# CS 1112 Fall 2015 Project 3 Part B — due Monday 10/5 at 11pm

Part A of this project is in a separate document on the course website. Both parts have the same deadline.

You must work either on your own or with one partner. If you work with a partner you must first register as a group in CMS and then submit your work as a group. *Adhere to the Code of Academic Integrity.* For a group, "you" below refers to "your group." You may discuss background issues and general strategies with others, but the work that you submit must be your own. In particular, you may discuss general ideas with others but you may not work out the detailed solutions with others. It is not OK for you to see or hear another student's code and it is certainly not OK to copy code from another person or from published/Internet sources. If you feel that you cannot complete the assignment on your own, seek help from the course staff.

## Objectives

Completing this project will solidify your understanding of user-defined functions and vectors, and you will also have the opportunity to use more graphics.

## Ground Rules

In this part of Project 3, you will use vectors for storing values, but the arithmetic operations that you write must be *scalar operations*. For example, let `v` be a vector: then `v(k)+5` is a scalar addition operation (assuming that `k` is a valid *index* of vector `v`), i.e. the *operands* of the addition operator are both scalars, not vectors. An example of a *vector operation*, which you should **not** use, could be `v+5`, which would add 5 to every element in `v`. Note that `v` is a *vector* while `v(k)` is a *scalar*.
Also, please do not use the `break` command.

## Flight of a Rocket

A rocket burns fuel to produce thrust, so it loses weight as time goes on until the carried fuel is burnt out. Given the rocket's mass without the fuel $m_R$, the fuel burn rate $q$, and the burn time $b$, we can calculate the rocket's initial mass including the fuel as:

$$m_0 = m_R + qb \tag{1}$$

Now given the exhaust velocity $u$ of the burned fuel relative to the rocket, the acceleration due to gravity $g$, and $m_0$, we can use Newton's law (neglecting air resistance) to derive the following equations. These equations describe velocity $v$ and height $h$, as a function of time $t$, of the rocket when it is launched vertically:

$$v(t) = u \ln \frac{m_0}{m_0 - qt} - gt \tag{2}$$

$$h(t) = \frac{u}{q}(m_0 - qt)\ln(m_0 - qt) + u(\ln m_0 + 1)t - \frac{gt^2}{2} - \frac{um_0}{q}\ln m_0 \tag{3}$$

After the burn time, the rocket no longer produces thrust and the following equations apply:

$$v(t) = v(b) - g(t-b) \tag{4}$$

$$h(t) = h(b) + v(b)(t-b) - \frac{g(t-b)^2}{2} \tag{5}$$

The following equations are used to calculate the peak height $h_p$, the time to reach peak height $t_p$, and the time to hit the ground $t_f$:

$$
\begin{aligned}
h_p &= h(b) + \frac{v^2(b)}{2g} && (6) \\
t_p &= b + \frac{v(b)}{g} && (7) \\
t_f &= tp + \sqrt{2h_p/g} && (8)
\end{aligned}
$$

Suppose we have a rocket that is carrying instruments to study the upper atmosphere and it carries just enough fuel for the planned burn time.

## 1 How long does the rocket fly at a certain range of altitudes?

Write a function `timeAtTargetAlt` that determines the duration of time that the rocket is between two altitudes.

Your function should take the following *arguments*:

1. `mR` - the mass of the rocket in slugs
2. `u` - the exhaust velocity of burned fuel relative to the rocket in ft/s
3. `b` - the burn time in seconds
4. `lo` - the low end of the desired altitude range
5. `hi` - the high end of the desired altitude range
6. `dt` - the time step for evaluating the height in the simulation, i.e. the unit of time by which you should split the time $t_f$
7. `makePlot` - a variable that can either be 0 or 1; if it is 1, a plot of time versus height (altitude) should be generated, i.e. a graph of the rocket's height at each time step `dt`.

Your function should give back the following *return values*:

1. `dur` - the duration in seconds that the rocket is in the target altitude range (i.e. between altitudes `lo` and `hi`)
2. `t` - a vector of total simulation time, from 0 to the total flight time ($t_f$) in steps of `dt`
3. `h` - a vector of the rocket's altitude (height) at each time in the vector `t`

Please use the following constants: $g$ (acceleration due to gravity in ft/s$^2$) = 32.2 ft/s$^2$, and $q$ (burn rate of fuel in slugs/second) = 1 (a slug is a mass that accelerates by 1 ft/s$^2$ when a force of one pound-force is exerted on it).

