Project CubeSat

Telemetry

CubeSat Telemetry Simulation Report

Task 2:

During a CubeSat flight, the avionics computer records telemetry parameters for the payload (CubeSat) until it abruptly stops. You are required to simulate such a scenario in which:

- The Payload launches with an initial thrust of 50 kN.
- The thrust lasts for a total of 10 seconds.
- Telemetry data transfer ceases at t = 16 s.
- The interval between each recorded value is 1 s.
- Drag can be neglected.

Tasks:

1. Plot the payload's *Altitude* and *Velocity* vs *Time* on two separate subplots.
2. Label the axes, and change the colour of each plot.
3. Highlight the point of max. *Altitude* and max. *Velocity* of the flight.

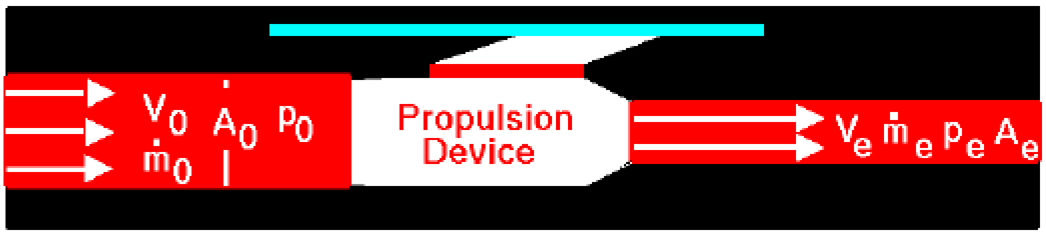
   The task can be done preferably using Python and the matplotlib module. (C++ has plotting options too, but it's a lot easier to work with the libraries in Python directly.) Bonus: Make the plot update in real time by using one loop to generate the values and another to read these inputs from the file and plot the data in real time.

Given parameters and Assumptions:

Thrust is 50KN, and duration of thrust is 10 second, telemetry ends at t = 16s, time interval is 1 second and drag is neglected.
We make some assumptions like the cubesat is being launched vertically, the mass of the cubesat is constant throughout the flight and no drag is considered, after 1o seconds the thrut stops and the cubesat coasts upwards until telemetry stops at 16 seconds.

Theoretical parameters:

We know that the general thrust equation is



Thrust is a force.

Force = change in momentum with time

$$F = \frac{([mV]_2 - [mV]_1)}{(t_2 - t_1)}$$

$\dot{m}$ = mass flow rate = mass / time

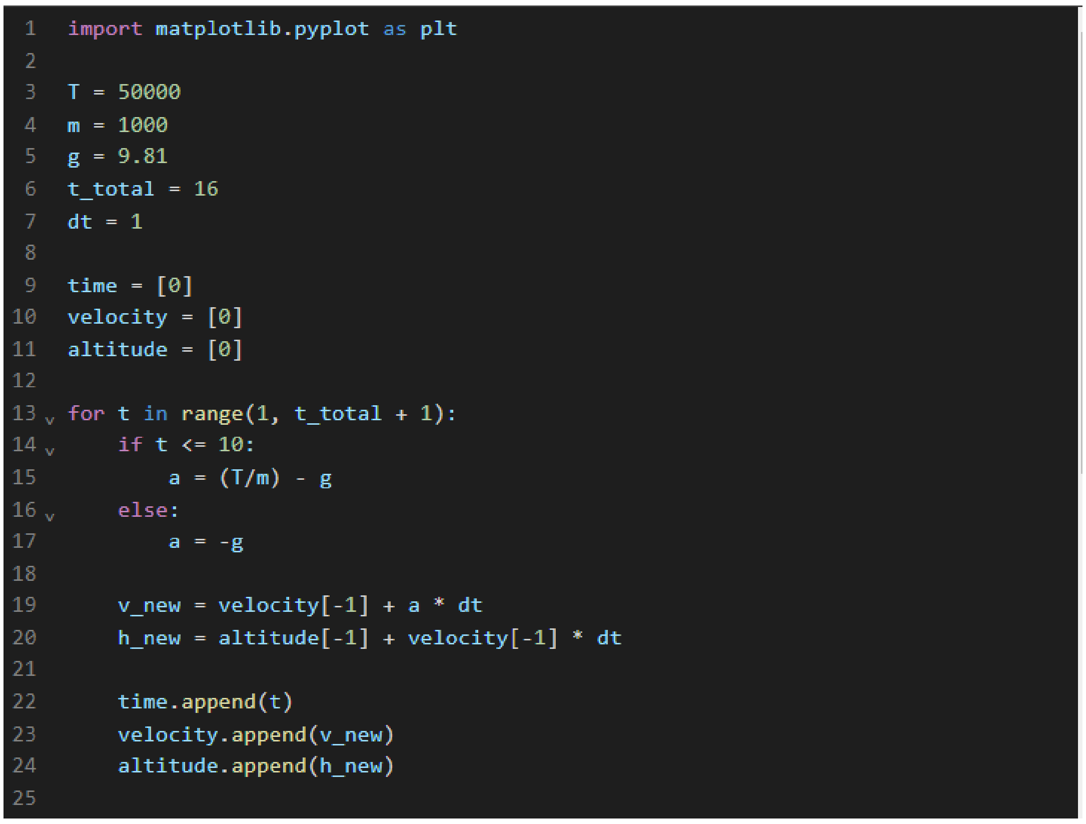
$\dot{m} = r \times V \times A$ where r = density, V = velocity, A = area

