3-D mesh plots

Three dimensional mesh surface plots are drawn with the function mesh. The command mesh(z) creates a three-dimensional perspective plot of the elements of the matrix z. The mesh surface is defined by the z-coordinates of points above a rectangular grid in the x-y plane. Try `mesh(eye(10))`.

To draw the graph of a function z = f(x,y) over a rectangle, one first defines vectors xx and yy which give partitions of the sides of the rectangle. With the function meshdom (mesh domain; called meshgrid in version 4.0) one then creates a matrix x, each row of which equals xx and whose column length is the length of yy, and similarly a matrix y, each column of which equals yy, as follows:

```
[x,y] = meshdom(xx,yy);
```

One then computes a matrix z, obtained by evaluating f entrywise over the matrices x and y, to which mesh can be applied.

You can, for example, draw the graph of z = e^(-x^2-y^2) over the square [-2,2] x [-2,2] as follows (try it):

```
xx = -2:.1:2;
yy = xx;
[x,y] = meshdom(xx,yy);
z = exp(-x.^2 - y.^2);
mesh(z)
```

Graphics:

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)

Printing Figures:

```
print -dps <filename>
```

prints the current figure into the postscript file specified.

Graphics

```
x = -4:.01:4; y = sin(x); plot(x,y);
x = -1.5:.01:1.5; y = exp(-x.^2); plot(x,y);
```

Parametric Plots:

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

Other commands:

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

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)
```

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.

Function files. Function files provide extensibility to MATLAB.

example:

```
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;
```

Colon Notation

[1:5] is the same as [1, 2, 3, 4, 5]

similarly,

0.2:0.2:1.2 is the same as [0.2, 0.4, 0.6, 0.8, 1.0, 1.2]

you can access matrices using the colon nocation:

```
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)
```

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

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
```

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
```

Relations:

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

Relations may be connected or quantified by the logical operators

&: and
|: or
~: not.

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
```

While. The general form of a while loop is

while relation
statements
end

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

If. The general form of a simple if statement is

if relation
statements
end

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

Statements, expressions, and variables

variable = expression;
or simply
expression

A statement is normally terminated with the carriage return.

Statements terminated by semi-colon dont print evaluated value.

The command save can be used to save the state of variables in matlab.
The command load retrieves the state.

Matlab Language Constructs:

For, while, if --- and relations

```
x = [];
for i = 1:10,
    x=[x,i^2],
end;
```

also try

```
x = [];
for i = 10:-1:1,
    x=[x,i^2],
end;
```

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

Try:

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

Matrix operations, array operations

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

x = A \ b; is the solution of A * x = b

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

Example: solving a system:

```
A = [
      1 2 3
      4 5 3
      7 8 5 ];

b = [5; 6; 7];
```

(Solution is x = [-4.3333; 4.6667; 0.0000]).

Entering matrices

```
A = [1 2 3; 4 5 6; 7 8 9];
```

or

```
A = [
       1 2 3
       4 5 6
       7 8 9 ];
```

`rand(n)` will create an n x n matrix with randomly generated entries distributed uniformly between 0 and 1

`rand(m,n)` will create an m x n one.

`magic(n)` will create an integral n x n matrix which is a magic square (rows and columns have common sum)

`hilb(n)` will create the n x n Hilbert matrix (m and n denote, of course, positive integers).

Accessing individual entries: A(2,3) is element of A at row 2, column 3.
x(3) is the third element of vector x.

Loading Files into Matlab:

Files ending with a .m extension are matlab files containing commands.

(Load all .m files in the directory http://www.cs.purdue.edu/homes/ayg/CS590C/matlab_samples)

The simplest matlab commands:

```
m1 = 100;
m2 = 250;
k1= 500000;
k2 = 10000;
m = (m1 + m2)/2;
k = (k1 + k2)/2;
s = sqrt(m*m + k*k);
```

CS590C: Using Matlab.

Lecture 5: Using Matlab.

This series of lectures is intended to familiarize you with:

- ❏ Various features of Matlab.
- ❏ Graphics capabilities of Matlab.
- ❏ Coding some applications in Matlab.