

Digital Signal Processing

Prof. Nizamettin AYDIN

naydin@yildiz.edu.tr

naydin@ieee.org

<http://www.yildiz.edu.tr/~naydin>

1

Course Details

- Course Code : 0113620
- Course Name: Digital Signal Processing
(Sayısal İşaret İşleme)
- Instructor : Nizamettin AYDIN

2

MATLAB Tutorial

3

Content...

- What is Matlab?
- MATLAB Parts
- MATLAB Desktop
- Matrices
 - Numerical Arrays
 - String Arrays
- Elementary Math
 - Logical Operators
 - Math Functions
 - Polynomials and Interpolation
- Importing and Exporting Data

4

...Content...

- Graphics Fundamentals
 - 2D plotting
 - Subplots
 - 3D plotting
 - Specialized Plotting
- Editing and Debugging M-files
- Script and Function Files
- Basic Parts of an M-file
- Flow Control Statements
- M-file Programming

5

...Content

- Data types
 - Multidimensional Arrays
 - Structures
 - Cell Arrays
- Nonlinear Numerical Functions
- Ordinary Differential Equations (ODE)
- Handle Graphics
- Graphic Objects
- Graphical User Interface (GUI)

6

MATLAB

- high-performance software
 - *Computation*
 - *Visualization*
 - *Easy-to-use environment.*
- high-level language
 - *Data types*
 - *Functions*
 - *Control flow statements*
 - *Input/output*
 - *Graphics*
 - *Object-oriented programming capabilities*

7

MATLAB Parts

- Developed Environment
- Programming Language
- Graphics
- Toolboxes
- Application Program Interface

8

Toolboxes

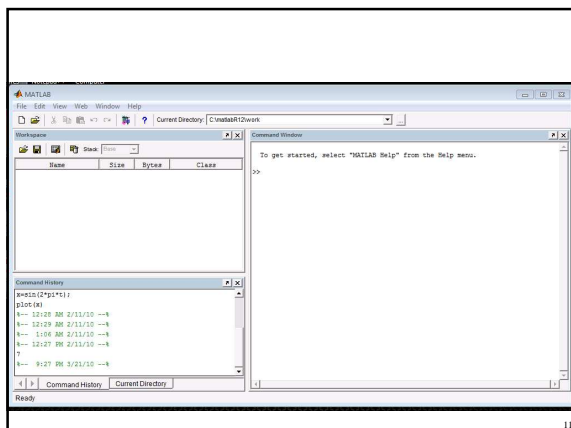
- Collections of functions to solve problems of several applications.
 - DSP Toolbox
 - Image Toolbox
 - Wavelet Toolbox
 - Neural Network Toolbox
 - Fuzzy Logic Toolbox
 - Control Toolbox
 - Communication Toolbox
 -

9

MATLAB Desktop Tools

- Command Window
- Command History
- Help Browser
- Workspace Browser
- Editor/Debugger
- Launch Pad

10



11

Calculations at the Command Line

MATLAB as a calculator

```

» -5/(4.8+5.32)^2
ans =
-0.0488
» (3+4i)*(3-4i)
ans =
32
» cos(pi/2)
ans =
6.1230e-017
» exp(acos(0.3))
ans =
3.5470
    
```

Assigning Variables

```

» a = 2;
» b = 5;
» a*b
ans =
32
» x = 5/2*pi;
» y = sin(x)
y =
1
» z = asin(y)
z =
1.5708
    
```

Semicolon suppresses screen output

Results assigned to "ans" if name not specified

() parentheses for function inputs

Numbers stored in double-precision floating point format

12

General Functions

- **whos**: List current variables
- **clear** : Clear variables and functions from memory
- **close**: Closes last figures
- **cd** : Change current working directory
- **dir** : List files in directory
- **echo** : Echo commands in M-files
- **format**: Set output format

13

Getting help

- **help** command (>>help)
- **lookfor** command (>>lookfor)
- Help Browser (>>doc)
- **helpwin** command (>>helpwin)
- Search Engine
- Printable Documents
 - "Matlabroot\help\pdf_doc"
- Link to The MathWorks

