MATLAB Summary

MATLAB (matrix algebra)

Matlab is a commercial "Matrix Laboratory" package which operates as an interactive programming environment. It is a mainstay of the Mathematics Department software lineup and is also available for PC's and Macintoshes and may be found on the CIRCA VAXes. Matlab is well adapted to numerical experiments since the underlying algorithms for Matlab's builtin functions and supplied m-files are based on the standard libraries LINPACK and EISPACK.

Matlab program and script files always have filenames ending with ".m"; the programming language is exceptionally straightforward since almost every data object is assumed to be an array. Graphical output is available to supplement numerical results.

Online help is available from the Matlab prompt (a double arrow), both generally (listing all available commands):

```
>> help
[a long list of help topics follows]
```

and for specific commands:

```
>> help fft
[a help message on the fft function follows].
```

Paper documentation is on the document shelf in compact black books and locally generated tutorials are available and are used in courses.

How to quit Matlab

The answer to the most popular question concerning any program is this: leave a Matlab session by typing

```
quit
```

or by typing

```
exit
```

to the Matlab prompt.

Batch jobs

Matlab is most often used interactively, but "batch" or "background" jobs can be performed as well. Debug your commands interactively and store them in a file (`script.m', for example). To start a background session from your input file and to put the output and error messages into another file (`script.out', for example), enter this line at the system prompt:

```
nice matlab < script.m >& script.out  &
```

You can then do other work at the machine or logout while Matlab grinds out your program. Here's an explanation of the sequence of commands above.

1. The "nice" command lowers matlab's priority so that interactive users have first crack at the CPU. You

must do this for noninteractive Matlab sessions because of the load that number--crunching puts on the CPU.
2. The "< script.m" means that input is to be read from the file script.m.
3. The ">& script.out" is an instruction to send program output and error output to the file script.out. (It is important to include the first ampersand (&) so that error messages are sent to your file rather than to the screen -- if you omit the ampersand then your error messages may turn up on *other* people's screens and your popularity will plummet.)
4. Finally, the concluding ampersand (&) puts the whole job into background.

(Of course, the file names used above are not important -- these are just examples to illustrate the format of the command string.)

A quick tutorial on Matlab is available in the next Info node in this file. (Touch the "n" key to go there now, or return to the menu in the Top node for this file.)

MATLAB Tutorial

MATLAB Tutorial (based on work of R. Smith, November 1988 and later)

This is an interactive introduction to MATLAB. I have provided a sequence of commands for you to type in. The designation `RET` means that you should type the "return" key; this is implicit after a command.

To bring up MATLAB from from the operating system prompt

```
lab%
```

you should type `matlab`

```
lab% matlab RET
```

This will present the prompt

```
>>
```

You are now in MATLAB.

If you are using the X Window system on the Mathematics Department workstations then you can also start MATLAB from the Main Menu by selecting "matlab" from the "Math Applications" submenu. A window should pop up and start MATLAB. When you run MATLAB under the window system, whether you start from the menu or a system prompt, a small MATLAB logo window will pop up while the program is loading and disappear when MATLAB is ready to use.

When you are ready to leave, type `exit`

```
>> exit RET
```

In the course of the tutorial if you get stuck on what a command means type

```
>> help <command name> RET
```

and then try the command again.

You should record the outcome of the commands and experiments in a notebook.

Remark: Depending on the Info reader you are using to navigate this tutorial, you might be able to cut and paste many of the examples directly into Matlab.

Building Matrices

Matlab has many types of matrices which are built into the system. A 7 by 7 matrix with random entries is produced by typing

```
rand(7)
```

You can generate random matrices of other sizes and get help on the `rand` command within matlab:

```
rand(2,5)

help rand
```

Another special matrix, called a Hilbert matrix, is a standard example in numerical linear algebra.

```
hilb(5)

help hilb
```

A 5 by 5 magic square is given by the next command:

```
magic(5)

help magic
```

A magic square is a square matrix which has equal sums along all its rows and columns. We'll use matrix multiplication to check this property a bit later.

Some of the standard matrices from linear algebra are easily produced:

```
eye(6)

zeros(4,7)

ones(5)
```

You can also build matrices of your own with any entries that you may want.

```
[1   2    3    5    7    9]

[1, 2, 3; 4, 5, 6; 7, 8, 9]

[1   2 RET   3   4 RET 5   6]
```

[Note that if you are using cut-and-paste features of a window system or editor to copy these examples into Matlab then you should *not* use cut-and-paste and the last line above. Type it in by hand, touching the Return or Enter key where you see `RET`, and check to see whether the carriage returns make any difference in

Matlab's output.]

Matlab syntax is convenient for blocked matrices:

```
[eye(2);zeros(2)]

[eye(2);zeros(3)]

[eye(2),ones(2,3)]
```

Did any of the last three examples produce error messages? What is the problem?

Variables

Matlab has built-in variables like `pi`, `eps`, and `ans`. You can learn their values from the Matlab interpreter.

```
pi

eps

help eps
```

At any time you want to know the active variables you can use `who`:

```
who

help  who
```

The variable `ans` will keep track of the last output which was not assigned to another variable.

```
magic(6)

ans

x = ans

x = [x, eye(6)]

x
```

Since you have created a new variable, x, it should appear as an active variable.

```
who
```

To remove a variable, try this:

```
clear x

x

who
```

Functions

```
a = magic(4)
```

Take the transpose of a:

```
a'
```

Note that if the matrix A has complex numbers as entries then the Matlab function taking A to A' will compute the transpose of the conjugate of A rather than the transpose of A.

