Matlab 学习笔记

(http://www.math.ucsd.edu/~bdriver/21d-s99/matlab-primer.html)

1 Entering matrices

Matrices can be introduced into MATLAB in several different ways:

- •Entered by an explicit list of elements,
- •Generated by built-in statements and functions,
- •Created in M-files (see sections 12 and 14 below),
- •Loaded from external data files (see User's Guide).

A(2,3) denotes the entry in the second row, third column of matrix **A** and **x(3)** denotes the third coordinate of vector **x**.

2 Matrix operations, array operations

The following matrix operations are available in MATLAB:

+	addition
-	subtraction
*	multiplication
^	power
'	transpose
\	left division
/	right division

x = A \ b is the solution of **A * x = b** and, resp.,

x = b / A is the solution of **x * A = b**.

The matrix operations of addition and subtraction already operate entry-wise but the other matrix operations given above do not---they are ***matrix* operations**.

It is important to observe that these other operations, *****, **^**, ****, and **/**, can be made to operate entry-wise by **preceding them by a period**. For example, either [1,2,3,4].*[1,2,3,4] or [1,2,3,4].^2 will yield [1,4,9,16].

3 Statements, expressions, and variables; saving a session

A statement can be continued to the next line with **three or more periods** followed by a carriage return.

MATLAB is case-sensitive in the names of commands, functions, and variables. For example, solveUT is not the same as solveut.

The command `who` will list the variables currently in the workspace. A variable can be cleared from the workspace with the command `clear variablename`. The command `clear` alone will clear all nonpermanent variables.

Invoking the command `save` before exiting causes all variables to be written to a non-human-readable diskfile named `matlab.mat`.

The command `load` will restore the workspace to its former state.

4 Matrix building functions

Convenient matrix building functions are

eye	identity matrix
zeros	matrix of zeros
ones	matrix of ones

diag	see below
triu	upper triangular part of a matrix
tril	lower triangular part of a matrix
rand	randomly generated matrix
hilb	Hilbert matrix
magic	magic square
toeplitz	see `help toeplitz`

For example, `zeros(m,n)` produces an m-by-n matrix of zeros and `zeros(n)`produces an n-by-n one; if A is a matrix, then `zeros(A)`produces a matrix of zeros of the same size as A.

If x is a vector, `diag(x)` is the diagonal matrix with x down the diagonal; if A is a square matrix, then `diag(A)` is a vector consisting of the diagonal of A. What is `diag(diag(A))`? Try it.

Matrices can be built from blocks. For example, if A is a 3-by-3 matrix, then

```
B = [A, zeros(3,2); zeros(2,3), eye(2)]
```

will build a certain 5-by-5 matrix. Try it.

5 For, while, if --- and relations

```
x = []; for i = 1:n, x=[x,i^2], end
```

will produce a certain n-vector and the statement.

```
x = []; for i = n:-1:1, x=[x,i^2], end
```

will produce the same vector in reverse order.

```
for i = 1:m
for j = 1:n
H(i, j) = 1/(i+j-1);
end
end
H
```

will produce and print to the screen the m-by-n hilbert matrix.

The general form of a `while` loop is

```
while relation
statements
end
```

The statements will be repeatedly executed as long as the relation remains true.

```
n = 0;
while 2^n < a
n = n + 1;
end
n
```

The general form of a simple `if` statement is

```
if relation
statements
end
```

The statements will be executed only if the relation is true. Multiple branching is also possible, as is illustrated by

```
if n < 0
parity = 0;
elseif rem(n,2) == 0
parity = 2;
else
parity = 1;
end
```

In two-way branching the `elseif` portion would, of course, be omitted.

The `relational operators` in MATLAB are

<	less than
>	greater than

<=	less than or equal
>=	greater than or equal
==	equal
~=	not equal.

`logical operators`

&	and
|	or
~	not.

When applied to matrices of the same size, a relation is a matrix of 0's and 1's giving the value of the relation between corresponding entries. Try `a = rand(5), b = triu(a), a == b`.