14

Matrices

- Entering and Generating Matrices
- Subscripts
- Scalar Expansion
- Concatenation
- Deleting Rows and Columns
- Array Extraction
- Matrix and Array Multiplication

15

Entering Numeric Arrays

```

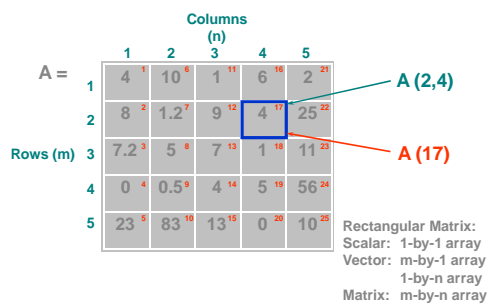
> a=[1 2;3 4]
a =
     1     2
     3     4
> b=[-2.8, sqrt(-7), (3+5+6)*3/4]
b =
-2.8000    0 + 2.6458i   10.5000
> b(2,5) = 23
b =
-2.8000    0 + 2.6458i   10.5000    0    0
         0             0         0    0 23.0000
    
```

Use square brackets []

- Any MATLAB expression can be entered as a matrix element
- Matrices must be rectangular. (Set undefined elements to zero)

16

The Matrix in MATLAB



17

Entering Numeric Arrays

```

Scalar expansion: > w=[1 2;3 4] + 5
w =
     6     7
     8     9

Creating sequences:
- colon operator (:): > x = 1:5
x =
     1     2     3     4     5
> y = 2:-0.5:0
y =
2.0000   1.5000   1.0000   0.5000   0
> z = rand(2,4)
z =
0.9501   0.6068   0.8913   0.4565
0.2311   0.4860   0.7621   0.0185
    
```

18

Numerical Array Concatenation

Use `[]` to combine existing arrays as matrix "elements"

Row separator:
- semicolon (`;`)

Column separator:
- space / comma (`,`)

```
> a=[1 2;3 4]
a =
     1     2
     3     4
> cat_a=[a, 2*a; 3*a, 4*a; 5*a, 6*a]
cat_a =
     1     2     2     4
     3     4     6     8
     5    10    12    16
     3     6     6     8
     9    12    12    16
     5    10     6    12
    15    20    18    24
```

Use square brackets `[]`

$4 \times a$

- The resulting matrix must be rectangular

19

Deleting Rows and Columns

```
> A=[1 5 9;4 3 2.5; 0.1 10 3i+1]
A =
    1.0000    5.0000    9.0000
    4.0000    3.0000    2.5000
    0.1000   10.0000   1.0000+3.0000i
> A(:,2)=[]
A =
    1.0000    9.0000
    4.0000    2.5000
    0.1000    1.0000 + 3.0000i
> A(2,2)=[]
??? Indexed empty matrix assignment is not allowed.
```

20

Array Subscripting / Indexing

		1	2	3	4	5
1	4	10	6	1	6	2
2	8	1.2	9	4	17	25
3	7.2	5	7	1	18	11
4	0	0.5	4	5	19	56
5	23	83	13	0	20	10

$A(1:5,5)$ $A(1:end,end)$
 $A(:,5)$ $A(:,end)$
 $A(21:25)$ $A(21:end)$

$A(3,1)$
 $A(3)$

$A(4:5,2:3)$
 $A(19\ 14;10\ 15)$

21

Matrix/Array Multiplication

Matrix Multiplication

```
> a = [1 2 3 4; 5 6 7 8];
> b = ones(4,3);
> c = a*b
c =
    10    10    10
    26    26    26
```

$[2 \times 4]$
 $[4 \times 3]$
 $[2 \times 4] \times [4 \times 3] \rightarrow [2 \times 3]$

$a(2nd\ row), b(3rd\ column)$

Array Multiplication

```
> a = [1 2 3 4; 5 6 7 8];
> b = [1:4; 1:4];
> c = a.*b
c =
     1     4     9    16
     5    12    21    32
```

$c(2,4) = a(2,4) \cdot b(2,4)$

22

Matrix Manipulation Functions...

