DOING PHYSICS WITH MATLAB

GETTING STARTED WITH MATLAB

ADVANCED XY PLOTTING TECHNIQUES

Ian Cooper
School of Physics, University of Sydney
ian.cooper@sydney.edu.au

DOWNLOAD DIRECTORY FOR MATLAB SCRIPTS

Advanced XY Plotting Techniques

To create a graph you will need to write your m-script that will perform some of the following functions.

1. Define and calculate the data to be plotted.
2. Determine the types of plots to be made and prepare the figure windows to display graphs.
3. Decide how the data is to be displayed.
4. Select suitable axes limits and tick marks or grid.
5. Annotate the graph with a title, axes labels, legend, text and markers.
6. Copy a plot and pasting it into other applications
7. Save a plot to a file.

All of the above functions can be achieved by the use of high-level graphics commands and handle graphics. The m-scripts are designed to be like a template containing re-useable code. Minimum changes to an existing m-script will only be needed to create a new m-script for plotting another function.

Data to be plotted

To illustrate some of the ways in which you can enhance or change the appearance of your plot we will consider a number of illustrative examples.

Firstly, consider plotting the sinusoidal function

$$d = A_0 \sin(\frac{2\pi t}{T} + \phi) \qquad (1)$$

This function represents the oscillatory motion of an object moving up and down on the end of a spring. The variables to be plotted are *d* (Y-axis) and are *t* (X-axis) where *d* is the position of the object about an equilibrium position (*d* = 0) and *t* is the time. *A* is called the amplitude and is the maximum distance from the equilibrium position and *T* is the period of oscillation or the time for one complete oscillation. The constant ϕ is known as the initial phase angle. The quantity ($2\pi t/T + \phi$) is the phase and is measured in radians.

An m-script template will be developed in stages showing how you can fully annotate the plot of this sinusoidal function.

Exercise 1

Figure 1 shows a plot of the sinusoidal function given by equation 1 produced by the m-script ex1.m. Open Matlab and create your own m-script version of the m-script to plot the sinusoidal function. The code for ex1.m is listed at the end.

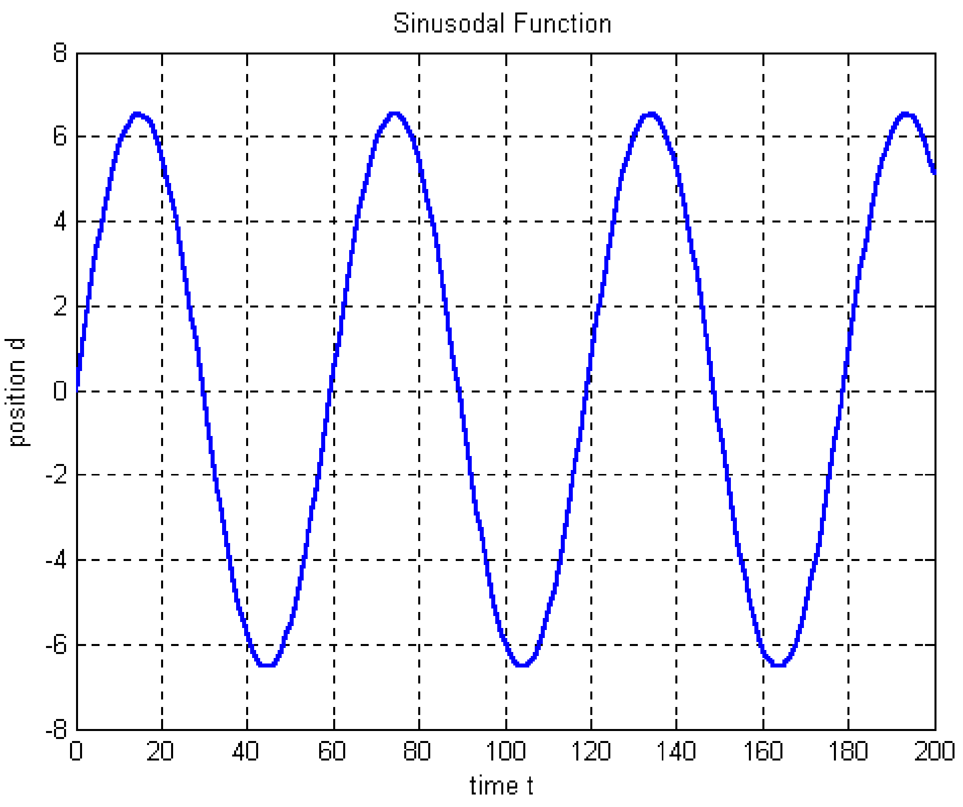

Fig. 1. Plot of sinusoidal function given by equation 1.

You will notice the m-script has been divided into the segments to give a structured format.

- *Starting comments* (Name of the m-script and general comments).
- *Setup* (**clear** clears all variables; **close all** closes all figure windows: **tic** starts a clock to record execution time).
- *Inputs* (Changes can be easily made of the constants in the program).
- *Initialize* (To increase the speed of execution of the m-script, array sizes can be set before any calculations are performed).
- *Calculations* (Calculations are done).
- *Graphics* (Instructions for the figure window used for plotting).
- *End* (**toc** prints execution time in seconds in command window and you can display any values in the command window)

Run your m-script and observe. Work through the m-script line by line so that you know what is happening.

GET and SET Commands and Handles

All the elements of a figure window possess a range of ***properties***. Each element can be though as having a ***handle*** attached to it. To view the properties assigned to an element, use **get** command and to change a property of an element use the **set** command.

GET(handle) displays all the property names and current property values for the graphics element associated with the handle.

GET(handle, 'Property Name') displays the current property values for property of the graphics element associated with the handle.

SET(handle, 'Property Name', 'Property Value') changes the values of the Property associated with the handle.