Other arithmetic operations are easy to perform.

```
3*a

-a

a+(-a)

b = max(a)

max(b)
```

Some Matlab functions can return more than one value. In the case of `max` the interpreter returns the maximum value and also the column index where the maximum value occurs.

```
[m, i] = max(b)

min(a)

b = 2*ones(a)

a*b

a
```

We can use matrix multiplication to check the "magic" property of magic squares.

```
    A = magic(5)

    b = ones(5,1)

    A*b

    v = ones(1,5)

    v*A
```

Matlab has a convention in which a dot in front of an operation usually changes the operation. In the case of multiplication, `a.*b` will perform entry-by-entry multiplication instead of the usual matrix multiplication.

```
a.*b  (there is a dot there!)

x = 5
```

```
x^2

a*a

a^2

a.^2    (another dot)

a

triu(a)

tril(a)

diag(a)

diag(diag(a))

c=rand(4,5)

size(c)

[m,n] = size(c)

m

d=.5-c
```

There are many functions which we apply to scalars which Matlab can apply to both scalars and matrices.

```
sin(d)

exp(d)

log(d)

abs(d)
```

Matlab has functions to round floating point numbers to integers. These are `round`, `fix`, `ceil`, and `floor`. The next few examples work through this set of commands and a couple more arithmetic operations.

```
f=[-.5 .1 .5]

round(f)

fix(f)

ceil(f)

floor(f)

sum(f)

prod(f)
```

Relations and Logical Operations

In this section you should think of 1 as "true" and 0 as "false." The notations &, |, ~ stand for "and,""or," and "not," respectively. The notation == is a check for equality.

```
a=[1 0 1 0]

b=[1  1  0  0]

a==b

a<=b

~a

a&b

a & ~a

a | b

a | ~a
```

There is a function to determine if a matrix has at least one nonzero entry, `any`, as well as a function to determine if all the entries are nonzero, `all`.

```
a

any(a)

c=zeros(1,4)

d=ones(1,4)

any(c)

all(a)

all(d)

e=[a',b',c',d']

any(e)

all(e)

any(all(e))
```

Colon Notation

Matlab offers some powerful methods for creating arrays and for taking them apart.

```
x=-2:1

length(x)

-2:.5:1

-2:.2:1

a=magic(5)

a(2,3)
```

Now we will use the colon notation to select a column of a.

```
a(2,:)

a(:,3)

a

a(2:4,:)

a(:,3:5)

a(2:4,3:5)

a(1:2:5,:)
```

You can put a vector into a row or column position within a.

```
a(:,[1 2 5])

a([2 5],[2 4 5])
```

You can also make assignment statements using a vector or a matrix.

```
b=rand(5)

b([1 2],:)=a([1 2],:)

a(:,[1 2])=b(:,[3 5])

a(:,[1 5])=a(:,[5 1])

a=a(:,5:-1:1)
```

When you a insert a 0-1 vector into the column position then the columns which correspond to 1's are displayed.

```
v=[0 1 0 1 1]

a(:,v)

a(v,:)
```

This has been a sample of the basic MATLAB functions and the matrix manipulation techniques. At the end of the tutorial there is a listing of functions. The functions that you have available will vary slightly from version to version of MATLAB. By typing

```
help
```

you will get access to descriptions of all the Matlab functions.

Miscellaneous Features

You may have discovered by now that MATLAB is case sensitive, that is

```
"a" is not the same as "A."
```

If this proves to be an annoyance, the command

```
casesen
```

will toggle the case sensitivity off and on.

The MATLAB display only shows 5 digits in the default mode. The fact is that MATLAB always keeps and computes in a double precision 16 decimal places and rounds the display to 4 digits. The command

```
format long
```

will switch to display all 16 digits and

```
format short
```

will return to the shorter display. It is also possible to toggle back and forth in the scientific notation display with the commands

```
format  short e
```

and

```
format long e
```

It is not always necessary for MATLAB to display the results of a command to the screen. If you do not want the matrix A displayed, put a semicolon after it, A;. When MATLAB is ready to proceed, the prompt >> will appear. Try this on a matrix right now.

Sometimes you will have spent much time creating matrices in the course of your MATLAB session and you would like to use these same matrices in your next session. You can save these values in a file by typing

```
save filename
```

This creates a file

```
filename.mat
```

which contains the values of the variables from your session. If you do not want to save all variables there are two options. One is to clear the variables off with the command

```
clear a b c
```

which will remove the variables a,b,c. The other option is to use the command

```
save x y z
```

which will save the variables x,y,z in the file filename.mat. The variables can be reloaded in a future session by typing

```
load filename
```

When you are ready to print out the results of a session, you can store the results in a file and print the file from the operating system using the "print" command appropriate for your operating system. The file is created using the command

```
diary filename
```

Once a file name has been established you can toggle the diary with the commands

```
diary on
```

and

```
diary off
```

This will copy anything which goes to the screen (other than graphics) to the specified file. Since this is an ordinary ASCII file, you can edit it later. Discussion of print out for graphics is deferred to the project "Graphics" where MATLAB's graphics commands are presented.

Some of you may be fortunate enough to be using a Macintosh or a Sun computer with a window system that allows you to quickly move in and out of MATLAB for editing, printing, or other processes at the system level. For those of you who are not so fortunate, MATLAB has a feature which allows you to do some of these tasks directly from MATLAB. Let us suppose that you would like to edit a file named myfile.m and that your editor executes on the command ed. The MATLAB command

```
!ed myfile.m
```

will bring up your editor and you can now work in it as you usually would. Obviously the exclamation point is the critical feature here. When you are done editing, exit your editor as you usually would, and you will find that you are back in your MATLAB session. You can use the ! with many operating system commands.

Programming in MATLAB

MATLAB is also a programming language. By creating a file with the extension .m you can easily write and run programs. If you were to create a program file myfile.m in the MATLAB language, then you can make the command myfile from MATLAB and it will run like any other MATLAB function. You do not need to compile the program since MATLAB is an interpretative (not compiled) language. Such a file is called an m-file.