$$F = \dot{m}_e V_e - \dot{m}_0 V_0$$

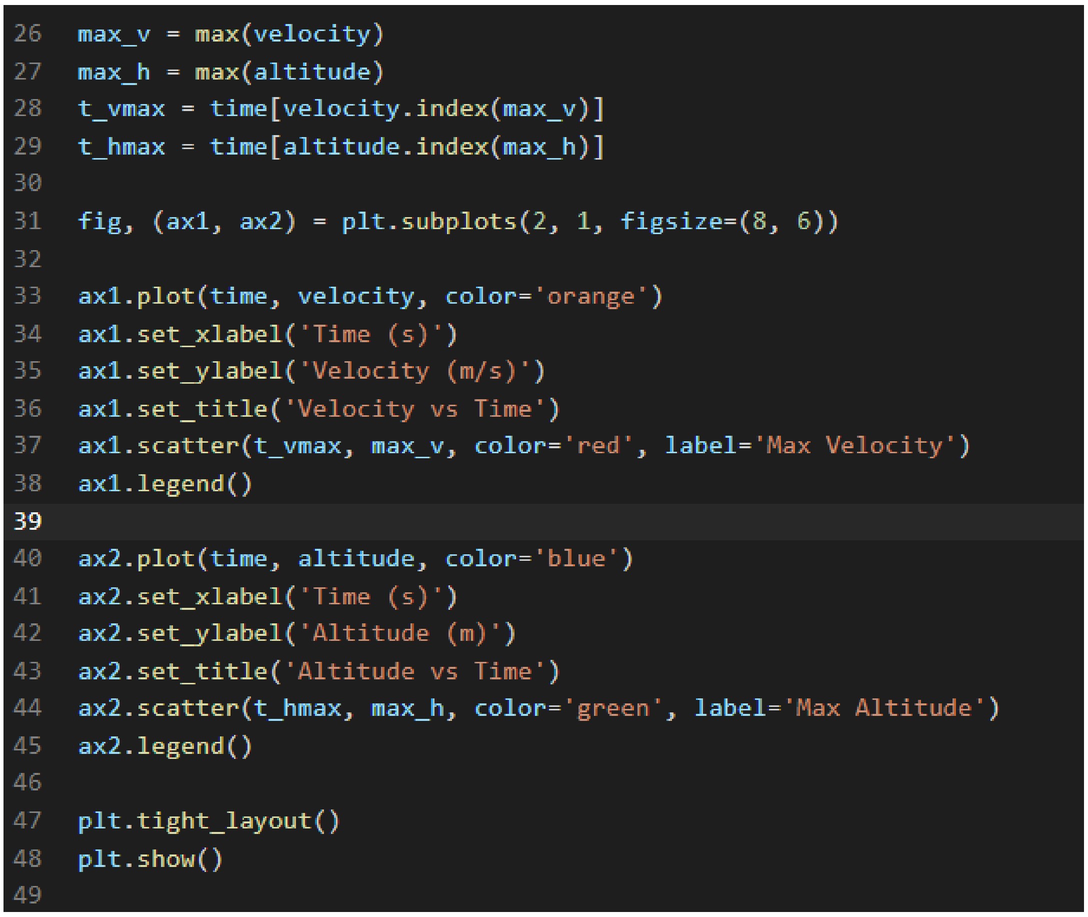
If $p_e \neq p_0$: $F = \dot{m}_e V_e - \dot{m}_0 V_0 + (p_e - p_0) A_e$

Motion after thrust is just the gravitational acceleration due to earth in the downward direction and some other basic kinematic equations.

Code:



