Plotting and Analyzing Graphs
Functions in MATLAB
Solving Dynamic Problems

LAB 6

ENGG100 - Spring 2024

Lab 6 Objectives

- To be able to create 2D graph plots, edit features and analyze data in MATLAB
- To be able to create modular sections of code using functions
- To solve dynamic complex problems using MATLAB

Plotting in MATLAB

- The most common type of plot is `plot[x,y]` where x and y are both vectors of the same length, e.g:

```
x = 0:pi/40:4*pi;
plot(x,sin(x));
```

- Straight-line graphs are drawn by providing x and y coordinates of the endpoints in two vectors. For example, to draw a line between the points (0,1) and (4,3), we would write:

```
plot([0 4], [1 3]);
```

LABELS:

- Graphs can be labelled by using `gtext('X marks the spot')` which writes a string in the graph window
- Labels can also be placed by clicking on **Tools > Edit Plot** in the figure window
- Labels can also be added to a specific coordinate by using `text(2,2,'information')` which would add a label to the point (2,2) on the graph
- Legends can be created by using `legend('plotlabel1','plotlabel2','plotlabel3');`

TITLES AND AXIS LABELS:

- Titles for graphs can be added using - `title('Straight Line Graph')`
- X and Y labels can be added using - `xlabel('Time'); ylabel('Speed');`

Plotting in MATLAB

MULTIPLE PLOTS ON SAME AXES:

1. Easiest way is to use hold to keep the current plot on the axes. All subsequent plots are added to the axes until hold is released, either with hold off or just hold which toggles the hold state:

```
plot([0 4], [1, 3]);
hold;
plot([1 3], [1, 3]);
hold off;
```

2. Second way to plot is by using multiple arguments within the plot function, e.g. `plot[x1, y1, x2, y2, x3, y3]` plots (vector) pairs (x1,y1), (x2,y2) and (x3, y3), etc. This way is helpful when you have vector pairs that are different lengths. MATLAB also automatically selects a different color style for each pair

3. Finally, `plot(x,y)` can be used when both x and y are matrices or one of them is a vector and the other is a matrix

MULTIPLE GRAPHS ON SAME FIGURE WINDOW

- Multiple graphs can be plotted in the same figure window by using subplot() function
  - `subplot(2,1,1); plot(x,y);` - this creates a figure window of size 2x1 and plots x&y as the first figure
  - `subplot(2,1,2); plot(x,y);` - this uses the previously created figure window of size 2x1 and plots x&y as the second figure
  - `subplot(2,2,1); plot(x,y);` - creates a 2x2 figure window and plots x&y as the first figure(1,1)
  - `subplot(2,2,2); subplot(2,2,3); subplot(2,2,4)` - plots the respective graphs in the same 2x2 figure window created above

Plotting in MATLAB

LINE STYLES, MARKERS & COLORS

- Line styles, markers and colors may be selected for a graph, e.g.
- `plot(x,y,'-')` joins the plotted points with dashes
- `plot(x,y,'o')` draws circles at the data points, with no lines joining them
- `plot(x, sin(x), x, cos(x), 'om-')` plots sin(x) in the default style and color, and cos(x) with circles joined with dashes in magenta
- Available colors are denoted by `c,m,y,k,r,g,b,w`

EDITING AXES LIMITS

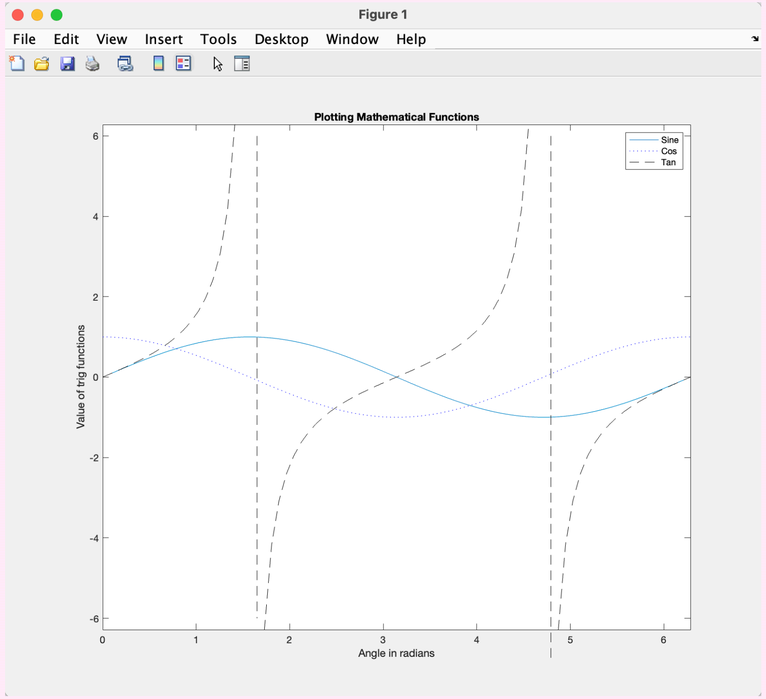
- By default, MATLAB automatically scales the axis to fit the data, this can be overridden by:

```
axis([xmin, xmax, ymin, ymax]);
```