I am going to describe the basic programming constructions. While there are other constructions available, if you master these you will be able to write clear programs.

Assignment

Assignment is the method of giving a value to a variable. You have already seen this in the interactive mode. We write x=a to give the value of a to the value of x. Here is a short program illustrating the use of assignment.

```
function r=mod(a,d)

% r=mod(a,d). If a and d are integers, then
% r is the integer remainder of a after
% division by d. If a and b are integer matrices,
% then r is the matrix of remainders after division
% by corresponding entries. Compare with REM.

r=a-d.*floor(a./d);
```

You should make a file named mod.m and enter this program exactly as it is written. Now assign some integer values for a and d. Run

```
mod(a,d)
```

This should run just like any built-in MATLAB function. Type

```
help mod
```

This should produce the five lines of comments which follow the % signs. The % signs generally indicate that what follows on that line is a comment which will be ignored when the program is being executed. MATLAB will print to the screen the comments which follow the "function" declaration at the top of the file when the help command is used. In this way you can contribute to the help facility provided by MATLAB to quickly determine the behavior of the function. Type

```
type mod
```

This will list out the entire file for your perusal. What does this line program mean? The first line is the "function declaration." In it the name of the function (which is always the same as the name of the file without the extension .m), the input variables (in this case a and d), and the output variables (in this case r) are declared. Next come the "help comments" which we have already discussed. Finally, we come to the meat of the program.

The variable r is being assigned the value of a-d.*floor(a./d); The operations on the right hand side of the assignment have the meaning which you have just been practicing (the / is division) with the "." meaning the entry-wise operation as opposed to a matrix operation. Finally, the ";" prevents printing the answer to the screen before the end of execution. You might try replacing the ";" with a "," and running the program again just to see the difference.

Branching

Branching is the construction

```
if <condition>, <program> end
```

The condition is a MATLAB function usually, but not necessarily with values 0 or 1 (later I will discuss when we can vary from this requirement), and the entire construction allows the execution of the program just in case the value of condition is not 0. If that value is 0, the control moves on to the next program construction. You should keep in mind that MATLAB regards a==b and a<=b as functions with values 0 or 1.

Frequently, this construction is elaborated with

```
if <condition1>, <program1> else <program2> end
```

In this case if condition is 0, then program2 is executed.

Another variation is

```
if <condition1>, <program1>
elseif <condition2>, <program2>
end
```

Now if condition1 is not 0, then program1 is executed, if condition1 is 0 and if condition2 is not 0, then program2 is executed, and otherwise control is passed on to the next construction. Here is a short program to illustrate branching.

```
function b=even(n)

% b=even(n). If n is an even integer, then b=1
% otherwise, b=0.
```

```
if mod(n,2)==0,
   b=1;
   else b=0;
end
```

For Loops

A for loop is a construction of the form

for i=1:n, <program>, end

Here we will repeat program once for each index value i. Here are some sample programs. The first is matrix addition.

```
function c=add(a,b)

% c=add(a,b). This is the function which adds
% the matrices a and b. It duplicates the MATLAB
% function a+b.

[m,n]=size(a);
[k,l]=size(b);
if m~=k | n~=l,
   r='ERROR using add: matrices are not the same size';
   return,
end
c=zeros(m,n);
for i=1:m,
   for j=1:n,
      c(i,j)=a(i,j)+b(i,j);
   end
end
```

The next program is matrix multiplication.

```
function c=mult(a,b)

% c=mult(a,b). This is the matrix product of
% the matrices a and b. It duplicates the MATLAB
% function c=a*b.

[m,n]=size(a);
[k,l]=size(b);
if n~=k,
   c='ERROR using mult: matrices are not compatible
      for multiplication',
   return,
end,
c=zeros(m,l);
for i=1:m,
   for j=1:l,
      for p=1:n,
         c(i,j)=c(i,j)+a(i,p)*b(p,j);
      end
   end
end
```

For both of these programs you should notice the branch construction which follows the size statements. This is included as an error message. In the case of add, an error is made if we attempt to add matrices of different sizes, and in the case of mult it is an error to multiply if the matrix on the left does not have the same number of columns as the number of rows of the the matrix on the right. Had these messages not been included and the error was made, MATLAB would have delivered another error message saying that the index exceeds the matrix dimensions. You will notice in the error message the use of single quotes. The words surrounded by the quotes will be treated as text and sent to the screen as the value of the variable c. Following the message is the command return, which is the directive to send the control back to the function which called add or return to the prompt. I usually only recommend using the return command in the context of an error message. Most MATLAB implementations have an error message function, either errmsg or error, which you might prefer to use.

In the construction

```
for i=1:n, <program>, end
```

the index i may (in fact usually does) occur in some essential way inside the program. MATLAB will allow you to put any vector in place of the vector 1:n in this construction.

Thus the construction

```
for i=[2,4,5,6,10], <program>, end
```

is perfectly legitimate. In this case program will execute 5 times and the values for the variable i during execution are successively, 2,4,5,6,10. The MATLAB developers went one step further. If you can put a vector in, why not put a matrix in? So, for example,

```
for i=magic(7), <program>, end
```

is also legal. Now the program will execute 7 (=number of columns) times, and the values of i used in program will be successively the columns of magic(7).

