

# Using Matlab at CUED

Tim Love

July 24, 2006

## Abstract

This document does *not* try to describe **matlab** comprehensively (see *matlab's HelpDesk* or the installed tutorial documents<sup>1</sup> for this), rather it introduces non-beginners to undocumented and/or local features of **matlab**. Suggestions and contributions for this document are welcomed. Mail [tpl@eng.cam.ac.uk](mailto:tpl@eng.cam.ac.uk).

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Info and help commands</b>	<b>2</b>
<b>3</b>	<b>Graphics</b>	<b>3</b>
3.1	Default Properties . . . . .	3
3.2	Labelling . . . . .	3
3.3	Creating User Interfaces . . . . .	4
<b>4</b>	<b>Sparse Arrays</b>	<b>4</b>
<b>5</b>	<b>File Input/Output</b>	<b>4</b>
<b>6</b>	<b>Examples</b>	<b>5</b>
6.1	Movies . . . . .	5
6.2	Bar and pie charts . . . . .	5
6.3	Displaying Surfaces . . . . .	5
6.4	Displaying data from a file . . . . .	5
6.5	Interpolation . . . . .	5
6.6	Generating solids from functions . . . . .	6
6.7	Making holes . . . . .	6
6.8	Adding buttons and menus . . . . .	6
6.9	Program-generated names and commands . . . . .	6
6.10	Vectorised code . . . . .	7
<b>7</b>	<b>User Interface Controls</b>	<b>7</b>
<b>8</b>	<b>Local Utilities</b>	<b>8</b>

---

<sup>1</sup><http://www-h.eng.cam.ac.uk/help/tpl/programs/matlab.html>

Copyright ©2002-6 by T.P.Love. This document may be copied freely for the purposes of education and non-commercial research. Cambridge University Engineering Department, Cambridge CB2 1PZ, England.

<b>9</b>	<b>Hardcopy</b>	<b>8</b>
9.1	Publication-quality output . . . . .	8
<b>10</b>	<b>Miscellaneous</b>	<b>8</b>
10.1	Customisation . . . . .	8
10.2	Setting your path . . . . .	9
10.3	Saving and Loading figures . . . . .	9
10.4	Optimising . . . . .	9
10.5	Numerical Analysis . . . . .	9
<b>11</b>	<b>More Information</b>	<b>9</b>
<b>12</b>	<b>Known bugs</b>	<b>10</b>

## 1 Introduction

**Matlab** is installed on many CUED research machines as well as on the teaching system. It has in-built facilities for data acquisition, processing and visualisation it also has toolboxes (sets of extra routines) to deal with symbolic maths, etc. Not all machines at CUED have the same matlab toolboxes.

**Matlab's** own help system contains all the standard information you need - just type `help` or better still `helpwin` inside **matlab** to access tutorials and reference manuals.

CUED's own documentation on matlab is far more extensive than can be provided in print. Rather than read the rest of this document on paper you may prefer to see the online versions at <http://www-h.eng.cam.ac.uk/help/tpl/programs/matlab.html>.

## 2 Info and help commands

There are several commands in **matlab** to help you get information and find out about your set-up.

**ver :-** Tells you which versions of which toolboxes (libraries) are installed.

**help command :-** tells you about the command.

**type command :-** tells you whether the command is a script or is built into matlab. If it's a script, it's displayed.

**which command :-** Locates functions and files. '`which fft`' tells you whether `fft` is a built-in command, a function or doesn't exist.

**lookfor :-** A keyword search. '`lookfor fft`' finds commands that deal with `fft`'s and briefly describes them.

**helpdesk :-** Starts a WWW browser to display matlab documentation.

**path :-** Prints the current matlab PATH. This is a list of directories. When you type a command, matlab looks in these directories for the corresponding file.

## 3 Graphics

Though it's not essential to know how **matlab** graphics are organised, it's likely to be useful sooner or later.

Figures are made of objects (lines, text, images etc). They have properties (like Color, Position, etc) which can be individually controlled. Each object has a *handle* (a unique identifier). To experiment with handles, create a simple graph by typing 'plot(1:3)'. Typing 'gcf' returns the handle of the current figure. 'get(*handle*)' lists the properties of the given object, so get(gcf) lists all the properties of the current figure. You can use set to change the values of these properties once you know the handle of the object and name of the property. For example, set(gcf, 'Color', 'green') sets the color of the current figure to green. If you type set(gcf) you get a list of the properties of the current figure, and alternative settings for these properties.