Exercise 2

Run the m-script ex1.m for *Exercise 1* used for plotting the sinusoidal function (ex1.m)

$$d = A_0 \sin(\frac{2\pi t}{T} + \phi)\,. \qquad (3.1)$$

Then type the following commands in the Matlab command window:

get(gcf) to view the current property values for the figure window

get(gca) to view the current t current axes properties.

Look through the list of properties for the elements **gcf** (current figure) and **gca** (current axes).

Color

Color is an important aspect of visualising data graphically with Matlab. A lookup table called colormap is used to determine the color of the elements of a figure window. A colormap matrix may have any number of rows, but it must have exactly 3 columns. Each row is interpreted as a color, with the first column specifying the intensity of red light, the second green, and the third blue. Each color is specified by a three element vector of RGB values ranging from 0 to 1, for example:

red [1 0 0]	green [0 1 0]	blue [0 0 1]
black [0 0 0]	white [1 1 1]	gray [.5 .5 .5]

The color of an element is most often specified by a single letter or the name of the color

Primary Colors	**Other Colors**
b blue	c cyan
g green	m magenta
r red	y yellow
	k black
	w white

Figure window properties with get(gcf) and set(gcf)

Some of the properties and their values are given in the following table for the figure window used for plotting your graph. Remember the property values are viewed in the command window with the command **get(gcf).** To change a property value use the command **set(gcf, 'Property Name', 'Property value')**.

Property Name	Property Values (default)
Color	**[r g b]** Controls the figure window background color.
MenuBar	(**figure**) **none** Display or hide the menu bar placed at the top of a figure window.
Name **NumberTitle**	'string' gives the name displayed at the top of the figure window. By default, the name is empty and the figure name is displayed as Figure No. 1, Figure No. 2, and so on. When you set this parameter to a string, the figure title becomes Figure No. 1: 'string'. (**on**) **off** Determines whether the string Figure No. N is prefixed to the figure window name.
Resize	(**on**) **off** Property determines if you can resize the figure window with the mouse. **on** means you can resize the window, **off** means you cannot. When Resize is off, the figure window does not display any resizing controls (such as boxes at the corners) to indicate that it cannot be resized.
Units	(**pixels**) **normalized inches centimeters points characters** Property specifies the units Matlab uses to interpret size and location data. All units are measured from the lower-left corner of the window. Normalized units map the lower-left corner of the figure window to (0,0) and the upper-right corner to (1.0,1.0). Inches, centimeters, and points are absolute units (one point equals 1/72 of an inch). The size of a pixel depends on screen resolution. Character units are defined by characters from the default system font; the width of one character is the width of the letter x, the height of one character is the distance between the baselines of two lines of text. If you change the value of Units, it is good practice to return it to its default value after completing your computation so as not to affect other functions that assume Units is set to the default value.
Position	**[left, bottom, width, height]** Specifies the size and location of the figure window by a four-element vector of the form: rect = [left, bottom, width, height] where left and bottom define the distance from the lower-left corner of the screen to the lower-left corner of the figure window. Width and height define the dimensions of the window. The size depends upon the Units specified.

Property Name	Property Values (default)
Visible	(**on**) **off** If the Visible property of a figure is off, the entire figure window is invisible.
BackingStore	(**on**) **off** **on** – improves speed of drawing plot. **off** – reduces system memory requirements and can increase the speed of animations because it eliminates the need to draw into both an off-screen buffer and the figure window.
PaperOrientation	(**portrait**) **landscape** Determines how printed figures are oriented on the page. **portrait** orients the longest page dimension vertically and **landscape** orients the longest page dimension horizontally.
PaperUnits	**normalized** (**inches**) **centimeters points** Hardcopy measurement units. This property specifies the units used to define the PaperPosition and PaperSize properties. All units are measured from the lower-left corner of the page. Normalized units map the lower-left corner of the page to (0, 0) and the upper-right corner to (1.0, 1.0). Inches, centimeters, and points are absolute units (one point equals 1/72 of an inch). If you change the value of PaperUnits, it is good practice to return it to its default value after completing your computation so as not to affect other functions that assume.
PaperPosition	**[left bottom width height]** Location on printed page. A rectangle that determines the location of the figure on the printed page. Specify this rectangle with a vector of the form: rect = [left, bottom, width, height] where left specifies the distance from the left side of the paper to the left side of the rectangle and bottom specifies the distance from the bottom of the page to the bottom of the rectangle. Together these distances define the lower-left corner of the rectangle. Width and height define the dimensions of the rectangle. The PaperUnits property specifies the units used to define this rectangle.
PaperPositionMode	(**manual**) **auto** WYSIWYG printing of figure. In manual mode, value is specified by the **PaperPosition** property. In auto mode, prints the figure the same size as it appears on the computer screen, centered on the page.
PaperType	(**usletter**) **A4** etc Selection of standard paper size.

Many of the above properties can be changed. For example the caption at the top of the figure window can be changed to

Figure No. 1: Sinusoidal Function with the command

```
fig = figure('Name','Sinusoidal Function')
```

Sinusoidal Function with the command

```
fig = figure('Name','Sinusoidal Function','NumberTitle','off')
```

Alternatively the property values can be changed using the set command:

```
set(gcf,'Name','Sinusoidal Fucntion')
set(gcf,'NumberTitle','off')
```

Exercise 2

Open the m-script ex1.m and save it immediately as ex3.m. Add to your m-script commands to format the figure window:

- Display "Sinusoidal Function" as the name of the figure window.
- Change the background color of the figure window.
- Change the size and position of the figure window.
- Format so that plot can be plotted on A4 paper.

A sample m-script ex3_3.m is at the end.

Axes properties with get(gca) and set(gca)

Some of the properties and their attributes are given in the following table for the axes of your plot. Remember the property values are viewed in the command window with the command **get(gca).** To change a property value use the **set(gca, 'Property Name', 'Property value')**.

