Final Project A

AERO423

Orbital Mechanics

Fall 23 Project

Benjamin Tollison

Tasks
Author

Friday 8th December, 2023

Contents

1 Project Constraints

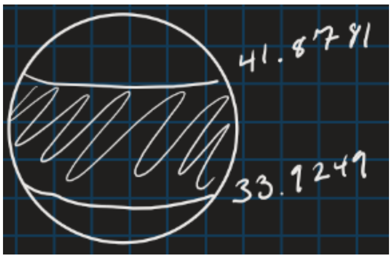
1. **UIN:** 428004920
2. **Orbit Altitude ($h_{\textbf{alt}}$):** 1100 km
3. **Minimum Grazing Angle ($\epsilon_{\min}$):** 12.5°
4. **Earth Radius (Re):** 6378.1365 km
5. **Locations:**
   (a) Chicago, IL: $41.8781°\mathrm{N}, 87.6298°\mathrm{W}$
   (b) Melbourne, Australia: $37.8136°\mathrm{S}, 144.9631°\mathrm{E}$
   (c) Cape Town, South Africa: $33.9249°\mathrm{S}, 18.4241°\mathrm{E}$
   (d) Mombasa, Kenya: $4.0435°\mathrm{S}, 39.6682°\mathrm{E}$

2 Initial Approximation

Starting from the minimum number of satellites equation:

$$N_{\text{sat,min}} = \frac{2\alpha}{1 - \cos \Lambda_{\min}}$$

and using the northernmost and southernmost latitudes to find the proportion of the Earth coverage that we want.



To create the $0 < \alpha < 1$:

$$\alpha = \frac{41.8781 + 33.9248}{180} = 0.42113$$

2.1 Finding $\Lambda_{\min}$

Starting from the definition of the Earth angle or half swath angle:

$$\Lambda = 90 - \epsilon - \eta$$

And η can be found with

$$\sin \eta = \frac{Re}{Re + h} \cos \epsilon$$

$$\eta = \arcsin\left(\frac{Re}{Re + h} \cos \epsilon\right)$$

$$\eta = 56.376°$$

Which then going back to the Λ equation

$$\therefore \Lambda = 21.124°$$

With all of the unknowns solved in the $N_{\text{sat,min}}$ equation produces

$$N_{\text{sat,min}} = 12.534 \to 13$$

2.2 Finding Number of planes

$$N_P = \frac{180}{2\Lambda + \omega_E \mathcal{TP}}$$

Where $\mathcal{TP} = \sqrt{\frac{a^3}{\mu}}$, $\omega_E = -0.2507$, and $a = Re + h$

$$N_P \to 12$$

Please be aware that the aforementioned approximation for the minimum number of satellites is specifically calculated under the condition of achieving 100% coverage without any intervals of downtime or gaps.

3 STK

With the newest update of STK 12 (12.7) the Pro version got the walker function removed and moved to Premium/Enterprise, which was the tool that I used previously. The work around that I found using the newer satellite collection tool based off of a parent satellite and sensor, a constellation of places, and a chain grouping. However, to check the gap times between the satellite and locations can be found by going to the constellation¿properties¿constraints¿logical restriction and changing both access positions to 'None Of'. This now only checks when there are now sensors availible and then will show that time value in the report.

