MATLAB: A Powerful Tool for Computation and visualization

Shuxia Zhang and Ravi Chityala
Sumpercomputing Institute
University of Minnesota
e-mail: szhang@msi.umn.edu,chityala@msi.umn.edu
Tel:612-624-8858

© 2009 Regents of the University of Minnesota. All rights reserved.

Outline of part I

Introduction
Basic Math Operations
Input and Output
Solving computational problems
M-files
Submit Matlab jobs to the queues
2d Graphics/3d Visualization/Image data
Hands-on
Reference

© 2009 Regents of the University of Minnesota. All rights reserved.

Introduction

- MATLAB handles a wide range of computing tasks in engineering and science, from data acquisition and analysis to application development.
- Focus on high-level technical concepts and ignore programming detail, built-in functions.
- One can interactively run line by line command. One can also write a MATLAB code (referred as M-file) and run it in a batch mode.
- Interactive language and programming environment. M-files require no compiling or linking, so you can edit and debug an M-file and test the changes immediately.

© 2009 Regents of the University of Minnesota. All rights reserved.

How to Start Matlab

Type:

module load matlab

matlab

Search information, type the followings on MATLAB window

>> help known-name

>> lookfor string

Demos:

>> demo

“>>” marks the commands that one can type on MATLAB window.

© 2009 Regents of the University of Minnesota. All rights reserved.

MATLAB Toolboxes

Bioinformatics Toolbox
Communications Toolbox
Control System Toolbox
Curve Fitting Toolbox
Data Acquisition Toolbox
Database Toolbox
Datafeed Toolbox
Excel Link
Filter Design Toolbox
Financial Toolbox
Financial Derivatives Toolbox
Financial Time Series Toolbox
Fixed-Income Toolbox
Fuzzy Logic Toolbox
GARCH Toolbox
Genetic Algorithm Toolbox
Image Acquisition Toolbox
Image Processing Toolbox
Instrument Control Toolbox
LMI Control Toolbox
Mapping Toolbox
Model-Based Calibration Toolbox
Model Predictive Control Toolbox
Mu-Analysis and Synthesis Toolbox
Neural Network Toolbox
Optimization Toolbox
Partial Differential Equation (PDE) Toolbox
Robust Control Toolbox
Signal Processing Toolbox
Spline Toolbox
Statistics Toolbox
Symbolic Math Toolbox
System Identification Toolbox
Virtual Reality Toolbox
Wavelet Toolbox

© 2009 Regents of the University of Minnesota. All rights reserved.

Elementary Math and Matrix Manipulation Functions

Elementary math functions

Trigonometric.

sin - Sine.
cos - Cosine.tan
tan - Tangent.
sec - Secant.

Exponential.

exp - Exponential.
log - Natural logarithm.
log10 - Common (base 10) logarithm.
log2 - Base 2 logarithm
sqrt - Square root

Complex.

abs - Absolute value.
complex - Construct complex data
conj - Complex conjugate.
imag - Complex imaginary part.
real - Complex real part.

Matrix initialization.

zeros - Zeros array.
ones - Ones array.
eye - Identity matrix.
rand - Uniformly distributed random
randn - Normally distributed random numbers.
linspace - Linearly spaced vector.
logspace - Logarithmically spaced vector.

>> help logspace

Basic array information.

size - Size of matrix.
length - Length of vector.
ndims - Number of dimensions.
disp - Display matrix or text
spy - View the matrix structure

>> lookfor key_word

© 2009 Regents of the University of Minnesota. All rights reserved.

Operators and special characters

Arithmetic operators:

plus	- Plus	+
minus	- Minus	-
mtimes	- Matrix multiply	*
times	- Array multiply	.*
mpower	- Matrix power	^
power	- Array power	.^

Relational and logical operators:

eq	- Equal	==
ne	- Not equal	~=
lt	- Less than	<
gt	- Greater than	>
le	- Less than or equal	<=
ge	- Greater than or equal	>=
and	- Element-wise logical AND	&
or	- Element-wise logical OR	|

Special characters:

punct	- Semicolon	;	Continue	….
punct	- Comment	%		

© 2009 Regents of the University of Minnesota. All rights reserved.

Follow Algebra rules
Inner products

dot is the dot product weight function.

