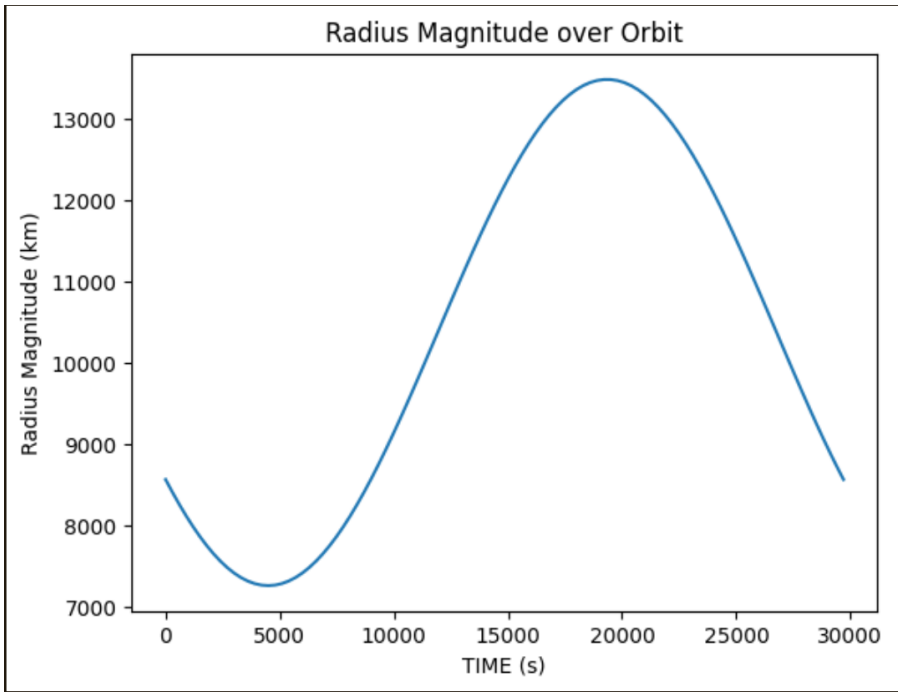
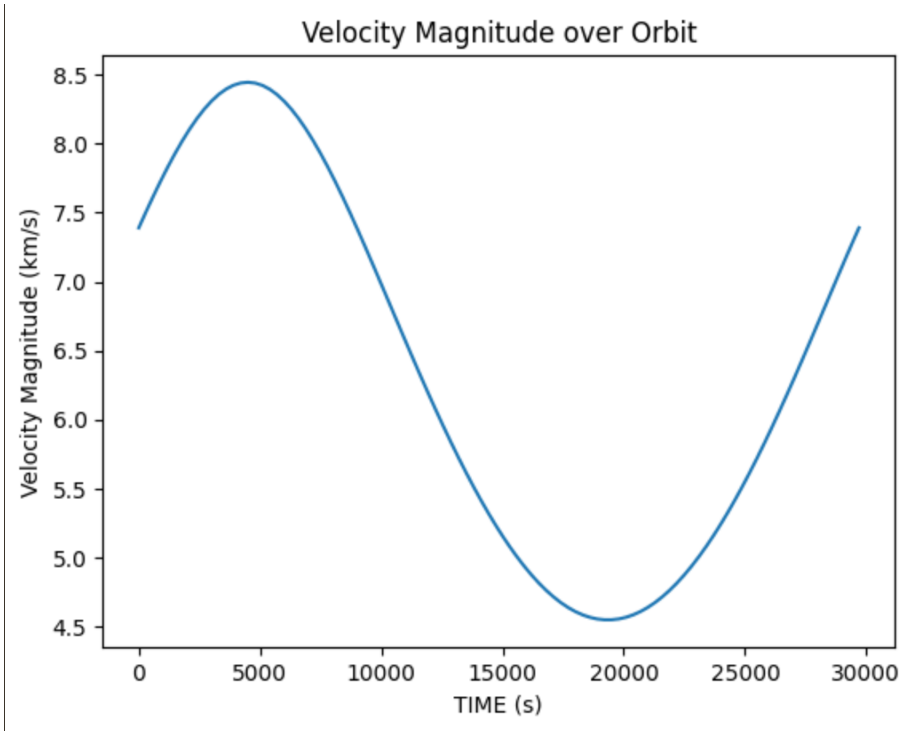
Matias Calderon
HW 0