- `zeros` : Create an array of all zeros
- `ones` : Create an array of all ones
- `eye` : Identity Matrix
- `rand` : Uniformly distributed random numbers
- `diag` : Diagonal matrices and diagonal of a matrix
- `size` : Return array dimensions
- `fliplr` : Flip matrices left-right
- `flipud` : Flip matrices up and down
- `repmat` : Replicate and tile a matrix

23

...Matrix Manipulation Functions

- `transpose (')` : Transpose matrix
- `rot90` : rotate matrix 90
- `tril` : Lower triangular part of a matrix
- `triu` : Upper triangular part of a matrix
- `cross` : Vector cross product
- `dot` : Vector dot product
- `det` : Matrix determinant
- `inv` : Matrix inverse
- `eig` : Evaluate eigenvalues and eigenvectors
- `rank` : Rank of matrix

24

Character Arrays (Strings)

- Created using single quote delimiter (')

```
> str = 'Hi there,'
str =
Hi there,
> str2 = 'Isn't MATLAB great?'
str2 =
Isn't MATLAB great?
```

- Each character is a separate matrix element
(16 bits of memory per character)

```
str = [ H i t h e r e , ] ← 1x9 vector
```

- Indexing same as for numeric arrays

25

String Array Concatenation

- Using [] operator:
 - Each row must be same length

```
> str = 'Hi there, '
> str1 = 'Everyone!'
> new_str = [str, ' ', str1]
new_str =
Hi there, Everyone!
> str2 = 'Isn't MATLAB great?'
> new_str2 = [new_str; str2]
new_str2 =
Hi there, Everyone!
Isn't MATLAB great?
```

- Row separator:
 - semicolon (;)

- Column separator:
 - space / comma (,)

- For strings of different length:
 - STRVCAT
 - char

```
> new_str3 = strvcat(str, str2)
new_str3 =
Hi there,
Isn't MATLAB great?
```

26

Working with String Arrays

- String Comparisons
 - strcmp : compare whole strings
 - strncmp : compare first 'N' characters
 - findstr : finds substring within a larger string
- Converting between numeric & string arrays:
 - num2str : convert from numeric to string array
 - str2num : convert from string to numeric array

27

Elementary Math

- Logical Operators
- Math Functions
- Polynomial and Interpolation

28

Logical Operations

==	equal to
>	greater than
<	less than
>=	Greater or equal
<=	less or equal
~	not
&	and
	or
isfinite(), etc. ...	
all(), any()	
find	

```
> Mass = [-2 10 NaN 30 -11 Inf 31];
> each_pos = Mass>=0
each_pos =
     0     1     0     1     0     1     1
> all_pos = all(Mass>=0)
all_pos =
     0
> all_pos = any(Mass>=0)
all_pos =
     1
> pos_fin = (Mass>=0)&(isfinite(Mass))
pos_fin =
     0     1     0     1     0     0     1
```

Note: 1 = TRUE 0 = FALSE

29

Elementary Math Function...

- abs : Absolute value
- sign : Signum Function
- sin, cos : Triangular functions (sinus, cosinus)
- asin, acos: Triangular functions (arcsinus,...)
- exp : Exponential
- log : Natural logarithm
- log10 : Common (base 10) logarithm
- ceil, floor: Round toward infinities
- fix : Round toward zero

30

...Elementary Math Function...

- `round` : Round to the nearest integer
- `gcd` : Greatest common divisor
- `lcm` : Least common multiple
- `sqrt` : Square root function
- `real, imag`: Real and Image part of complex
- `rem` : Remainder after division

31

...Elementary Math Function

- `max, min`: Maximum and Minimum of arrays
- `mean, median`: Average and Median of arrays
- `std, var`: Standard deviation and variance
- `sort`: Sort elements in ascending order
- `sum, prod`: Summation & Product of Elements
- `trapz`: Trapezoidal numerical integration
- `cumsum, cumprod`: Cumulative sum, product
- `diff, gradient`: Differences and Numerical Gradient

32

Polynomials and Interpolation

- Polynomials
 - Representing
 - Roots (`>> roots`)
 - Evaluation (`>> polyval`)
 - Derivatives (`>> polyder`)
 - Curve Fitting (`>> polyfit`)
 - Partial Fraction Expansion (`residue`)
- Interpolation
 - One-Dimensional (`interp1`)
 - Two-Dimensional (`interp2`)

