Rocket Launch and Orbit Simulation using Python

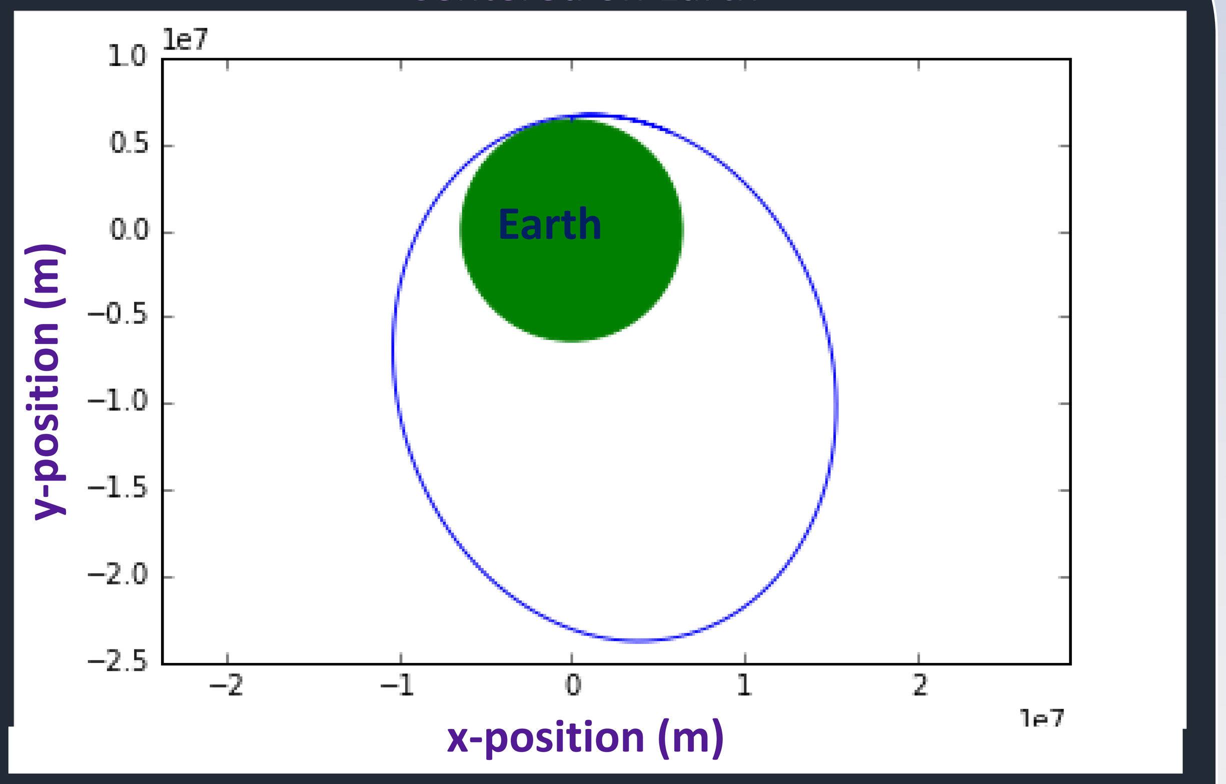
Matthew Owens, Prof. Nicholas Truncale

PHYS 386H Rocket Science Honors Tutorial, The University of Scranton

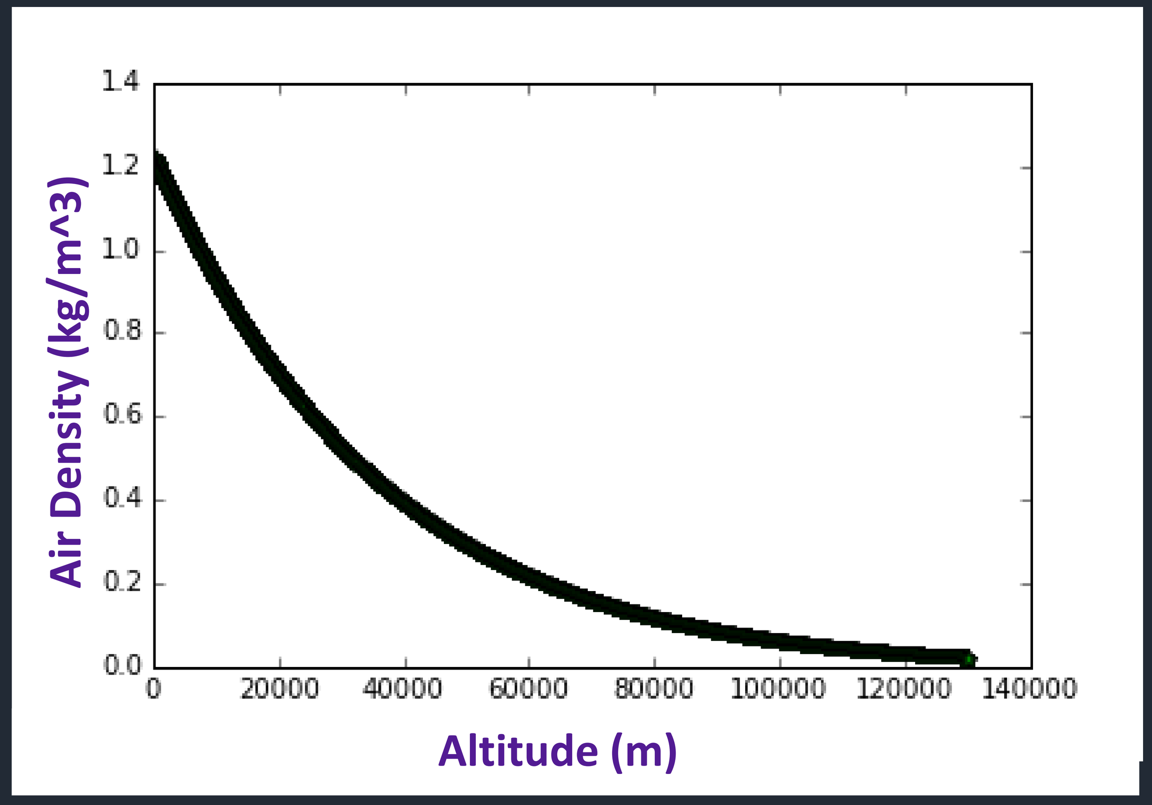
Introduction

The purpose of this project is a to create a two-dimensional simulation of rocket launches and orbits that makes it possible to both replicate real-life launches and see the results of changes in various parameters of the launch vehicle and flight path. We started by creating a simulation of Newton's Cannonball thought experiment and proceeded by adding other forces acting on the spacecraft individually (air resistance, thrust, etc.). The simulation calculates the rocket's two-dimensional position over a set number of discrete time steps and plots the orbital path of travel in the (x,y) plane.

