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))
```

$\eta = 56.375785193364 \; \left[deg\right]$

$\Lambda = 21.124214806636 \; \left[deg\right]$

$\alpha = 0.421127777777778$

$N_{sat,min} = 13.0$

[ ]:
```
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')
```

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

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

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