Self Study 3 – Which Planet Am I On?

Introduction

In the 2nd and 3rd labs we looked at logic operators, for and while loops and if/else structures. Your task in this exercise will require all these tools to solve a problem.

Although we will be using ScriptCheck to test on your computer you will need to download `self_study_3.mat` that contains the variables we need for the exercise and the file `check_escape.p` which is a user defined function. These should be placed in your working directory.

The file extension .p denotes a MATLAB .m file that has been protected and cannot be read by hand but can still be called and operate as a normal function taking input arguments and giving an output.

The Scenario

You awake on a planet somewhere in our solar system, but you have no idea which one (because for strange reasons probably related to the currently unknown mechanism that brought you here you're oblivious to temperature, atmosphere, pressure etc….). The only way you can determine which one is by using a rather convenient projectile launcher that will fire us into space at a specified initial velocity and then checking to see whether we make it to space or fall back to the planet surface.

Luckily you have some information about the 8 planets plus you remember you covered this in XJME1520 back on earth as an example on solving differential equations. Thankfully you recall that the velocity required to escape a planets gravitational field can be calculated by:

$$U_{esc} = \sqrt{\frac{2GM}{R}} \quad (1)$$

where U_{esc} is the escape velocity, G is the gravitational constant in N m^2 kg^{-2} , M is the mass of the planet in kg and R is its radius in meters.

Unfortunately as we're not sure what planet we're on, we are unsure what this will need to be. We are going to start off with low velocity and check if we can successfully escape. If unsuccessful, we are going to iteratively increase our velocity until we can escape from the planet. Once we know what the escape velocity was, we'll be able to compare it to our calculated values and determine which planet we were stranded on. It all makes perfect sense and you'll be happy you know how to do this should this ever happen to you.

To solve this, we're going to use several of the concepts we covered this week and last week which is the real point of the exercise and is why this whole thing is rather contrived.

Instructions

1. Create a program .m file called `which_planet_am_i_on.m`. Don't forget to add a comment at the start of the script containing the file name and some useful comments describing what the file will do, i.e. works out which planet you're on by trying to get off it.

The variables you will need to use are below, remember you can cut and paste them to avoid typos:

```
escape_velocity
test_velocity
increment
escape
planet_index
planet_name
```

2. Programmatically clear the workspace and command window. (QG 0)

3. In your program load the `self_study_3.mat` file into the workspace. (QG 11.1) The .mat file contains a cell array called `planets` that contains the planet names in order from the Sun. A cell array allows us to put the character strings of different lengths into an array format such that we can index each name. It also contains 2 row vectors containing the `mass` and `radius` of each of those planets in the same order and a scalar containing the gravitational constant `G`.

4. Using a `for` loop, step through the data for each planet in turn, calculating the escape velocity using equation (1) and assign it to a vector variable `escape_velocity`. (QG 9) (hint: if you use a counter that counts from 1 to the number of planets, you can use that counter to index each value in the vectors). `escape_velocity` should grow each iteration of the loop but using the loop counter as an index you can place each new value in the vector (QG 2.3 but you are creating the vector rather than replacing a value).

5. Within the for loop you can use `disp` to display a line of text that says something along the lines of 'The escape velocity for (insert `planet_name`) is (insert `escape_velocity`) m/s' for each planet in turn. (QG 12.2) (Note: to be able to display the planet name correctly we need to use a slightly different method to get the value from the cell in the cell array. Instead of indexing with parentheses i.e. `planets(index)`, we can use the curly brackets `planets{index}`) however this is optional for your submission to ScriptCheck as ScriptCheck won't read it but it is a good way to check if your code is working.

The result of the for loop should be 8 lines of text printed to the command window stating escape velocity for each planet plus a new variable in the workspace called `escape_velocity` which will be a row vector containing the 8 calculated escape velocities.

6. Define three new variables, they will be scalars called `test_velocity` , `increment`, and `planet_index`, set them to 100, 1 and 1 respectively. `test_velocity` will be your initial guess for the escape velocity of the planet, `increment` will be the amount you will increase `test_velocity` by on each successive attempt, and `planet_index` will change later depending on which planet you are on.

7. You are now going to use the protected function you downloaded from the VLE. You can't see the planet parameters contained within the .p file. The function takes an input that is `test_velocity` and checks if it is greater or equal to the planet you are on's escape velocity. It will set `escape` to true if this is the case, otherwise false. (note that false and 0, and true and 1 are interchangeable).

Use the code `escape = check_escape(test_velocity)` to call the function with the initial `test_velocity`, this will also initialise the variable `escape` which in this first instance will be false.

8. You are now going to use a while loop that will increment the `test_velocity` and check if `escape` is true by calling `check_escape` on each iteration of the loop. It should stop as soon as `escape` becomes true (or 1). (QG 7) The result of this loop should be a value in `test_velocity` that is equal or just above the escape velocity of the planet you are on. Your code should resemble the following:

```
while (condition to be met)

      test_velocity = test_velocity + ???;
      escape = check_escape(test_velocity)

end
```

9. Now we have an approximation of the escape velocity of the planet you were on. We can now use a for loop again to step through the values in the vector `escape_velocity`. Within the for loop we want to use an if statement to check if the `test_velocity` returned from the while loop above is within an range of one (increment value) of the escape velocities we calculated for the 8 planets in step 5. Think about the conditional logic statement required for this. The result should be an index for the particular planet that matches set to `planet_index`. This bit of code will be of the form:

```
for x = 1:???

    if (conditional statement comparing velocity and escape_velocity (QG4))
        planet_index = x;
    end

end
```

10. Finally you have the index of the planet on which you have strangely found yourself. Index the cell array `planets` using the result of section 8 to get the name of the planet as a string (think about type of brackets), setting it to the variable `planet_name`. Remember `planet_name` should be string data rather than a cell (check workspace for data type).

11. Test your code in MATLAB before uploading to scriptcheck, ensure you have loaded the data in programmatically and your variable names are correct. https://scriptcheck.leeds.ac.uk