Inputs:

W – input (row) 1x n matrix

P - input (n columns) vectors.

dot(W,P) returns the dot product of W and P.

```
>> w=[1   3   5   7];
>> p=[2;   4;   6;   8]
>> dot(w,p)
   ans =100
>> dot(p,w)
   ans =
        2    6   10   14
        4   12   20   28
        6   18   30   42
        8   24   40   56
```

© 2009 Regents of the University of Minnesota. All rights reserved.

Function have multiple arguments

NORM: Matrix or vector norm

For matrices...

NORM(X) is the 2-norm of X.
NORM(X,2) is the same as NORM(X).
NORM(X,1) is the 1-norm of X.
NORM(X,inf) is the infinity norm of X.
NORM(X,'fro') is the Frobenius norm of X.
NORM(X,P) is available for matrix X only if P is 1, 2, inf or 'fro'.

For vectors...

NORM(V,P) = sum(abs(V).^P)^(1/P).
NORM(V) = norm(V,2).
NORM(V,inf) = max(abs(V)).
NORM(V,-inf) = min(abs(V)).

© 2009 Regents of the University of Minnesota. All rights reserved.

One function - more functionalities

SPARSE

SPARSE(X) converts a sparse matrix to sparse form by
squeezing out any zero elements. Save memory
But it may use more memory if the matrix is dense.

Example:

```
>> p=rand(10,20);
>> for i=1:10
   for j=1:20
   if (p(i,j) <0.5 ) T(i,j) = 0;
   else
   T(i,j) = p(i,j);
    end;end;end
>> sparse(T);
>> whos
Name      Size        Bytes  Class

T        10x20         1356  sparse array
i        1x1              8  double array
j        1x1              8  double array
p        10x20         1600  double array
```

© 2009 Regents of the University of Minnesota. All rights reserved.

One function - More functionalities

SPARSE

Create Sparse Matrix:

S = sparse(i,j,s,m,n,nzmax)
S: created m-by-n sparse matrix
nzmax: allocated zeros
i and j: integer index vectors
s: real or complex vector (non-zeros);
i,j, and s all have the same length
m=max(i)
n=max(j)

Example

```
>> s=rand(10,1);
>> i=[1,3,4,6,8,9,14,18,24,26];
>> j=[1,2,5,7,14,11,13,9,18,26];
>> m=max(i);
>> n=max(j);
>> S=sparse(i,j,s,m,n) %It will create a 26-by-26 sparse matrix.
>> spy(S)     % will show the structure of the sparse matrix
```

© 2009 Regents of the University of Minnesota. All rights reserved.

Basic operations

Create an array or an vector:

```
>> a = [1  2  3   4   5   6   7   8   9]

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

plus:

```
>> b = a+2

b = 3    4    5    6    7    8    9   10   11
```

Creating a matrix is as easy as making a vector

```
>> A = [1 2 0; 2 5 -1; 4  10  1]
   A =
         1    2    0
         2    5   -1
         4   10    1
```

© 2009 Regents of the University of Minnesota. All rights reserved.

Basic operations

```
>> a = [1  2  3   4   5   6   7   8   9]

            a = 1 2 3 4 5 6 7 8 9
>> a = [1  2  3   4   5   6   7   8   9]'   % transpose

a =

    1
    2
    3
    4
    5
    6
    7
    8
    9
```

Use of colon sign - a vector of sequential values.

```
x = [0:0.1:100]

x =  Columns 1 through 7

       0    0.1000    0.2000    0.3000    0.4000    0.5000    0.6000
  .......

 Columns 995 through 1001

  99.4000   99.5000   99.6000   99.7000   99.8000   99.9000  100.0000
```

© 2009 Regents of the University of Minnesota. All rights reserved.

Basic matrix operations

INV(X) is the inverse of the square matrx X

Given: A=[1 2 0; 2 5 -1; 4 10 1];

```
>>  X = inv(A)

