OENG1207 – Digital Fundamentals

Laboratory Exercise 1
Introduction to MATLAB

Laboratory Exercise 1: Introduction to MATLAB

Learning Outcomes

Upon successful completion of this practical task, you will be able to:

- Use the basic MATLAB functions to solve a simple physics problem.
- Become familiar with using the MATLAB plotting tools for visualising data.

Introduction to the practical

American daredevil stuntman Evel Knievel was most well- known for his world records in motorcycle jumping. One of his most famous feats was his successful world record jump over 19 cars on February 28th, 1971. This world record stood for 27 years until 1998.

This practical is going to consider some of the physics behind projectile motion which can be used to simplistically model this type of scenario. We will also look at how MATLAB can be used to model physical systems and display information about these scenarios.

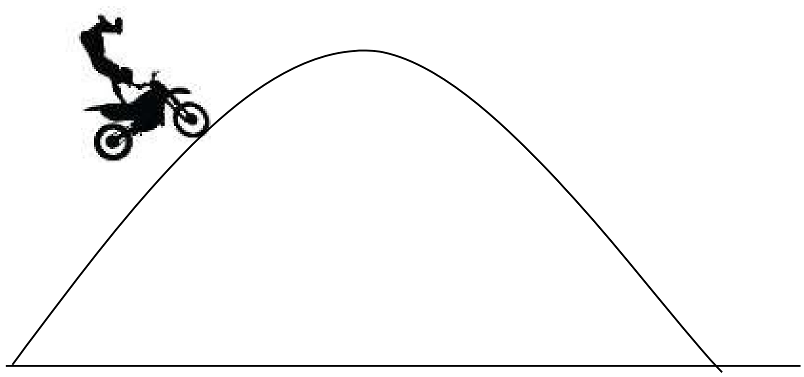

Figure 1: Projectile motion

Part 1 (week 2): The Physics of Projectile motion

One of the aspirations of Evel Knievel during his career was to motorcycle- jump over the Grand Canyon. As it happened, he never achieved this feat, but let's consider if this was even theoretically possible.

Let's start this problem by assuming a level surface between start and end points, as shown in the trajectory illustrated in Figure 2.

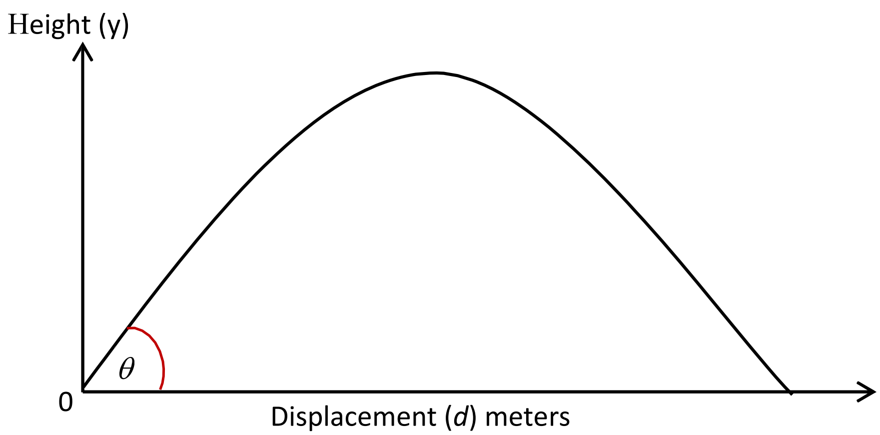

Figure 2: Model of projectile motion

To find the distance travelled we need to know some information about the **initial velocity** (v_0) of the projectile, its **angle of launch** (θ, in radians) and the **acceleration due to gravity** (g = ~9.8 ms^{-2}):

$$\boxed{d = \frac{{v_0}^2}{g}\sin(2\theta)\text{ meters}}$$

Where *d* is the **displacement in metres**.

Begin this practical by working out **how fast (initial velocity) Evel Knievel would have had to have been going to cross the Grand Canyon (in km/h)**. The point where the Grand Canyon is at its narrowest is at Marble Canyon where it's ~183 metres wide. Also assume the angle of launch is 45^o, which will maximize the distance travelled.

For the problems in this practical we'll ignore factors such as **air resistance** and the way the motorcycle rider may change the trajectory using his/her body (**changes to centre of gravity**), this will give us a simplified model of this scenario. You should discuss all these assumptions in your practical write-up.

