

Matlab for Research session 3

Data visualisation and plotting

Colin Goldblatt, c.goldblatt@uea.ac.uk

October 21, 2007

1 Introduction

This handout and a m-file containing the code herein are available at <http://researchpages.net/people/colin-goldblatt/teaching/matlab/>.

Matlab will produce a very wide range of graphics with a high level of customisability. It is possible to change the properties of a plot both interactively, using graphical tools in the figure window, or by writing a script. The latter approach is advocated here, it is generally quicker and, critically, is repeatable.

Matlab is accompanied by very comprehensive help for graphics; however this may be inaccessible to a user who does not understand the general principles of Matlab figures or know what is available. The aim of this class is to give an overview of some of the Matlab plotting tools and to explain the principles of how to manipulate figures. In general, to access the help use

```
help filename
```

or

```
doc filename
```

for text or graphical (recommended) help respectively. If you dont know the function name, either try searching for what you want to do or look in the contents. Or ask your favourite Matlab geek¹.

2 Creating and modifying plots

2.1 A simple plot

We will start with simple 2-D plots to look at how to create pretty graphs in Matlab.

```
x = [-20:0.01:20];  
y = sin(x);  
z = cos(x);
```

```
plot(x,y,'r',x,z,'g')
```

¹This is often quicker, but note that it is important to look after your geek properly, with feeding and watering being especially helpful. Ask your geek what they prefer: pizza and coffee are a common diet, but regional variation means that cake and beer are common requirements in Norwich.

Add a title, legend and axis labels:

```
xlabel('x')
ylabel('sin(x) and cos(x)')
title('A simple graph')
legend('sin(x)', 'cos(x)')
```

Now output to a sensible format (pdf in this case)

```
print -dpdf graph1.pdf
```

To see a list of the available file formats and options

```
doc print
```

Graphs are vector graphics (as opposed to raster (bitmap)), so the best results are obtained by saving as vector format, such as **eps** or **pdf**. For display on the web, it will be necessary to use a raster format; **png** usually works best.

Note to Linux users: I find the the Matlab pdf driver does not produce very nice results, so I use `'-depsc'` then convert the eps file to a pdf file by running `epstopdf --outfile filename.pdf filename.eps` in a terminal.

*Note to Windows users: You may find that Word does not handle vector graphics well, for example using the `tiff` preview associated with an **eps** rather than the file itself. I would recommend using different software, with better graphics handling, such as \LaTeX .*

2.2 Controlling the properties of the plot

The most powerful way of controlling the properties of the plot is by using the **set** command with the handle of the graph. In the example above, we could have written

```
h1 = plot(x,y,'r')
hold on
h2 = plot(x,z,'g')
```

Here, **h1** and **h2** are the handles of the two lines. Hold on stops the first plot being overwritten by the first. To change the properties of one of the lines we could, for example, type

```
set(h1,'color',[ 1 0.5 0],'linewidth',4)
```

To see the full set of available options, search for *line properties* in the Matlab help. Most of these are very obscure and you will never use them!

Similarly, we can control the *axis properties* (again, see help). Use the handle **gca** which means “get current axis”

```
set(gca,'xlim',[-pi pi],...
      'xtick',[-pi:pi/2:pi],...
      'xticklabel',{'-pi','-pi/2','0','pi/2','pi'})
```

Note that `...` continues the current line; I put a single property/value pair on each line to make my code easier to read.

Give the plot a better **xlabel**

```
xlabel('-\pi \leq \Theta \leq \pi')
```

Here, we have used \TeX commands in the label to give special symbols. This is only available in the functions `title`, `xlabel`, `ylabel`, `zlabel`, and `text`, and not all \TeX / \LaTeX characters are available (look at *string* under *text properties* help for what is). Unfortunately, it is not possible to use special characters in tick labels (Grrrr!).

One can also control the *figure properties* (again, see help), using the handle `gcf`. For example, to get graphs to be the right size to put in my thesis, I use the following code:

```
width = 12;  
height = 10;  
set(gcf,'paperunits','centimeters',...  
      'PaperSize',[width height],...  
      'PaperPosition',[0 0 width height])
```

```
print -dpdf graph2.pdf
```

2.3 Annotating plots

Lets plot $y = x^2$ and annotate the plot...

```
figure
```

```
a = [1:1:10]  
b = a.^2;  
plot(a,b,'go--')
```

```
hold on
```

Here, `figure` opens a new figure window (your last figure is still there). We are setting some line properties within plot here; for more options,

```
doc linespec  
doc plot
```

Now add a text annotation to the plot

```
text(2,20,'hello!','color','b')
```

Here, we are setting the property/value in the text function but we could have equally written

```
h5 = text(2,40,'hello!');  
set(h5,'color','m')
```

Now add a filled shape

```
c1 = [4 4 5 5];  
c2 = [10 30 30 10];  
h3 = fill(c1,c2,'c');
```

Note how this covers up the line. We can plot a transparent shape too

```

d1 = [7 7 8 8];
d2 = [50 60 60 50];
h4 = fill(d1,d2,'r','linestyle','none');
alpha(h4,0.3)

```

For more information about setting transparency,

doc alpha

3 Multiple plots

Multiple plots can drawn in the same figure window with the `subplot` command. This has the general syntax

```
subplot(rows,columns,index)
```

For example

```

a = [0:1:100];
b = randn(size(a));

```

figure

```

subplot(2,2,1)
plot(a,b,'r-')

```

```

subplot(2,2,2)
stem(a,b)

```

```

subplot(2,2,3:4)
scatter(a,b,40,abs(b),'filled')

```

It is also possible to control precisely where subplots are located with the syntax

```
subplot('position',[left bottom width height])
```

