Using Matlab 4

Tim Love

June 11, 1996

Abstract

This document does *not* try to describe **matlab 4.2** comprehensively (type `help matlab` for this), rather it introduces users to undocumented and/or local features of **matlab 4.2**. Suggestions and contributions for this document are welcomed. Mail `tpl@eng.cam.ac.uk`.

Contents

1 Introduction

Matlab 4.0 was released in 1992. **Matlab 4.2** was released in April 1994. **Matlab 4.2** is installed on some SUN and DEC-Alpha research machines and the series 9000/700 HP teaching machines.

The full set of **matlab 4.0** manuals are in the DPO machine room along with the **matlab 4.2** release notes which describe how **matlab 4.2** differs from **matlab 4.0**. **Matlab 4.2** also has its own help system (just type `help` inside **matlab 4.2**).

The online documentation on matlab is far more extensive than can be provided on paper. Rather than read the rest of this document you may prefer to type `help` from the unix command line, and click on the `matlab`link - you'll find this handout and *much* else besides.

2 Info and help commands

cputime :- This returns the CPU time in seconds that has been used by the **matlab 4.2** process.

expo :- - demos.

history :- If you type the first few characters of an old line then hit up-arrow, that line is retrieved.

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

more :- Scrolls text output.

path :- Prints the current matlab `PATH` and lets you change it.

tic, toc :- `tic` starts a timer, `toc` stops the time and prints time taken.

type *command* :- tells you whether the command is a script or is built into matlab.

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

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

3 Graphics

Figures are trees of objects (lines, text, images etc). Each of these objects can have parent and child objects properties (like Color, Position, etc) which can be individually controlled. One of these properties (called `ButtonDownFcn`) describes what happens if there's a mouse click on the object, thus permitting complex user-interaction.

Each object has a *handle* (a unique identifier). 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.

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.

Typing `getpref` gives the list of properties for which default values can be set. To set the default for any of these, 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.

Setting default values can sometimes save much typing. As of **matlab 4.2** , `DefaultLineLineStyle` has gone. Instead, there's a `LineStyleOrder` which works rather like the `ColorOrder` facility. Consequently the example on page 2-112 of the *User's Guide* no longer works. Here's a corrected version

```
figure('Position',[360 250 950 400])
set(gcf,'DefaultAxesBox','on')
subplot(121)
set(gcf,'DefaultAxesLineStyleOrder',':')
plot(sin(0:pi/20:2*pi));
hold on
plot(cos(0:pi/20:2*pi))
text('Position',[20 .4],'String','sine')
text('Position',[15 -.3],'String','cosine','HorizontalAlignment','right')

subplot(122)
set(gcf,'DefaultAxesLineStyleOrder','-')
set(gca,'DefaultTextRotation',90)
plot(sin(0:pi/20:2*pi))
hold on
plot(cos(0:pi/20:2*pi))
text('Position',[20 .4],'String','sine')
text('Position',[15 -.3],'String','cosine','HorizontalAlignment','right')
```

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 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,'ZTickLabels',-8:0.5:8)
% Label the Z axis
zlabel = get(gca,'ZLabel');
set (zlabel,'String','D(b/3)')
set (zlabel,'FontName','Symbol')
```

We have 2 facilities installed to help add maths to graphs. `stextfun` is newer than `textfcn` and is probably easier to use (see section 9), but `textfcn` has been used more. Add `/export/matlab/toolbox/local/textfcns/` to your `MATLABPATH` and type `help mergtex` for details.

Note that the `textfcns` routines can't cope with subfigures – the text disappears when `subplot` is used! Here are two solutions from *Dan Braithwaite*. For both solutions enter all of your subplot commands before you attempt to put text on the figure or else subplot will delete everything.

- use mergtex (with the 'rot=90' optional argument for y-labels) to put up all of your text strings. To fine tune the placement of the strings, use movtex to manually shift them around.
- you can alter the xlabtex, ylabtex, and titltex functions by removing the line `delprev('xlb or ylb or tit')` and altering the lines that look like

  ```
  set(pvec(i),'position',[x,y],...,'userdata','xlb or ylb or tit')
  ```

  to use the same userdata scheme as mergtex which calls the nxnum function to get a unique number to put in the userdata property. NOTE: the reason the xlab/ylab/titltex functions were originally written so that they delete the previous string is to eliminate overwriting the strings. In future versions of the textfcns suite a different identifier will be added, such as xl1, xl2, yl1, yl2 etc.

More colorful 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.

See the examples below for more sophisticated usage. Perhaps the easiest way to produce a complete user interface is to use the `guimaker` facility (see the **Local Utilities** below).

4 Sparse Arrays

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

```
% comparison of sparse vs full op speed
% 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

2 functions undocumented in the paper manuals are `fgets`, which reads a line, stripping it of its newline character, and `fgetl` which doesn't strip the newline. Online help is available 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.

