Introduction To Matlab Features and Capabilities

MATLAB Main Components

Built-in Functions

- Computations
- Graphics
- External interface (dll and mex-files)
- Computer controlled hardware

External Functions or Tool Boxes

- Signal Processing
- Image Processing
- Simulink Control Design
- Statistics Toolbox
- Communications
- Image Acquisition
- Control System
- Image Processing
- Optimization
- Symbolic mathematics
- ….. and many more

MATLAB Main Features

- Variables are inherently arrays
- Detects variable type (real or complex) and chooses correct operator
- Variables are case-sensitive
- No dimension statements are required
- Very **efficient** in manipulating matrices
- Performs array and matrix operations in a single command
- Very **inefficient** in working with do **"loops"** and **"if"** statements
- Slower execution language, relative to Fortran and C
- Provides an extensive collection of **graphics** and **animation** functions
- M-files: ASCII text files
- Mat-files: MATLAB binary files
- Mex-files: MATLAB callable Fortran and C programs
- Platform independent (PC, Mac, Unix)

Simple Math Operations in Matlab

Operation	Symbol	Example
addition, a+b	+	3 + 4.2
subtraction, a-b	-	4-2.4
multiplication, a . b	*	3*5
division, a / b	/	56/8
exponential function, a power b	^	2^7

Expressions are always evaluated from left to right with the following order:
1- parenthesis
2- exponential function("^")
3- multiplication (*), division(/)
4- addition (+), subtraction (-)

Parenthesis are evaluated from innermost parenthesis to the outer most parenthesis.

Use parenthesis as much as possible for better readability and understandability.

Matlab Variables

- **Variable names case sensitive and may contain up to 31 characters.**
- **Variables must start with a letter followed by character(s), number(s) or under score.**
- **Punctuation marks are not allowed in the variable names.**
- **Matlab has some built in variables, some of them are**

Special variables	Description
pi	Ratio of circumference to diameter of a circle (22/7)
i (and) j	Square Root (-1)
nargin	Number of function input arguments used.
nargout	Number of function output arguments used.

Note that:
- **Values stored in a variable is erased when a new value is assigned to it.**
- **Special variables can be assigned any value but when Matlab is restarted or after execution of the clear command the original values are restored.**

Other Features of Matlab Programming

- **A comment can be written by using a "%" at the beginning of the comment.**
  Example: a = 4 % Radius of a circle.
- **Two or more Matlab statements can be placed on the same line if they are separated by a comma ",", or a semicolon";".**
- **A semicolon after a statement suppresses the value to be printed at the command line.**
  Example: a = 5; (value of a is not displayed in the command window)
  a = 5 (value of a is displayed in the command window)
- **To stop the Matlab processing, press "Ctrl + C".**

Matlab Workspace

- **Matlab remembers the commands and variables as they are typed in a workspace called "Matlab Workspace".**
- **The command "who" displays all the variables present in the workspace at any instant of time.**
- **The cursor arrows can be used at the command prompt to scroll through the commands typed in the particular session.**
- **The command "clear" deletes the variables present in the workspace.**

Example:

Type "a =3 , b=4" at the command prompt.

Type "who" at command prompt (that will displays the variables present in the session)

Scroll "UP" arrow key of the keyboard (the commands typed earlier are displayed at the command prompt)

Now type "a" and press enter (value of "a" is displayed)

Now type "clear" (clears the variables present in the memory)

For a check now type "who" (nothing is displayed).

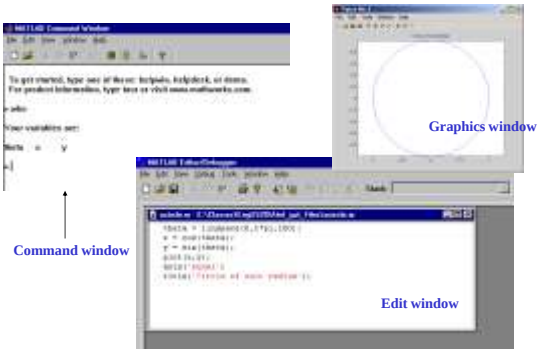
Getting Help For Matlab Programming

On-line Help

• help	lists topics on which help is available
• helpwin	opens the interactive help window
• helpdesk	opens the web browser based on help facility
• help *topic*	provides help on topic
• lookfor *string*	lists help topic containing string
• demo	Runs the demo program