While Loops

A while loop is a construction of the form

```
while <condition>, <program>, end
```

where condition is a MATLAB function, as with the branching construction. The program program will execute successively as long as the value of condition is not 0. While loops carry an implicit danger in that there is no guarantee in general that you will exit a while loop. Here is a sample program using a while loop.

```
function l=twolog(n)

% l=twolog(n). l is the floor of the base 2
% logarithm of n.

l=0;
m=2;
while m<=n
   l=l+1;
   m=2*m;
end
```

Recursion

Recursion is a devious construction which allows a function to call itself. Here is a simple example of recursion

```
function y=twoexp(n)
```

```
% y=twoexp(n). This is a recursive program for computing
% y=2^n. The program halts only if n is a nonnegative integer.

if n==0, y=1;
   else y=2*twoexp(n-1);
end
```

The program has a branching construction built in. Many recursive programs do. The condition n==0 is the base of the recursion. This is the only way to get the program to stop calling itself. The "else" part is the recursion. Notice how the twoexp(n-1) occurs right there in the program which is defining twoexp(n)! The secret is that it is calling a lower value, n-1, and it will continue to do so until it gets down to n=0. A successful recursion is calling a lower value.

There are several dangers using recursion. The first is that, like while loops, it is possible for the function to call itself forever and never return an answer. The second is that recursion can lead to redundant calculations which, though they may terminate, can be time consuming. The third danger is that while a recursive program is running it needs extra space to accomodate the overhead of the recursion. In numerical calculations on very large systems of equations memory space is frequently at a premium, and it should not be wasted on program overhead. With all of these bad possibilities why use recursion? It is not always bad; only in the hands of an inexperienced user. Recursive programs can be easier to write and read than nonrecursive programs. Some of the future projects illustrate good and poor uses of recursion.

Miscellaneous Programming Items

It is possible to place a matrix valued function as the condition of a branching construction or a while loop. Thus the condition might be a matrix like ones(2),zeros(2), or eye(2). How would a construction like

```
if <condition>, < program1>,
else <program2>, end
```

behave if condition=eye(2)? The program1 will execute if all of the entries of condition are not 0. Thus if condition=magic(2), program1 will execute while if condition=eye(2) control will pass to the "else" part and program2 will execute.

A problematic construction occurs when you have

```
if A ~= B, <program>, end.
```

You would like program to execute if the matrices A and B differ on some entry. Under the convention, program will only execute when they differ on all entries. There are various ways around this. One is the construction

```
if A ==B  else <program>, end
```

which will pass control to the "else" part if A and B differ on at least one entry. Another is to convert A==B into a binary valued function by using all(all(A==B)). The inside all creates a binary vector whose i--th entry is 1 only if the i--th column of A is the same as the i--th column of B. The outside all produces a 1 if all the entries of the vector are 1. Thus if A and B differ on at least one entry, then all(all(A==B))=0. The construction

```
if ~ all(all(A==B)), <program>, end
```

then behaves in the desired way.

Essentially, the same convention holds for the while construction.

```
while <condition>, <program>, end.
```

The program program will execute successively as long as every entry in condition is not 0, and the control passes out of the loop when at least one entry of condition is 0.

Another problem occurs when you have a conjunction of conditions, as in

```
if  <condition1> & < condition2>,
<program>,  end
```

Of course, program will execute if both condition1 and condition2 are nonzero. Suppose that condition1=0 and condition2 causes an error message. This might happen for

```
i<=m &  A(i,j)==0
```

where m is the number of columns of A. If i>m, then you would like to pass the control, but since A(i,j) makes no sense if i>m an error message will be dished up. Here you can nest the conditions.

```
if i<=m,
   if A(i,j)==0,
      <program>
   end
end
```

Scripts

A script is an m-file without the function declaration at the top. A script behaves differently. When you type who you are given a list of the variables which are in force during the current session. Suppose that x is one of those variables. When you write a program using a function file and you use the variable x inside the program, the program will not use the value of x from your session (unless x was one of the input values in the function), rather x will have the value appropriate to the program. Furthermore, unless you declare a new value for x, the program will not change the value of x from the session. This is very convenient since it means that you do not have to worry too much about the session variables while your program is running. All this has happened because of the function declaration. If you do not make that function declaration, then the variables in your session can be altered. Sometimes this is quite useful, but I usually recommend that you use function files.

Suggestions

These are a few pointers about programming and programming in MATLAB in particular.

1) I urge you to use the indented style that you have seen in the above programs. It makes the programs easier to read, the program syntax is easier to check, and it forces you to think in terms of building your programs in blocks.

2) Put lots of comments in your program to tell the reader in plain English what is going on. Some day that reader will be you, and you will wonder what you did.

3) Put error messages in your programs like the ones above. As you go through this manual, your programs will build on each other. Error messages will help you debug future programs.

4) Always structure your output as if it will be the input of another function. For example, if your program has "yes-no" type output, do not have it return the words "yes" and "no," rather return 1 or 0, so that it can be used as a condition for a branch or while loop construction in the future.

5) In MATLAB, try to avoid loops in your programs. MATLAB is optimized to run the built-in functions. For a comparison, see how much faster A*B is over mult(A,B). You will be amazed at how much economy can be achieved with MATLAB functions.

6) If you are having trouble writing a program, get a small part of it running and try to build on that. With reference to 5), write the program first with loops, if necessary, then go back and improve it.

MATLAB demonstrations

Matlab is shipped with a number of demonstration programs. Use `help demos` to find out more about these (the number of demos will depend upon the version of Matlab you have).