Sample Launch and Elliptical Orbital Path Plot
Centered on Earth



Air Density Model Used in Simulation



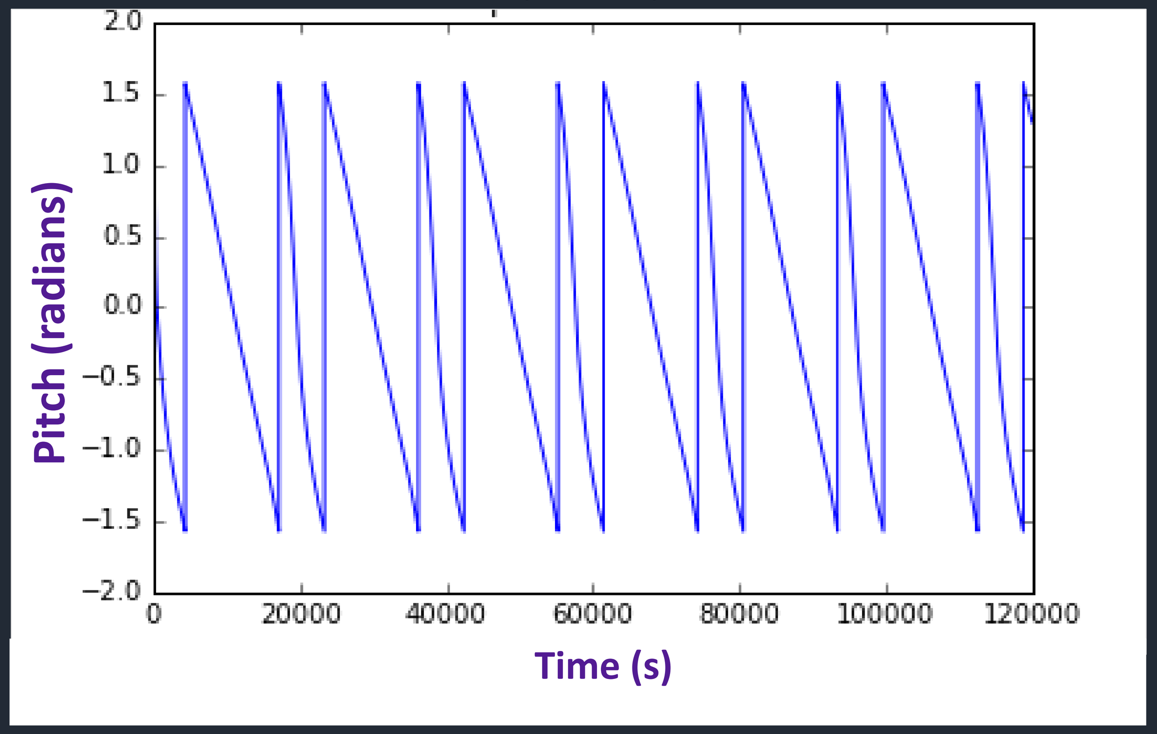
Spacecraft Configuration:

```
{
'name' : 'atlas',
  'boosterMass': 120000, #kg
  'capsuleMass':1360, #kg
  'burnTimes':[134, 570], #two-stage rocket burn times (seconds)
  'thrustValues':[2600000,2600000], #two-stage rocket thrust values (N)
  'gravityTurnTime':80, #Pitch-program time (seconds)
  'area':7.0685834705770345, #cross-sectional area used for air resistance
  'fuelLapseRate':27 #kg of fuel burned/second
}
```