Property Name	Property Value (default)
Box	(**on**) **off** Property specifies whether to enclose the axes extent in a box.
Color	(**[1 1 1]**) **[r g b]** **none** Determines the color of the plot area. **None** means the axes is transparent and the figure color shows through.
Units	(**normalised**) **points** **pixels** **characters** The units used to interpret the Position property. All units are measured from the lower-left corner of the figure window. **Normalized** units map the lower-left corner of the figure window to (0,0) and the upper-right corner to (1.0, 1.0). Character units are defined by characters from the default system font; the width of one character is the width of the letter x, the height of one character is the distance between the baselines of two lines of text.
Position	**[left bottom width height]** Position of axes. A four-element vector specifying a rectangle that locates the axes within the figure window. The vector is of the form where **left** and **bottom** define the distance from the lower-left corner of the figure window to the lower-left corner of the rectangle. **Width** and **height** are the dimensions of the rectangle. All measurements are in units specified by the Units property.
Visible	(**on**) **off** Visibility of axes. Setting this property to off prevents axis lines, tick marks, and labels from being displayed.
FontAngle	(**normal**) **italic** Axes text normal or italic.
FontName	(**Helvetica**) The font family name specifying the font to use for axes labels.
FontSize	**[10]** The font size in points or in font units use for axes labels
FontUnits	(**points**) **normalized** **inches** **centimeters** **pixels** Units used to interpret the FontSize property. When set to normalized, Matlab interprets the value of FontSize as a fraction of the height of the axes. A normalized FontSize of 0.1 sets the text characters to a font whose height is one tenth of the axes height.
FontWeight	(**normal**) **bold** Normal or bold for axes text
GridLineStyle	(:) **-** **- -** **-.** **none** Line style used to draw grid lines. The line style is a string consisting of a character, in quotes, specifying solid lines (**-**), dashed lines (**--**), dotted lines(**:**), or dash-dot lines (**-.**). To turn on grid lines, use the grid command.

LineWidth	**[0.5]** linewidth in points Specifies the width of axes lines in points, of the X-, Y-, and Z-axis lines.
MinorGridLineStyle	(:) **-** **- -** **-.** **none**
TickLength	**[2DLength 3DLength]** A two-element vector specifying the length of axes tick marks. The first element is the length of tick marks used for 2-D views and the second element is the length of tick marks used for 3-D views. Specify tick mark lengths in units normalized relative to the longest of the visible X-, Y-, or Z-axis annotation lines.
TickDir	(**in**) **out** Direction of tick marks
Xcolor **Ycolor**	**[0 0 0]** Propert sets color of X and Y-axis lines
Xdir **Ydir**	(**normal**) reverse Direction of increasing X and Y values.
Xlabel **yLabel**	Axes labels
XaxisLocation	(**bottom**) **top** Property controls the display of the X-axis tick marks and labels. Setting this property to **top** moves the X-axis to the top of the plot from its default position at the bottom.
YaxisLlocation	(**bottom**) **right** Property controls the display of the Y-axis tick marks and labels. Setting this property to **right** moves the Y-axis to the right of the plot from its default position at the left.
Xlim **Ylim**	**[minimum maximum]** Axis limits. A two-element vector specifying the minimum and maximum values of the respective axis.
XminorGrid **YminorGrid**	(**off**) **on** Enable or disable minor gridlines
XminorTick **YminorTick**	(**off**) **on** Enable or disable minor tick marks
Xscale **Yscale**	(**linear**) **log** Linear or logarithmic scaling for the respective axis.
Xtick **Ytick**	[ ] A vector of X and Ydata values that determine the location of tick marks along the respective axis. If you do not want tick marks displayed, set the respective property to the empty vector, [ ]. These vectors must contain monotonically increasing values.
XtickLabel **YtickLabel**	[ ] string A matrix of strings to use as labels for tick marks along the respective axis. These labels replace the numeric labels generated by Matlab. If you do not specify enough text labels for all the tick marks, Matlab uses all of the labels specified, then reuses the specified labels. For example, the statement, set(gca,'XTickLabel',{'One';'Two';'Three';'Four'}) labels the first four tick marks on the X-axis and then reuses the labels until all ticks are labeled.

Text displayed on graph

A graph can be annotated with the Matlab commands **title**, **xlabel**, **ylabel**, **text** and **gtext**. The command **line** adds a straight line to the graph.

Property	Comment
title	**title('string')** adds string at the top of the current axis. **title('string','Property1',PropertyValue1,'Property2',PropertyValue2,...)** Sets the values of the specified properties of the title. **h_title = title(...)** returns the handle to the text object used as the title.
xlabel	**xlabel('string')** adds string beneath X-axis on the current axis. **xlabel('string','Property1',PropertyValue1,'Property2',PropertyValue2,...)** Sets the values of the specified properties of the X-axis title. **h_xlabel = xlabel(...)** returns the handle to the text object used as the X-axis title.
ylabel	**ylabel('string')** adds string on left of Y-axis on the current axis. **ylabel('string','Property1',PropertyValue1,'Property2',PropertyValue2,...)** Sets the values of the specified properties of the Y-axis title. **h_ylabel = ylabel(...)** returns the handle to the text object used as the X-axis title.
text	**text(x,y,'string')** adds the text in the quotes to location (x,y) on the current axes, where (x,y) is in units from the current plot. If x and y are vectors, **text** writes the string at all locations given. If 'string' is an array the same number of rows as the length of x and y, **text** marks each point with the corresponding row of the 'string' array.
gtext	**gtext('string')** displays cross-hair in the figure window and waits for a mouse button or keyboard key to be pressed. The cross-hair can be positioned with the mouse (or with the arrow keys on some computers). Pressing a mouse button or any key writes the text string onto the graph at the selected location. **gtext(C)** places the multi-line strings defined by each row of the cell array of strings C. **gtext(...,'PropertyName',PropertyValue,...)** sets the value of the specified text property. Multiple property values can be set with a single statement.

You can change the appearance of the text displayed on the graph with the commands in the table below.