Some of the standard demos may be especially useful to users who are beginners in linear algebra:

- demo - Demonstrate some of MATLAB's capabilities.
- matdemo - Introduction to matrix computation in MATLAB.
- rrefmovie - Computation of Reduced Row Echelon Form

Some MATLAB built-in functions

This is a list of functions available in Matlab as of 1984, which should be taken as a quick reminder of the most basic tools available. See the Matlab help screens and excerpts from those screens reprinted in section Some MATLAB function descriptions. In any case, your version of Matlab may vary slightly.

```
intro   <       chol    end     function lu      quit     sprintf
help    >       clc     eps     global   macro   qz       sqrt
demo    =       clear   error   grid     magic   rand     startup
[       &       clg     eval    hess     max     rcond    string
]       |       clock   exist   hold     memory  real     subplot
(       ~       conj    exit    home     mesh    relop    sum
)       abs     contour exp     ident    meta    rem      svd
.       all     cos     expm    if       min     return   tan
,       ans     cumprod eye     imag     nan     round    text
;       any     cumsum  feval   inf      nargin  save     title
%       acos    delete  fft     input    norm    schur    type
!       asin    det     filter  inv      ones    script   what
:       atan    diag    find    isnan    pack    semilogx while
'       atan2   diary   finite  keyboard pause   semilogy who
+       axis    dir     fix     load     pi      setstr   xlabel
-       balance disp    floor   log      plot    shg      ylabel
*       break   echo    flops   loglog   polar   sign     zeros
\       casesen eig     for     logop    prod    sin
/       ceil    else    format  ltifr    prtsc   size
^       chdir   elseif  fprintf ltitr    qr      sort
```

```
acosh       demo        hankel      membrane    print       table1
angle       demolist    hds         menu        quad        table2
asinh       dft         hilb        meshdemo    quaddemo    tanh
atanh       diff        hist        meshdom     quadstep    tek
bar         eigmovie    histogram   mkpp        rank        tek4100
bench       ergo        hp2647      movies      rat         terminal
bessel      etime       humps       nademo      ratmovie    toeplitz
bessela     expm1       idft        nelder      readme      trace
besselh     expm2       ieee        neldstep    residue     translate
besseln     expm3       ifft        nnls        retro       tril
```

```
blanks        feval        ifft2        null         roots        triu
cdf2rdf       fft2         info         num2str      rot90        unmkpp
census        fftshift     inquire      ode23        rratref      vdpol
citoh         fitdemo      int2str      ode45        rratrefmovie versa
cla           fitfun       invhilb      odedemo      rref         vt100
compan        flipx        isempty      orth         rsf2csf      vt240
computer      flipy        kron         pinv         sc2dc        why
cond          funm         length       plotdemo     sg100        wow
conv          gallery      log10        poly         sg200        xterm
conv2         gamma        logm         polyfit      sinh         zerodemo
corr          getenv       logspace     polyline     spline       zeroin
cosh          ginput       matdemo      polymark     sqrtm
ctheorem      gpp          matlab       polyval      square
dc2sc         graphon      mean         polyvalm     std
deconv        hadamard     median       ppval        sun
```

```
addtwopi buttap   cov      fftdemo  freqz    kaiser   specplot
bartlett butter   decimate filtdemo fstab    numf     spectrum
bilinear chebap   denf     fir1     hamming  readme2  triang
blackman chebwin  detrend  fir2     hanning  remez    xcorr
boxcar   cheby    eqnerr2  freqs    interp   remezdd  xcorr2
                                                      yulewalk
```

