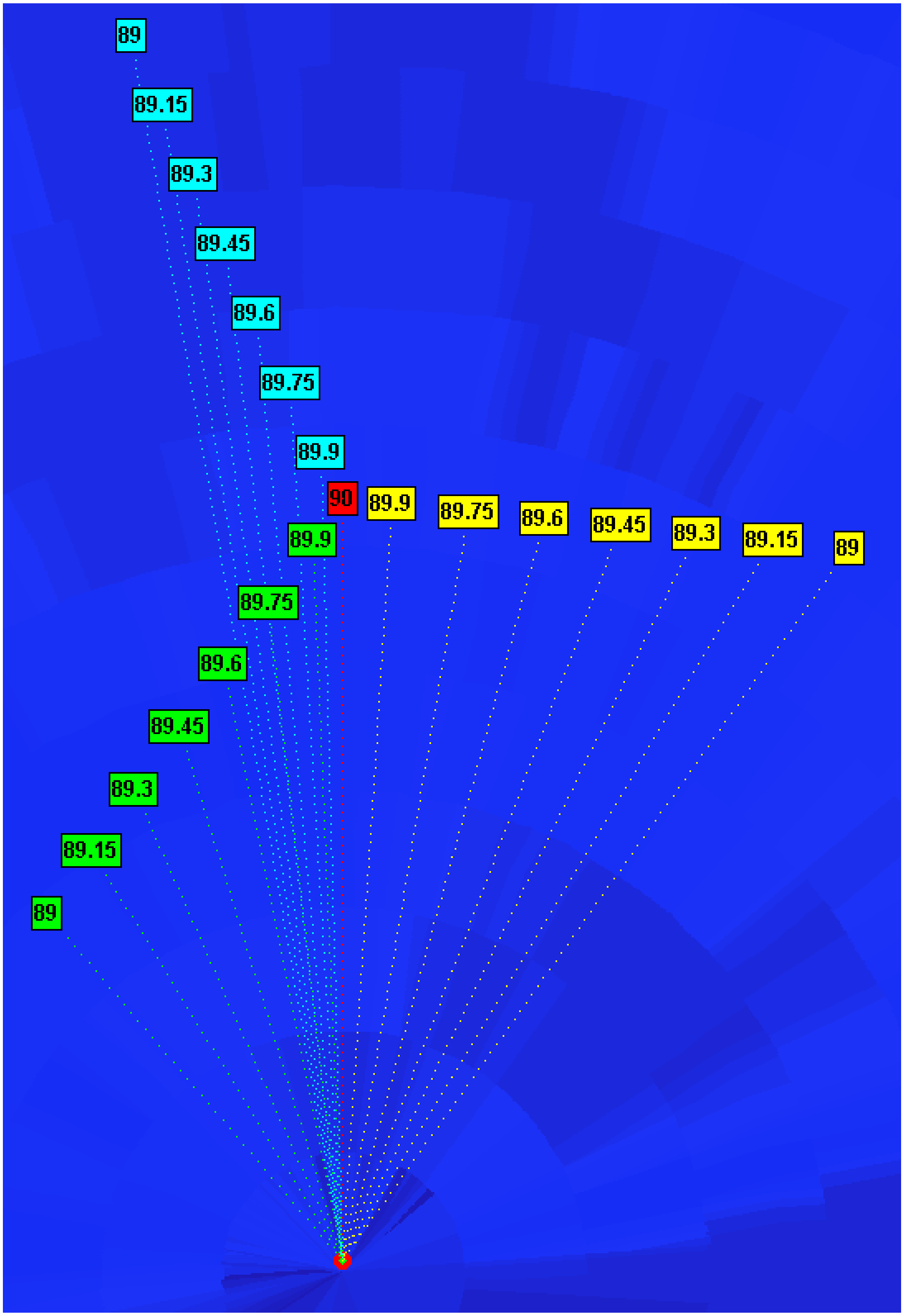


In every latitudinal shift of our three satellites, the Newton-Raphson method is able to accurately converge on our user position in exactly three iterations.

Repeating the simulation with a smaller iterative step and latitudinal range evaluated provides us with the following plot and no change in the Newton-Raphson iterations or accuracy.



Lastly we look at a case where all 3 satellites occupy nearly the same plane. In all iterations the Newton-Raphson method is able to converge accurately in 3 iterations.



We are seeking a user position of 0, 0, 6700 kilometers and a clock bias of 1e-7 seconds. Here we will look at the individual iterations of a simulation between 89 latitude and 90 latitude, iterating at .25 latitude degrees.

Latitude 89				
	X (km)	Y (km)	Z (km)	Clock Bias (s)
Guess 1:	-110	190	5700	0
Guess 2:	0	0	6693.449397	-6.48e-5
Guess 3:	0	0	6700.002126	1.21e-7
Latitude 89.25				
	X (km)	Y (km)	Z (km)	Clock Bias (s)
Guess 1:	-110	190	5700	0
Guess 2:	0	0	6693.449325	-6.48e-5
Guess 3:	0	0	6700.002126	1.21e-7
Latitude 89.50				
	X (km)	Y (km)	Z (km)	Clock Bias (s)
Guess 1:	-110	190	5700	0
Guess 2:	0	0	6693.449273	-6.48e-5
Guess 3:	0	0	6700.002126	1.21e-7
Latitude 89.75				
	X (km)	Y (km)	Z (km)	Clock Bias (s)
Guess 1:	-110	190	5700	0
Guess 2:	0	0	6693.449242	-6.48e-5
Guess 3:	0	0	6700.002126	1.21e-7

Examining data from our previous plots presents us with similar results. The first initial provided guess is evaluated to be incorrect, and our first iteration of Newton-Raphson converges immediately to our X, Y values, only needing one more additional run to converge with enough accuracy on the Z and one more run to acquire the clock bias. The fourth guess is not mentioned as it meets our tolerance levels.

6. Conclusions

Though we expected to see a degradation in the Newton-Raphson's ability to converge on an accurate user position and clock bias as the satellite constellation is positioned closer to the user, we actually observed that the change in satellite constellation had no impact at all. As long as we continue to constraint ourselves as described, the Newton-Raphson method is able to reliably converge on our desired value in exactly three iterations.

Additional research could be explored by loosening the constraints on the satellite constellations, perhaps considering cases where there is no satellite overhead from the user or examining the impact of satellite's approaching each other longitudinally. Requiring stricter distance and clock bias tolerances also may present different cases. One could also consider increasing the error on the initial guesses. Also, due to the previously observed integrity issues of this model, we believe that one could consider this model to be a first step in modeling real GPS behavior, but some additional equations and modifications to existing equations would be needed to continue on.