A relation between matrices is interpreted by `while` and `if` to be true if each entry of the relation matrix is nonzero. Hence, if you wish to execute statement when matrices A and B are equal you could type

```
if A == B
statement
end
```

but if you wish to execute statement when A and B are not equal, you would type

```
if any(any(A ~= B))
statement
end
```

or, more simply,

```
if A == B else
statement
end
```

Note that the seemingly obvious

```
if A ~= B, statement, end
```

will not give what is intended since statement would execute only if each of the corresponding entries of A and B differ.

Two any's are required above sinceany is a vector operator.

The for statement permits any matrix to be used instead of 1:n. See the User's Guide for details of how this feature expands the power of the for statement.

6 Scalar functions

Certain MATLAB functions operate essentially on scalars, but operate element-wise when applied to a matrix. The most common such functions are

sin	asin	exp	abs	round
cos	acos	log (natural log)	sqrt	floor
tan	atan	rem (remainder)	sign	ceil

7 Vector functions

Other MATLAB functions operate essentially on a vector (row or column), but act on an m-by-n matrix (m >= 2) in a column-by-column fashion to produce a row vector containing the results of their application to each column. Row-by-row action can be obtained by using the transpose; for example, `mean(A')'`. A few of these functions are

max	sum	median	any
min	prod	mean	all
sort		std	

For example, the maximum entry in a matrix A is given by `max(max(A))` rather than `max(A)`. Try it.

8 Matrix functions

Much of MATLAB's power comes from its matrix functions. The most useful ones are

eig	eigenvalues and eigenvectors
chol	cholesky factorization
svd	singular value decomposition
inv	inverse
lu	LU factorization
qr	QR factorization
hess	hessenberg form
schur	schur decomposition
rref	reduced row echelon form
expm	matrix exponential
sqrtm	matrix square root
poly	characteristic polynomial
det	determinant
size	size
norm	1-norm, 2-norm, F-norm, <infinity>-norm*
cond	condition number in the 2-norm
rank	rank

MATLAB functions may have single or multiple output arguments. For example,

```
y = eig(A), or simply eig(A)
```

produces a column vector containing the eigenvalues of A while

```
[U,D] = eig(A)
```

produces a matrix U whose columns are the eigenvectors of A and a diagonal matrix D with the eigenvalues of A on its diagonal. Try it.

9 Submatrices and colon notation

Vectors and submatrices are often used in MATLAB to achieve fairly complex data manipulation effects. "Colon notation" (which is used both to generate vectors and reference submatrices) and subscripting by vectors are keys to efficient manipulation of these objects.

The expression `1:5` (met earlier in for statements) is actually the row vector [1 2 3 4 5]. The numbers need not be integers nor the increment one. For example,

```
0.2:0.2:1.2
```

gives [0.2, 0.4, 0.6, 0.8, 1.0, 1.2], and

```
5:-1:1
```

gives [5 4 3 2 1]. The following statements will, for example, generate a table of sines. Try it.

```
x = [0.0:0.1:2.0]';
y = sin(x);
[x y]
```

Note that since sin operates entry-wise, it produces a vector y from the vector x.

The colon notation can be used to **access submatrices of a matrix**. For example,

```
A(1:4,3)
```

is the column vector consisting of the first four entries of the third column of A. A colon by itself denotes an entire row or column:

```
A(:,3)
```

is the third column of A, and A(1:4,:) is the first four rows. Arbitrary integral vectors can be used as subscripts:

```
A(:,[2 4])
```

contains as columns, columns 2 and 4 of A. Such subscripting can be used on both sides of an assignment statement:

```
A(:,[2 4 5]) = B(:,1:3)
```

replaces columns 2,4,5 of b with the first three columns of B. Note that the entire altered matrix A is printed and assigned. Try it.

Columns 2 and 4 of A can be multiplied on the right by the 2-by-2 matrix [1 2;3 4]:

```
A(:,[2,4]) = A(:,[2,4]) * [1 2;3 4]
```

Once again, the entire altered matrix is printed and assigned.

If x is an n-vector, what is the effect of the statement `x = x(n:-1:1)`? Try it.

To appreciate the usefulness of these features, compare these MATLAB statements with a Pascal, FORTRAN, or C routine to effect the same.

10 M-files

There are two types of M-files: `script files` and `function files`.

