mas.622j Matlab Help

Originally written by Tom Minka, and modified by Yuan Qi and Ashish Kapoor, Bo Morgan Sept. 2006

Getting started

Open an editor window next to your matlab window. You'll often find yourself mousing text in the editor window and pasting it into the matlab window.

If you're a complete matlab novice, type "intro" or, if you're real fond of splashy colors, type "demo". You can also read help from matlab help menu.

You can also type "helpwin" or "helpdesk" to get a help window. The matlab help command is "help". Try "help general" or just plain "help". The apropos command is "lookfor". Try, e.g. "lookfor tangent" or "lookfor random".

Syntax

The matlab continuation code is "..." i.e. if you have a long formula that is several lines long, end each line with ... and continue the formula on the next line.

The matlab comment character is "%". The semicolon ";" is useful too; it makes the command on that line operate silently. For example, "A = B;" copies matrix B into matrix A. "A = B" does the same thing, but prints out the whole contents of matrix B while copying.

Data structures

The most useful data strucutre in Matlab is matrix. Scalars is considered 1 by 1 matrices, and strings (delimited by 'single quotes') are considered vectors (which in turn are just skinny matrices). Check out

```
  sprintf  sscanf  num2str  int2str
```

Matlab5 and 6 added the very useful "cell array" and "struct" data types. A cell array is just like a matrix except each entry can be any data type, not just a number. For example:

```
>> c = {'joe' 5 [1 2 3]}

c =

    'joe'    [5]    [1x3 double]

>> c{3}

ans =

     1     2     3
```

A struct is similar except its contents are addressable by name only.

```
>> s = struct('name', 'joe', 'age', 30)

s =

    name: 'joe'
     age: 30

>> s.name

ans =

joe

>> s.age

ans =

    30
```

Plotting and I/O

Matlab is very good at plotting your data. See our plotting examples and/or get matlab's help on:

```
  plot  grid  hold  drawnow  axis  axes  orient
  subplot  mesh  meshgrid  plot3  rotate3d
```

Then you'll want to print your plot to a file. To print the current plot to a postscript file, type

```
  orient tall   %% this line is optional
  print -deps myfilename.ps
```

You can use C style file I/O. Get help on commands

```
  fopen  fclose  fscanf  fprintf  printf
```

The matlab parser is not too clever, so don't put "-" in a filename; poor matlab will think you want to subtract!

Matlab has a native data file format, plus it can save and load data from ASCII files. See the stock answers and check out

```
  save filename.dat -ascii
  load filename
```

Subroutines and objects

Yes, you can write subroutines in matlab. Each subroutine lives in its own file, with a name ending with .m, and you call it by filename. Matlab will sometimes not notice that you have created a new subroutine. Use `path(path)` to make it rescan the directories.

In matlab, subroutines can be associated with specific classes to simulate "methods" for object-orientation. To make a method for class `myclass`, all you have to do is put the subroutine in a subdirectory called `@myclass`.

You must always have a subroutine in `@myclass` which is just called `myclass`. This subroutine is called to create instances of the class. Get matlab's help on `class`. If you redefine a class, e.g. by editing `@myclass/myclass`, Matlab will complain unless you clear the old definition with `clear myclass`. Don't ask me why.

`clear x` can also be used to remove the variable *x* from the workspace, e.g. to reclaim memory. Careful: `clear` alone removes all variables!

Some Basic Examples of Variable Manipulations in MATLAB

```
>> v = ones(1,3)

v =
     1     1     1

>> v = [ 1 1 1 ];

>> w = 2:4;

>> w = [2 3 4];

>> x =  x = [[1];[4];[9]]

x =
     1
     4
     9

>> x = [1 4 9]';

>> v * x
```

```
ans =
    14

>> v*3

ans =
     3     3     3

>> v .* x
??? Error using ==> .*
Matrix dimensions must agree.

>> v .* x'

ans =
     1     4     9

>> y = exp(w)

y =
    7.3891   20.0855   54.5982

>> z = [v' w' x]

z =
     1     2     1
     1     3     4
     1     4     9

>> w*z

ans =
     9    29    50
```

