**MIE 124**

**Homework 8**

After noticing your phenomenal performance in MIE 124 Elon Musk has personally recruited you to come work for Space X. Today just so happens to be your first day on the job, it also happens to be the day of a big launch. Unfortunately for you, all your engineering colleagues skipped work today because they were out all night boozing in celebration for the big launch. Right before liftoff Elon Musk rushes to your side. With panic in his voice he informs you that someone has deleted the rockets autopilot program. He knows you’re a MATLAB genius, so he pleads for your help. All the weight is now lying on your shoulders to help make this mission a success.

In this assignment, you will need to create three functions. The first function will simulate the launch of a falcon 9 rocket and track its position, velocity, and acceleration until it reaches a certain altitude. The second function will control, and land the rocket’s booster stage as it plummets back to earth after separating from the rest of the rocket. Your third function will calculate air drag and will be called by the two previous functions.

**Air Drag Function**

The function to calculate drag force will be called *force\_of\_air\_drag\_yourinitials*. This function will have three scalar inputs, *velocityX, velocityY,* and *altitude*. This function will output two scalar quantities that should be called *dragforceX* and *dragforceY*.

Within this function you will need to calculate the force from air drag in each axis. This is the equation you will need to use:

For the X and Y direction, assume the drag coefficient is equal to .25. However, the X direction has an area of 148.29m^2 and the Y direction has an area of 48.29m^2. For the sake of this problem we are going to assume air density is only a function of altitude (decreasing as you get higher) and can be modeled with this equation:

NOTE: if the altitude is greater than 100,000 meters this equation will produce a negative result. Within your function set the density to zero if the altitude is 100,000 meters or more.

**Launch Simulation Function**

The second function will have no inputs and should be called *launch\_simulation\_your\_initials*. There are 9 outputs of this function and they should be named as such: *fuelmass, Xdisplacement, altitude, velocityX, velocityY, accelerationX, accelerationY, theta, and time,* all of which should be stored as vectors.

Initial Conditions

The rocket you will be flying has an empty mass of 300,000 kg (without any fuel in it), and burns 1000 kg of fuel per second when the thrusters are on. You will track the amount of fuel in the tank using the vector *fuelmass,* which should be initiated with a value of 400,000, representing the amount of fuel you have at launch. The thrusters produce an earth shaking 7,000,000 newtons of thrust.

Simulation

The function starts at the very moment the rockets fire (time=0). In this moment, the rocket is pointing straight up at the sky so theta=90 and velocity, altitude, x displacement all equal 0. There is a net force coming from the thrusters and an associated acceleration, as well as force due to gravity. Theta is the direction the front of the rocket is pointing, measured counterclockwise from the horizontal (pointing straight up is 90 degrees, pointing in the positive x-direction is 0 degrees, pointing in the negative x-direction is 180 degrees, etc). This is therefore the direction that thrust acts.

Your function should use a while loop that continues while the rocket’s altitude is below 100,000 meters. The time should increase by .1 seconds for each iteration of the while loop.

For each interaction of the while loop, do the following:

1 – Use the current values (at the jth iteration) of the variables to calculate the total force on the rocket, which is the sum of the thrust, gravity, and drag forces (all of which have an x and y component). The thrust will be on the entire time during the launch, and you can assume the gravitational acceleration is constant. Drag depends on altitude, velocity, and theta and has two components in x and y.

2 – Use the total force and the total mass to calculate the current acceleration (jth iteration).

3 – Use Euler integration to calculate the velocity at the next time step (j+1th iteration).

4 – Calculate the next position (j+1th iteration) using the average of the current and next velocity.

5 – Update the fuel mass of the rocket (j+1th iteration).

6 – Update theta. Once the rocket has reached an altitude of 50,000 meters it will begin to slowly rotate in the clockwise direction. This is designed to create an elliptical trajectory, making it easier for the rockets second stage to enter orbit once it has separated. The angle between the vertical axis and the rocket is theta. At this stage theta can only change 5 degrees per second. After reaching 50,000 meters your value of theta should increase until it reaches 55 degrees where it will remain for the rest of the flight.