The axes are children of the figure - and consequently inherit the figure's properties. If you type get(gcf, 'Children') you'll get the axes handle, which you could also get by typing 'gca'. The axes in turn can have children.

'set' is a powerful mechanism. Often **matlab** has commands that spare you needing to use set directly, but particularly if you want to retrospectively fix something, the facility is useful.

### 3.1 Default Properties

It's also possible to set default property values for objects. When setting default values, the *handle* must be an ancestor of the object for which you are changing the default property value. This is because the object will inherit the property values from an ancestor.

To set the default values, prepend Default to the property name and use the set command. For example, set(gcf, 'DefaultAxesFontName', 'helvetica') will make helvetica the font for all future axes which are part of the current figure. set(0, 'DefaultAxesFontName', 'helvetica') would do the same for all figures subsequently created, because 0 is the handle of the root object - the ancestor of all other objects.

Setting default values can sometimes save much typing.

### 3.2 Labelling

People often want to change the axes and labels that the default setting provides. The size and style of fonts, axes labels and the numbering along the axes can all be controlled using menu options or the set command. Here's an example that also shows how to use greek characters.

```
surf(peaks(10));
% Make the axis numbering smaller than usual
set(gca, 'FontSize', 8)
% Have z tic marks at intervals of 0.5
set(gca, 'ZTick', -8:0.5:8)
set(gca, 'ZTickLabel', -8:0.5:8)
% Label the Z axis
zlabel = get(gca, 'ZLabel');
set(zlabel, 'String', 'D(b/3)')
set(zlabel, 'FontName', 'Symbol')
```

The text command can deal with maths and changes of font styles. For example,

```
text(0,1,'{\bf bold} \alpha=x^3. \Sigma x \leq \int y \forall x')
```

works as L<sup>A</sup>T<sub>E</sub>X users might expect. The `title` and `labelling` commands are similar. More colourful information can be added by using

- `colorbar` - Adds a color scale to axes.
- `legend` - Creates legends.

### 3.3 Creating User Interfaces

There are commands to add simple dialog and information boxes to applications

`errordlg` – displays an error message.

`helpdlg` – displays a help message.

`questdlg` – displays a yes/no/cancel box.

`warndlg` – displays a warning message.

The easiest way to produce a complete user interface is to use `guide`. The *Guide* window you get after typing `guide` has its own help menu.

## 4 Sparse Arrays

If you know that your matrices will stay nearly empty, then make them sparse.

```
% comparison of sparse vs full matrices
% Create sparse and full versions of the same matrix.
% Note that the density is 0.05. The break-even point
% for this operation seems to be about 0.25.
S = sprandn(60,60,0.05);
F = full(S);
% Compare speed.
disp('Sparse')
t0=cputime;
B = S * S;
cputime-t0
disp('Full')
t0=cputime;
C = F * F;
cputime-t0
```

## 5 File Input/Output

Online help on this is available from within **matlab** if you type `help iofun`.

`load` and `save`, if given a filename without an extension assume that binary `.mat` files mean to be used, otherwise ASCII files are used.

`reshape` is a useful command: you can read a stream of data then reformat it into the required shape.

See the Matlab page on Experimental data<sup>2</sup> for details.

---

<sup>2</sup>[http://www-h.eng.cam.ac.uk/help/tpl/programs/Matlab/experimental\\_data.html](http://www-h.eng.cam.ac.uk/help/tpl/programs/Matlab/experimental_data.html)

## 6 Examples

Here are some commonly use features of **matlab**

### 6.1 Movies

```
% movie demo
M=moviein(16);
for j = 1:16
    plot(fft(eye(j+16)))
    M(:,j)= getframe;
end
movie(M,5)
```

### 6.2 Bar and pie charts

```
y=1:5;
subplot(1,2,1)
bar3(y)
subplot(1,2,2)
pie(y)
```

### 6.3 Displaying Surfaces

To display a surface without a mesh, where the colors of the facets are smoothly graduated, use the shading command.

```
surf(peaks(20))
shading interp    %graduated color.
colorbar          %Adds a color key.
```

### 6.4 Displaying data from a file

Suppose that you have a text file of data (called `foo`, say) produced by another program. The data is in 4 columns. The 1st column contains  $x$  coordinates and the other columns contain 3 sets of  $y$  coordinates. You want to display 3 lines on a graph.