Property	Properties values (default)
FontName	eg (**Helvetica**), **Arial**, **Times**
FontUnits	(**points**), **normalized**, **inches**, **centimeters**, **pixels**
FontSize	(**12**)
FontWeight	(**normal**), **bold**, **light**, **demi**
FontAngle	(**normal**), **italic**, **oblique**
Color	(**k**), **b**, **g**, r, **c**, **m**, **y**, **w** or by the array **[r g b]** where **r**, **g** and **b** are numbers from 0 to1.
BackGroundColor	(**none**), **k**, **b**, **g**, r, **c**, **m**, **y**, **w** or by the array **[r g b]** where **r**, **g** and **b** are numbers from 0 to1. Color for the rectangle that encloses the text.
EdgeColor	(**none**), **k**, **b**, **g**, r, **c**, **m**, **y**, **w** or by the array **[r g b]** where **r**, **g** and **b** are numbers from 0 to1. Color of the rectangle's edge.
LineStyle	(:) **-** **- -** **-.** Style of the rectangle's edge line (first set EdgeColor).
LineWidth	**[0.5]** linewidth in points Width of the rectangle's edge line (first set EdgeColor).
Margin	???? Increase the size of the rectangle by adding a margin to the existing text extent.
HorizontalAlignment	(**left**), **center**, **right**
Rotation	(**0**) Text orientation of the text string. Specify values of rotation in degrees with positive angles giving a counter-clockwise rotation.
Line	**line(x,y)** adds the line in vectors X and Y to the current axes. **h_line = line(x,y)** return the handle to the line and can be used to set the property values for the line.

When the text Interpreter property is set to Tex (the default), you can use a subset of TeX commands embedded in the string to produce special characters such as Greek letters and mathematical symbols. The following table lists these characters and the character sequences used to define them.

String modifies

You can also control the font used by string modifiers. The first four modifiers are mutually exclusive. However, you can use \fontname in combination with one of the other modifiers:

\bf	bold font
\it	italics font
\sl	oblique font (rarely available)
\rm	normal font
\fontname{fontname}	name of the font family to use
\fontsize{fontsize}	font size in FontUnits.

String modifiers remain in effect until the end of the string or only within the context defined by braces { }. The subscript character "_" and the superscript character "^"

modify the character or sub-string defined in braces immediately following. To print the special characters used to define the Tex strings when Interpreter is Tex, prefix them with the backslash "\" character.

Character Sequence	**Symbol**	**Character Sequence**	**Symbol**	**Character Sequence**	**Symbol**
\alpha	α	\beta	β	\gamma	γ
\delta	δ	\epsilon	ε	\zeta	ζ
\eta	η	\theta	θ	\iota	ι
\kappa	κ	\lambda	λ	\mu	µ
\nu	ν	\xi	ξ	\omicron	ο
\pi	π	\rho	ρ	\sigma	σ
\tau	τ	\upsilon	υ	\phi	ϕ
\chi	χ	\psi	ψ	\omega	ω
\vartheta	υ	\varsigma	ς	\varpi	ϖ
\Alpha	Α	\Beta	Β	\Gamma	Γ
\Delta	Δ	\Epsilon	Ε	\Zeta	Ζ
\Eeta	Η	\Theta	Θ	\Iota	Ι
\Kappa	Κ	\Lambda	Λ	\Mu	Μ
\Xi	Ξ	\Omicron	Ο	\Pi	Π
\Rho	Ρ	\Sigma	Σ	\Tau	Τ
\Upsilon	T	\Phi	Φ	\Chi	Χ
\Psi	Ψ	\Omega	Ω		
\Im	ℑ	\Re	ℜ	\wp	℘
\aleph	ℵ				
\otimes	⊗	\oplus	⊕	\oslash	∅
\cap	∩	\cup	∪	\supseteq	⊇
\supset	⊃	\subseteq	⊆	\subset	⊂
\in	∈	\exists	∃	\forall	∀
\int	∫	\surd	√	\partial	∂
\times	×	\div	÷	\bullet	∗
\propto	∝	\geq	≥	\leq	≤
\pm	±	\circ	°	\infty	∞
\sim	~	\equiv	≡	\approx	≈
\neq	≠		\ni	∋	
\downarrow	↓	\uparrow	↑		
\leftarrow	←	\rightarrow	→	\leftrightarrow	↔
\spadesuit	♠	\heartsuit	♥	\diamondsuit	♦
\clubsuit	♣				
\rfloor	⌋	\lceil	⌈	\lfloor	⌊
\rceil	⌉	\mid	|		
\wedge	∧	\vee	∨	\prime	'
\langle	>	\rangle	<	\copyright	©
\perp	⊥	\neg	¬		
\cdot	.	\ldots	…		

ANNOTATING A PLOT

The plot of the sinusoidal function can be annotated, by adding a title, axes labels and text comments.

For the title, we want to display the equation (3.1) and the numerical values for the amplitude A_0 in millimeters and the period T in seconds.

$$d = A_0 \sin(\frac{2\pi t}{T} + \phi) \qquad \text{where } A = 6.55 \text{ mm} \quad T = 59.561 \text{ s}$$

The string for the title is made from an array for each component of the title and we have to convert the numbers representing the amplitude and period into strings with the number to string command, **num2str**.

strX = num2str(X) converts the matrix, X into a string, strX
e.g. **X = pi = 3.141592654…** **strX** = **num2str(X)** = 3.1416

strX = num2str(X,N) converts the matrix, X into a string with a maximum N digits of precision
e.g. **X = pi** **strX** = **num2str(X,3)** = 3.14

strX = num2str(X,format) converts the matrix, X into a string with a format specified by the **sprintf** command.

The table below shows how π can be displayed in different formats with the **sprintf** command.