33

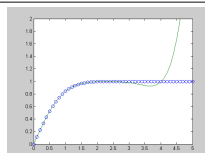
Example

```
polysam=[1 0 0 8];
roots(polysam)
ans =
    -2.0000
    1.0000 + 1.7321i
    1.0000 - 1.7321i
Polyval(polysam,[0 1 2.5 4 6.5])
ans =
    8.0000    9.0000   23.6250   72.0000  282.6250
polyder(polysam)
ans =
     3     0     0
[r p k]=residue(polysam,[1 2 1])
r = 3     7
p = -1    -1
k = 1    -2
```

34

Example

```
x = [0: 0.1: 2.5];
y = exp(x);
p = polyfit(x,y,6)
p =
    0.0084   -0.0983    0.4217   -0.7435    0.1471    1.1064    0.0004
```



```
interp1(x,y,[0.45 0.95 2.2 3.0])
ans =
    0.4744    0.8198    0.9981    NaN
```

35

Importing and Exporting Data

- Using the Import Wizard
- Using `Save` and `Load` command

```
save fname
save fname x y z
save fname -ascii
save fname -mat
```

```
load fname
load fname x y z
load fname -ascii
load fname -mat
```

36

Input/Output for Text File

- Read formatted data, reusing the format string N times.

```
> [A1...An]=textread(filename,format,N)
```

- Import and Exporting **Numeric** Data with General ASCII delimited files

```
> M = dlmread(filename,delimiter,range)
```

37

Input/Output for Binary File

- fopen** : Open a file for input/output
- fclose** : Close one or more open files
- fread** : Read binary data from file
- fwrite** : Write binary data to a file
- fseek** : Set file position indicator

```
> fid = fopen('mydata.bin', 'wb');
> fwrite(fid,eye(5), 'int32');
> fclose(fid);
> fid = fopen('mydata.bin', 'rb');
> M = fread(fid, [5 5], 'int32');
> fclose(fid);
```

38

Graphics

- Basic Plotting
plot, title, xlabel, grid, legend, hold, axis
- Editing Plots
Property Editor
- Mesh and Surface Plots
meshgrid, mesh, surf, colorbar, patch, hidden
- Handle Graphics

39

2-D Plotting

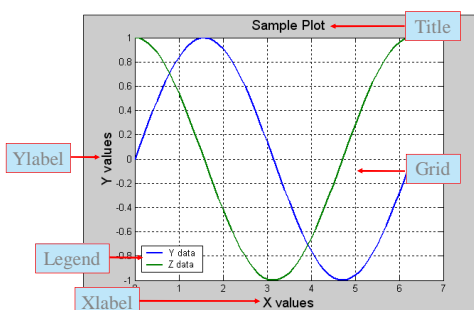
- Syntax:

```
plot(x1, y1, 'c1m1', x2, y2, 'c2m2', ...)
```

```
x=[0:0.1:2*pi];
y=sin(x);
z=cos(x);
plot(x,y,x,z,'linewidth',2);
title('Sample Plot','fontsize',14);
xlabel('X values','fontsize',14);
ylabel('Y values','fontsize',14);
legend('Y data','Z data');
grid on
```