Some MATLAB function descriptions

These lists are copied from the help screens for MATLAB Version 4.2c (dated Nov 23 1994). Only a few of the summaries are listed -- use Matlab's `help` function to see more.

```
>> help


HELP topics:

matlab/general       -  General purpose commands.
matlab/ops           -  Operators and special characters.
matlab/lang          -  Language constructs and debugging.
matlab/elmat         -  Elementary matrices and matrix manipulation.
matlab/specmat       -  Specialized matrices.
matlab/elfun         -  Elementary math functions.
matlab/specfun       -  Specialized math functions.
matlab/matfun        -  Matrix functions - numerical linear algebra.
matlab/datafun       -  Data analysis and Fourier transform functions.
matlab/polyfun       -  Polynomial and interpolation functions.
matlab/funfun        -  Function functions - nonlinear numerical methods.
matlab/sparfun       -  Sparse matrix functions.
matlab/plotxy        -  Two dimensional graphics.
matlab/plotxyz       -  Three dimensional graphics.
matlab/graphics      -  General purpose graphics functions.
matlab/color         -  Color control and lighting model functions.
matlab/sounds        -  Sound processing functions.
matlab/strfun        -  Character string functions.
matlab/iofun         -  Low-level file I/O functions.
matlab/demos         -  The MATLAB Expo and other demonstrations.
toolbox/chem         -  Chemometrics Toolbox
toolbox/control      -  Control System Toolbox.
fdident/fdident      -  Frequency Domain System Identification Toolbox
fdident/fddemos      -  Demonstrations for the FDIDENT Toolbox
toolbox/hispec       -  Hi-Spec Toolbox
toolbox/ident        -  System Identification Toolbox.
toolbox/images       -  Image Processing Toolbox.
toolbox/local        -  Local function library.
toolbox/mmle3        -  MMLE3 Identification Toolbox.
```

```
mpc/mpccmds          -  Model Predictive Control Toolbox
mpc/mpcdemos         -  Model Predictive Control Toolbox
mutools/commands     -  Mu-Analysis and Synthesis Toolbox.: Commands directory
mutools/subs         -  Mu-Analysis and Synthesis Toolbox -- Supplement
toolbox/ncd          -  Nonlinear Control Design Toolbox.
nnet/nnet            -  Neural Network Toolbox.
nnet/nndemos         -  Neural Network Demonstrations and Applications.
toolbox/optim        -  Optimization Toolbox.
toolbox/robust       -  Robust Control Toolbox.
toolbox/signal       -  Signal Processing Toolbox.
toolbox/splines      -  Spline Toolbox.
toolbox/stats        -  Statistics Toolbox.
toolbox/symbolic     -  Symbolic Math Toolbox.
toolbox/wavbox       - (No table of contents file)
simulink/simulink    -  SIMULINK model analysis and construction functions.
simulink/blocks      -  SIMULINK block library.
simulink/simdemos    -  SIMULINK demonstrations and samples.
toolbox/codegen      -  Real-Time Workshop

For more help on directory/topic, type "help topic".

>> help elmat

 Elementary matrices and matrix manipulation.

 Elementary matrices.
   zeros       - Zeros matrix.
   ones        - Ones matrix.
   eye         - Identity matrix.
   rand        - Uniformly distributed random numbers.
   randn       - Normally distributed random numbers.
   linspace    - Linearly spaced vector.
   logspace    - Logarithmically spaced vector.
   meshgrid    - X and Y arrays for 3-D plots.
   :           - Regularly spaced vector.

 Special variables and constants.
   ans         - Most recent answer.
   eps         - Floating point relative accuracy.
   realmax     - Largest floating point number.
   realmin     - Smallest positive floating point number.
   pi          - 3.1415926535897....
   i, j        - Imaginary unit.
   inf         - Infinity.
   NaN         - Not-a-Number.
   flops       - Count of floating point operations.
   nargin      - Number of function input arguments.
   nargout     - Number of function output arguments.
   computer    - Computer type.
   isieee      - True for computers with IEEE arithmetic.
   isstudent   - True for the Student Edition.
   why         - Succinct answer.
   version     - MATLAB version number.

 Time and dates.
   clock       - Wall clock.
   cputime     - Elapsed CPU time.
   date        - Calendar.
   etime       - Elapsed time function.
   tic, toc    - Stopwatch timer functions.

 Matrix manipulation.
   diag        - Create or extract diagonals.
   fliplr      - Flip matrix in the left/right direction.
   flipud      - Flip matrix in the up/down direction.
```

```
    reshape     - Change size.
    rot90       - Rotate matrix 90 degrees.
    tril        - Extract lower triangular part.
    triu        - Extract upper triangular part.
    :           - Index into matrix, rearrange matrix.

>> help specmat

 Specialized matrices.

    compan      - Companion matrix.
    gallery     - Several small test matrices.
    hadamard    - Hadamard matrix.
    hankel      - Hankel matrix.
    hilb        - Hilbert matrix.
    invhilb     - Inverse Hilbert matrix.
    kron        - Kronecker tensor product.
    magic       - Magic square.
    pascal      - Pascal matrix.
    rosser      - Classic symmetric eigenvalue test problem.
    toeplitz    - Toeplitz matrix.
    vander      - Vandermonde matrix.
    wilkinson   - Wilkinson's eigenvalue test matrix.

>> help elfun

 Elementary math functions.

 Trigonometric.
    sin         - Sine.
    sinh        - Hyperbolic sine.
    asin        - Inverse sine.
    asinh       - Inverse hyperbolic sine.
    cos         - Cosine.
    cosh        - Hyperbolic cosine.
    acos        - Inverse cosine.
    acosh       - Inverse hyperbolic cosine.
    tan         - Tangent.
    tanh        - Hyperbolic tangent.
    atan        - Inverse tangent.
    atan2       - Four quadrant inverse tangent.
    atanh       - Inverse hyperbolic tangent.
    sec         - Secant.
    sech        - Hyperbolic secant.
    asec        - Inverse secant.
    asech       - Inverse hyperbolic secant.
    csc         - Cosecant.
    csch        - Hyperbolic cosecant.
    acsc        - Inverse cosecant.
    acsch       - Inverse hyperbolic cosecant.
    cot         - Cotangent.
    coth        - Hyperbolic cotangent.
    acot        - Inverse cotangent.
    acoth       - Inverse hyperbolic cotangent.

 Exponential.
    exp         - Exponential.
    log         - Natural logarithm.
    log10       - Common logarithm.
    sqrt        - Square root.

 Complex.
    abs         - Absolute value.
    angle       - Phase angle.
    conj        - Complex conjugate.
    imag         - Complex imaginary part.
    real         - Complex real part.

  Numeric.
    fix          - Round towards zero.
    floor        - Round towards minus infinity.
    ceil         - Round towards plus infinity.
    round        - Round towards nearest integer.
    rem          - Remainder after division.
    sign         - Signum function.

 >> help specfun

  Specialized math functions.

    besselj      - Bessel function of the first kind.
    bessely      - Bessel function of the second kind.
    besseli      - Modified Bessel function of the first kind.
    besselk      - Modified Bessel function of the second kind.
    beta         - Beta function.
