

Mathematica versus MATLAB Refresher*

Samuel A. Hambleton

This is a short Mathematica tutorial on plotting functions in Mathematica for those who may have used MATLAB. I have intentionally copied verbatim text from MATLAB Refresher * (adapted from material developed by Mrs Alise Fox) so that you may compare the syntax and style. In the pdf version I have merged the two pdfs.

Place your mouse cursor at the end of the line 1+1 below and left-click. Then press Shift and Enter at the same time.

In[60]:= **1 + 1**

1. Plotting data points

Say we wish to plot the following coordinates:

(0, 1), (3, 2), (8, 3), (15, 4), (24, 5).

One method is to enter the x-values as a vector, and then the y-values, and use the inbuilt Mathematica function to plot them against each other. To do this, copy the following lines into Mathematica, hitting Shift and Enter at the same time.

```
(* this creates a vector for the x-values *)
x = {0, 3, 8, 15, 24};
(* this creates a vector for the y-values *)
y = {1, 2, 3, 4, 5};
(* this forms the list of points {{0,1},{3,2},{8,3},{15,4},{24,5}} *)
A = Transpose[{x, y}]
(* this plots the list of points without any options other
   than to join the points with line segments as MATLAB does *)
ListPlot[A, Joined -> True]
```

With the above option Joined -> True, the ListPlot function connects the points with a straight line. Copy the following line into Mathematica to specify that points are plotted instead of lines:

```
ListPlot[A]
```

You can then add axis labels and titles to your graph.

In[81]:= **ListPlot[A, AxesLabel -> {"x values", "y values"},
PlotLabel -> "A plot of the provided data"]**

You can also change the colour and shape of the data points by editing the input at the end of the plot

function. Copy and paste each instruction one at a time to see the effect.

```
(* this makes the points turn black *)
ListPlot[A, PlotStyle → Black]

(* this turns the points into red circles *)
ListPlot[A, PlotStyle → {Red}, PlotTheme → "Business"]
```

```
In[91]:= (* this turns the points into connected green circles *)
ListPlot[A, Joined → True, PlotStyle → {Green}, PlotTheme → "Business"]
```

2. Plotting functions

In some cases, you may need to plot a function over a continuous domain, instead of a set of points. In this case, Plot can be used. For example, to plot the function $y = \frac{1}{2}x^4 - x^2$ between $x = -5$ and $x = 5$, enter the following into Mathematica:

```
In[93]:= (* clears x and y from memory *)
ClearAll[x, y]
(* inputs the function x *)
f[x_] := (1/2) * x^4 - x^2;
(* plots the function that was previously assigned, over the desired domain *)
Plot[f[x], {x, -5, 5}]
```

You will notice that every time you enter the Plot command, it overrides the previous graph. You can avoid this by putting another Plot below this first one...

```
In[96]:= f[x_] := (1/2) * x^4 - x^2;
Plot[f[x], {x, -5, 5}]
g[x_] := 4 * x^3 - 3 * x^2 + 5 * x - 1;
Plot[g[x], {x, -5, 5}]
```

Let $y_1 = 4x^3 - 3x^2 + 5x - 1$, and y_2 , y_3 and y_4 be the subsequent derivatives. Work out each of the derivatives and enter them into Mathematica as functions. Start with

```
(* Now we enter a symbolic expression for y1 rather than a function of x *)
y1 = 4 * x^3 - 3 * x^2 + 5 * x - 1;
```

(Note that in Mathematica you should NOT enter the powers on the variables with a full stop between the x and the ^, to avoid getting a warning message.) Remember in MATLAB the . between the x and the ^ is necessary.

Plot the first function y_1 , between $x = -10$ and $x = 10$ by entering the following line:

```
In[103]:= Plot[y1, {x, -10, 10}]
```

Now we signify to Mathematica that we want to continue plotting on the same graph, by putting { and } around y_1 and any other curve we wish to plot on the same graph.

From here, we can add more graphs to our plot without it overriding the previous plot. So add the y_2

function to the plot:

```
(* takes the first derivative of y1 with respect to x *)
y2 = D[y1, x]
(* plots y1 and y2 on the same graph *)
Plot[{y1, y2}, {x, -10, 10}]
```

Continue by adding y3 and y4 to your plot.

You can add as many functions to the plot as you like, and Mathematica will continue to add them to the same graph.