X = 5.0000   -0.6667   -0.6667
-2.0000    0.3333    0.3333
         0   -0.6667    0.3333

>> I = inv(A)*A

I =
      1  0  0
      0  1  0
      0  0  1
```

© 2009 Regents of the University of Minnesota. All rights reserved.

Basic matrix operations

E= eig(X) -- a vector containing the eigenvalues of a square matrix X
[V,D] = eig(X) -- a diagonal matrix D of eigenvalues and a full matrix V, whose columns are the corresponding eigenvectors so that X*V = V*D.

Example:

Given: A=[1 2 0; 2 5 -1; 4 10 1];

```
>> E=eig(A)
E =
      0.1911
      3.4045 + 2.0270i
      3.4045 - 2.0270i
>> [V,D]=eig(A)
V =  -0.9258          -0.1555 + 0.0304i  -0.1555 - 0.0304i
      0.3745          -0.2178 - 0.1212i  -0.2178 + 0.1212i
     -0.0510          -0.9041 + 0.3088i  -0.9041 - 0.3088i
D = 0.1911                0                  0
      0          3.4045 + 2.0270i          0
      0                0           3.4045 - 2.0270i
```

© 2009 Regents of the University of Minnesota. All rights reserved.

Loop and If statements - Syntax

```
if expression
   statements
elseif expression
   statements
else
   statements
end
```

Example

```
for r = 1:nrows
   for c = 1:ncols
      if r == c
         myData(r,c) = 2;
      elseif abs(r - c) == 1
         myData(r,c) = -1;
      else
         myData(r,c) = 0;
      end
   end
end
```

© 2009 Regents of the University of Minnesota. All rights reserved.

Input and Output

Save: saves workspace or varaiables to disk:

```
>> save          % saves all variables to the defaulti binary file "matlab.mat"
>> save fname    % saves all variables to the file with the given name.
>> save fname X Y % save only variables X and Y.
>> save fname X Y -append % adds the variables to an existing mat-file.
>> save fname -ascii        % uses 8-digit ASCII form instead of binary.
>> save fname -ascii -double % uses 16-digit ASCII form.
```

To read in the mat-files, one need to use the load command, i.e.,

```
>> load fname
```

To delete a variable in the memory

```
>> clear X
```

© 2009 Regents of the University of Minnesota. All rights reserved.

Input and Output

I/O formats ---- compatible with other computer languages.

The commonly used ones include:

fopen - Open file.
fclose - Close file.

Input:

fread - Read binary data from file.
textread - Read formatted data from text file.
fscanf - Read formatted data from file.
load - Load workspace from MAT-file or ASCII file.

Output

save - Save workspace variables to disk
fwrite - Write binary data to file.
fprintf - Write formatted data to file.

© 2009 Regents of the University of Minnesota. All rights reserved.

I/O examples

fread: >> fid = fopen('input.dat','r');
>> [A, COUNT] = fread(fid,size,precision,skip)

size -- optional; if not specified, the entire file is read; else it can be:

N	read N elements into a column vector.
inf	read to the end of the file.
[M,N]	read elements to fill an M-by-N matrix, in column order. N can be inf, but M can't.

precision-- type of data that MATLAB supports, like schar, int8, single, double, etc,…

skip -- the number of bytes or bits to skip, dependent on the data type.

fprintf :

```
>> x = 0:.1:1; y = [x; exp(x)];
>> fid = fopen('exp.txt','w');
>> fprintf(fid,'%6.2f  %12.8f\n',y);
>> fclose(fid);
```

© 2009 Regents of the University of Minnesota. All rights reserved.

Many special built-in functions

Hilbert Matrtx

hilb(N) produces the N by N matrix with elements 1/(i+j-1).

```
>> n = 12;
>> A = hilb(n);
>> b = A * ones(n,1);
>> x = A \ b; err = ones(n,1)-x;
```

© 2009 Regents of the University of Minnesota. All rights reserved.

Function Definition -Syntax

```
function [output_list] = function_name(input_list)
input_list and output_list
       comma-separated lists of matlab variable.
```

```
function [r,theta] = cart2plr(x,y)
 %   [r,theta] = cart2lr(x,y)
 %   computes r and theta with
 %       r = sqrt(x^2 + y^2);
 %       theta = atan2(y,x);

 r = sqrt(x^2 + y^2);
 theta = atan2(y,x);
```

```
%Use of function
 a=rand(20);
 b=rand(20);
 [c the] = cart2plr(a,b)
 plot(c,the,'*r')
 Print -dpng test
 quit
```

© 2009 Regents of the University of Minnesota. All rights reserved.

Solving A X = B