Additional information can be found at **http://www.Mathworks.com**

Basics Windows of MATLAB



General Commands

On-line Help:

help	list topics on which help is available
helpwin	opens the interactive help window
helpdesk	opens the web browser based help facility
help *topic*	provides help on *topic*
lookfor *string*	lists help topics containing *string*
demo	runs the demo program

Workspace Information:

who	lists variables currently in the workspace
whos	lists variables currently in the workspace and their size
what	listsm-, mat-, and mex-files on the disk
clear	clears the workspace, all variables are removed
clear x y z	clears only the variables *x, y* and *z*
clear all	clears all variables and functions from workspace
mlock *fun*	locks function *fun* so that **clear** cannot remove it
munlock *fun*	unlocks function *fun* so that **clear** can remove it
clc	clears the command window, command history is lost
home	same as **clc**
clf	clears figure window

More General Commands

Directory Information:

pwd	shows the current working directory
cd	changes the current working directory
dir	lists the contents of the current directory
ls	lists the contents of the current directory, same as **dir**
path	gets or sets MATLAB search path
editpath	modifies the MATLAB search path
copyfile	copies a file
mkdir	creates a new directory

General Information:

computer	tells you the computer type you are using
clock	gives you the wall clock time and date as a vector
date	tells you the date as a string
more	controls the paged output according to the screen size
flops	shows the number of floating point operations used so far
ver	gives the license and version information about MATLAB

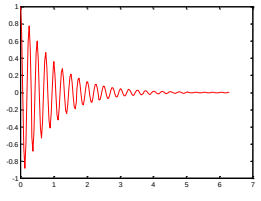
Creating 2D Plots in Matlab

```
% Generating 2D Plot

t1=linspace(0,2*pi,200); % Generate time array
ft = cos(10e5.*t1);      % Generate a function of time
ft = ft.*exp(-t1);
plot(t1(:),ft(:),'r-');  % Plots function vs. time
```

Syntax: plot(t1(:), ft(:), 'r--')

X Data, Y Data, Line Style (optiona



Creating Polar Plots in Matlab

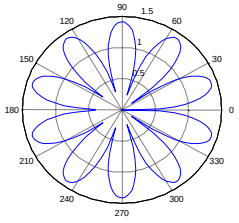
```
% Polar Plot of Function

t=linspace(0,2*pi,200); % Generate Angle Array
r=sqrt(abs(2*sin(5.*t))); % Radius (function)
polar(t,r);  % Plot of radius vs. angle
```

Syntax: polar(t,r);

Angle Array

Radius Array



Adding Legends, Labels, and Titles in Matlab

Main Title: plot(x,zampx); Title ('Pattern Parallel to X axis');

Plot Command

Adding a title to the current Plot

Labels: plot(x,zampx); Title ('Pattern Parallel to X axis');
xlabel('X values'); ylabel('Amplitude');

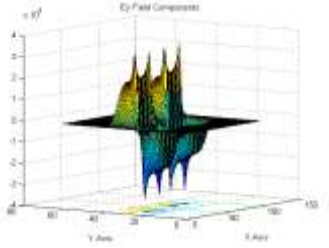
Title example with X and Y axis labels

Legends: plot(xlo,E15(501:1001),'r-.');
plot(xlo,E3(501:1001),'b-');
plot(xlo,E45(501:1001),'k--');
legend('t=1.5ms','t=3ms','t=4.5ms');

Generate 3 Plots

Add Legend

Creating 3D Plots in Matlab

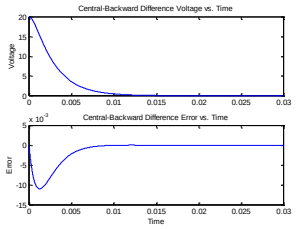


Syntax: figure; surfc(Ey);

Surface Plot with a 2D Array

title('Ey Field Components');
xlabel('X Axis'); ylabel('Y Axis');