Some Basic Examples of Language Structure in MATLAB

```
%%      help  is a very useful matlab command.  So is  lookfor
%%      try 'help' and 'help general' to see what help topics are
available.

>> help if

 IF     Conditionally execute statements.
        The general form of an IF statement is:
                IF variable, statements, END
        The statements are executed if the real part of the variable
        has all non-zero elements. The variable is usually the result
of
        expr rop expr where rop is ==, , <=, >=, or ~=.
        For example:

               IF I == J
```

```
                 A(I,J) = 2;
              ELSEIF ABS(I-J) == 1
                 A(I,J) = -1;
              ELSE
                 A(I,J) = 0;
              END
>> help for

 FOR    Repeat statements a specific number of times.
        The general form of a FOR statement is:

           FOR variable = expr, statement, ..., statement END

        The columns of the expression are stored one at a time in
        the variable and then the following statements, up to the
        END, are executed. The expression is often of the form X:Y,
        in which case its columns are simply scalars. Some examples
        (assume N has already been assigned a value).

             FOR I = 1:N,
                 FOR J = 1:N,
                     A(I,J) = 1/(I+J-1);
                 END
             END

        FOR S = 1.0: -0.1: 0.0, END steps S with increments of -0.1
        FOR E = EYE(N), ... END  sets E to the unit N-vectors.


%% do 'help lang' for more on flow-of-control constructs.
%% do 'help elfun' for more on elementary math functions.
%% do 'help ops' for operators and special characters.
%% do 'help matfun' for matrix functions.
%%
%%  each of these gives a list of topics for further help; e.g.

>> help eig

 EIG    Eigenvalues and eigenvectors.
        EIG(X) is a vector containing the eigenvalues of a square
        matrix X.

        [V,D] = EIG(X) produces a diagonal matrix D of
        eigenvalues and a full matrix V whose columns are the
        corresponding eigenvectors so that X*V = V*D.

        [V,D] = EIG(X,'nobalance') performs the computation with . . .
```

Some Basic MATLAB Plotting Examples

This assumes you've gotten matlab started and are staring at a matlab prompt. You can copy and paste text into the matlab window. Do that with each block of statements, one block at a time, so you can see the results of each plot.

```
%%  make a row vector from -3 to 3
%% you can see the contents of xvals by typing 'xvals' with no trailing
';'
xvals = -3:.25:3;

%% make a row vector of gaussian densities.
yvals = 1/(sqrt(2*pi))*exp(-(xvals.^2)/2);

%%%% The FOR loop below produces the same result as the statement above
%% for i = 1:length(xvals);
%%    yvals(i) = 1/(sqrt(2*pi))*exp(-(xvals(i)^2)/2);
%% end;

%% make a simple 2D plot
plot(xvals, yvals);

%% dress it up a little
clf;
h = axes('position',[.2 .1 .6 .8]);  %% also try "help subplot"
set(h,'fontsize', 7);    %% type "set(h)" to get huge list of options %%
plot(xvals, yvals), axis([-3.1 3.1 0 .5]), grid on;
xlabel('Standard Deviations'), ylabel('Probablility Density'),
legend('The curve'), text(-.4, .42, 'The peak');
   title('Your Basic Gaussian Density')

%%  make another row vector
zvals = log10(abs(yvals))/2;

%% simple 2-color plot
plot(xvals, yvals, ':', xvals, zvals, '--');
legend('The curve', 'Its log');

%% Clear, then put three plots on the same graph
clf;
plot(xvals, yvals,'-'), hold on,
plot(xvals, zvals,'o'), hold on,
plot(xvals, xvals/7,'+'), hold off;

%% save it in a file in postscript format
print -deps killmeplot.ps

%% example from 'help quiver'
xord = -2:.2:2;
yord = -2:.2:2;
[x,y] = meshgrid(xord,yord);
z = x .* exp(-x.^2 - y.^2);
[px,py] = gradient(z,.2,.2);
contour(x,y,z),hold on, quiver(x,y,px,py), hold off

%% one kind of 3D plot
plot3(x,y,z), grid on;
% spin it with the mouse
rotate3d on;

%% another kind of 3D plot
mesh(xord, yord, z), view(10,20);
rotate3d on;
```