6 Examples

6.1 Movies

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

6.2 Displaying Surfaces

To display a surface without a mesh, where the colors of the facets are smoothly graduated, use the following commands

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

6.3 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

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))
```

6.4 2D interpolation

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. You can display this data using `griddata`

```
% first define a regular grid. Suppose x and y are between 0 and 100
steps = 0:.5:100;
[XI,YI] = meshgrid(steps, steps);

% now 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
```

6.5 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');
```

6.6 Making holes

If a point is given the value `NaN` (Not-A-Number), then that point isn't display. 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)
```

6.7 Adding buttons and menus

```
% This displays a simple figure and some buttons
% It needs a command called button_callbacks to work
buttons = buttonv([.05 .95], 0:-1, 0, 'button_callbacks', ...
   'left', 'right', 'up', 'down');
mesh(peaks(10))
```

Put the callbacks into another file called `button_callbacks.m`

```
% button_callbacks.m
% callbacks for buttons
[az, el ] = view;
if button == 1
   az = az +45;
end
if button == 2
   az = az - 45;
end
if button == 3
   el = el + 45;
end
if button == 4
   el = el - 45;
end
view(az, el)
```

6.8 Miscellaneous

```
% 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)

% 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
```

7 User Interface Controls

`uicontrol` and `uimenu` can produce a variety of objects with more control than `buttonv`. The example below uses them directly. A utility to make their use easier is available. Do

```
path(path,'/export/matlab/toolbox/local/guimaker');
```

to make the utility accessible, then type `help guimaker` or just `guimaker`.

```
% 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');
disp('I can''t get more than 1 radiobutton to appear - tpl');
uicontrol('Style','Radiobutton',  'Position', [20, 140, 50,30],...,
          'Callback','disp(''Radio'')','Max',3, 'Value',2, ...
          'HorizontalAlignment','left', 'String','first|second|third');
uicontrol('Style','Slider', 'Position', [20, 180, 100,30],...,
          'Callback','disp(''Slider'')','Max',100,'Min',10);
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. Type `sigdemo1` to see an example.

8 Using Matlab 4.2 on the Teaching System

The new 3D features work best on color terminals like the newer ones in the DPO. The older terminals can only show 64 colors simultaneously. The HP 9000/712 machines (`twg`XX) in a suite at the end of the DPO can show 256 colors. To use **matlab 4.2** on these just type `matlab`.

You might find, especially if you're on an old workstation, that the screen colors change as you move the pointer in and out of the **matlab 4.2** graphics window. This is because when you started **matlab 4.2**, other applications had already used up many of the available colors, so **matlab 4.2** created for itself a private *colormap* which is switched in when the pointer is moved into its window, making the colors in the other windows false. This effect is distracting but harmless.

There is a process size limit of 32 Megs imposed on the Teaching System machines. If your process grows bigger than this (run the `top` program to see if that's the case) it will crash. Try freeing memory breaking your job into smaller sections.

9 Local Utilities

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

guimaker Creates Graphical User Interfaces. (needs `/export/matlab/toolbox/local/guimaker` on the matlab path).

plotyy Allows 2 y axes on a graph.

waitbar Lets you indicate how much of a long calculation has been done.

odesuite This improves and extends the standard matlab facilities. Type `handout odesuite` to read about the theory. Run matlab and do

```
path('/export/matlab/toolbox/local/funfun',path)
help funfun
```

to find out how to use it.

arrow Draws arrows.

textfcn A collection of routines that help in producing strings with sub- and superscripts as well as greek characters. (needs `/export/matlab/toolbox/local/textfcns` on the matlab path). Type `help mergtex` for details.

stext (needs `/export/matlab/toolbox/local/stextfun` on the matlab path). Similar to `textfcn` but probably easier to use. Type `help stext` for details.

polarhg provides more control over polar plots.