Some notes for designing your algorithm:

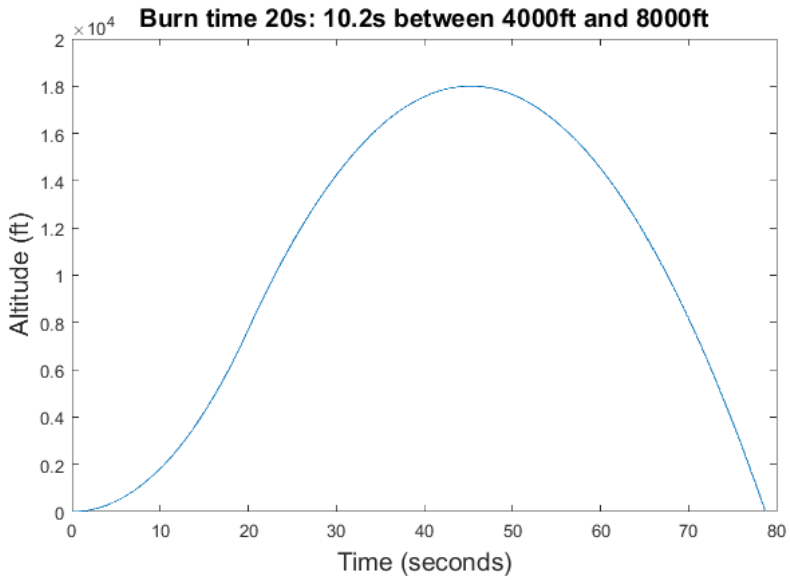
- You'll need to use a loop to calculate the rocket's height at each time step. Recall that the height is calculated using different formulas depending on whether or not the rocket has finished burning all of its fuel - see equations (3) and (5).
- Once you've calculated the height at each time step, you'll need to find the duration of time for which the rocket was between the heights `lo` and `hi`, but this can only be done if the peak height $h_p$ is greater than the lower height `lo`. The number of time steps for which the rocket was between these heights, multiplied by the length of each time step, gives the duration of time that you need.

Some notes for the graphics: use the following commands to create the plot.

```
close all   % close all figure windows
figure      % create new figure window
hold on     % hold all calls to plot on current axes

plot(vector1, vector2)  % plot values of vector1 on x-axis, vector2 on y-axis
xlabel('...')   % add a label to x-axis
ylabel('...')   % add a label to y-axis
```

Below is an example of what the plot would look like for `mR` = 100, `u` = 8000, `b` = 20, `lo` = 4000, `hi` = 8000, and `dt` = 0.1.




## 2 Sensitivity Analysis - Varying burn times

A *sensitivity analysis* is used to explore how the result of a model changes when the inputs to the model are systematically varied, i.e., we ask the question "how sensitive is the result to variations in the model parameters?"

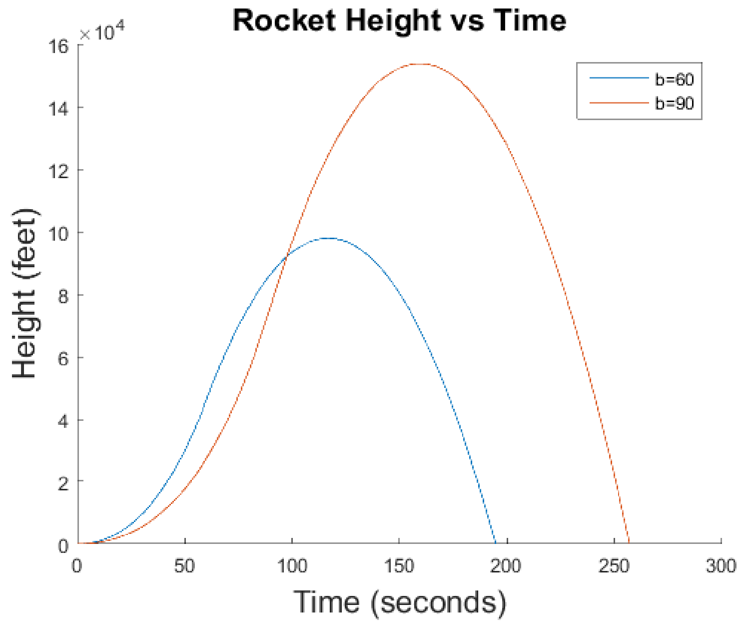
You will explore the sensitivity of our rocket flight model to changes in the parameter $b$, the burn time. Use the following values of $b$: 10, 15, 30, 60, 90, and 120 seconds. Specifically, use a loop and your function `timeAtTargetAlt` from part 1 to generate variables `dur`, `t`, and `h` (which are described in the information for part 1 on the previous page) for each of the different burn time values.