This is very useful for producing multiple plot figures for printing (thesis, papers...) which need to look *right!* Here is an example stripped out of one of the plotting scripts for my thesis:

```

% this produces a 5 x 2 set of subplots which looked right for my thesis
sleftoffset = 0.10;
srightoffset = 0.04;
sbottomoffset = 0.05;
stopoffset = 0.023;
swidth = 0.5 - sleftoffset - srightoffset;
sheight = 0.2 - sbottomoffset - stopoffset;
vpos = [0.8 0.6 0.4 0.2 0];
hpos = [0 0.5];

a = [0:1:100];

```

```

figure

for ih = 1:length(hpos)
    for iv = 1:length(vpos)
        subplot('Position',...
            [hpos(ih)+sleftoffset vpos(iv)+sbottomoffset swidth sheight])
        plot(a,randn(size(a)),'r-')
        xlabel('Time')
        ylabel('Made up data')
    end
end

% add labels to the plot
% this uses another axes overlaid on the figure
h = axes('Position',[0 0 1 1],'Visible','off');
set(gcf,'CurrentAxes',h)
set(h,...
    'DefaultTextVerticalAlignment','Cap',...
    'DefaultTextHorizontalAlignment','left',...
    'DefaultTextFontSize',12,...
    'DefaultTextFontWeight','bold');

hold on
tposv = [1 0.8 0.6 0.4 0.2];
tposh = [0 0.5];
lab = ['ab';'cd';'ef';'gh';'ij'];
for vv = 1:length(tposv)
    for hh=1:2
        text(tposh(hh),tposv(vv),lab(vv,hh))
    end
end

% setup the page size and print the graph
width = 12.6;
height = 20;
set(gcf,'paperunits','centimeters',...
    'PaperSize',[width height],...
    'PaperPosition',[0 0 width height])

print -dpdf graph3.pdf

```

Note that this graph looks poorly spaced when displayed on screen, with some of the labels falling out of the edge of the figure window. However, it appears as I intended when printed. This is important! If you are trying to get a figure to look right when exported, you need to make adjustments in your scripts comparing to the exported figure, not what you see in the Matlab figure window.

4 Regression

Fitting a polynomial to some data can be done with the `polyfit` function. The example here fits a first order polynomial (a line) to some pseudodata:

```
a = [0:0.1:10];
b = 3*a+2 + randn(size(a));

figure
plot(a,b,'rx')
hold on

p=polyfit(a,b,1);
plot(a,polyval(p,a),'k-')
text(3,5,['y = ',num2str(p(1)), 'x + ',num2str(p(2))])
```

For more information, see the *Help Contents*,
MATLAB → *Data analysis* → *Linear regression analysis* → *Programmatic fitting*.

To work out what is an appropriate fit to some data, the *Basic Fitting GUI* (graphical user interface) is quite useful. From *Help Contents*,
MATLAB → *Data analysis* → *Linear regression analysis* → *Interactive fitting*.

5 Plotting matrix data

5.1 Begin with a note of caution...

Its very easy to produce some pretty pictures, but its important to remember that different graphics functions process your data in different ways. Ignoring this can lead to misleading plots! Look at some different options to plot a 3×3 magic square.

```
A = magic(3)
```

Using the `pcolor` function,

```
figure
pcolor(A)
shading flat
```

`pcolor` with `shading flat` or the default `shading faceted` ignores a row and column of the matrix. I have yet to meet someone who thinks this is a useful feature! A work around is add a row and column of NaNs to the data

```
B = NaN*ones(4,4)
B(1:3,1:3) = A
```

```
figure
pcolor(B)
shading flat
```

One can also use the `imagesc` function:

```
figure
imagesc(A)
axis xy
```

By default, this plots with a reversed y -axis, `axis xy` sets to normal (cf. `axis ij`). However, one problem with `imagesc` occurs when plotting with given (x, y) data. Try plotting all the following graphs:

```
x = 1:10;
y = 1:20;
Z = x'*y;
```

```
figure
pcolor(x,y,Z')
figure
pcolor(y,x,Z')
```

```
figure
imagesc(x,y,Z')
figure
imagesc(y,x,Z')
figure
imagesc(1:1:3,50:-10:-250,Z')
```

`pcolor` requires that matrix dimensions agree whereas `imagesc` lets you get away with anything; the biggest risk here is plotting your matrix the wrong way round and not realising!

Also, try some other options, for plotting the magic square:

```
figure
pcolor(A)
shading interp
```

```
figure
contourf(A)
shading flat
```

Do these well represent the data?

5.2 Now lets make some pretty pictures...

Try a few different ways of plotting some data:

```
A = peaks(50);
```

```
figure
imagesc(A)
axis xy
```

```
figure
contourf(A)
shading flat
```

```
figure
contourf(A,50)
shading flat
```

We can add a `colorbar` at a given location around or inside the plot

```
colorbar('southoutside')
```

and change the limits of the color axis

```
caxis([-2 2])
```

Note that on old versions of Matlab, it was necessary to redraw the colorbars after changing the `caxis`, but this seems to work automatically on v7.4. We can also change the `colormap`

```
colormap gray
```

Matlab can draw a range of 3-D graphics too, for example

```
figure
surf(A)
```

6 More exciting plots

There are lots of plot types in Matlab. To see what is available, look in the help. From the *Help Contents*,
MATLAB → *Graphics* → *Plots and plotting tools* → *Figures, plots and graphs*
→ *Types of plots available in Matlab*.

General help topics available from the *Help Contents* are often useful when working out how to plot a new type of graph. Have a look under
MATLAB → *Graphics* → *Creating specialised plots*.

Have a play!