40

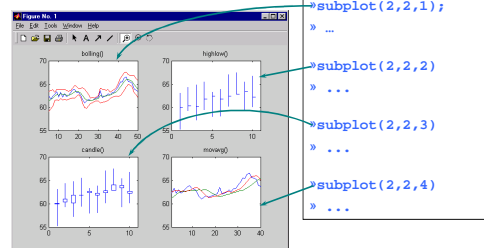
Sample Plot



41

Subplots

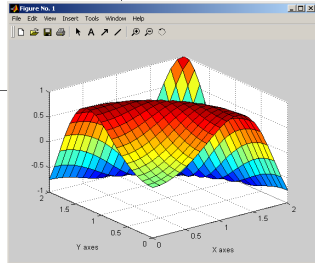
```
subplot(rows,cols,index)
```



42

Surface Plot Example

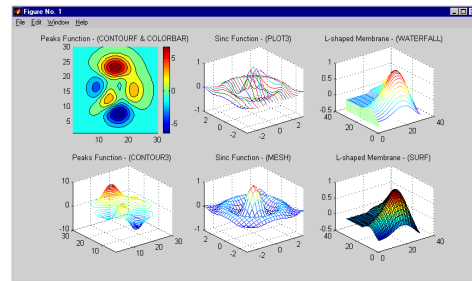
```
x = 0:0.1:2;
y = 0:0.1:2;
[xx, yy] = meshgrid(x,y);
zz=sin(xx.^2+yy.^2);
surf(xx,yy,zz);
xlabel('X axes');
ylabel('Y axes');
```



43

3-D Surface Plotting

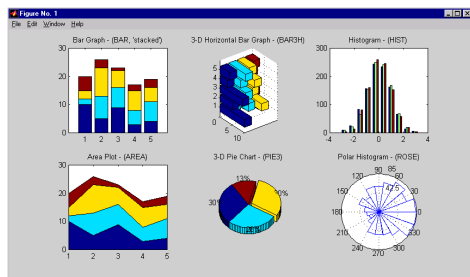
contourf-colorbar-plot3-waterfall-contour3-mesh-surf



44

Specialized Plotting Routines

bar-bar3h-hist-area-pie-3d-rose



45

Editing and Debugging M-Files

- What is an M-File?
- The Editor/Debugger
- Search Path
- Debugging M-Files
 - Types of Errors (*Syntax Error and Runtime Error*)
 - Using *keyboard* and ";" statement
 - Setting Breakpoints
 - Stepping Through
 - Continue, Go Until Cursor, Step, Step In, Step Out
 - Examining Values
 - Selecting the Workspace
 - Viewing *Datatypes* in the Editor/Debugger
 - Evaluating a Selection

46

Debugging

```
8: sweepLE=
9: xpos=xpos
10: ypos=ypos
11: vert=1;
12: for j=1:num
13:     newrib=[coord(j)*coord(:,1:j)+xpos(j) ypos(j)*ones(size(coo
14:     vert=[vert; coord =
15:     newri
16: end
17: index=size(coord,1);
18: for j=1:num-1, % this loop calling all patterns
19:     for i=1:(index-1),
20:         fac(2*i+2*(j-1)*index, :)=(j-1)*index+[i i+1 index+i
21:         fac(2*i+2*(j-1)*index, :)=(j-1)*index+[i index+i index
22:     end
23:     fac(2*i+index-1+2*(j-1)*index, :)=(j-1)*index+j*index 1 in
24:     fac(2*i+index+2*(j-1)*index, :)=(j-1)*index+j*index 2*inde
25: end
26: hpatch('faces',fac,'vertices',vert);
27: x]
```

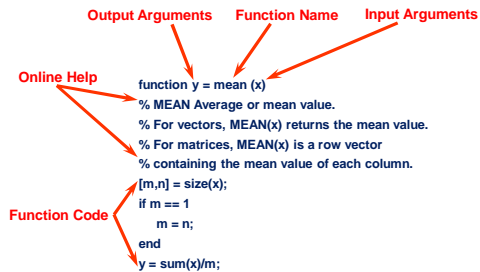
47

Script and Function Files

- Script Files
 - Work as though you typed commands into MATLAB prompt
 - Variable are stored in MATLAB workspace
- Function Files
 - Let you make your own MATLAB Functions
 - All variables within a function are *local*
 - All information must be passed to functions as parameters
 - Subfunctions are supported

48

Basic Parts of a Function M-File



49

Flow Control Statements...