Remember that all your variables will be vectors with length equal to the number of time steps to reach 100,000 meters. You should have no problem reaching the desired altitude of 100,000 meters with plenty of fuel left. At that point your while loop should stop and you should produce the following figures (all with appropriate axes, etc).

First, plot the trajectory of your rocket in Figure 1, i.e. plot x displacement versus altitude. Plot a red circle at the location where theta begins to increase (about 50,000 meters). Fix the axis limits so that the x and y scale have the same extent (hint use “axis equal”).

Next on a new figure, plot X acceleration vs time and Y acceleration vs time, in blue and black. Additionally, plot a red circle at the location where theta begins to increase (about 50,000 meters).

If you have made it this far you have successfully launched your rocket into space! Here comes the hard part. After the rocket reached 100,000 meters in altitude, the booster stage separated and began to plummet back to earth. These booster stages are extremely expensive and difficult to build so it’s extremely important that you bring it back in one piece. Your boss is counting on you!

**Booster Landing**

For this last step create a new function called *booster\_landing\_yourinitials*. It has no inputs and two outputs named *success,* and *distancefromtarget*. Success will return a 1 if the mission was a success, and 0 if the mission was a failure. *distacefromtarget*  will report how far away the rocket landed from its target landing site.

Initial Conditions

Your mission is to land the booster stage at a spot called *targetX. targetX*  is equal to 1/4th the x-position of the rocket when it reached 100,000 meters (the last value in your *Xdisplacement* vector after the *launch\_simulator* function ended). For example: if your rocket reached 100,000 meters in altitude at an X displacement of 8,000 meters in the positive X direction, then your target would be 2,000 meters in the positive X direction form the original launch site.

The first line of your function should be to call your *launch\_simulator* function. The initial positions and velocities of the booster are simply the final values of the positions and velocities of the launch simulation. Fortunately for you, after the stages separated a 150,000 kg fuel tank was made available within the booster stage. The booster stage now has a mass of 150,000 kg (empty mass after separation) + 150,000 kg of fuel + whatever fuel you had left over from the launch. The thrust can be either 7,000,000 N or 0, and due to the first stage being detached the booster can now rotate up to 10 degrees/second where it was only 5 degrees/second before stage separation.

Simulation

You have two options for the simulation.

Option 1 – No Control. In this case, you do not use the thrust of the rocket, and the only forces are due to gravity, drag, and wind (more below on this). The rocket will eventually reach an altitude of 0 m due to gravity forces. In a similar manner to the launch, simulate your rockets path until it reaches the ground and calculate the distance between its x-positions and the target x-position for your output, with *success = 0*. In this case, the total mass is constant. The time step is the same, and you can reuse the air drag function. Likewise, the method for determining the velocity and position on each time step is the same. Include a force from wind that can exert a random force between +/- 800 N in the X direction at each time step.

Option 2 – In this case, you can control both theta and thrust at each time step. Thrust is either on or off, and there is a limit of 10 deg/sec for the rate of change of theta. You can only have positive thrust while the fuel mass is greater than 0. Your mission will result in a success if you can land the rocket within 500 meters of the target site while having an X and Y velocity that does not exceed +/- 200 m/s. The lower your X and Y velocity, the better however. Try to minimize them!

State in a comment at the top of the code which option you choose. Set Opt = 1 or Opt = 2 at the top of the code to indicate what you are doing. If you do Option 1 perfectly, you can get up to a 90. If you do Option 2, include an if statement that executes Option 1 if Opt == 1 and executes Option 2 is Opt == 2 (Note you do not need to repeat every line of code, you can just have the keep theta and Ft unchanged if Opt == 1). Thus to do Option 2, you must do Option 1 first. If you choose to do Option 2, you will get a 90 if Option 1 works perfectly, and up to 20 additional points depending on how well your controller in Option 2 works. Include a lengthy comment describing the approach you use to control your rocket in Option 2.

After landing produce two figure. First plot the x-position versus altitude. On a new figure, plot the x and y velocities versus time. Use labels, legends, etc.