First, remove from `foo` any title headings, etc. Then type `'load foo -ascii'` (type `'help load'` for more info). This creates a matrix called `foo`. To check that the matrix has the expected number of rows and columns you can type `size(foo)`. The following command will now produce the required graph.

```
plot(foo(:,1),foo(:,2), foo(:,1),foo(:,3), foo(:,1),foo(:,4))
```

### 6.5 Interpolation

Most of **matlab**'s 3D routines require the values to be on a regular 2D grid. Suppose you have experimental data  $z$  for a function of 2 variables,  $x$  and  $y$ , but those results don't lie on a regular grid.

```
% first define a regular grid. Suppose the x and y values lie between
% 0 and 100, and that we want 20 grid-lines each way
steps = linspace(0,100,20);
[XI,YI] = meshgrid(steps, steps);
```

```

% now extrapolate - find z values for these grid points
ZI = griddata(x,y,z,XI, YI);
% display this mesh with the original data
mesh(XI,YI,ZI);
hold
plot3(x,y,z);
hold off

```

Matlab has 3D interpolation facilities too.

## 6.6 Generating solids from functions

Use cylinder

```

t = 0:pi/10:2*pi;
[X,Y,Z] = cylinder(2 + cos(t));
surf(X,Y,Z);
axes('Box','on');

```

## 6.7 Making holes

If a point is given the value NaN (Not-A-Number), then that point isn't displayed. This is useful if you want a 'hole' in your picture.

```

z = peaks(20);
z(4:8,4:8) = NaN*z(4:8,4:8);
surf(z)

```

## 6.8 Adding buttons and menus

One of the properties of graphical objects (called ButtonDownFcn) describes what happens if there's a mouse click on the object, thus permitting complex user-interaction.

```

% This displays a simple figure and some buttons
% It needs a command called rotating to work
uicontrol('style','pushbutton','string','rotate',...
'ButtonDownFcn','rotating','Enable','off');
mesh(peaks(10))

```

Put this code into rotating.m

```

% rotaing.m
[az, el ] = view;
az=az+45;
view(az, el)

```

## 6.9 Program-generated names and commands

This shows how to generate variable names 'on the fly'.

```

% A1, A2, A3, and A4 are created and set to 3
for i=1:4
s=sprintf('A%d= 3',i);
eval(s);
end;

```

```
% This shows how to run a command that's stored in a string
str=input('Type in a command ', 's')
eval(str)
```

## 6.10 Vectorised code

Code can be made shorter and faster by exploiting vectorisation - removing for loops.

```
% This creates a 10x10 matrix using the magic command, then finds
% the mean of each column, ignoring any element less than 10.
% Note that no explicit loops are used.
array=magic(10)
keep = (array>=10);
colSums = sum(array .* keep);
counts = sum(keep);
means = colSums ./ counts
```

For many more examples see Vectorisation Tricks<sup>3</sup>

## 7 User Interface Controls

uicontrol and uimenu can produce a variety of objects. The example below uses them directly, but guide is easier.

```
% uidemo
title('A demo of User Interface Facilities');
uicontrol('Style','Pushbutton', 'Position', [20, 20, 100,30],...
    'Callback','disp(''Pushbutton'')','String','Push me');
uicontrol('Style','Checkbox', 'Position', [20, 60, 100,30],...
    'Callback','disp(''Checkbox'')','String','Push me too');
uicontrol('Style','Popup', 'Position', [20, 100, 100,30],...
    'Callback','disp(''Popup'')','String','first|second|third');
uicontrol('Style','Radiobutton', 'Position', [20, 140, 50,30],...,
    'Callback','disp(''Radio'')','Min',0,'Max',3, 'Value',3, ...
    'HorizontalAlignment','left', 'String','first|second|third');
uicontrol('Style','Slider', 'Position', [20, 180, 100,30],...,
    'Callback','disp(''Slider'')','Max',100,'Min',10,'Value',50);
uicontrol('Style','Edit', 'Position', [20, 220, 100,30],...,
    'Callback','disp(''Edit'')','String','Change me');

top1 = uimenu('Label','Calculator');
uimenu(top1,'Label','add','Callback','disp(''add'')');
uimenu(top1,'Label','subtract','Callback','disp(''subtract'')');
uimenu(top1,'Label','multiply','Callback','disp(''multiply'')');
uimenu(top1,'Label','divide','Callback','disp(''divide'')');
top2 = uimenu('Label','Roots');
uimenu(top2,'Label','square','Callback','disp(''square'')');
uimenu(top2,'Label','cube','Callback','disp(''cube'')');
```

Each graphical object can have a 'callback' (a routine that's called when the object is clicked-on, etc). Type sigdemo1 to see an example.