    betainc      - Incomplete beta function.
    betaln       - Logarithm of beta function.
    ellipj       - Jacobi elliptic functions.
    ellipke      - Complete elliptic integral.
    erf          - Error function.
    erfc         - Complementary error function.
    erfcx        - Scaled complementary error function.
    erfinv       - Inverse error function.
    expint       - Exponential integral function.
    gamma        - Gamma function.
    gcd          - Greatest common divisor.
    gammainc     - Incomplete gamma function.
    lcm          - Least common multiple.
    legendre     - Associated Legendre function.
    gammaln      - Logarithm of gamma function.
    log2         - Dissect floating point numbers.
    pow2         - Scale floating point numbers.
    rat          - Rational approximation.
    rats         - Rational output.
    cart2sph     - Transform from Cartesian to spherical coordinates.
    cart2pol     - Transform from Cartesian to polar coordinates.
    pol2cart     - Transform from polar to Cartesian coordinates.
    sph2cart     - Transform from spherical to Cartesian coordinates.

 >> help matfun

  Matrix functions - numerical linear algebra.

  Matrix analysis.
    cond         - Matrix condition number.
    norm         - Matrix or vector norm.
    rcond        - LINPACK reciprocal condition estimator.
    rank         - Number of linearly independent rows or columns.
    det          - Determinant.
    trace        - Sum of diagonal elements.
    null         - Null space.
    orth         - Orthogonalization.
    rref         - Reduced row echelon form.

  Linear equations.
    \ and /      - Linear equation solution; use "help slash".
    chol         - Cholesky factorization.
    lu           - Factors from Gaussian elimination.
    inv          - Matrix inverse.
    qr           - Orthogonal-triangular decomposition.
    qrdelete    - Delete a column from the QR factorization.
    qrinsert    - Insert a column in the QR factorization.
    nnls        - Non-negative least-squares.
    pinv        - Pseudoinverse.
    lscov       - Least squares in the presence of known covariance.

  Eigenvalues and singular values.
    eig         - Eigenvalues and eigenvectors.
    poly        - Characteristic polynomial.
    polyeig     - Polynomial eigenvalue problem.
    hess        - Hessenberg form.
    qz          - Generalized eigenvalues.
    rsf2csf     - Real block diagonal form to complex diagonal form.
    cdf2rdf     - Complex diagonal form to real block diagonal form.
    schur       - Schur decomposition.
    balance     - Diagonal scaling to improve eigenvalue accuracy.
    svd         - Singular value decomposition.

  Matrix functions.
    expm        - Matrix exponential.
    expm1       - M-file implementation of expm.
    expm2       - Matrix exponential via Taylor series.
    expm3       - Matrix exponential via eigenvalues and eigenvectors.
    logm        - Matrix logarithm.
    sqrtm       - Matrix square root.
    funm        - Evaluate general matrix function.

>> help general

  General purpose commands.
  MATLAB Toolbox  Version 4.2a 25-Jul-94

  Managing commands and functions.
    help        - On-line documentation.
    doc         - Load hypertext documentation.
    what        - Directory listing of M-, MAT- and MEX-files.
    type        - List M-file.
    lookfor     - Keyword search through the HELP entries.
    which       - Locate functions and files.
    demo        - Run demos.
    path        - Control MATLAB's search path.

  Managing variables and the workspace.
    who         - List current variables.
    whos        - List current variables, long form.
    load        - Retrieve variables from disk.
    save        - Save workspace variables to disk.
    clear       - Clear variables and functions from memory.
    pack        - Consolidate workspace memory.
    size        - Size of matrix.
    length      - Length of vector.
    disp        - Display matrix or text.

  Working with files and the operating system.
    cd          - Change current working directory.
    dir         - Directory listing.
    delete      - Delete file.
    getenv      - Get environment value.
    !           - Execute operating system command.
    unix        - Execute operating system command & return result.
    diary       - Save text of MATLAB session.

  Controlling the command window.
    cedit       - Set command line edit/recall facility parameters.
    clc         - Clear command window.
    home        - Send cursor home.
    format      - Set output format.
    echo        - Echo commands inside script files.
    more        - Control paged output in command window.

  Starting and quitting from MATLAB.
    quit        - Terminate MATLAB.
    startup     - M-file executed when MATLAB is invoked.
    matlabrc    - Master startup M-file.

  General information.
    info        - Information about MATLAB and The MathWorks, Inc.
    subscribe   - Become subscribing user of MATLAB.
    hostid      - MATLAB server host identification number.
    whatsnew    - Information about new features not yet documented.
    ver         - MATLAB, SIMULINK, and TOOLBOX version information.

 >> help funfun

  Function functions - nonlinear numerical methods.

    ode23       - Solve differential equations, low order method.
    ode23p      - Solve and plot solutions.
    ode45       - Solve differential equations, high order method.
    quad        - Numerically evaluate integral, low order method.
    quad8       - Numerically evaluate integral, high order method.
    fmin        - Minimize function of one variable.
    fmins       - Minimize function of several variables.
    fzero       - Find zero of function of one variable.
    fplot       - Plot function.

  See also The Optimization Toolbox, which has a comprehensive
  set of function functions for optimizing and minimizing functions.

 >> help polyfun

  Polynomial and interpolation functions.

  Polynomials.
    roots       - Find polynomial roots.
    poly        - Construct polynomial with specified roots.
    polyval     - Evaluate polynomial.
    polyvalm    - Evaluate polynomial with matrix argument.
    residue     - Partial-fraction expansion (residues).
    polyfit     - Fit polynomial to data.
    polyder     - Differentiate polynomial.
    conv        - Multiply polynomials.
    deconv      - Divide polynomials.

  Data interpolation.
    interp1     - 1-D interpolation (1-D table lookup).
    interp2     - 2-D interpolation (2-D table lookup).
    interpft    - 1-D interpolation using FFT method.
    griddata    - Data gridding.

  Spline interpolation.
    spline      - Cubic spline data interpolation.
    ppval       - Evaluate piecewise polynomial.

 >> help ops

  Operators and special characters.

   Char    Name                         HELP topic
```

```
    +      Plus                         arith
    -      Minus                        arith
    *      Matrix multiplication        arith
    .*     Array multiplication         arith
    ^      Matrix power                 arith
    .^     Array power                  arith

    \      Backslash or left division   slash