TASK 1 - SCRIPT

- Create a 2D graph by plotting multiple data series on the same axes

```
y = sin(x)
y = cos(x)
Y = tan(x)
```



- Make sure to add x and y axis labels, figure title, legend, and all 3 functions should be a **different** marker, style and color
- Use x range from 0 to 4*pi, with iterations of pi/12
- Use plotting axis range for x-axis from 0 to pi*2 and y-axis from -2*pi to 2*pi
- Make sure your code is **commented** properly to explain your code

Marker	Description
o	Circle
+	Plus sign
*	Asterisk
.	Point
x	Cross
s	Square
d	Diamond
^	Upward-pointing triangle
v	Downward-pointing triangle
>	Right-pointing triangle
<	Left-pointing triangle
p	Pentagram
h	Hexagram

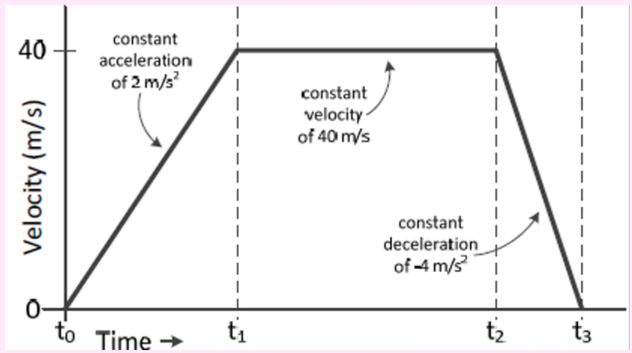
Line Style	Description
-	Solid line (default)
--	Dashed line
:	Dotted line
-.	Dash-dot line

Color	Description
y	yellow
m	magenta
c	cyan
r	red
g	green
b	blue
w	white
k	black

TASK 2 - SCRIPT

A car starts at rest, and follows the following pattern:

- Constant acceleration 2 m/s^2 until the velocity reaches 40 m/s
- Then, constant velocity until the displacement reaches 800 m
- Then, constant deceleration of -4 m/s^2 until rest



In MATLAB:

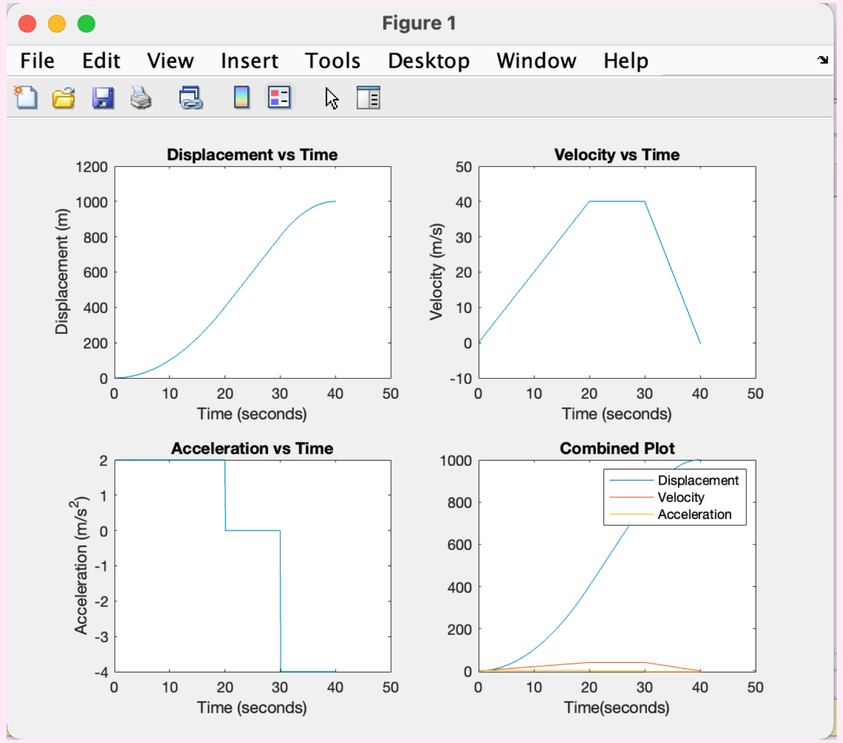
- Create arrays of displacement, velocity, acceleration and time using a time step of 0.1 seconds
- Once the arrays are created, use the plot function to draw graphs of displacement, velocity and acceleration over time