Script files. A script file consists of a sequence of normal MATLAB statements. If the file has the filename, say, rotate.m, then the MATLAB command rotate will cause the statements in the file to be executed. Variables in a script file are global and will change the value of variables of the same name in the environment of the current MATLAB session.

Script files are often used to enter data into a large matrix; in such a file, entry errors can be easily edited out. If, for example, one enters in a diskfile data.m

A = [

1 2 3 4

5 6 7 8

];

then the MATLAB statement data will cause the assignment given in data.m to be carried out.

An M-file can reference other M-files, including referencing itself recursively.

Function files. Variables in a function file are by default local. However, version 4.0 permits a variable to be declared global.

We first illustrate with a simple example of a function file.

```
function a = randint(m,n)
%RANDINT Randomly generated integral matrix.
% randint(m,n) returns an m-by-n such matrix with entries
% between 0 and 9.
a = floor(10*rand(m,n));
```

A more general version of this function is the following:

```
function a = randint(m,n,a,b)
%RANDINT Randomly generated integral matrix.
% randint(m,n) returns an m-by-n such matrix with entries
% between 0 and 9.
```

```
% rand(m,n,a,b) return entries between integers a and b.
if nargin < 3, a = 0, b = 9; end
a = floor((b-a+1)*rand(m,n)) + a;
```

Note that use of nargin ("number of input arguments") permits one to set a default value of an omitted input variable---such as a and b in the example.

A function may also have multiple output arguments. For example:

```
function [mean, stdev] = stat(x)
% STAT Mean and standard deviation
% For a vector x, stat(x) returns the
% mean and standard deviation of x.
% For a matrix x, stat(x) returns two row vectors containing,
% respectively, the mean and standard deviation of each column.
[m n] = size(x);
if m == 1
m = n; % handle case of a row vector
end
mean = sum(x)/m;
stdev = sqrt(sum(x.^2)/m - mean.^2);
```

Once this is placed in a diskfile stat.m, a MATLAB command [xm, xd] = stat(x), for example, will assign the mean and standard deviation of the entries in the vector x to xm and xd, respectively. Single assignments can also be made with a function having multiple output arguments. For example, xm = stat(x) (no brackets needed around xm) will assign the mean of x to xm.

The following function, which gives the greatest common divisor of two integers via the Euclidean algorithm, illustrates the use of an error message (see the next section).

```
function a = gcd(a,b)
% GCD Greatest common divisor
% gcd(a,b) is the greatest common divisor of
% the integers a and b, not both zero.
a = round(abs(a)); b = round(abs(b));
```

```
if a == 0 & b == 0
error('The gcd is not defined when both numbers are zero')
else
while b ~= 0
r = rem(a,b)
a = b; b = r;
end
end
```

Some more advanced features are illustrated by the following function. As noted earlier, some of the input arguments of a function---such as tol in the example, may be made optional through use of nargin ("number of input arguments"). The variable nargout can be similarly used.

```
function [b, steps] = bisect(fun, x, tol)
%BISECT Zero of a function of one variable via the bisection method.
% bisect(fun,x) returns a zero of the function. fun is a string
% containing the name of a real-valued function of a single
% real variable; ordinarily functions are defined in M-files.
% x is a starting guess. The value returned is near a point
% where fun changes sign. For example,
% bisect('sin',3) is pi. Note the quotes around sin.
%
% An optional third input argument sets a tolerence for the
% relative accuracy of the result. The default is eps.
% An optional second output argument gives a matrix containing a
% trace of the steps; the rows are of form [c f(c)].

% Initialization
if nargin < 3, tol = eps; end
trace = (nargout == 2);
if x ~= 0, dx = x/20; else, dx = 1/20; end
a = x - dx; fa = feval(fun,a);
b = x + dx; fb = feval(fun,b);
```

```
% Find change of sign.
while (fa > 0) == (fb > 0)
dx = 2.0*dx;
a = x - dx; fa = feval(fun,a);
if (fa > 0) ~= (fb > 0), break, end
b = x + dx; fb = feval(fun,b);
end
if trace, steps = [a fa; b fb] end

% Main loop
while abs(b - a) > 2.0*tol*max(abs(b),1.0)
c = a + 0.5*(b - a); fc = feval(fun,c);
if trace, steps = [steps; [c fc]];
if (fb > 0) == (fc > 0)
b = c; fb = fc;
else
a = c; fa = fc;
end
end
```

Some of MATLAB's functions are built-in while others are distributed as M-files. The actual listing of any M-file---MATLAB's or your own---can be viewed with the MATLAB command `type functionname`. Try entering `type eig`, `type vander`, and `type rank`.