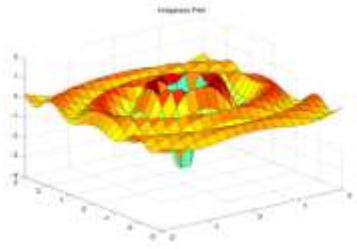
Creating Subplots in Matlab



Synxax: subplot(2,1,1), plot(tarray(:),cbd(:)) ← First Plot
subplot(2,1,2), plot(tarray(:),error(:)) ← Second Plot

Number of Rows, Number of Columns, Plot Number

Creating JPEG Output for Figures in Matlab



Syntax: g4=figure; surf(x,y,zimag); Title ('Imaginary Part');
print -djpeg90 d:\TempDir\fig4.jpg;

Generating the figure

Printing the figure in JPEG format

Creation of Publication Quality Figures in Matlab

- To create figures in Matlab, that are suitable for reports and general publications, do the following:
- create a file called Setfigures.m with the following code:

  set(0,'DefaultAxesFontSize',18)

  set(0,'DefaultLineLineWidth',2)

  set(0,'DefaultTextFontSize',18)
- run that file before plotting a figure by typing: Setfigures at the Matlab prompt (make sure you're in the right directory) or include it at the beginning of your m-file by typing: Setfigures at the beginning of your file, before any figures are plotted
- After your figures are plotted on the screen, save them to eps format by selecting the figure and typing the following at the Matlab prompt:

  save -deps filename.eps % for black and white

  save -depsc filename.eps % for color
- or save them to tiff format by selecting the following figure menu options: file->export->Save as type: TIF image->type in filename->Save
- or if you need smaller graphics (half the size), save as a jpg: file->export->Save as type: JPEG image->type in filename->Save
- When plotting multiple data sets on the same figure, use different line shapes and colors to distinguish the lines, since almost all publications are in black and white.

Reading Data from a File

Format of file to be read into Matlab:

```
 0.0000000E+00  5.6953170E+09
 9.9999994E-02 -5.6088530E+09
 0.2000000     5.3722378E+09
 0.3000000    -5.0399273E+09
 0.4000000     4.6702193E+09
 0.5000000    -4.3059026E+09
 0.6000000     3.9702057E+09
 0.7000000    -3.6714665E+09
 0.8000000     3.4094346E+09
 0.9000000    -3.1799450E+09
 1.000000      2.9776438E+09
 1.100000     -2.7972936E+09
 1.200000      2.6343206E+09
 1.300000     -2.4849421E+09
 1.400000      2.3461217E+09
 1.500000     -2.2154775E+09
 1.600000      2.0911688E+09
 1.700000     -1.9717802E+09
  . . .
```

Matlab routine to read and plot data:

```
% Program to read a file and graph its contents

load C:\somedirectory\datafile.dat

% File data is converted into a array
% named datafile.  In this case, the
% array has N rows and two columns

% Plot the array

plot(datafile(:,1), datafile(:,2));

% This plots the first column as the X
% axis and the second column as the Y
% axis
```

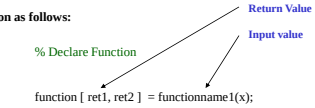

Matlab M File Functions

Matlab has the ability to be programmed by creating **scripts** or a text file of commands that Matlab can run when called. Usually each script represents a function. An example is shown below:

```
function [khi,gamma]=polarization(ax, ay, delta)
% returns the rotation angle, gamma, and the eplipticity angle, khi in degrees
% ax and ay are scalars while delta is the phase angle difference in degrees
deltar=delta*pi/180;
psir=atan(ay/ax);
gammar=1/2*(atan((tan(2*psir)*cos(deltar))));
khir=1/2*asin(sin(2*psir)*sin(deltar));
khi=180*khir/pi;
%convert from radians to degrees
gamma=180*gammar/pi;
if cos(deltar)>0 & gamma<0
  gamma=gamma+90;
elseif cos(deltar)<0 & gamma>0
  gamma=gamma-90;
else
  gamma=gamma;
end
```