```
>>A = [1.0000   0.5000   0.3333   0.2500   0.2000   0.1667   0.1429;
       0.5000   0.3333   0.2500   0.2000   0.1667   0.1429   0.1250;
       0.3333   0.2500   0.2000   0.1667   0.1429   0.1250   0.1111;
       0.2500   0.2000   0.1667   0.1429   0.1250   0.1111   0.1000;
       0.2000   0.1667   0.1429   0.1250   0.1111   0.1000   0.0909;
       0.1667   0.1429   0.1250   0.1111   0.1000   0.0909   0.0833;
       0.1429   0.1250   0.1111   0.1000   0.0909   0.0833   0.0769]
```

```
>> B= [2.5929 1.7179 1.3290 1.0956 0.9365 0.8199 0.7301]';
```

To get a solution for X

```
>> C = inv (A);
>> X = C*B
```

What is X ?

© 2009 Regents of the University of Minnesota. All rights reserved.

M-file

The M-file - capture your command-line explorations as permanent, reusable MATLAB functions.

Suppose you have a M-file, or you saved the command-line operations into a M-file, named as matlab_test.m, to run the code interactively, just type:

```
module load matlab  % on Linux machines
matlab < matlab_test.m > output
```

One can also submit the MATLAB jobs to the queue.

© 2009 Regents of the University of Minnesota. All rights reserved.

An example of PBS script file

```
#PBS -l ncpus=1,mem=1gb,walltime=1:30:00
#PBS –q lab
#PBS -m abe
cd /home/smpb/szhang/matlab_batch

module add matlab
matlab –nojvm –nodisplay < particles.m
```

© 2009 Regents of the University of Minnesota. All rights reserved.

2D Graphics

2D graphics consists of 2d plots and images

There are two basic ways to create graphs in MATLAB:

-- Use plotting tools to create graphs interactively.
-- Use the command interface to enter commands in the Command Window.

Tip: Combine both approaches eg. issue a plotting command to create a graph and then modify the graph using one of the interactive tools

© 2009 Regents of the University of Minnesota. All rights reserved.

2D graphics

Plot(X,Y,S) plots vector Y versus vector X. S is a character string and specifies the line types, plot symbols and/or colors, made from one element from any or all the following 3 columns:

Colors		**Symbols**		**Linestyles**	
y	yellow	.	point	-	solid
m	magenta	o	circle	:	dotted
c	cyan	x	x-mark	-.	dashdot
r	red	+	plus	--	dashed
g	green	*	star		
b	blue	s	square		
w	white	d	diamond		
k	black	v	triangle (down)		

© 2009 Regents of the University of Minnesota. All rights reserved.

2D graphics

bar(X,Y,WIDTH) draws the columns of the M-by-N matrix Y as M groups of N vertical bars with a value of WIDTH. default WIDTH is 0.8.

polar(THETA,RHO,S) makes a plot using polar coordinates of the angle THETA, in radians, versus the radius RHO with the linestyle specified in string S.

stairs(X,Y,S) draws a stairstep graph of the elements in vector Y at the locations specified in X with linestyle specified by the string S.

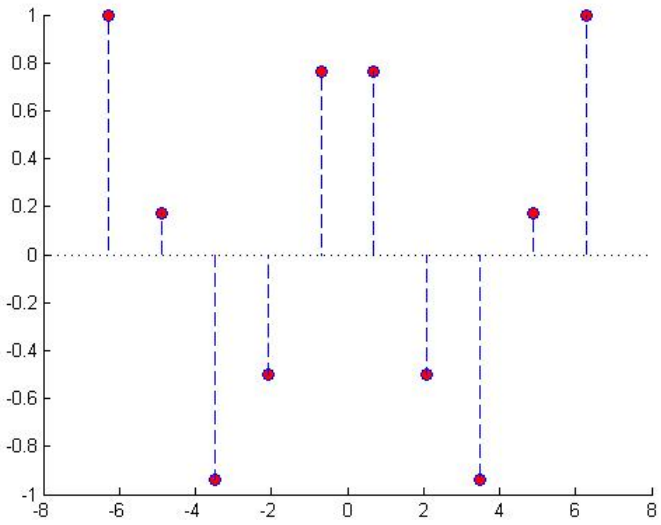
stem(X,Y,'filled',S) plots the data sequence Y as stems from the x axis terminated with the filled symbols. String S determines the linestyle of the stem.

© 2009 Regents of the University of Minnesota. All rights reserved.

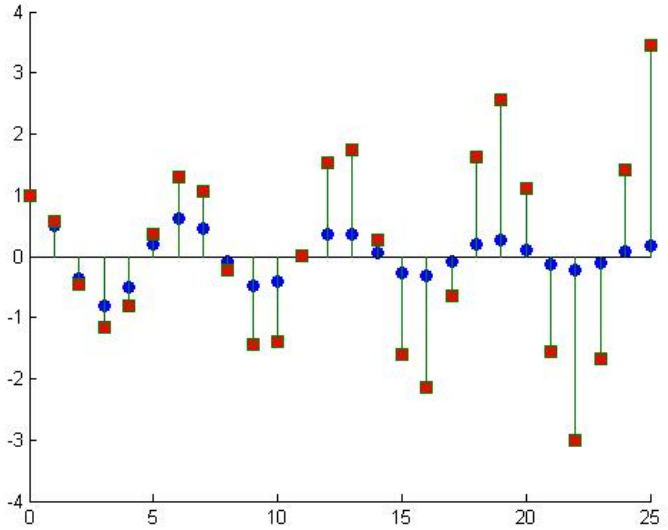
2D graphics

stem(X,Y,'filled',S) plots the data sequence Y as stems from the x axis terminated with the filled symbols. String S determines the linestyle of the stem.