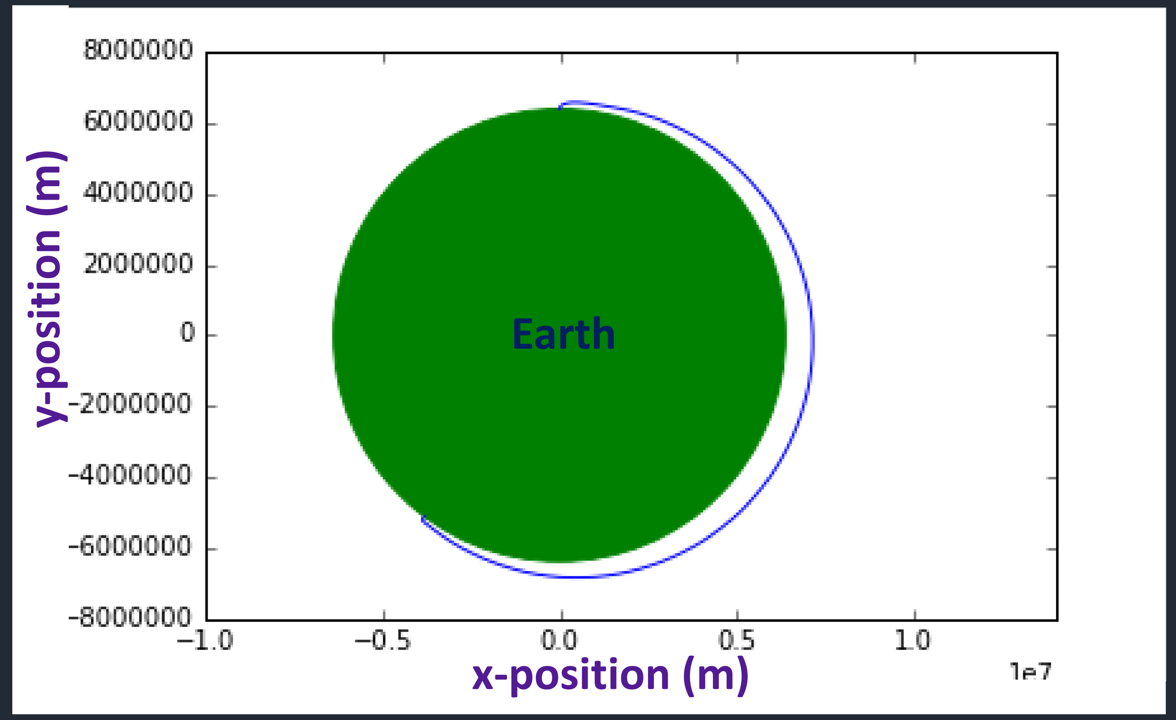
Spacecraft Design

The Spacecraft class reads a configuration file containing parameters for the rocket launch including spacecraft mass, cross-sectional area, duration and thrust for each burn, fuel lapse rate, and time to execute a gravity turn. This allows users to change the flight profile and experiment with different values by loading a different file. The example configuration file and its resulting launch plot are shown above.