Writing Functions in Matlab

1. **Create a new m-file named** *functionname.m*
2. **Write function as follows:**




```
% Declare Function

function [ ret1, ret2 ]  = functionname1(x);
```

or

```
function [ ret]  = functionname2(x);

% Write the mathmatical operations here

ret = sin(x)/x;
% That is it!
```

3. **Make sure** *functionname.m* **is in MATLAB's search Path**
4. **Here is how to use this function in your Matlab program**

```
[a,b]= functionname1(m);        a >>> ret1 , b >>> ret 2
xnew = functionname2(m);     xnew >>> ret1
```

The Commands "path" and "addpath"

- **Path command allows for access to Matlab search path**
- **To add a path to the search paths use the following command:**
  path ('*newpath*',path) which adds a "*newpath*" directory to the current Matlab search path
- **This will ensure that Matlab looks at your directory, "*newpath*" before using the other default directories**
- **The faster and safer way is to use the addpath command. To add your path "*newpath*" use addpath like this:**
  addpath ('*newpath*') which adds this specified directory to MATLAB's current search path
  addpath ('*dir1*','*dir2*','*dir3*',...) adds all the specified directories to the path.

Example # 1

Consider an isotropic radiator located at the origin (0,0,0) of a three dimensional space. Calculate the radiation over the area -2 <= x <= 2, -3 <= y <= 3, where the grid points are located at
-2, -2+Δx, …, - Δx/2, Δx/2, …, 2-Δx, 2 for x and
-3, -3+Δy, …, - Δy/2, Δy/2, …, 3-Δy, 3 for y.
Choose Δx, and Δy to give you 20 and 30 points along the x and y axes, respectively. (hint: investigate the *linspace* command).

Create **3-D charts for the radiation**
a) magnitude b) phase
c) real part d) imaginary part
and 2-D charts for the magnitude of the radiation
e) parallel to the x-axis at y= 5.5Δy
f) parallel to the y-axis at x= Δx/2

The radiation pattern is given by $\frac{e^{-jkR}}{R}$
where $k = 2\pi$, and R is the distance from the radiator.

Functions you May Use For Example # 1

1. Set a range *var = begin: delta: end*
2. Create a 2-D mesh *[X,Y]=meshgrid(x,y)*
3. Convert from Cartesian to polar *[Th,R]=cart2pol(X,Y)*
4. Element-by-element matrix multiply and divide
   *C=A.*B D=A./B*
5. Plot
   mesh(abs(z)) , *plot3 (x,y,z)* , *plot (x,y)*
6. Use *addpath* to set path to your m-files directory

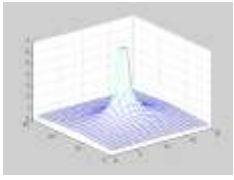
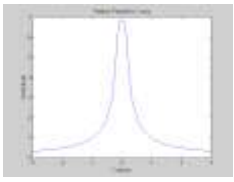
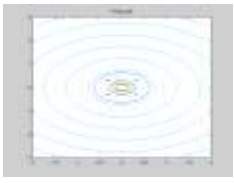
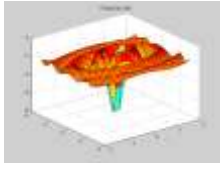
Solution to Example # 1

```
clear all; close all; clc;
x=linspace(-2,2,20);
y=linspace(-3,3,30);
[X,Y]=meshgrid(x,y);
[Th,R]=cart2pol(X,Y);
z=exp(-j*2*pi*R)./R;
zamp=abs(z);
zphase=angle(z)*180/pi;
zreal=real(z); zimag=imag(z);

% 3D figures
h1=figure; mesh(zamp)      % without x or y axes
h2=figure; mesh(x,y,zamp)  % with x and y axes
h3=figure; mesh(x,y,zphase)
h4=figure; contour(x,y,zreal); Title ('Real part')
h5=figure; surf(x,y,zimag);
Title (' Imaginary part')

% 2D figures
zampx = zamp(21,:); ampx = zampx';
zampy = zamp(:,11); ampy = zampy';
h6=figure; plot (x,ampx);
h7=figure; plot (y,ampy);
Title('Pattern Parallel to Y axis');
xlabel('Y values');ylabel('Amplitude');
% cd c:\classes;
print -djpeg90 fig2.jpg
```


Samples of Example # 1 Results

Homework # 1

1-a) For the following two equations, 4x – 6y = 4, and 10x -8y = 24, use Matlab matrix inversion function to find the values of x and y. Verify Matlab results with your analytical solution.