11 Text strings, error messages, input

Text strings are entered into MATLAB surrounded by single quotes. For example,

```
s = 'This is a test'
```

Text strings can be displayed with the function `disp`. For example:

```
disp('this message is hereby displayed')
```

Error messages are best displayed with the function `error`

```
error('Sorry, the matrix must be symmetric')
```

since when placed in an M-File, it causes execution to exit the M-file.s

In an M-file the user can be prompted to interactively enter input data with the function `input`. When, for example, the statement

```
iter = input('Enter the number of iterations: ')
```

12 Output format

While all computations in MATLAB are performed in double precision, the format of the displayed output can be controlled by the following commands.

`format short`	fixed point with 4 decimal places (the default)
`format long`	fixed point with 14 decimal place
`format short e`	scientific notation with 4 decimal places
`format long e`	scientific notation with 15 decimal places

Once invoked, the chosen format remains in effect until changed.

The command `format compact` will suppress most blank lines allowing more information to be placed on the screen or page. It is independent of the other format commands.

13 Hardcopy

Hardcopy is most easily obtained with the `diary` command. The command

```
diary filename
```

causes what appears subsequently on the screen (except graphics) to be written to the named diskfile (if the filename is omitted it will be written to a default file named diary) until one gives the command `diary off`; the command `diary on` will cause writing to the file to resume, etc. When finished, you can edit the file as desired and print it out on the local system. The !-feature (see section 14) will permit you to edit and print the file without leaving MATLAB.

14 Graphics

When in the graphics screen, pressing any key will return you to the command screen while the command shg (show graph) will then return you to the current graphics screen.

You can draw the graph of y = e^(-x^2) over the interval -1.5 to 1.5 as follows:

```
x = -1.5:.01:1.5; y = exp(-x.^2); plot(x,y)
```

Note that one must precede ^ by a period to ensure that it operates entrywise.

Plots of parametrically defined curves can also be made. Try, for example,

```
t=0:.001:2*pi; x=cos(3*t); y=sin(2*t); plot(x,y)
```

The command grid will place grid lines on the current graph.

The graphs can be given titles, axes labeled, and text placed within the graph with the following commands which take a string as an argument.

title	graph title
xlabel	x-axis label
ylabel	y-axis label
gtext	interactively-positioned text
text	position text at specified coordinates

By default, the axes are auto-scaled. This can be overridden by the command axis.

Two ways to make multiple plots on a single graph are illustrated by

```
x=0:.01:2*pi;y1=sin(x);y2=sin(2*x);y3=sin(4*x);plot(x,y1,x,y2,x,y3)
```

and by forming a matrix Y containing the functional values as columns

```
x=0:.01:2*pi; Y=[sin(x)', sin(2*x)', sin(4*x)']; plot(x,Y)
```

Another way is with hold. The command hold freezes the current graphics screen so that subsequent plots are superimposed on it. Entering hold again releases the "hold." The commands

hold on and hold off are also available in version 4.0. One can override the default linetypes and

pointtypes. For example,

```
x=0:.01:2*pi; y1=sin(x); y2=sin(2*x); y3=sin(4*x);
plot(x,y1,'--',x,y2,':',x,y3,'+')
```

renders a dashed line and dotted line for the first two graphs while for the third the symbol + is placed at each node. The line- and mark-types are

```
Linetypes: solid (-), dashed (\tt--). dotted (:), dashdot (-.)
Marktypes: point (.), plus (+), star (*), circle (o), x-mark (x)
```

See `help plot` for line and mark colors.

The command `subplot` can be used to partition the screen so that up to four plots can be viewed simultaneously. See `help subplot`.

In version 4.0, the 3-D graphics capabilities of MATLAB have been considerably expanded. Consult the on-line help for `plot3`, `mesh`, and `surf`.