Formulas to use:

- Final velocity = Initial velocity + acceleration * time
- Displacement = velocity * time

Your Code:

- Define variables to be used within the code
- Initialize 3 arrays for displacement, velocity and acceleration
- 3 while loops (one for each part of the graph), each loop should calculate velocity, displacement, acceleration and time as needed and save the values to their respective arrays
- Each of the arrays (velocity, displacement & acceleration) should be plotted against the time array and displayed in 1 figure window (use subplot to create a 2x2 or 3x1 figure window)
- Make sure your code is **commented**



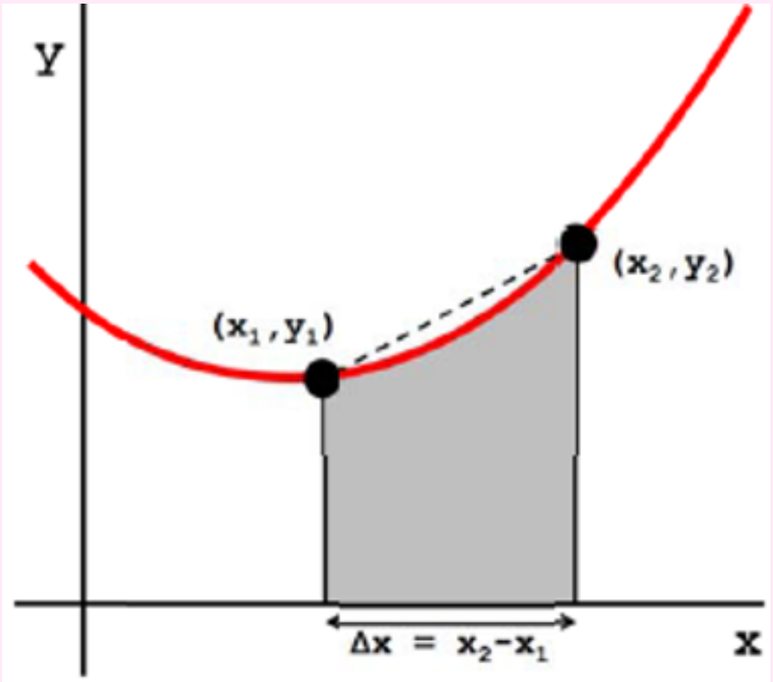
Creating Functions

- Functions allow for easy re-use of code by applying the same operations to many values
- You have already used built-in functions in MATLAB like sqrt() where you pass in a value and get a value back
- When a function is called, the variables in the workspace are separate from the variables inside a function, unless those variables are passed to or from the function as inputs
- Functions cannot be called by clicking “Run” in MATLAB, because inputs have to be defined. Instead, a function must be called in the command window/script using function name & parentheses
- The input and output variable names can be different inside and outside the function, only the position matters
- Functions can also be called from other functions
- Function syntax is:
  - If your function gives an output value: `function [output1, output2, …] = function_name (input1, input2, …)`
  - If your function does not give an output value: `function function_name(input1, input2, …)`

TASK 3 - SCRIPT

- Create a function called **trapArea** that estimates the area under a curve bounded by two points (x1,y1) and (x2, y2). Estimate the area by computing the area of the trapezium as shown
- The function should have four inputs **(x1, y1, x2, y2)** and one output called **area.** Make sure you test your function to ensure it works as expected
- Call the function inside your script with the test values below
- Formula to use for area of trapezoid: area = **height/2 * (side A + side B)** or in this case **area = (x2-x1)/2 * (y1+y2)**
- **Test Values: (1,3,4,7)**
- Results:

```
The area under the curve is: 15.00
```



TASK 4 (BONUS) - SCRIPT

- Create a second function called areaUnderCurve that will call your **trapArea** function created in Task 3 several times in a loop
- Your new function should have two inputs **(x,y)** which are both 1D arrays representing the x-axis and y-axis points on a curve.
- There should be one output called **totalArea**
- **Test Values: Use your new function to compute the below integrals:**

- $\int_0^3 x^2 dx = 9$
- $\int_{-\pi/2}^{\pi/2} \cos x \, dx = 2$