1-b) Create a surface plot and a contour plot of the function

$$z = 8xe^{-[(2x-4y^2)^3+6x^2]} \text{ for } -1 \le x \le 1 \text{ and } -1 \le y \le 1$$

1-c) Use Matlab to sketch the function

$$v(t) = 9e^{-t/2.5}\sin(3\pi t) \text{ for } -4 \le t \le 6$$

End of Lecture # 1

Solution to Homework # 1-a

1-a) For the following two equations, 3x – 4y = 5, and 6x -10y = 2, use Matlab matrix inversion function to find the values of x and y. Verify Matlab results with your analytical solution.

Program listing

```
% AHW_1a
% This program uses the matrix inverse function
% to find the values for x and y.
% By Dr. Atef Elsherbeni
% Last update August 25, 2004

clear all;
A=[3,-4 ; 6,-10]
AA=inv(A)
B=[5 ; 2]
XandY=AA*B
```

Program output

```
A =
   3   -4
   6   -10

AA =
   1.6667   -0.6667
   1.0000   -0.5000

XandY =
   7.0000
   4.0000
```


Solution to Homework # 1-b

1-b) Create a surface plot and a contour plot of the function

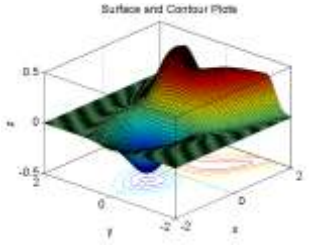
$$z = xe^{-[(x-y^2)^2+y^2]} \text{ for } -2 \le x \le 2 \text{ and } -2 \le y \le 2$$

```
% AHW_1b
% Surface and contour plot of a function z.
% By Dr. Atef Elsherbeni
% Last update August 25, 2004

clear all;
setfigures    % default parameters for figures

x=linspace(-2,2,100);
y=linspace(-2,2,100);
[X,Y]=meshgrid(x,y);
z=X.*exp(-[(X-Y.^2).^2+Y.^2]);

figure(1);
surfc(X,Y,z); box on
Title('Surface and Contour Plots');
xlabel('x'); ylabel('y'); zlabel('z')
```



Solution to Homework # 1-c

1-c) Use Matlab to sketch the function

$$v(t) = 10e^{-t/0.3}\sin(\pi t) \text{ for } 0 \le t \le 2$$

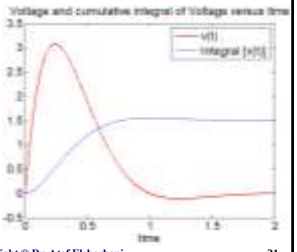
Then use Matlab to find the bounded by the function *v(t)* and the *t* axis.

```
% AHW_1c
% Integral of a function v(t)
% By Dr. Atef Elsherbeni
% Last update August 25, 2004

clear all;
setfigures      % defines default parameters for figures

t = linspace(0,2,200);
v = 10*exp(-t./3).*sin(pi*t);
f = inline('10*exp(-t./3).*sin(pi*t)')
format long
Area_using_quad = quad(f,0,2)
Area_using_trapz =trapz(t,v)
At = cumtrapz(t,v);

figure(1)
plot(t,v, 'r-',t,At,'b:')
title('Voltage and cumulative integral of Voltage versus time');
xlabel('time'); legend('v(t)', 'Integral |v(t)|')
```



Old Homework # 1

Consider a radiator located at the origin (0,0,0) of a three dimensional space. Using the spherical coordinates system, calculate the radiation in the x-y plane over the area -3 <= x <= 3, -2 <= y <= 2, where the grid points are located at -3, -3+Δx, …, - Δx/2, Δx/2, …, 3-Δx, 3 for x and -2, -2+Δy, …, - Δy/2, Δy/2, …, 2-Δy, 2 for y.
Choose Δx, and Δy to give you grid points in the order of 20 and 30 along the x and y axes, respectively.