Task 1 – State the Problem and Determine the Input/Output

Exercise:

Using the information given on the previous page state the problem concisely and determine the input you have/need to solve the problem and what you need to output to answer the question.

Points to be addressed:

- The problem statement should be **clear and concise**, double-check with your tutor if you are not sure of anything written in the problem statement.
- What input/output **data types** do you need (numeric, words, images, graphical output?).
- What **assumptions** are you making to help you solve/simplify the problem

Task 2 – Calculate the Simple Physics for the Scenario

Exercise:

Using the information given on the previous page, manually find:

- The **initial velocity** required to jump over the Grand Canyon at its narrowest point.

Points to be addressed:

- Make sure the velocity you present in your write-up is in km/h, you will need to convert your answer to this unit.
- Make sure you show all the working for this part and ensure this working is neatly typed (not handwritten) in your write-up.

Task 3 – Use MATLAB to Find Velocity

Exercise:

Now, in a new MATLAB script file enter all the parameters you've been given and use these in the equation for displacement to find and display the **initial** v**elocity** required (again in km/h).

Points to be addressed:

- Compare the answer from MATLAB to your handworked answer from task 2. They should be the same, if not troubleshoot your maths/script to find where the mistake is.
- When using MATLAB to do this, make sure you store all the information given to you (e.g. angle of launch, acceleration due to gravity and required displacement) in **appropriately named variables**, then reference those variable names in your equation to find velocity. Do not use '*magic numbers*' in yoursolution.

Part 2 (Week 3): Change Initial Height

The problem could also be approached by including an initial height (as illustrated in Figure 3), this would add in an extra term *H*. This will now change the total x-axis displacement that can be achieved.

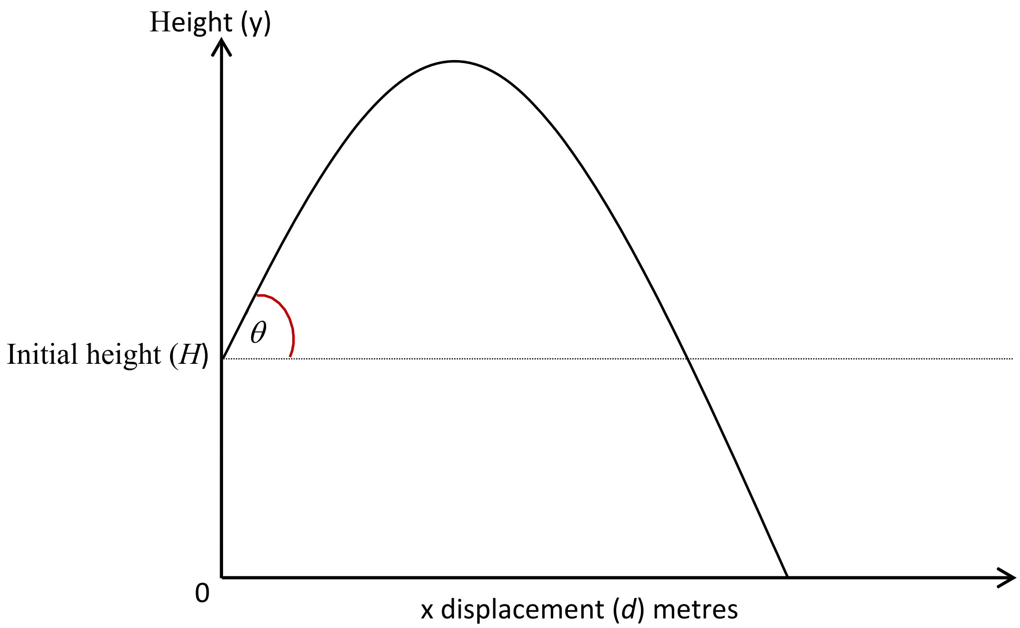

Figure 3: Model of projectile motion with a non—zero initial height

$x = v_0 . t . \cos(\theta)$ (x-displacement in meters)

$y = H + v_0 \, t . \sin(\theta) - \frac{1}{2} g t^2$ (y-displacement in meters)

To describe the motion of this projectile we can break it into the displacement in the x and y direction and then put all this together to show total displacement.

Where *t* is the **time** in seconds and *H* is the **initial height** in metres.