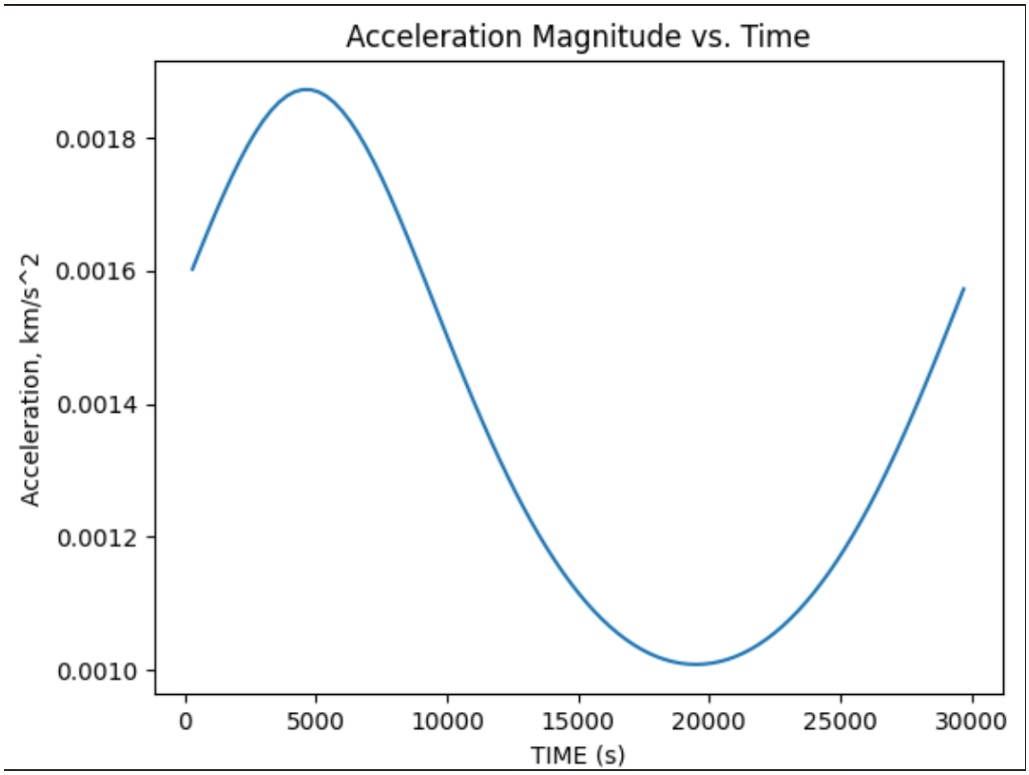
1. **Plots of radius vs. Time**



2. **Plot of velocity vs. time**



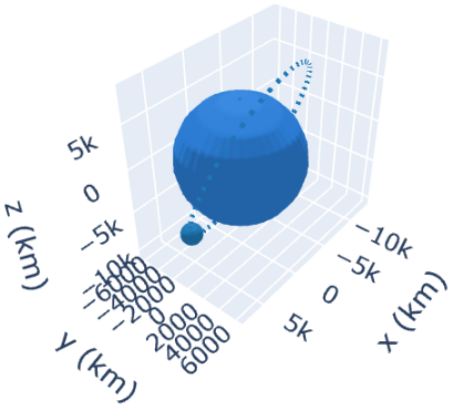
3. **Plot of acceleration vs. time**



4. This behavior for the values of the radius, velocity, and acceleration make sense given the equation of motion. If the radius graph is first observed, it can be seen that when the radius of orbit is smaller (and thus the force due to gravity larger as it is divided by radius squared) the corresponding acceleration value is also higher. F = ma (larger f, same mass = larger a). The converse is also true for areas of the radius graph where the radius is larger. The force due to gravity is smaller, and thus the resulting acceleration is also smaller.
5.

```
FINAL RADIUS VALUES: [ 5801.39136362 -1261.23267091 -6180.37558875]
FINAL VELOCITY VALUES: [3.89753799 2.76981155 5.63488805]
```