<sup>3</sup><http://www-h.eng.cam.ac.uk/help/tpl/programs/Matlab/tricks.html>

## 8 Local Utilities

Some extra facilities have been installed at CUED. Use the `help` command to find out more.

**arrow** - Draws arrows.

**polarhg** provides more control over polar plots.

**ezlegend** - menu-driven control over legend properties.

**b\_w** - converts lines on plots with default colour order lines to black lines

**laprint** - better control over output that's to be used with LaTeX.

## 9 Hardcopy

The print facility in the Teaching System's **matlab** has been set up to print to your default print, but please use it sparingly. The `cuedpr` command adds your name to the top of the page. Use it rather than `print` whenever possible.

Note that by default a color postscript file is converted to greys when the file is sent to the laserjet. If you want solid coloured lines to remain solid on hardcopy, do

```
print pic -dps
! lp -dljmr1 -opostscript pic.ps
```

To produce postscript files ready to insert into L<sup>A</sup>T<sub>E</sub>X documents or `xfig` use `print pic -depssc`

For color prints use something like `print pic -dpssc` to produce a file called `pic.ps`. See the printing page<sup>4</sup> in the help system for further details on colour printing.

### 9.1 Publication-quality output

As well as the `b_w` mentioned above, `laprint` is installed. See MATLAB Graphics in LaTeX Documents: Some Tips and a Tool<sup>5</sup> for details.

Sometimes (especially with filled contours) the picture online will be different from the printed picture. To produce a postscript file that is sure to be an accurate representation of the picture at the usual resolution try `print -zbuffer -r864 -dpssc`.

To control the size and position of the result picture on the page you can set the figure's `PaperPosition` property. The following produces a picture whose left-bottom corner is at (0.25ins, 2.5ins), width 5ins, height 3ins.

```
set(gcf, 'PaperPosition', [0.25 2.5 5 3])
```

## 10 Miscellaneous

### 10.1 Customisation

If you have a file called `startup.m` it will be read when **matlab** starts.

---

<sup>4</sup><http://www-h.eng.cam.ac.uk/help/tpl/printing.html>

<sup>5</sup><http://cgi.hrz.uni-kassel.de/linne/mdownl.cgi?matlatex.ps>

## 10.2 Setting your path

When you type the name of a script matlab looks in a sequence of directories (called the path) for the script. You can print the current path by typing `path`. The `path` command can also be used to change the path.

## 10.3 Saving and Loading figures

Figures can be saved and recreated using

```
print -dmfile filename
```

which creates 2 files – `filename.m`, the commands to recreate the image, and `filename.mat`, the data for the figure.

## 10.4 Optimising

- Functions are faster than scripts – they get compiled internally the first time they are run.
- Use vectorised operations instead of `for` loops.
- If you know the maximum size that a matrix will be, create it that size, rather than letting it grow incrementally.
- Use `pack` every so often to tidy up memory usage.
- Use sparse matrices when you can. They may save you a great deal of space and time.

## 10.5 Numerical Analysis

It helps to know some numerical analysis terms.

**Inf** :- The IEEE representation of positive infinity.

**NaN** :- The IEEE representation of 'Not-a-Number', used when an operation has an undefined result (e.g.  $0/0$ ). All logical operations involving NaN return false, except for  $\neq$ . If you try to plot points with the value NaN, nothing appears. This can be useful for clipping, making holes in surfaces etc.

**Condition Number** :- The condition number of a matrix (calculated using `cond`) gives a fair indication of how well the matrices can be operated upon.

## 11 More Information

Matlab's 'Help Navigator' gives access to demos, video tutorials and extensive documentation.

The CUED help system's matlab page<sup>6</sup> gives you access to many types of matlab help, including a matlab discussion group.

Many matlab scripts are available from other sites. See the *Toolboxes* section of the CUED help system's matlab page.

---

<sup>6</sup><http://www-h.eng.cam.ac.uk/help/tpl/programs/matlab.html>

## 12 Known bugs

If you discover a bug, first try to repeat the problem then contact Tim Love (tpl@eng) saying what machines you were using, when you were using them, and giving a description of the problem, preferably with a script that shows up the problem. Here's a list of known bugs.

**April 2000** - legends are difficult to customise