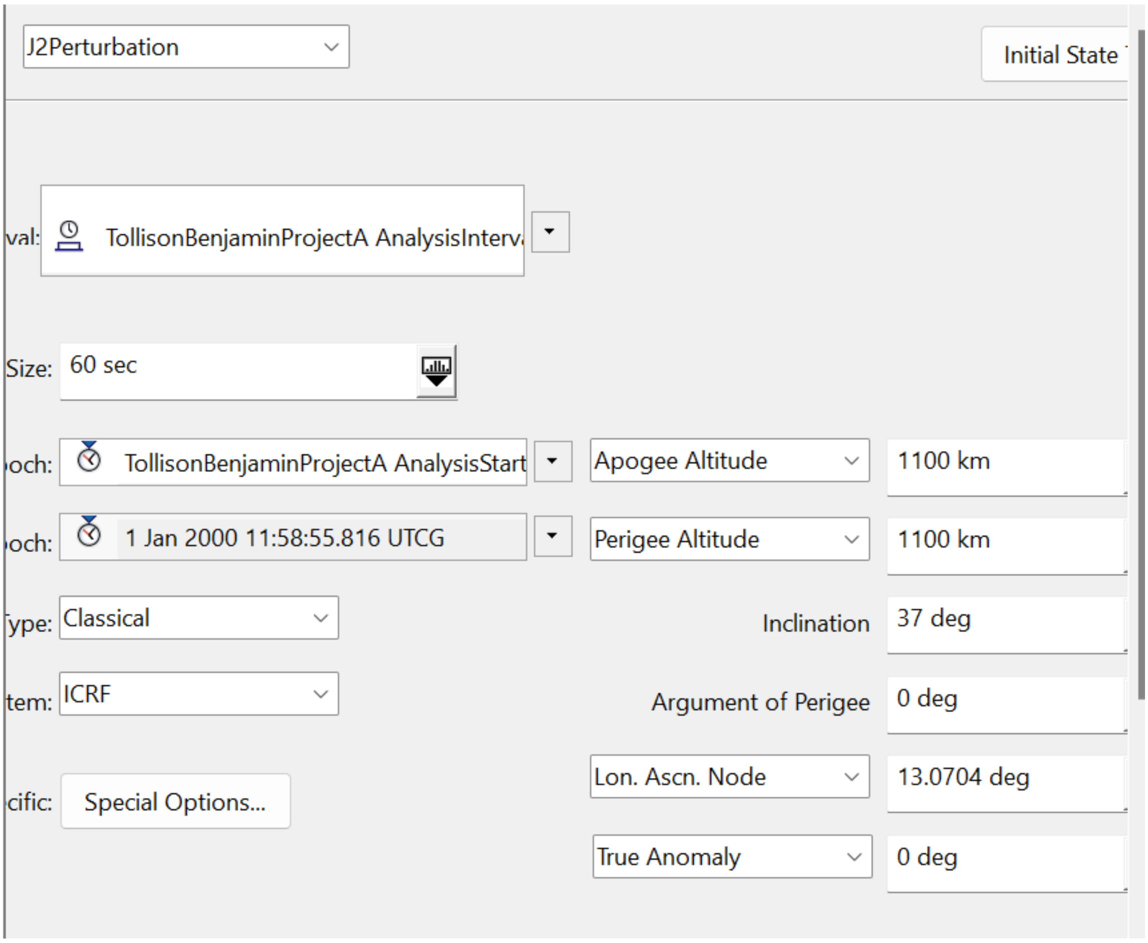
I also converted the gap times to seconds because that is what STK outputs to

$$900 < Gap_{avg} < 2700, Gap_{\max} = 6000$$

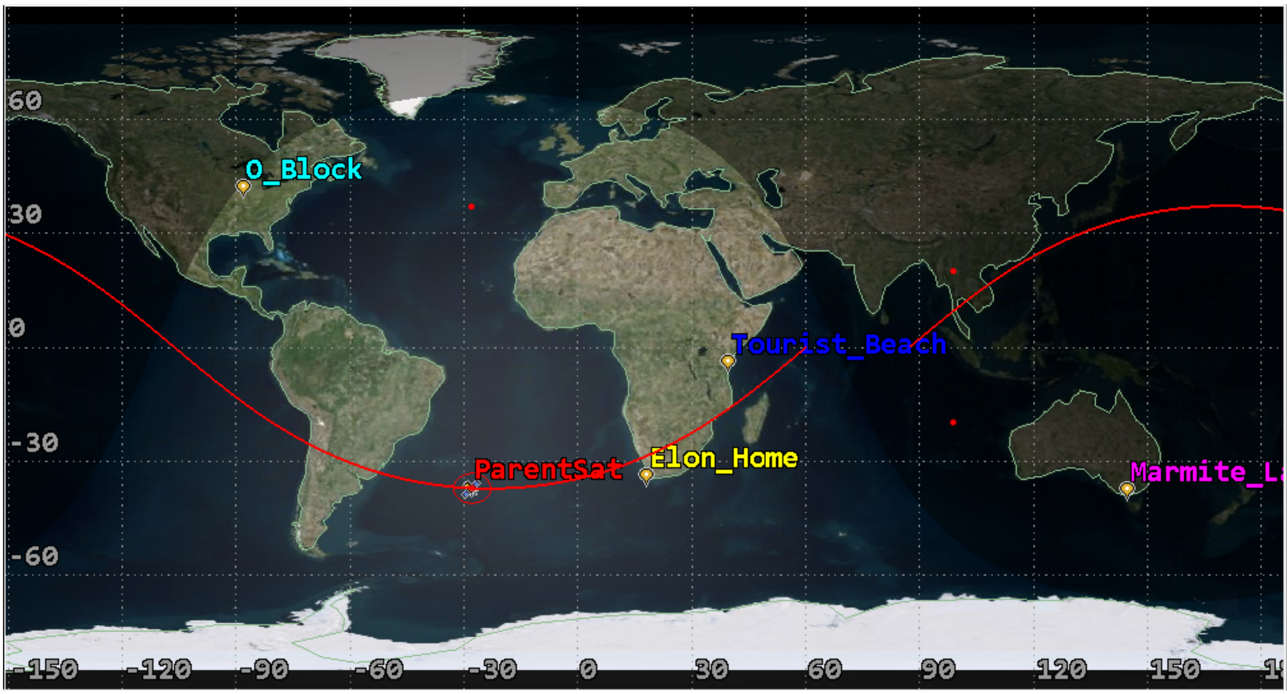
Here is a table of my guesses I used to narrow down to the final result of 6 satellites.