Pitch Attitude Dimension
for Multiple Elliptical Orbits

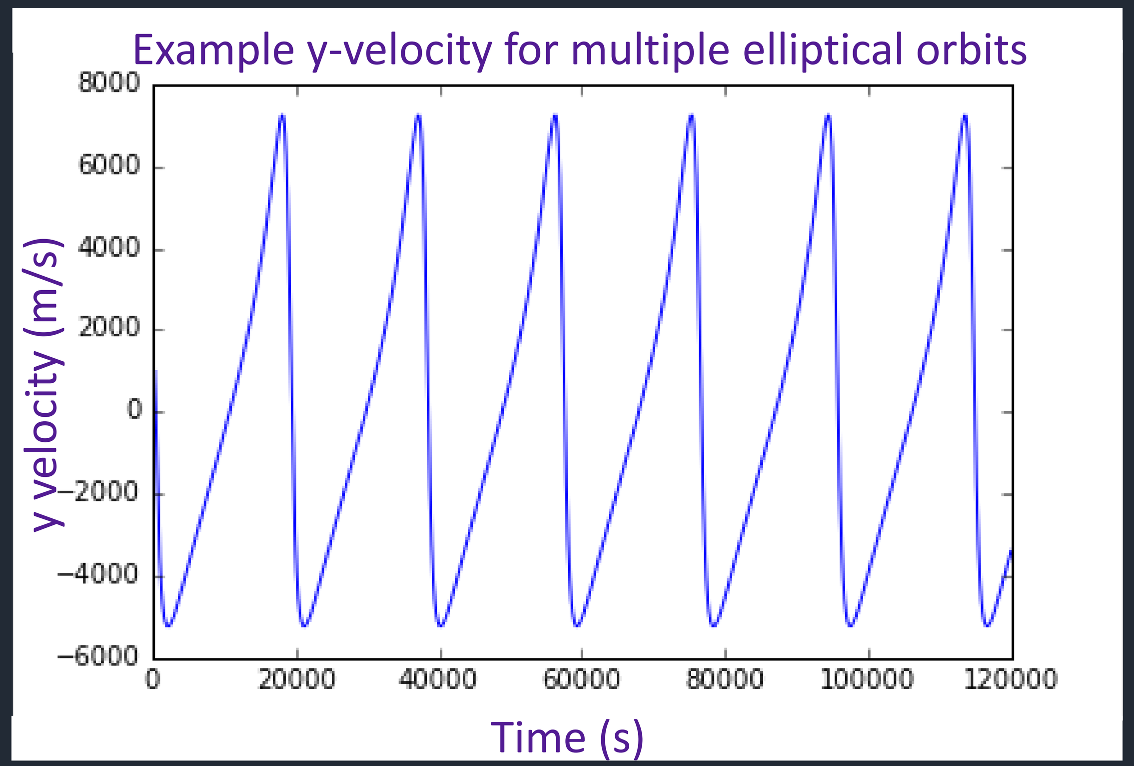


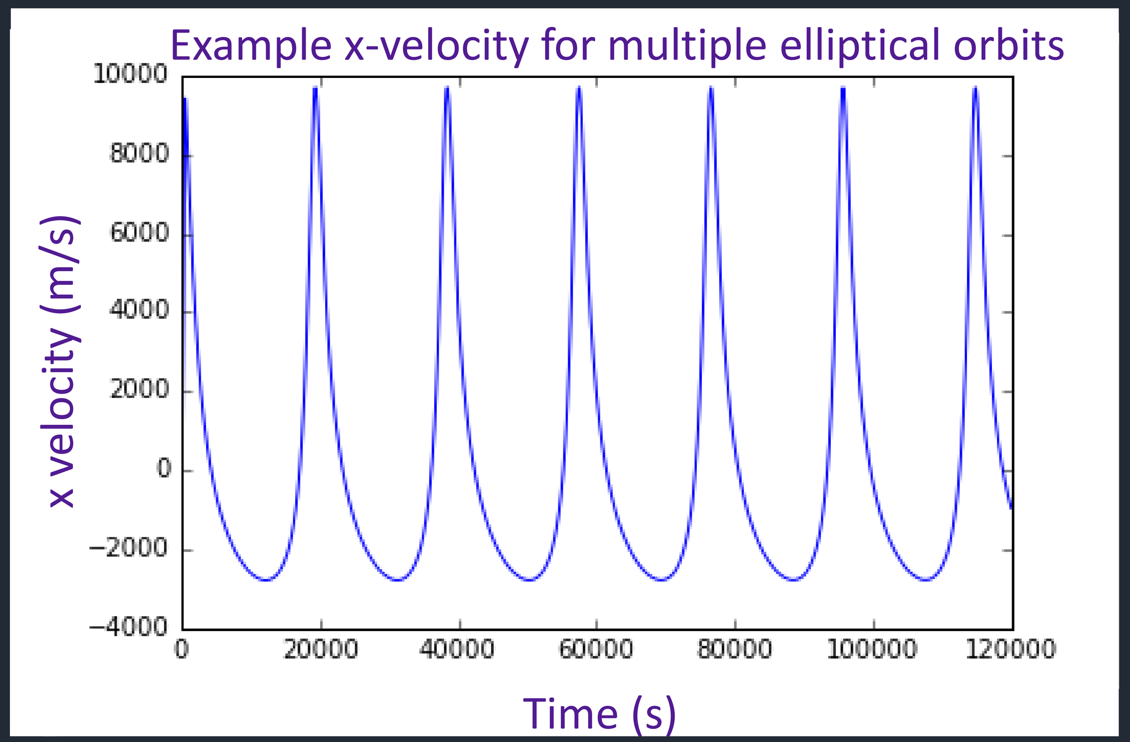
Sample Launch and Re-entry Orbital Path Plot
Centered on Earth



Forces on Spacecraft

Each force acting on the spacecraft is calculated in a separate function providing the main acceleration function "getAcceleration" with force values. This function returns the x and y acceleration due to that force. Currently the system models gravity, thrust, and air resistance. Currently, the acceleration due to gravity is only calculated in relation to the Earth. We intend to expand the simulation to include spacecraft missions near the moon, whose gravitational effects will be simulated.