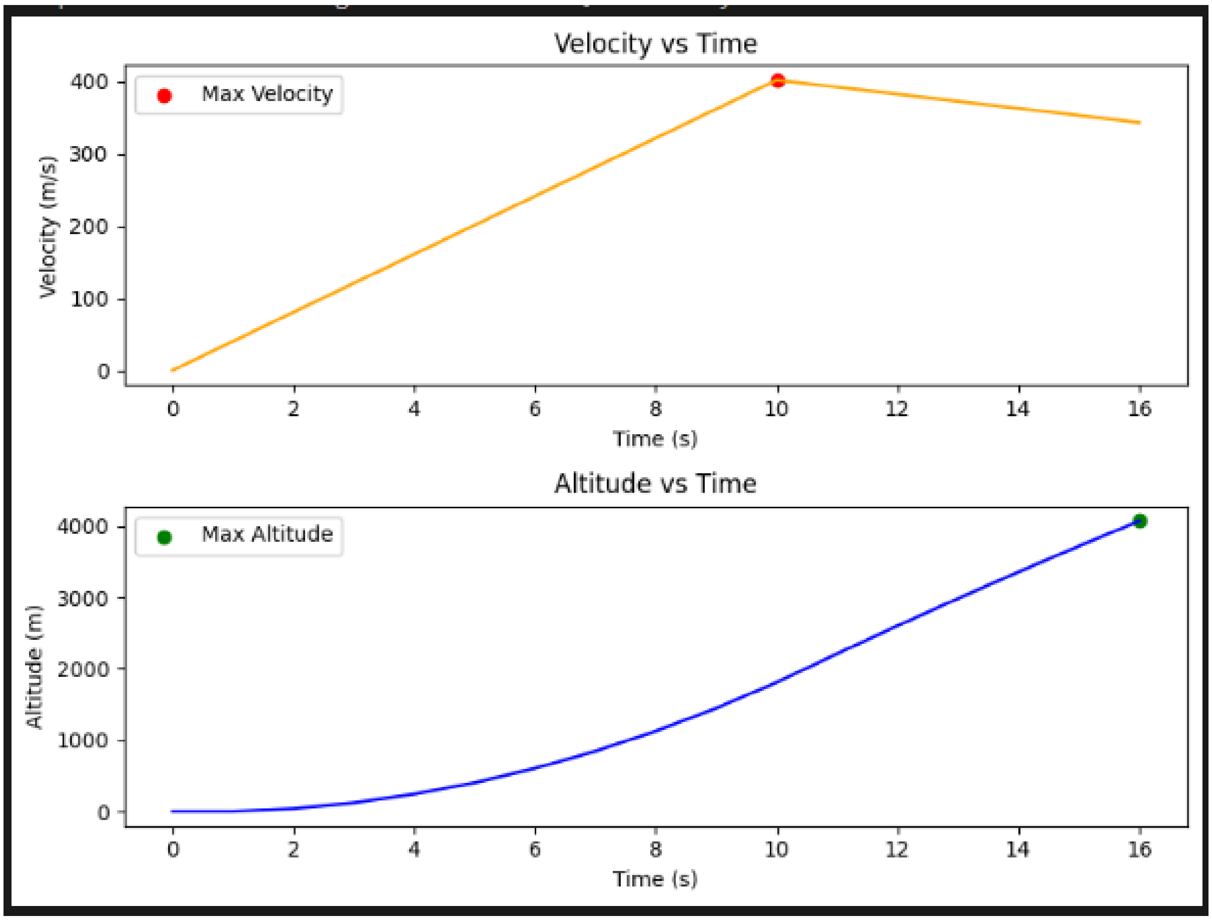
```python
import matplotlib.pyplot as plt

T = 50000
m = 1000
g = 9.81
t_total = 16
dt = 1

time = [0]
velocity = [0]
altitude = [0]

for t in range(1, t_total + 1):
    if t <= 10:
        a = (T/m) - g
    else:
        a = -g

    v_new = velocity[-1] + a * dt
    h_new = altitude[-1] + velocity[-1] * dt

    time.append(t)
    velocity.append(v_new)
    altitude.append(h_new)

```



```python
max_v = max(velocity)
max_h = max(altitude)
t_vmax = time[velocity.index(max_v)]
t_hmax = time[altitude.index(max_h)]

fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(8, 6))

ax1.plot(time, velocity, color='orange')
ax1.set_xlabel('Time (s)')
ax1.set_ylabel('Velocity (m/s)')
ax1.set_title('Velocity vs Time')
ax1.scatter(t_vmax, max_v, color='red', label='Max Velocity')
ax1.legend()

ax2.plot(time, altitude, color='blue')
ax2.set_xlabel('Time (s)')
ax2.set_ylabel('Altitude (m)')
ax2.set_title('Altitude vs Time')
ax2.scatter(t_hmax, max_h, color='green', label='Max Altitude')
ax2.legend()

plt.tight_layout()
plt.show()

```

So what exactly is happening in the code? We first take the constant value for thrust, mass, gravity and time parameters. Then there is a loop which gives different results after a time gap of 10 seconds as the thrust is ceasing, after 10 seconds only gravity acts right. The velocity and height are also being calculated at the same time using elementary kinematics equations for two dimensional motion. Then the data of time velocity and altitude is being stored and kept in track each and every second of the entire motion. We fine the maximum velocity and altitude and mark them on the graph, and those are the peaks. Finally we plot two subplots, one for velocity and another onoe for altitude.

Results:



Conclusion:

The simulation demonstrates how telemetry parameters (altitude and velocity) evolve during a CubeSat launch and coast phase. The Python-based approach provides a simple yet effective way to visualize real-time telemetry behavior and identify key flight moments such as **maximum velocity** and **maximum altitude**.

The code has been provided in the file as .py document. :)

THANK YOU

Anant Nagari - 251ec109