Number of Planes	Sat per plane	Gap_{avg} [s]	$Gap_{\max}$ [s]	Total Satellites
12	18	0	0	216
12	4	297.572	518.891	48
4	4	1122.903	1636.637	16
3	4	1131.256	5633.492	12
2	3	2363.21	16205.976	6
6	1	2352.093	4648.74	6
5	1	2866.598	7241.465	5
3	2	2345.584	10589.942	6

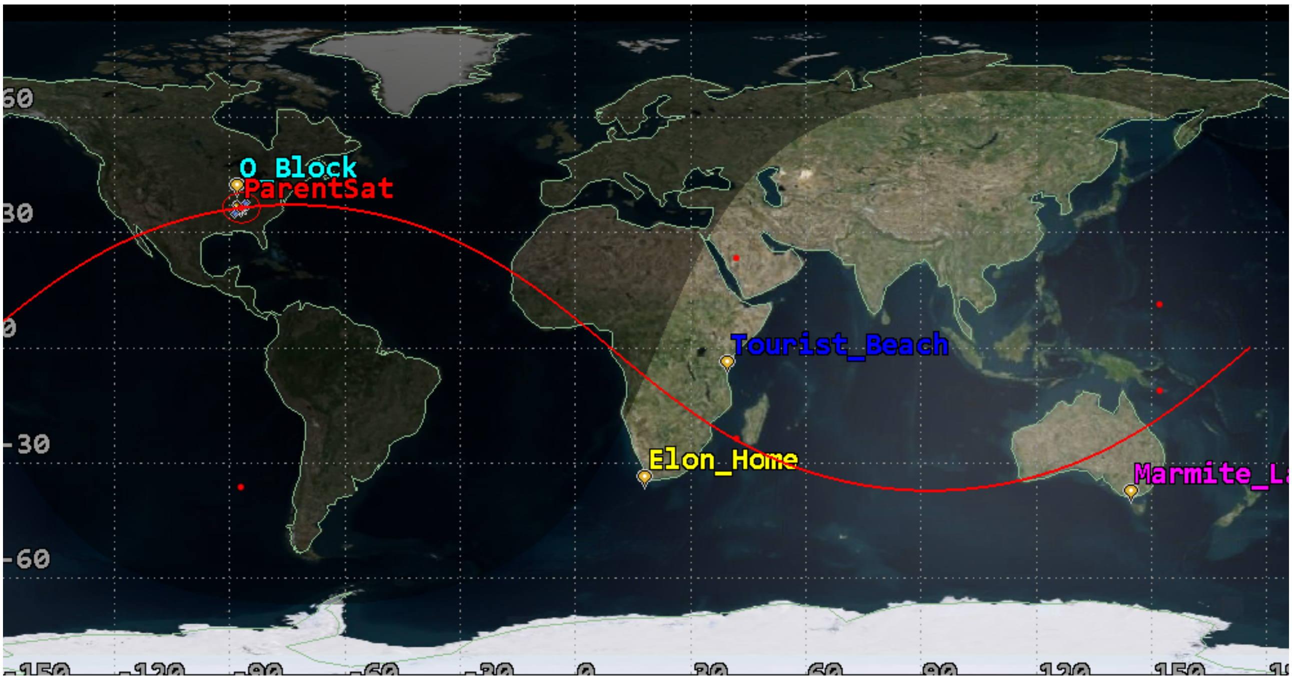
The peculiar thing that happens with the circular orbit, and an inclination of 37 degrees, is the the locations, almost naturally, fall in line with the orbit path such that to have the gap times is half of the predicted 100% coverage of the α. Below is the parent satellite that the constellation is based off of.