```
>> t = linspace(-2*pi,2*pi,10);
h = stem(t,cos(t),'fill','--');
set(get(h,'BaseLine'),'LineStyle',':')
set(h,'MarkerFaceColor','red')
```




```
>> x = 0:25;
y = [exp(-.07*x).*cos(x);exp(.05*x).*cos(x)]';
h = stem(x,y);
set(h(1),'MarkerFaceColor','blue')
set(h(2),'MarkerFaceColor','red','Marker','square')
```



served.

semilogx(...) is the same as plot(...), except a logarithmic (base 10) scale is used for the X-axis.

```
>> x=[1.0,5.0,9.7,15.6,23.7,32.9];
>> y=[-6.2,-3.5, -1.0, 1.8,4.6, 8.9];
>> semilogx(x,y);
```

loglog(...) is the same as plot(...), except Logarithmic scales are used for both the X- and Y- axes.

```
>> x=[1.0,5.0,9.7,15.6,23.7,32.9];
>> y=x+5;
>> loglog(x,y);
```

Question: how to make a plot that a logarithmic (base 10) scale is used for the Y-axis?

© 2009 Regents of the University of Minnesota. All rights reserved.

Simple XY plots with solid, dashed lines

```
>> x=0:0.05:5;
>> y=sin(x.^2);
>> z=sin(x.^2-1.5);
>> plot(x,y, x,z,'r');
```

Bar graph

```
>> x = -2.9:0.2:2.9;
>> y = exp(-x.*x);
>> bar(x,y);
```

Polar graph

```
>> t=0:.01:2*pi;
>> y=abs(sin(2*t).*cos(2*t));
>> polar(t,y);
```

Stem graph