    /      Slash or right division      slash
    ./     Array division               slash
    kron   Kronecker tensor product     kron

    :      Colon                        colon

    ( )    Parentheses                  paren
    [ ]    Brackets                     paren

    .      Decimal point                punct
    ..     Parent directory             punct
    ...    Continuation                 punct
    ,      Comma                        punct
    ;      Semicolon                    punct
    %      Comment                      punct
    !      Exclamation point            punct
    '      Transpose and quote          punct
    =      Assignment                   punct

    ==     Equality                     relop
    <,>    Relational operators         relop
    &      Logical AND                  relop
    |      Logical OR                   relop
    ~      Logical NOT                  relop
    xor    Logical EXCLUSIVE OR         xor

  Logical characteristics.
    exist       - Check if variables or functions are defined.
    any         - True if any element of vector is true.
    all         - True if all elements of vector are true.
    find        - Find indices of non-zero elements.
    isnan       - True for Not-A-Number.
    isinf       - True for infinite elements.
    finite      - True for finite elements.
    isempty     - True for empty matrix.
    isreal      - True for real matrix.
    issparse    - True for sparse matrix.
    isstr       - True for text string.
    isglobal    - True for global variables.

>> help lang

  Language constructs and debugging.

  MATLAB as a programming language.
    script      - About MATLAB scripts and M-files.
    function    - Add new function.
    eval        - Execute string with MATLAB expression.
    feval       - Execute function specified by string.
    global      - Define global variable.
    nargchk     - Validate number of input arguments.
    lasterr     - Last error message.

  Control flow.
    if          - Conditionally execute statements.
    else        - Used with IF.
    elseif      - Used with IF.
    end         - Terminate the scope of FOR, WHILE and IF statements.
    for         - Repeat statements a specific number of times.
    while       - Repeat statements an indefinite number of times.
    break       - Terminate execution of loop.
    return      - Return to invoking function.
    error       - Display message and abort function.

  Interactive input.
    input       - Prompt for user input.
    keyboard    - Invoke keyboard as if it were a Script-file.
    menu        - Generate menu of choices for user input.
    pause       - Wait for user response.
    uimenu      - Create user interface menu.
    uicontrol   - Create user interface control.

  Debugging commands.
    dbstop     - Set breakpoint.
    dbclear    - Remove breakpoint.
    dbcont     - Resume execution.
    dbdown     - Change local workspace context.
    dbstack    - List who called whom.
    dbstatus   - List all breakpoints.
    dbstep     - Execute one or more lines.
    dbtype     - List M-file with line numbers.
    dbup       - Change local workspace context.
    dbquit     - Quit debug mode.
    mexdebug   - Debug MEX-files.

 >> help plotxy
  Two dimensional graphics.

  Elementary X-Y graphs.
    plot      - Linear plot.
    loglog    - Log-log scale plot.
    semilogx  - Semi-log scale plot.
    semilogy  - Semi-log scale plot.
    fill      - Draw filled 2-D polygons.

  Specialized X-Y graphs.
    polar     - Polar coordinate plot.
    bar       - Bar graph.
    stem      - Discrete sequence or "stem" plot.
    stairs    - Stairstep plot.
    errorbar  - Error bar plot.
    hist      - Histogram plot.
    rose      - Angle histogram plot.
    compass   - Compass plot.
    feather   - Feather plot.
    fplot     - Plot function.
    comet     - Comet-like trajectory.

  Graph annotation.
    title     - Graph title.
    xlabel    - X-axis label.
    ylabel    - Y-axis label.
    text      - Text annotation.
    gtext     - Mouse placement of text.
    grid      - Grid lines.

  See also PLOTXYZ, GRAPHICS.

 >> help plotxyz

  Three dimensional graphics.
  Line and area fill commands.
    plot3      - Plot lines and points in 3-D space.
    fill3      - Draw filled 3-D polygons in 3-D space.
    comet3     - 3-D comet-like trajectories.

  Contour and other 2-D plots of 3-D data.
    contour    - Contour plot.
    contour3   - 3-D contour plot.
    clabel     - Contour plot elevation labels.
    contourc   - Contour plot computation (used by contour).
    pcolor     - Pseudocolor (checkerboard) plot.
    quiver     - Quiver plot.

  Surface and mesh plots.
    mesh       - 3-D mesh surface.
    meshc      - Combination mesh/contour plot.
    meshz      - 3-D Mesh with zero plane.
    surf       - 3-D shaded surface.
    surfc      - Combination surf/contour plot.
    surfl      - 3-D shaded surface with lighting.
    waterfall  - Waterfall plot.

  Volume visualization.
    slice      - Volumetric visualization plots.

  Graph appearance.
    view       - 3-D graph viewpoint specification.
    viewmtx    - View transformation matrices.
    hidden     - Mesh hidden line removal mode.
    shading    - Color shading mode.
    axis       - Axis scaling and appearance.
    caxis      - Pseudocolor axis scaling.
    colormap   - Color look-up table.

  Graph annotation.
    title      - Graph title.
    xlabel     - X-axis label.
    ylabel     - Y-axis label.
    zlabel     - Z-axis label for 3-D plots.
    text       - Text annotation.
    gtext      - Mouse placement of text.
    grid       - Grid lines.

  3-D objects.
    cylinder   - Generate cylinder.
    sphere     - Generate sphere.

  See also COLOR, PLOTXY, GRAPHICS.

>> help strfun

  Character string functions.

  General.
    strings     - About character strings in MATLAB.
    abs         - Convert string to numeric values.
    setstr      - Convert numeric values to string.
    isstr       - True for string.
    blanks      - String of blanks.
    deblank     - Remove trailing blanks.
    str2mat     - Form text matrix from individual strings.
    eval        - Execute string with MATLAB expression.

  String comparison.
    strcmp      - Compare strings.
    findstr     - Find one string within another.
    upper       - Convert string to uppercase.
    lower       - Convert string to lowercase.
    isletter    - True for letters of the alphabet.
    isspace     - True for white space characters.
    strrep      - Replace a string with another.
    strtok      - Find a token in a string.

  String to number conversion.
    num2str     - Convert number to string.
    int2str     - Convert integer to string.
    str2num     - Convert string to number.
    mat2str     - Convert matrix to string.
    sprintf     - Convert number to string under format control.
    sscanf      - Convert string to number under format control.

  Hexadecimal to number conversion.
    hex2num     - Convert hex string to IEEE floating point number.
    hex2dec     - Convert hex string to decimal integer.
    dec2hex     - Convert decimal integer to hex string.
```