This is an example of how the Satellite's ground path naturally align for easier coverage.



Below is another path that covers all the orbits in 3 revolutions around Earth because of the J2 drift



3.1 Final Proposed Design

The ultimate configuration for the satellite constellation involves six planes, each accommodating a single satellite. These planes are strategically positioned with a 60-degree offset from one another. As a result of this arrangement, the average gap time is anticipated to approach 39 minutes, while the maximum gap time is estimated to be approximately 77.5 minutes. Despite the fact that the proposed design aligns with the upper bounds of the specified constraints, it remains within the defined limitations.

4 Code

Code

December 8, 2023

[ ]:
```python
### Benjamin Tollison ###
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import scipy
import sympy as sp
from IPython.display import Latex, Math, display
from sympy import (
    Eq,
    Function,
    Matrix,
    cos,
    cosh,
    exp,
    integrate,
    lambdify,
    pi,
    sin,
    sinh,
    symbols,
)
from decimal import Decimal
from sympy.solvers.pde import pdsolve
from sympy.solvers.solveset import linsolve
def displayEquations(LHS,RHS):
    left = sp.latex(LHS)
    right = sp.latex(RHS)
    display(Math(left + '=' + right))
    np.set_printoptions(suppress=True)
def displayVariable(variable:str,RHS):
    left = sp.latex(symbols(variable))
    right = sp.latex(RHS)
    display(Math(left + '=' + right))
def displayVariableWithUnits(variable:str,RHS,units):
    left = sp.latex(symbols(variable))
    right = sp.latex(RHS)
    latexUnit = sp.latex(symbols(units))
```

```
    display(Math(left + '=' + right + '\\;' +'\\left['+ latexUnit + '\\right]'))
def format_scientific(number:float):
    a = '%E' % number
    return a.split('E')[0].rstrip('0').rstrip('.') + 'E' + a.split('E')[1]
deg2rad = np.pi/180
rad2deg = 180/np.pi
```

[ ]:
```
x = 2
y = 0
h_alt = 100*x + 900 # km
epsilon_min = (2.5*(y+1) + 10)*deg2rad # radians
Re = pd.read_csv('AstroConstants.csv').to_dict()['Earth'][1]
mu = pd.read_csv('AstroConstants.csv').to_dict()['Earth'][0]
nadir = np.arcsin((Re/(Re+h_alt))*np.cos(epsilon_min))
displayVariableWithUnits('\\eta',nadir*rad2deg,'deg')
angular_half_swath = (np.pi/2)-epsilon_min-nadir
displayVariableWithUnits('\\Lambda',angular_half_swath*rad2deg,'deg')
alpha_proportion = (41.8781+33.9249)/180
displayVariable('\\alpha',alpha_proportion)
satellite_min = (2*alpha_proportion)/(1-np.cos(angular_half_swath))
displayVariable('N_{sat\\,min}',np.ceil(satellite_min))
```

$\eta = 56.375785193364 \; [deg]$

$\Lambda = 21.124214806636 \; [deg]$

$\alpha = 0.421127777777778$

$N_{sat,min} = 13.0$

[ ]:
```
time_period = (2*np.pi*np.sqrt((Re+h_alt)**3/mu))/(60) # minutes
Number_planes = np.ceil(180/(2*angular_half_swath*rad2deg - 0.2507*time_period))
displayVariable('N_P',Number_planes)
Number_sat_per_plane = np.ceil(2*np.pi/angular_half_swath)
displayVariable('N_{SatPerOrbit}',Number_sat_per_plane)
Total_min = Number_sat_per_plane*Number_planes
displayVariable('N_{minTotal}',Total_min)
print(Re+h_alt)
```

$N_P = 12.0$

$N_{SatPerOrbit} = 18.0$

$N_{minTotal} = 216.0$

7478.1365

[ ]:
```
displayVariableWithUnits('Gap_{avg}',15*60,'sec')
displayVariableWithUnits('Gap_{avg}',45*60,'sec')
displayVariableWithUnits('Gap_{max}',100*60,'sec')
```

$Gap_{avg} = 900 \; [sec]$

$Gap_{avg} = 2700 \ [sec]$

$Gap_{max} = 6000 \ [sec]$