Command	Result	Command	Result
sprintf('% .0e \n',pi)	3e+00	**sprintf('% .3e \n',pi)**	3.142e+00
sprintf('% .0f \n',pi)	3	**sprintf('% .4f \n',pi)**	3.1416
sprintf('% .0g \n',pi)	3	**sprintf('% .3g \n',pi)**	3.14

\n line termination character.

e exponential notation (number to right of decimal point gives number of digits displayed after the decimal point in the number).

f fixed-point notation (number to right of decimal point gives number of digits displayed after the decimal point in the number).

g general notation: fixed-point or exponential notation, whichever is shorter (number to right of decimal point gives the total number of digits displayed).

e.g. **X = pi** **strX** = **num2str(X,'% .5f \n'**) = 3.14159

Exercise 4

Open your m-script ex3.m and immediately save it as ex4.m. The m-script can be extended to annotate the graph of the sinusoidal function as shown in figure 1 with:

- Title with equation and numerical values for A and T.
- Axes labelled.
- Lines and text added to show amplitude and period.
- Numerical values displayed for angular velocity ω and frequency *f*.

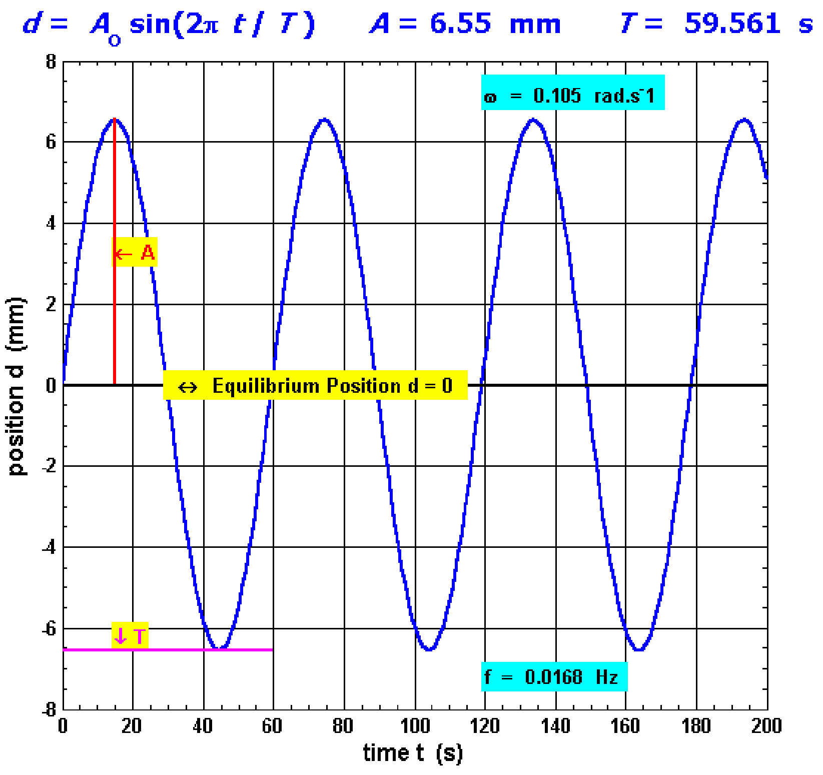

Fig. 2. Annotated plot of sinusoidal function of equation (3.1).

The complete m-script ex4.m for figure 1 is at the end of this Chapter. Work through it line by line making sure you understand each part of the code. The handles to figure window elements are given by the prefix **h_** in the code. Change the program and observe the figure window to test your understanding code.

Data displayed on the graph and the plot window

The way in which data is displayed on the graph is through the line properties. You can change the appearance of the plotted data using the following Matlab commands.

Property	Properties values (default)
Color	(**k**), **b**, **g**, **r**, **c**, **m**, **y**, **w** or by the array **[r g b]** where **r**, **g** and **b** are numbers from 0 to1
LineStyle	(**-**) solid line, **--** dashed line, **:** dotted line , **-.** dash-dot line, **none** no line
LineWidth	Width of line in points, 0.5 is the default.
Marker	(**none**) + plus sign, **o** circle, * asterisk, **.** point, **x** cross, **s** square, **d** diamond, **^** upward pointing triangle, **v** downward pointing triangle, **>** right pointing triangle, **<** left pointing triangle, **p** five-pointed star (pentagram), **h** six-pointed star (hexagram). Marker specifies the appearance of the data points.
MarkerEdgeColor	**none**, (**auto**), (**k**), **b**, **g**, **r**, **c**, **m**, **y**, **w** or by the array **[r g b]** where **r**, **g** and **b** are numbers from 0 to1. The color of the marker or the edge color for filled markers (circle, square, diamond, pentagram, hexagram, and the four triangles). Auto sets **MarkerEdgeColor** to the same color as the line's **Color** property.
MarkerFaceColor	(**none**), **auto**, (**k**), **b**, **g**, **r**, **c**, **m**, **y**, **w** or by the array **[r g b]** where **r**, **g** and **b** are numbers from 0 to1 The fill color for markers that are closed shapes (circle, square, diamond, pentagram, hexagram, and the four triangles). **None** makes the interior of the marker transparent, allowing the background to show through. **Auto** sets the fill color to the axes color, or the figure color, if the axes **Color** property is set to none (which is the factory default for axes).
MarkerSize	Size of marker in points.
EraseMode	(**normal**), **none**, **xor**, **background**. Controls the technique Matlab uses to draw and erase line objects. Alternative erase modes are useful for creating animated sequences, where control of the way individual objects redraw is necessary to improve performance of the desired effect..