Now add a title and axis labels to your graph:

```
In[108]:= Plot[{y1, y2}, {x, -10, 10}, PlotLabels → "Expressions", AxesLabel → {"x", "y(x)"}]
```

3. Plotting 3D graphs

Plotting 3-dimensional functions takes not much more effort because we can do this in the easy way in Mathematica using Plot3D or in a way more analogous to MATLAB. For the latter style, instead of just creating a function, we need to create an array (grid) of (x,y) points that we can enter into our function, which we then plot.

Say we want to plot the function $z = 2 - \sin(x^2) - \cos(y^2)$ between $-3 \leq x \leq 3$ and $-4 \leq y \leq 4$.

Begin by defining the function. Copy the following into the cell:

```
In[111]:= zfunction[x_, y_] := 2 - Sin[x^2] - Cos[y^2];
```

Now create vectors for both the x and y domains.

```
In[127]:= (* create a vector of x and y values, starting at x=-3, incrementing by 1,
and ending at 3, and y = -4, incrementing by 1, and ending at 4 *)
points = Table[{x, y, zfunction[x, y]}, {x, -3, 3, 1.0}, {y, -4, 4, 1.0}];
(* Now reshape to form triples *)
B = Flatten[points];
points3d = Partition[B, 3]
```

Plot the grid of x, y and z values together.

```
In[130]:= ListPointPlot3D[points3d]
```

You can increase the resolution of the graph by increasing the resolution of the grid. Repeat the process, but increment the x and y values by 0.05 instead of 1.

```
In[139]:= points = Table[{x, y, zfunction[x, y]}, {x, -3, 3, 0.05}, {y, -4, 4, 0.05}];
B = Flatten[points];
points3d = Partition[B, 3];
ListPointPlot3D[points3d]
```

Now the easy way of plotting 3D surfaces in Mathematica is as follows:

```
In[143]:= Plot3D[2 - Sin[x^2] - Cos[y^2], {x, -3, 3}, {y, -4, 4}]
```

Follow a similar process to graph $z = 2 - \frac{\sin(x^2)}{\exp(y)}$. Since the function has extreme values, part of the plot is chopped off.

```
In[146]:= Plot3D[2 - Sin[x^2] / Exp[y], {x, -5, 5}, {y, 0, 10}, PlotPoints -> 50]
```

MATLAB Refresher *

SESSION 1 (30 minutes)

1. Plotting data points

Say we wish to plot the following coordinates:

(0,1), (3,2), (8,3), (15,4) and (24,5).

One method is to enter the x -values as a *vector*, and then the y -values, and use the inbuilt MATLAB `plot` function to plot them against each other. To do this, copy the following lines into MATLAB, hitting <Enter> after each line.

```
x = [0,3,8,15,24]    % this creates a vector for the x-values
y = [1,2,3,4,5]      % this creates a vector for the y-values
plot(x,y)
```

Without specifying otherwise, the `plot` function automatically connects the points with a straight line. Copy the following line into MATLAB to specify that points are plotted instead of lines:

```
plot(x,y, ' .')
```

You can then add axis labels and titles to your graph.

```
xlabel('x values')
ylabel('y values')
title('A plot of the provided data')
```

You can also change the colour and shape of the data points by editing the input at the end of the `plot` function. Copy and paste each instruction one at a time to see the effect.

```
plot(x,y,'k.') % this makes the points turn black
plot(x,y,'rp') % this turns the points into red stars
plot(x,y,'go-') % this turns the points into connected green circles
```

2. Plotting functions

In some cases, you may need to plot a function over a continuous domain, instead of a set of points. In this case, `fplot` can be used. For example, to plot the function $y = \frac{1}{2}x^4 - x^2$ between $x = -5$ and $x = 5$, enter the following into MATLAB:

```
y = @(x) 1/2*x.^4 - x.^2 % inputs the anonymous function
fplot(y,[-5,5]) % plots the anonymous function that was previously
assigned, over the desired domain
```

You will notice that every time you enter the `plot` or `fplot` command, it overrides the previous graph. You can avoid this by using the `hold` function...

Let $y_1 = 4x^3 - 3x^2 + 5x - 1$, and y_2 , y_3 and y_4 be the subsequent derivatives. Work out each of the derivatives and enter them into MATLAB as anonymous functions. Start with

```
y1 = @(x) 4*x.^3 + 3*x.^2 + 5*x - 1
```

(Note that you should enter the powers on the variables with a full stop between the x and the $^$, to avoid getting a warning message.)