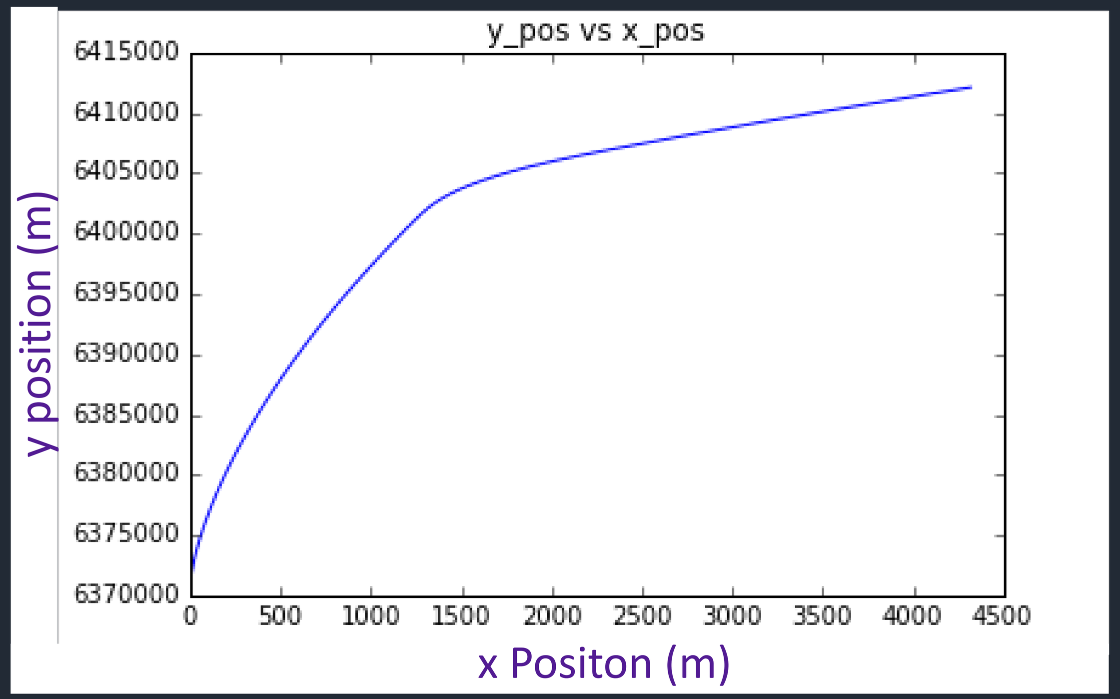




Flight Recorder

The class FlightRecorder keeps track of the spacecraft ephemeris (position, velocity) and attitude (one pitch dimension currently), radial distance from the center of the Earth, and each acceleration acting on the spacecraft. This information is saved to a csv file for later viewing and analysis. This class is also responsible for generating the graphs shown on this poster using PyPlot.

Gravity Turn – "Pitch Program" - Example



Gravity Turn – Pitch Program

The "gravityTurnTime" value in the spacecraft configuration file specifies how many seconds after launch to begin the gravity turn "Pitch Program". At that point in the flight, the simulation forces the pitch angle to change to nearly horizontal over a period of ~10 seconds. This changes the direction of the thrust vector, resulting in an increase in horizontal velocity. This was how we phased both the launch phase and orbital phases of the simulation.

Future Simulation Additions

This project is still in its early stages. The biggest goal moving forward is to make as many of the parameters of flight customizable as possible. Currently, burns are assumed to be continuous and it is not possible to execute more than one burn phase after the initial launch (two-stage only) and the only configurable option for the gravity turn is the time after launch at which is starts. Both of these situations would be more configurable by including a list of commands in the spacecraft configuration which dictate actions such as engine ignition, engine cutoff, stage separation, and pitch changes and the time at which they occur. This would give users more control over the flight and give more possibilities for mission simulations.