The way in which the axes are displayed and the scaling are controlled through the **axis** command and the size of the plot window through the **axes** command

Property	Comments
axis([xmin xmax ymin ymax])	Sets scaling for the X- and Y-axes on the current plot.
axis auto	Returns the axis scaling to its default, automatic mode where, for each dimension, 'nice' limits are chosen based on the extents of all objects.
axis manual	Freezes the scaling at the current limits, so that if **hold on** is used, subsequent plots will use the same limits.
axis tight	Sets the axis limits to the range of the data.
axis fill	Sets the axis limits and **PlotBoxAspectRatio** so that the axis fills the position rectangle. This option only has an effect if **PlotBoxAspectRatioMode** or **DataAspectRatioMode** are manual.
axis IJ	Puts Matlab into its "matrix" axes mode. The coordinate system origin is at the upper left corner. The I-axis is vertical and is numbered from top to bottom. The J- axis is horizontal and is numbered from left to right.
axis XY	Puts Matlab into its default "Cartesian" axes mode. The coordinate system origin is at the lower left corner. The X-axis is horizontal and is numbered from left to right. The Y-axis is vertical and is numbered from bottom to top.
axis equal	Sets the aspect ratio so that equal tick mark increments on the X and Y axes are equal in size. This makes a circle look like a circle instead of an ellipse and a square, a square, not a rectangle.
axis image	The same as axis equal except that the plot box fits tightly around the data.
axis square	Makes the current axis box square in size.
axis normal	Restores the current axis box to full size and removes any restrictions on the scaling of the units. This undoes the effects of axis square and axis equal.
axis on **axis off**	turns axis labeling, tick marks,grid and background back on. turns off all axis labeling, tick marks, grid and background.
axis(H,...)	Changes the axes handles listed in vector H. ???
axes **axes('position', rect)**	Create axes in arbitrary positions. Opens up an axis at the specified location and returns a handle to it. **Axes**, by itself, creates the default full-window axis and returns a handle to it. rect = [left, bottom, width, height] specifies the location and size of the side of the axis box, relative to the lower-left corner of the Figure window, in normalized units where (0,0) is the lower-left corner and (1.0,1.0) is the upper-right. **axes(H)** makes the axis with handle H current. Execute **get(H)** to see a list of axes object properties and their current values

Exercise 5

Lissajous figures are created when two sinusoidal functions are plotted against each other. Let the two sinusoidal functions (called wave functions) be given by equations (2) and (3)

$$y_1 = A_1 \sin(\omega_1 t + \phi_1)\,, \tag{2}$$

$$y_2 = A_2 \sin(\omega_2 t + \phi_2)\,, \tag{3}$$

where $\omega = 2\pi$ f = $2\pi/T$. Changing the values assigned to both the frequencies f_1 and f_2 and the phases ϕ_1 and ϕ_2 produces interesting changes to the Lissajous figure.

Create two figure windows for the plots

Figure	X Data	Y Data
1: Sinusoidal Functions	t	y_1
2: Lissajous Figure	y_1	y_2

You can try varying some of the following:

- Figure window properties.
- Axis properties. The Lissajous plot must have **axis** set to **square** so that a circle appears as a circle.
- Line properties.
- Marker properties.
- Location of tick marks along axes.
- Annotation of the plot.

  Figures 3 shows a plot for the wave functions against time and figure 4, the wave functions plotted against each other.

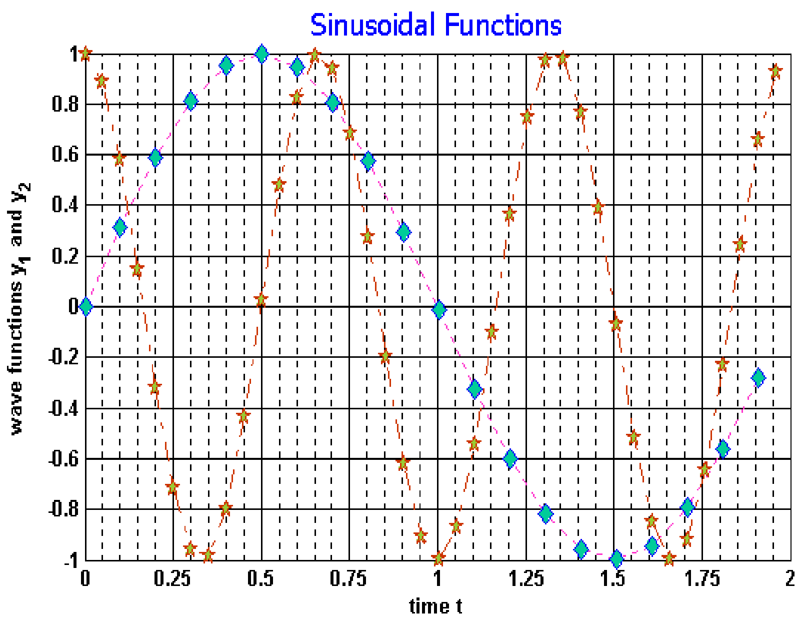

Fig. 3. Plot of wave functions against time for equation (2) and (3).

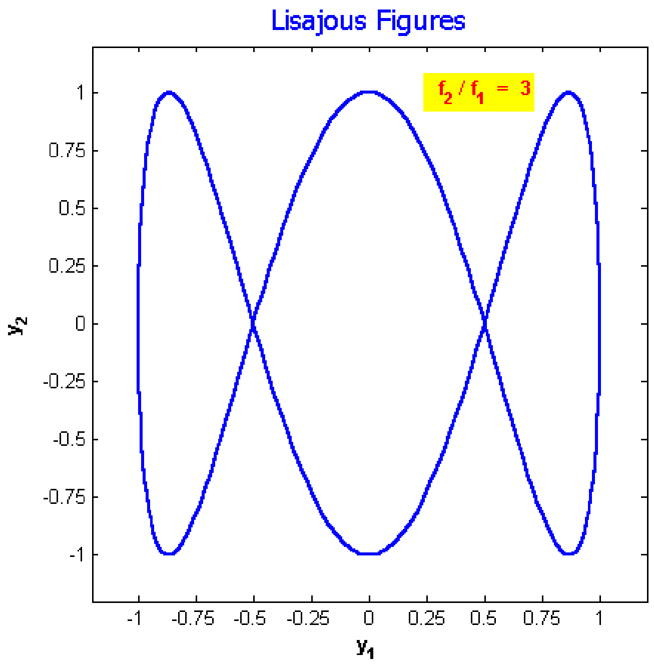

Fig. 4. Plot of wave function: y_1 against y_2.

Sample code ex5.m for the plots in figures 3 an 4 is found at the end of this Chapter.