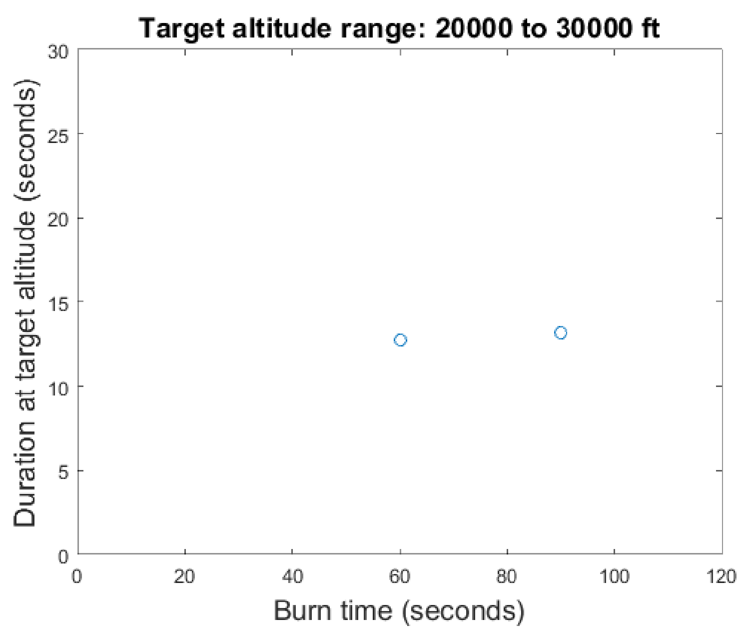
Use the following values as inputs to `timeAtTargetAlt`: `mR` = 100, `u` = 8000, `lo` = 20000, `hi` = 30000, `dt` = 0.01, and `makePlot` = 0 (set `makePlot` to 0 because this script will generate its own graphics, as described below). Recall that the parameter `b` to `timeAtTargetAlt` is changing on each of your loop iterations.

Write your code for this sensitivity analysis in a script `varyBurnTime`. Your script produce two plots:

1. A plot of time versus the rocket's height for each of the values of $b$ listed above. That is, this plot should have six curves on it, one for each value of $b$. On the next page is an example plot showing only two of the curves (for $b$ = 60 and $b$ = 90) - you'll see what the rest looks like after completing this question!

2. A plot of duration of flight spent at the target altitude range versus burn time. The second plot on the next page is an example showing only two of the points (for $b = 60$ and $b = 90$).




The following code outline shows the relevant graphics commands for plotting the six curves on the first plot. Note that your script should produce two figure windows for the two plots described above.

```
close all
figure
hold all
```

```
legendText = {};   % the curly brace notation is used here for storing text
                   % strings and will be explained later in the course

for k = 1:___      % loop over the different values of b

    % calculate [dur, t, h] using the timeAtTargetAlt
    % function for the current value of b

    % plot t on x-axis, h on y-axis using plot()

    legendText{k} = sprintf('b = %.0f', ___);

end

legend(legendText) % place strings in legendText in the plot legend
hold off           % needed before we create the second plot

% create the duration vs. time plot here!

% note: to create a plot that has open circular points, as in
% the example, use plot(___, ___, 'o').
```

Submit your files `timeAtTargetAlt.m` and `varyBurnTime.m` on CMS.