Now examine **what initial height** would be required to cross the Grand Canyon if the **angle of launch** stays at 45^{o} but the **initial velocity** is now restricted to a <u>maximum</u> of 125 km/h.

Lastly, examine what displacements would be achieved with this initial velocity and height, but launch angles of 35^{o} and 55^{o}, <u>show as a plot</u> and **present the angle that results in the greatest distance being achieved**.

Task 4 – State the Problem and Determine the Input/Output

Exercise:

Using the information given on the previous page state the new problem concisely and determine the input you have/need to solve the problem and what you need to output to answer the question.

Points to be addressed:

- The problem statement should be **clear and concise**, double-check with your tutor if you are not sure of anything written in the problem statement.
- What input/output **data types** do you need (numeric, words, images, graphical output?).
- What **assumptions** are you making to help you solve/simplify the problem.

Task 5 – Calculate the Physics for the new Scenarios

Exercise:

Using the information given on the previous page, manually find:

- The **total time** it would take to travel the required distance, use the formula for x-displacement to help you find this.
- Use this time to now find the **value of initial height** (*H*) that would be required to make the jump. Hint: the y-displacement formula can help you with this. Remember, at the point y = 0 the rider has touched the ground.
- Now, work out **how far the rider would get** if the angles of launch were 35^o and 55^o.

Points to be addressed:

- Make sure you show all the working for this part and ensure this working is neatly typed (not handwritten) in your write-up.

Task 6 – Use MATLAB to Solve Problem

Exercise:

Now, in your MATLAB script file enter all the parameters you've been given and use these in the equations for x and y displacement to find the value of initial height required to reach the other side of the Grand Canyon.

Also make sure to display graphically the x vs. y displacement graphs for the three angles examined (35^{o}, 45^{o} and 55^{o}).

<u>**Points to be addressed**</u>**:**

- Compare the answers from MATLAB to your handworked answers from task 5. They should all be the same, if not troubleshoot your maths/script to find where the mistake is.
- **Make sure you appropriately label your x and y axes on your graphs** (use the xlabel()and ylabel()functions to do this). This is really very important; your tutor needs to know what all your axes mean when assessing your work.

Table I: Some useful MATLAB functions

Function	Description
Trigonometric Functions	

`sin(x)`
Sinusoid
	Calculates the **sine** of the value(s) in `x`. MATLAB's `sin()` function operates on <u>radians</u> as inputs by default (use `sind()` if working in degrees).
	Syntax example:
	`y = sin(x); % sine of x`
	The inverse ($\sin^{-1}()$) is:
	`x = asin(y); % Inverse sine`

`cos(x)`
Cosinusoid
	Calculates the **cosine** of the value(s) in `x`. MATLAB's `cos()` function operates on <u>radians</u> as inputs by default (use `cosd()` if working in degrees).
	Syntax example:
	`y = cos(x); % cosine of x`
	The inverse ($\cos^{-1}()$) is:
	`x = acos(y); % inverse cosine`

`tan(x)`
Tangent
	Calculates the **tangent** of the value(s) in `x`. MATLAB's `tan()` function operates on <u>radians</u> as inputs by default (use `tand()` if working in degrees).
	Syntax example:
	`y = tan(x); % tangent of x`
	The inverse ($\tan^{-1}()$) is:
	`x = atan(y); % inverse tangent`

Data output tools	
`plot(t,x)` `plot(t,x, '*')`	Plots a graph of `x` vs. `t`. `plot(t, x,'*')` will plot `x` vs `t` as points rather than a line. Other markers include: `'*'`, `'.'`, `'o'`, `'p'`, `':'`, `'--'`.

`xlabel('x-axis text')`	Adds the text string in the brackets as a label for the x-axis.
`ylabel('y-axis text')`	Adds the text string in the brackets as a label for the y-axis.
`title('Figure title')`	Adds a heading to the top of a figure.
`axis([`x_{min} x_{max} y_{min} y_{max}`])`	Sets the axis of a figure to the (x,y) values entered in to x_{min}, x_{max}, y_{min} and y_{max}.

`subplot(n, m, k)`	Creates subplots within a figure window.
	`n` is the number of rows of subplots.
	`m` is the number of columns of subplots.
	`k` is the current plot being referred to (from left to right and/or from top to bottom).
	Syntax example:
	`figure(1)`
	`subplot(2,1,1), plot(t,x)`
	`subplot(2,1,2), plot(t,y)`