Charles Wszalek

PYTHON CODE

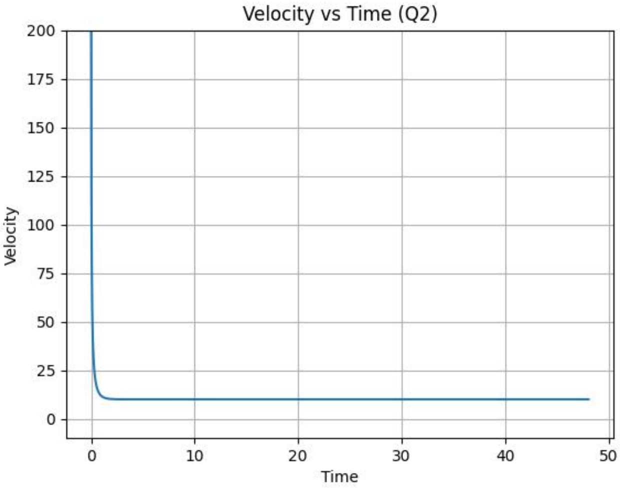
```
from lib.Header import *

rho =  0.00065 # Air Density of Mars: 0.00065 slug/ft3
g = 32.174 # ft/s/s
gmars = .38 * g # Acceleration due to Gravity ft/s/s
wearth = 680 # Weight of the Lander on Earth lb
v0 = 200 # ft/s
m = wearth/g # slug

'''
Q1) (5 points) Calculate the parachute radius required to achieve a terminal velocity of 10 ft/s
'''

rho = 0.002377 # slug/ft^3
Cd = 1.75

v = 10 # ft/s
Area = m * gmars / (.5 * rho * Cd * v**2)
print2(f"Area: {Area}")


'''
Q2) (5 points) Create plots showing the time-dependent variations in velocity and altitude
above the ground based on the following parameters:
    a) Initial Height: 500 ft
    b) Halt the simulation upon ground impact.
    c) Keep the retrorocket deactivated.
'''
CD = Cd * .5 * rho * Area

h1 = 1e-4
time1 = np.arange(0, 50, h1)

x1 = np.zeros(len(time1))
v1 = np.zeros(len(time1))
x1[0] = 500
v1[0] = v0

def acc(v):
    return gmars - ( CD * v**2 / m )

stop1 = 0
for i in range(len(time1)-1):
    if (x1[i] > 0):
        vhalf = v1[i] + .5 * h1 * acc(v1[i])
        v1[i + 1] = v1[i] + h1 * acc(vhalf)
        x1[i + 1] = x1[i] - h1 * v1[i]
        stop1 = i
```

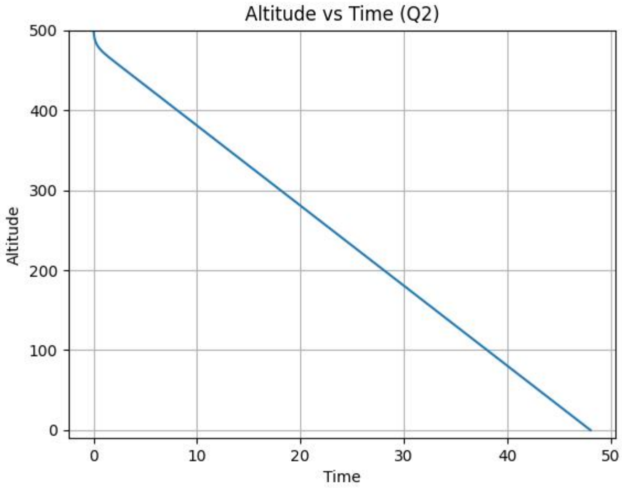
```
    else:
        break

plt.figure()
plt.title("Velocity vs Time (Q2)")
plt.xlabel("Time")
plt.ylabel("Velocity")
plt.plot(time1[0:stop1], v1[0:stop1], label='v1')
plt.ylim(-10, 200)
plt.grid()
SAVE(1)
plt.show()

plt.figure()
plt.title("Altitude vs Time (Q2)")
plt.xlabel("Time")
plt.ylabel("Altitude")
plt.plot(time1[0:stop1], x1[0:stop1], label='x1')
plt.ylim(-10, 500)
plt.grid()
SAVE(2)
plt.show()

'''
Q3) (10 points) Create plots showing the time-dependent variations in velocity and altitude
above the ground based on the following parameters:
    a) Initial Height: 500 ft
    b) Terminate the simulation upon impact with the ground.
    c) Engage the retrorocket when the vehicle's altitude falls below 50 ft AND the lander's
    velocity exceeds 4 ft/s.
    d) Disengage the rocket when the lander's velocity drops below 4 ft/s.
    e) Assume the rocket thrust as T = 0.9 * wmars
'''

h2 = 1e-4
time2 = np.arange(0, 100, h2)

x2 = np.zeros(len(time2))
v2 = np.zeros(len(time2))
x2[0] = 500
v2[0] = v0

def acc1(v, T):
    return gmars - ( CD * v**2 / m ) - T/m

stop2 = 0
for i in range(len(time2)-1):
    if (x2[i] > 0):
        if (abs(x2[i]) < 50) & (v2[i] > 4):
            T = .9 * m*gmars
        elif (abs(x2[i]) < 50) & (v2[i] < 4):
            T = 0
        else:
```