Chapter 3 m-scripts

m-script ex1.m for Exercise 1

```
% ex1.m
% m-script for Exercise 1
% Plot sinusoidal function given by Eq.(1)
%
% setup -------------------------------------------------------------
clear                      % clears all variables
close all                  % closes all figure windows
tic                        % starts clock

% x ---> d (position)
% y ---> t (time)

% inputs -------------------------------------------------------------
A = 6.54760;                % amplitude
T = 59.56100;               % period

n = 200;                    % number of data points to be plotted
minx = 0;                   % first point for x to be plotted
maxx = 200;                 % last point for x to be plotted

% initialise ---------------------------------------------------------
x = zeros(n,1);             % setup array for x data
y = zeros(n,1);             % setup array for y data

x = linspace(minx,maxx,n);     % n equally spaced points for x data

% calculations -------------------------------------------------------
w = 2*pi/T;                 % angular frequency
f = 1/T;                    % frequency

y = A.*sin(w.*x);            % array for y data

% graphics -----------------------------------------------------------

% open figure window
figure(1)                   % open figure window for plotting
tm = 'Sinusodal Function';     % Main title
tx = 'time t';               % X-axis title
ty = 'position d'             % Y-axis title

plot(x,y,'LineWidth',2)        % XY plot

title(tm)
xlabel(tx)
ylabel(ty)
grid on

% end ----------------------------------------------------------------
toc                   % execution time in seconds shown in command window
```

m-script ex3.m for Exercise 3

First part same as m-script 1

```
% graphics --------------------------------------------------------

%open and format figure window
figure(1)                    % open figure window for plotting
set(gcf,'Name','Sinusoidal Function')
set(gcf,'NumberTitle','off')
set(gcf,'Color',[0.8 0.8 0])
set(gcf,'Units','Normalized')
set(gcf,'Position',[0.2 0.2 0.6 0.6])
set(gcf,'PaperType','A4')

%plot titles
tm = 'Sinusodal Function';     % Main title
tx = 'time t';                 % X-axis title
ty = 'position d'              % Y-axis title

%plot
plot(x,y,'LineWidth',2)        % XY plot

title(tm)
xlabel(tx)
ylabel(ty)
grid on


% end ----------------------------------------------------------------
toc                        % execution time in seconds shown in command window
```

---

m-script ex4.m for Exercise 4

```
% ex3_1.m
% m-script for Exercise 3.1
% Plot sinusoidal function given by Eq.(1)
%
% setup --------------------------------------------------------------
clear                        % clears all variables
close all                    % closes all figure windows
tic                          % starts clock

% x ---> d (position)
% y ---> t (time)

% input constants

% inputs -------------------------------------------------------------
A = 6.54760;                 % amplitude
T = 59.56100;                % period

n = 200;                     % number of data points to be plotted
minx = 0;                    % first point for x to be plotted
maxx = 200;                  % last point for x to be plotted

% initialise -------------------------------------------------------
x = zeros(n,1);              % setup array for x data
```

```
y = zeros(n,1);                 % setup array for y data

x = linspace(minx,maxx,n);     % n equally spaced points for x data

% calculations -----------------------------------------------------
w = 2*pi/T;                    % angular frequency
f = 1/T;                       % frequency

y = A.*sin(w.*x);              % array for y data

% graphics ---------------------------------------------------------

%open and format figure window
figure(1)                      % open figure window for plotting
set(gcf,'Name','Sinusoidal Function')
set(gcf,'NumberTitle','off')
set(gcf,'Color',[0.8 0.8 0])
set(gcf,'Units','Normalized')
set(gcf,'Position',[0.2 0.2 0.6 0.7])
set(gcf,'PaperType','A4')


%plot
plot(x,y,'LineWidth',2)        % XY plot
set(gca,'FontWeight','b')
set(gca,'GridLineStyle','-');
set(gca,'XminorTick','on');
set(gca,'YminorTick','on');
grid on

%plot titles
tm1 = '{\it d} = {\it A}_o sin(2\pi{\it t} /{\it T} )';
tm2 = '     {\it A} = '
tm3 = num2str(A,3);
tm4 = '  mm';
tm5 = '     {\it T} =  ';
tm6 = num2str(T,'% .3f \n');
tm7 = '  s';
tm = [tm1 tm2 tm3 tm4 tm5 tm6 tm7]; % string for main title

tx = 'time t  (s)';                % X-axis title
ty = 'position d  (mm)'              % Y-axis title

h_title = title(tm);
set(h_title,'FontSize',14,'Color','b','FontName','Tahoma') % main plot title

h_xlabel = xlabel(tx)                  % title for X axis
h_ylabel = ylabel(ty)                  % title for Y axis
set(h_xlabel,'FontSize',12,'FontWeight','b')
set(h_ylabel,'FontSize',12,'FontWeight','b')

%plot annotation: amplitude
h_text = text(T/4,A/2,'\leftarrow A');
set(h_text,'FontWeight','b','Color','r','BackGroundcolor','y' );
h_line = line([T/4 T/4],[0 A]);
set(h_line,'Color','r','LineWidth',2);

%plot annotation: period
h_text = text(T/4,-0.95*A,'\downarrow T')
set(h_text,'FontWeight','b','Color','m','BackGroundcolor','y' );
```

```
h_line = line([0 T],[-A -A],'LineWidth',2)
set(h_line,'Color','m')

%plot annotation: equilibrium position
h_line = line([0 maxx],[0 0],'LineWidth',2)
set(h_line,'Color','k')
h_text = text(T/2,0,'  \leftrightarrow  Equilibrium Position d = 0  ')
set(h_text,'FontWeight','b','Color','k','BackGroundcolor','y' );

%plot annotation: angular velocity
str1 = '\omega  =  '
str2 = num2str(w,3)
str3 = '  rad.s^-1 '
h_text = text(0.60*maxx,1.1*A,[str1 str2 str3])
set(h_text,'FontWeight','b','Color','k','BackGroundcolor','c' );

%plot annotation: frequency
str1 = 'f  =  '
str2 = num2str(f,3)
str3 = '  Hz '
h_text = text(0.60*maxx,-1.1*A,[str1 str2 str3])
set(h_text,'FontWeight','b','Color','k','BackGroundcolor','c','Margin',2);

% end ---------------------------------------------------------------
toc                        % execution time in seconds shown in command window
```