6. 3-D Orbit plot



7. **Python Code:**

```
from astropy import units as u
from astropy.coordinates import CartesianRepresentation as cr
from poliastro.plotting import OrbitPlotter3D as op3
import datetime as d
from poliastro.bodies import Earth, Sun
from poliastro.twobody import Orbit
from poliastro.constants import J2000
from pandas import read_csv, DataFrame
from matplotlib import pyplot as plt
plt.ion()
import plotly.express as px
import numpy as np

#READING IN .ENV FILE------------------------------>
def readEnvironment():
    radi = []
    vel = []
    with open('orbits.env', 'r+') as orbitfile:
        A = read_csv(orbitfile)
        orbits = A.values
        for orbits in orbits:
            radius = orbits[:3]
            velocity = orbits[3:]
            radi.append(radius)
            vel.append(velocity)
    return radi,vel
```

```
#PROPOGATES AN ORBIT BY A GIVEN TIME BELOW------CAN REWRITE FOR USER INPUT

def orbitTest(radius, velocity):
    radius = radius << u.km
    velocity = velocity << u.km / u.s
    #INITIALIZING ORBIT OBJECT HERE [can also change tested orbit here]-------->
    orb = Orbit.from_vectors(Earth, radius[2], velocity[2])

    #HERE PROPOPGATED BASED ON TIME ADDITION
    orb2 = orb.propagate(30000 << u.s)

    #SAMPLING RADIUS SET FROM PROPOGATED ORBIT
    PATH = orb2.sample()
    finalrad = PATH[-1]
    return finalrad, PATH, orb, orb2

#DRIVER--------------------------------------->
radius,velocity = readEnvironment()
finalrad, PATH, orb, orb2 = orbitTest(radius, velocity)
orb3 = orb2

#WRITING=============================================================================================
#TAKING SAMPLE OF R, V VALUES-------------------------->
OrbEphem = orb3
V = OrbEphem.sample()
vel = V.differentials
```

```
#DATA COMPOSITION FOR RADIUS AND VELOCITY-------------------------->
RADS = DataFrame(V, columns = ['x,y,z'], index = range(0,30000,300))
vel2 = DataFrame(vel['s'], columns = ['dx,dy,dz'], index = range(0,30000,300))


#WRITING/READING TO/FROM FILES================================>
with open('orbit_radius.csv', 'w+') as orbitrads:
    orbitrads.write('x,y,z\n')
    for row in RADS['x,y,z']:
        row = str(row).replace('(','').replace(')','').strip(f' km')+'\n'

        orbitrads.write(row)
with open('orbit_velocity.csv', 'w+') as orbitvels:
    orbitvels.write('dx,dy,dz\n')
    for row in vel2['dx,dy,dz']:
        row = str(row).replace('(','').replace(')','').strip(f' km / s')+'\n'

        orbitvels.write(row)
with open('orbit_radius.csv', 'r') as radfile:
    r = read_csv(radfile)
    radius = r.values
    radius_magnitude = []
    for row in radius:
        row = np.array(row)
        row = np.linalg.norm(row)
        radius_magnitude.append(row)
with open('orbit_velocity.csv', 'r') as velfile:
    v = read_csv(velfile)
    velocity = v.values
    velocity_magnitude = []
    for row in velocity:
        row = np.array(row)
        row = np.linalg.norm(row)
        velocity_magnitude.append(row)
#=============================================================================================
```

```
#2-D CHARTS

#RADIUS---------------------------------->
plt.figure(1)
plt.title('Radius Magnitude over Orbit')
plt.plot(range(0,30000,300),radius_magnitude)
plt.xlabel('TIME (s)')
plt.ylabel('Radius Magnitude (km)')
#VELOCITY-------------------------------->
plt.figure(2)
plt.title('Velocity Magnitude over Orbit')
plt.plot(range(0,30000, 300), velocity_magnitude)
plt.xlabel('TIME (s)')
plt.ylabel('Velocity Magnitude (km/s)')
plt.show()
#EXTRAPOLATING ACCELERATION VALUES------------>
accels = []
for i in range(1,len(velocity)):
    vnew = np.array(velocity[i])
    vold = np.array(velocity[i-1])
    delv = vnew - vold
    accel = delv / 300
    accel = np.linalg.norm(accel)
    accels.append(accel)
#PLOTTING ACCELERATIONS
plt.figure(3)
plt.title('Acceleration Magnitude vs. Time')
plt.plot(range(300,30000,300),accels)
plt.xlabel('TIME (s)')
plt.ylabel('Acceleration, km/s^2')
plt.show()
#3-D CHARTS
frame = op3()
frame.plot(orb3, label = Earth)

frame.show()
```