ezlegend - menu-driven control over legend properties

contourfill, contoursurf, extcontour - improved contour facilities

10 Hardcopy

The `print` facility in the Teaching System's **matlab 4.2** has been set up to print to `ljmr1` by default, but please use it sparingly. 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
```

If you have access to a research group's postscript printer, use it.

To produce postscript files ready to insert into LaTeX documents use `print pic -deps`

For color prints use `print pic -dpaintjet` to produce a file called `pic.pjet` that you can send to the paintjet printer in the machine room by doing

```
lp -dpjmr1 pic.pjet
```

but note that you'll be charged for such output. See the `printing` page in the help system for further details

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])
```

11 Miscellaneous

11.1 Customisation

If you have a file called `startup.m` it will be read when **matlab 4.2** starts. **Matlab 4.2** can also be customised using the Xdefaults mechanism. See `/usr/lib/X11/app-defaults/Matlab` for details.

11.2 Using monochrome displays

There are 3 things you can do to ensure **matlab 4.2** works appropriately on a monochrome display

- Type `system_dependent(7)` at the MATLAB prompt. This is MATLAB's system flag which lets MATLAB know that it is a monochrome monitor.
- Type the following command at the MATLAB prompt:

  ```
  set(0, 'blackandwhite','on')
  ```

  If this works, then place the following code in your `startup.m` file:

  ```
  set(0,'defaultrootblackandwhite','on')
  ```

- Place the following in your `startup.m` file. This one will definitely work even if the above 2 don't.

  ```
  set(0,'defaultaxescolororder',[1 1 1])
  ```

11.3 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.

11.4 Optimising

- Functions are faster than scripts – they get compiled internally the first time they are run.
- Use matrix 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.

11.5 Faster Plotting

Plotting many lines individually with **matlab 4.2** can be slow, apparently because each line object has a certain (large) overhead involved with it. However, there is a (fairly) easy solution available.

Say you have row vectors $x1, x2, x3, ...$ to be plotted against $y1, y2, y3, ...$

The slow way to plot is:

```
plot(x1,y1,x2,y2,x3,y3, ... );
```

The faster way is:

```
plot([x1,NaN,x2,NaN,x2,NaN, ... ],[y1,NaN,y2,NaN,y3,NaN, ... ]);
```

This can make this difference of orders of magnitude.

Now the next problem is how to prevent **matlab 4.2** from redrawing everything every time you add a new line (aka Matlab3.5 'hold on' effect). The good news is that you *can* do this, the bad news is that you can't exactly! The sequence of events is

1. plot your data setting the 'erasemode' property to 'none', and save the handles. *All* the lines must have `erasemode=none`.

2. change the xdata and ydata properties to your new line each time you want to add one.

e.g.

```
hh=plot(x1,y1);
set(hh,'erasemode','none');
for [some kind of loop]
  [change y1]
  set(hh,'ydata',y1);drawnow;
end;
```

(from the command line you won't need the drawnow)

This will work as long as you don't want to a) refresh (i.e. redraw) the screen, or b) make a hardcopy output. For an example, see the code in `ode23p.m` or run `lorenz.m`).

Alternatively if you know you will be plotting say 100 lines, you pre-plot 100 lines with empty data vectors and `erasemode=none`, save the handles, andthen change the `xdata` and `ydata` properties as you need to. This will be a little slower, but you will be able to make hardcopies.

(from Rich Pawlowicz - `rich@boreas.whoi.edu`)

11.6 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.

12 More Information

Typing `help matlab` gives you access to many types of matlab help. You might also find it useful to look at the provided demo scripts in `/export/matlab/toolbox/matlab/demos`. Some other scripts are in `/export/Examples/matlab`. There's also a newsgroup devoted to matlab issues: `comp.soft-sys.matlab`.

Many matlab scripts are available by `ftp` from `ftp.mathworks.com`. The Matlab User Group files are archived at `research.att.com` in `/netlib/matlab`.

13 Known bugs

Note that the HP 700 series machines are currently running **matlab 4.2**. A set of release notes for this is in the DPO machine room. It lists some errors in the manuals (which are for `matlab4.0`) and lists a few new features and clarifications.

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 major bugs using **matlab 4.2** on the HP workstations.

Wed Oct 19, 1994 In subfigures, the `textfcns` don't work properly (see page 3). Also, the position of titles, etc are returned to the default after the user has made changes.

Wed Mar 6, 1995 If the x label uses a big font, the resulting postscript file clips it. Set the Figure's Position to have the same width and height of its PaperPosition rectangle to see what your Axes looks like in the size it will print.

Fri Sept 22, 1995 `guimaker` can sometimes crash. A new version exists but it's shareware so we haven't installed it.