m-script ex5.m for Exercise 5

```
%ex5.m

% m-script to plot Lisajous figues
%
% setup ---------------------------------------------------------------
clear                        % clears all variables
close all                    % closes all figure windows
tic                          % starts clock

% inputs --------------------------------------------------------------
n = 200;                     % number of data points to be plotted
tmin = 0;                    % first point to be plotted
tmax = 2;                    % last point to be plotted

A(1) = 1.0;                       % amplitude
A(2) = 1.0;

f(1) = 100;                    % period
f(2) = 300;

phi(1) = 0;
phi(2) = pi/2;

% initialise ----------------------------------------------------------
t = zeros(n,1);              % setup array for time t
y = zeros(2,n);              % setup array for position h


t = linspace(tmin,tmax,n);     % n equally spaced points for t from tmin to tmax

% calculations
% --------------------------------------------------------------------
w = 2*pi.*f;

for c = 1 : 2
y(c,:) = A(c).*sin(w(c).*t + phi(c));           % calculation of array for position h
end

% graphics ------------------------------------------------------------

% open figure window
figure(1)                    % open figure window for plotting
set(gcf,'Name','  Lisajous Figures')
set(gcf,'Units','Normalized','Position', [0.1 0.1 0.6 0.5])
set(gcf,'Color',[0.9 0.9 0.5])
set(gcf,'PaperType','A4')
h_plot1 = plot(t(1:10:n),y(1,1:10:n));                 % plot XY graph with line
                        and marker style S
set(h_plot1,'LineStyle',':','LineWidth',1,'Color',[1.0 0.2 0.8])
set(h_plot1,'Marker','d','MarkerSize',8,'MarkerFaceColor',[0 0.8 0.6])
set(h_plot1,'MarkerEdgeColor','k')

hold on
h_plot2 = plot(t(1:5:n),y(2,1:5:n));
set(h_plot2,'LineStyle','-.','LineWidth',1,'Color',[0.8 0.2 0])
set(h_plot2,'Marker','p','MarkerSize',8,'MarkerFaceColor',[0.6 0.8 0.2])
set(h_plot1,'MarkerEdgeColor','b')
```

```
% Plot titles
%plot title
tm = 'Sinusoidal Functions';
h_title = title(tm);
set(h_title,'FontSize',14,'Color','b','FontName','Tahoma') % main plot title

%plot labels for axes
tx = 'time t';                % string for X axis title
ty = 'wave functions y_1 and y_2' ;            % string for Y axis title
xlabel(tx,'FontWeight','b');                   % title for X axis
ylabel(ty,'FontWeight','b')                    % title for Y axis
%
% Plot axes
set(gca,'FontWeight','b','Color',[0.9 0.9 0.9])
set(gca,'GridLineStyle','-')
set(gca,'Xtick',[0:0.25:tmax])
set(gca,'XMinorGrid','on')

%

grid on                       % toggle grid on or off: grid on or grif off

% open figure window
figure(2)                     % open figure window for plotting
set(gcf,'Name','  Lisajous Figures')
set(gcf,'Units','Normalized','Position', [0.1 0.1 0.6 0.6])
set(gcf,'Color',[0.9 0.9 0.7])
set(gcf,'PaperType','A4')
plot(y(1,:),y(2,:),'LineWidth',2)
axis equal
axis([-1.2 1.2 -1.2 1.2])
set(gca,'Xtick',[-1 : 0.25 : 1])
set(gca,'Ytick',[-1 : 0.25 : 1])

% Plot titles
%plot title
tm = 'Lisajous Figures';
h_title = title(tm);
set(h_title,'FontSize',14,'Color','b','FontName','Tahoma') % main plot title
%
%plot labels for axes
tx = 'y_1';                % string for X axis title
ty = 'y_2' ;             % string for Y axis title
xlabel(tx,'FontWeight','b');                   % title for X axis
ylabel(ty,'FontWeight','b')                    % title for Y axis
%
% plot annotation
str1 = '  f_2 / f_1  =  ';
str2 = num2str(f(2)/f(1),1);
str = [str1 str2]
h_text = text(0.25*A(1),1.0*A(2),str);
set(h_text,'FontWeight','b','Color','r','BackGroundcolor','y' );

axis on
% %plot annotation: period
% h_text = text(T/4,-0.95*A,'\downarrow T')
% set(h_text,'FontWeight','b','Color','m','BackGroundcolor','y' );
% h_line = line([0 T],[-A -A],'LineWidth',2)
% set(h_line,'Color','m')
%
```

```
% %plot annotation: equilibrium position
% h_line = line([0 tmax],[0 0],'LineWidth',2)
% set(h_line,'Color','k')
% h_text = text(T/2,0,'  \leftrightarrow  Equilibrium Position d = 0  ')
% set(h_text,'FontWeight','b','Color','k','BackGroundcolor','y' );
%
% %plot annotation: angular velocity
% str1 = '\omega  =  '
% str2 = num2str(w,3)
% str3 = '  rad.s^-1 '
% h_text = text(0.75*tmax,1.0*A,[str1 str2 str3])
% set(h_text,'FontWeight','b','Color','k','BackGroundcolor','c' );
%
% %plot annotation: frequency
% str1 = 'f  =  '
% str2 = num2str(f,3)
% str3 = '  Hz '
% h_text = text(0.6*tmax,-1.0*A,[str1 str2 str3])
% set(h_text,'FontWeight','b','Color','k','BackGroundcolor','c' );
%
%


% end ---------------------------------------------------------------
toc
```