Create the following 3-D charts for the radiation pattern
a) magnitude b) phase
c) real part d) imaginary part
and the 2-D charts for the magnitude of the radiation pattern at the plane
e) parallel to the x-axis at y= 1.5Δy
f) parallel to the y-axis at x= -1.5Δx.

The **radiation pattern** is given by ⟹ $\frac{e^{-jkR}}{2R}\left[1-\frac{2}{jkR}+\frac{j}{kR^2}\right](2-\cos^2\phi)$
where $k = 2\pi$.

Trapezoidal Integration Function

```
function [area,error]=trapint(polynomial,upper_limit,lower_limit,resolution)
% Trapezoidal Numerical Integration Method
% Usage:
% polynomial -- array of coefficients of polynomial to be integrated
% upper_limit -- upper limit of definite integral
% lower_limit -- lower limit of definite integral
% resolution -- number of points in integral
%
% Returns:
% area -- area under the curve
% error -- error term
% BEGIN
%
% Variable Declaration
% A -- Area under curve; initially set to 0
% f0 -- first point
% f1 -- second point
% numpoints -- number of points between a and b with given resolution
% funct -- array of points at vertices of trapezoids
%       when evaluated by function
% increment -- increment step;  = (b-a)/numpoints
%
% deriv -- array of derivatives corresponding to points
% Initialization
area=0;
f0=lower_limit;
numpoints = 10;
% Calculate points  array
increment=(upper_limit-lower_limit)/numpoints;
funct=zeros(numpoints,1);
for i=1:numpoints+1
  funct(i)=polyval(polynomial,f0);
  f0=f0+increment;
end
% Calculate Area
for i=1:numpoints
  f0=funct(i);
  f1=funct(i+1);
  area=area+(f0+f1)/2*increment;
end
% Calculate the Error
```


Numerical Methods in Electromagnetics
ENGR 626 or ENGR 597 - Fall 2004

Part I: by Dr. Atef Elsherbeni, Finite Difference Techniques (50%)

Part II: by Dr. Allen Glisson , Method of Moments (50%)

References:

- ***Mastering Matlab 6*, by Duane Hanselman and Bruce Littlefield, Prentice Hall, 2001.**
- **Field Computation by Moment Methods, by Roger F. Harrington, IEEE Press, 1993.**
- **Computational Methods for Electromagnetics, by Andrew F. Peterson, Scott L. Ray, and Raj Mittra, IEEE Press, 1998.**
- **Computational Magnetics, Edited by Jan K. Sykulski, Chapman and Hall, 1995.**
- **Numerical Techniques in Electromagnetics, by Matthew N. O. Sadiku, CRC Press, 2001.**

Numerical Methods in Electromagnetics
ENGR 626 - Fall 2004
Part II- Method of Moments

Topics to be covered:

1. The equivalence principle and formulation of integral equations.
2. Numerical solution of integral equations using the MoM.
3. Numerical solution of differential equations using the MoM.
4. Numerical evaluation of integrals.
5. The method of moments applied to general surfaces.

50% of Course Final Grade:

Homework	35%
Project Written Report	15% **Due December 5**

Matlab M File Functions

- This is a simple function that we can use to learn the rules of programming for Matlab **functions.**
- File Name -- The file name of any function is the name of the file. So in our exampe function *polarization* the file name is *polarization.m.*
- Comment lines (denoted by the %) up to the first noncomment line are displayed when you type *help function.* This will be very helpful in remembering the calling parameters and what each function does. For example:

  help polarization

  returns the rotation angle, gamma, and the eplipticity angle, khi in degrees
  ax and ay are scalars while delta is the phase angle difference in degrees

- The function will terminate when it either reaches the end of the file or it encounter the command *return.*
- Matlab is capable of doing **recursion** with functions.
- Functions can share **global variables** with the Matlab Command Window, other functions and recursive calls to itself with the variable(s) declared as **global**.
- Functions can be nested. That is a function may contain a call to another function whether it be different or itself.