Plot the first function, y_1 , between $x = -10$ and $x = 10$ by entering the following line:

```
fplot(y1, [-10, 10])
```

Now we signify to MATLAB that we want to continue plotting on the same graph, by typing the words:
`hold on`

(You only have to type this once per graph.)

From here, we can add more graphs to our plot without it overriding the previous plot. So add the y_2 function to the plot:

```
fplot(y2, [-10, 10])
```

Continue by adding y_3 and y_4 to your plot.

You can add as many functions to the plot as you like, and MATLAB will continue to add them to the same graph, until you tell it to stop by typing `hold off`.

Add a title and axis labels to your graph.

3. Plotting 3D graphs

Plotting 3-dimensional functions takes a little more effort. Instead of just creating a function, we need to create an *array* (grid) of (x, y) points that we can enter into our function, which we then plot.

Say we want to plot the function $z = 2 - \sin(x^2) - \cos(y^2)$ between $-3 \leq x \leq 3$ and $-4 \leq y \leq 4$.

Begin by defining the anonymous function. Copy the following into the Command Window.

```
zfunction = @(x,y) 2 - sin(x.^2) - cos(y.^2);
```

Now create vectors for both the x and y domains. Copy the following into the Command Window.

```
xvec = -3:1:3 % create a vector of x values, starting at -3,  
incrementing by 1, and ending at 3  
yvec = -4:1:4 % create a vector of y values, starting at -4,  
incrementing by 1, and ending at 4
```

Turn both vectors into grids of values. Copy the following into the Command Window.

```
[x,y] = meshgrid(xvec,yvec)
```

Enter these grids of x and y values into the function to calculate a corresponding grid of z values. Copy the following into the Command Window.

```
z = zfunction(x,y)
```

Plot the grid of x , y and z values together. Copy the following into the Command Window.

```
surf(x,y,z)
```

You can increase the resolution of the graph by increasing the resolution of `xvec` and `yvec`. Repeat the process, but increment the `xvec` and `yvec` values by 0.05 instead of 1. Copy the following into the Command Window.

```
xvec = -3:0.05:3;  
yvec = -4:0.05:4;  
[x,y] = meshgrid(xvec,yvec)
```

```
z = zfunction(x,y)
surf(x,y,z)
```

Follow a similar process to graph $z = 2 - \frac{\sin(x^2)}{\exp(y)}$ between $-5 \leq x \leq 5$ and $0 \leq y \leq 10$. Use a vector resolution of 0.1 for both the x and y vectors.

SESSION 2 (30 minutes)

Sometimes, if you have to enter a series of calculations over-and-over, it's easier to write them all in one place and run them at once. These sequences of calculations are called *scripts*.

In MATLAB, click the button in the top-left that says "New Script".

Copy and paste the following code into the blank script.

```
f = @(x) 2*x.^4 - 15*x.^2 + 3*x;

x = 6;
h = 1;


f_dash_approx = (f(x+h) - f(x)) / h;

text = ['When h = ', num2str(h), ', the approximate slope of the tangent at x = ',
num2str(x), ' is ', num2str(f_dash_approx), '.'];
disp(text)

f_derivative = @(x) 8*x.^3 - 30*x + 3;
f_dash_truth = f_derivative(x);

text = ['The true slope of the tangent at x = ', num2str(x), ' is ',
num2str(f_dash_truth), '.'];
disp(text)
```

Click the down arrow below the Save Icon, and select Save As....

Save the script as a file called "script_example" in a folder that you'll be able to find later. Sometimes, the folder in which you save the script won't match the folder that MATLAB is currently working in. (Did script_example.m just pop up in the Current Folder window to the left? If not, you'll need to tell MATLAB where to look, by clicking this button just above it () and searching for the correct folder.)

Now that your script is saved, and you are working in the correct folder, you can run the script by calling its name. Type `script_example` into the command window and press Enter. What does MATLAB output?

Now edit the script so that $h = 0.01$ and execute the script (you can either click Run, above the script, or type `script_example` in to the Command Window again.)

Edit the script again so that $h = 0.001$. What do you notice of the approximation of the slope of the tangent each time you reduce the value of h ? Edit the script again so that $h = 0.01$ and $x = 2$.

Finally, type the word `why` into the Command Window and hit enter. 😊