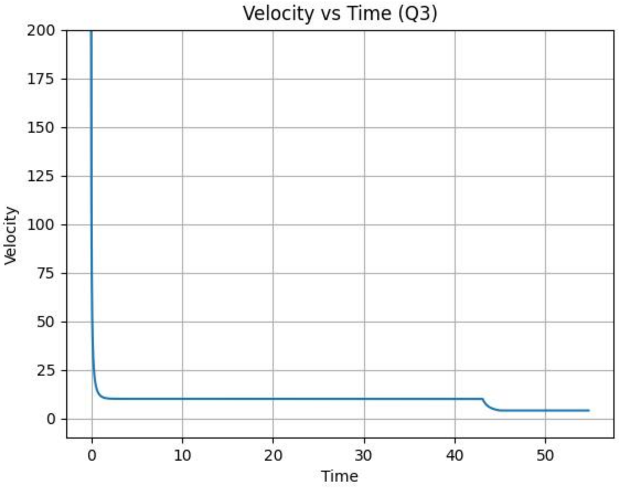
```
            T = 0
        vhalf = v2[i] + .5 * h2 * acc1(v2[i], T)
        v2[i+1] = v2[i] + h2 * acc1(vhalf, T)
        x2[i+1] = x2[i] - h2 * v2[i]
        stop2 = i
    else:
        break

print(v2[stop2])
print2(f"Note that the thruster doesn't turn off in Q3 because the final velocity is {round(v2[stop2], 6)} ft/s at

plt.figure()
plt.title("Velocity vs Time (Q3)")
plt.xlabel("Time")
plt.ylabel("Velocity")
plt.plot(time2[0:stop2], v2[0:stop2], label='v2')
plt.ylim(-10, 200)
plt.grid()
SAVE(3)
plt.show()

plt.figure()
plt.title("Altitude vs Time (Q3)")
plt.xlabel("Time")
plt.ylabel("Altitude")
plt.plot(time2[0:stop2], x2[0:stop2], label='x2')
plt.ylim(-10, 500)
plt.grid()
SAVE(4)
plt.show()

PDF()
```

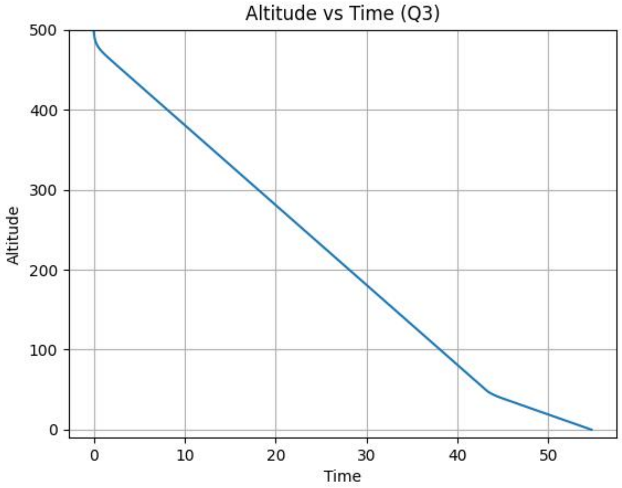
OUTPUT

Plots









Prints

Area: 1242.3823547088166
Note that the thruster doesn't turn off in Q3 because the final velocity is 4.000405 ft/s at 0.000206 ft and time 547339.0 s.