```
>> x = 0:0.1:4;
>> y = sin(x.^2).*exp(-x);
>> stem(x,y)
```

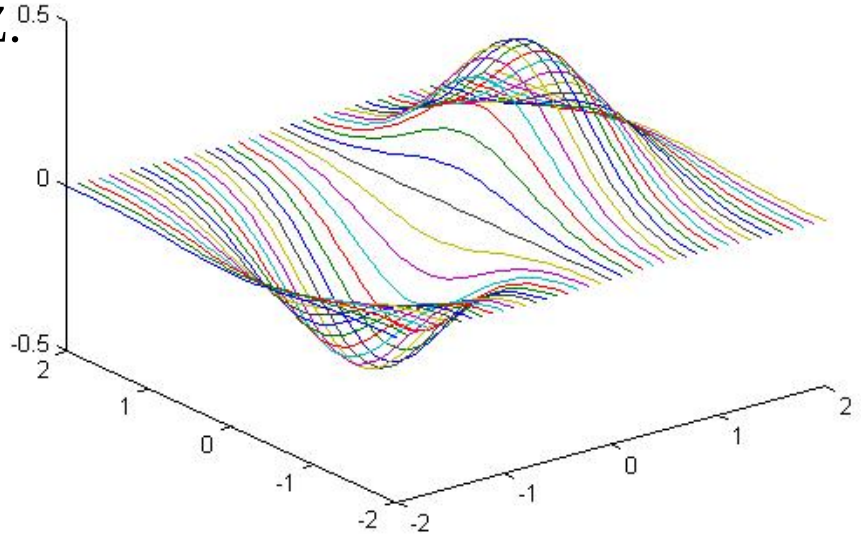
© 2009 Regents of the University of Minnesota. All rights reserved.

3D graphics

PLOT3(x,y,z,S), where x, y and z are three vectors, plots a line in 3-space the same length, through the points whose coordinate are the elements of x, y and z. S is a character string and specifies the plot style, curve color, style and/or symbols.

If the arguments to plot3 are matrices of the same size, MATLAB plots lines obtained from the columns of X, Y, and Z.

```
[X,Y] = meshgrid([-2:0.1:2]);
Z = X.*exp(-X.^2-Y.^2);
plot3(X,Y,Z)
grid on
```



3D Graphics

SURF(X,Y,Z,C) plots the colored parametric surface defined by four matrix arguments. The axis labels are determined by the range of X, Y and Z, or by the current setting of AXIS. The color scaling is determined by the range of C.

CONTOUR(X,Y,Z,N) a contour plot of matrix Z treating the values in Z as heights above a plane of X and Y coordinates. N can be a scalar or a vector. If N is a scalar, it specifies the number of contour lines. If N is a vector, it requires the contour lines to be drawn at the values specified in the vector.

© 2009 Regents of the University of Minnesota. All rights reserved.

3D Graphics

3-D colored surface

```
>> z=peaks(25); size(z);
>> surf(z);
>> colormap(jet);
```

Contour

```
>> z=peaks(25);
>> contour(z,16);
```

Vector Arrows

```
>> x = -2:.2:2; y = -1:.2:1;
>> [xx,yy] = meshgrid(x,y);
>> zz = xx.*exp(-xx.^2-yy.^2);
>> [px,py] = gradient(zz,.2,.2);
>> quiver(x,y,px,py,2);
```

3-D Stem Plots

```
>> th = (0:127)/128*2*pi;
>> x = cos(th); y = sin(th);
>> f = abs(fft(ones(10,1),128))';
>> stem3(x,y,f,'d','fill')
>> view([-65 30])
```

© 2009 Regents of the University of Minnesota. All rights reserved.

3-D Quiver Plots

```
>> vz = 10;          % Velocity
>> a = -32;          % Acceleration
>> t = 0:.1:1;
>> z = vz*t + 1/2*a*t.^2;
>> vx = 2;
>> x = vx*t;
>> vy = 3;
>> y = vy*t;
>> u = gradient(x);
>> v = gradient(y);
>> w = gradient(z);
>> scale = 0;
>> quiver3(x,y,z,u,v,w,scale)
>> axis square
```

© 2009 Regents of the University of Minnesota. All rights reserved.

Hands On

- Create a Hibert Matrix A for dimension i=j=12 , a vector
  B=[1, 0.5, 0.4, -2.0, -4.6, -1.2, 0.9, 1.0, 0.0, 0.0, 0.0, 1.0]'
  and solve the equation for unknown X
  AX =B
- Generate a m-file for above example and run the m-file
- Given t=0:.2:5, a=10, and u=2, write a program for calculating

  $$s = ut + \tfrac{1}{2}at^2$$

  and generate a plot of s as function of t

- Practice the examples presented in the lecture.

© 2009 Regents of the University of Minnesota. All rights reserved.

On-line help

www.mathworks.com/products/matlab
Www.msi.umn.edu/tutorial

Help at MSI-

help@msi.umn.edu
612-626-0802(help line)

© 2009 Regents of the University of Minnesota. All rights reserved.