- *if* Statement

```

if ((attendance >= 0.90) & (grade_average >= 60))
    pass = 1;
end;
    
```

- *while* Loops

```

eps = 1;
while (1+eps) > 1
    eps = eps/2;
end
eps = eps*2
    
```

50

...Flow Control Statements

- *for* Loop

```

a = zeros(k,k) % Preallocate matrix
for m = 1:k
    for n = 1:k
        a(m,n) = 1/(m+n -1);
    end
end
    
```

- *switch* Statement

```

method = 'Bilinear';
switch lower(method)
    case {'linear','bilinear'}
        disp('Method is linear')
    case 'cubic'
        disp('Method is cubic')
    otherwise
        disp('Unknown method.')
end
Method is linear
    
```

51

M-file Programming Features

- SubFunctions
- Varying number of input/output arguments
- *Local* and *Global* Variables
- Obtaining User Input
 - Prompting for Keyboard *Input*
 - Pausing During Execution
- Errors and Warnings
 - Displaying *error* and *warning* Messages
- Shell Escape Functions (*!* Operator)
- Optimizing MATLAB Code
 - Vectorizing loops
 - Preallocating Arrays

52

Function M-file

```

function r = ourrank(X,tol)
% rank of a matrix
s = svd(X);
if (nargin == 1)
    tol = max(size(X)) * s(1) * eps;
end
r = sum(s > tol);

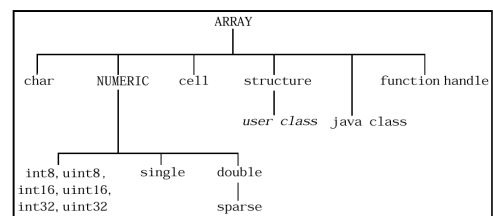
function [mean,stddev] = ourstat(x)
[m,n] = size(x);
if m == 1
    m = n;
end
>[m std]=ourstat(1:9); end
mean = sum(x)/m;
stddev = sqrt(sum(x.^2)/m - mean.^2);
    
```

Multiple Input Arguments use ()

Multiple Output Arguments, use []

53

Data Types



- Numeric Arrays
- Multidimensional Arrays
- Structures and Cell Arrays

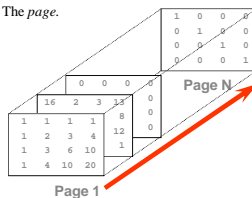
54

Multidimensional Arrays

The first references array dimension 1, the row.

The second references dimension 2, the column.

The third references dimension 3, the page.



```

» A = pascal(4);
» A(:, :, 2) = magic(4)
A(:, :, 1) =
    1     1     1     1
    1     2     3     4
    1     3     6    10
    1     4    10    20

A(:, :, 2) =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1

» A(:, :, 9) =
    diag(ones(1,4));
    
```

55

Structures

- Arrays with named data containers called *fields*.

patient	<pre> » patient.name='John Doe'; » patient.billing = 127.00; » patient.test = [79 75 73; 180 178 177.5; 220 210 205]; </pre>
<pre> - name _____ 'John Doe' - billing _____ 127.00 - test _____ </pre>	<pre> 79 75 73 180 178 177.5 220 210 205 </pre>

- Also, Build structure arrays using the *struct* function.
- Array of *structures*

```

» patient(2).name='Katty Thomson';
» Patient(2).billing = 100.00;
» Patient(2).test = [69 25 33; 120 128 177.5; 220
210 205];
    
```

56

Cell Arrays

- Array for which the elements are *cells* and can hold other MATLAB arrays of different types.

```

» A(1,1) = {[1 4 3;
0 5 8;
7 2 9]};
» A(1,2) = {'Anne Smith'};
» A(2,1) = {3+7i};
» A(2,2) = {-pi:pi/10:pi};
    
```

cell 1.1 1 4 3 0 5 8 7 2 9	cell 1.2 Anne Smith
cell 2.1 3+7i	cell 2.2 [-pi:pi/10:pi]

- Using braces `{}` to point to elements of cell array
- Using *celldisp* function to display cell array

57

Nonlinear Numerical Functions

- inline* function

– Use *char* function to convert *inline* object to *string*

```

» f = inline('3*sin(2*x.^2)','x')
f =
    Inline function:
    f(x) = 3*sin(2*x.^2)
» f(2)
ans =
    2.9681
    
```

- Numerical Integration using *quad*

```

» Q = quad('1./(x.^3-2*x-5)',0,2);
» F = inline('1./(x.^3-2*x-5)');
» Q = quad(F,0,2);
» Q = quad('myfun',0,2)
    
```

- Note: *quad* function use adaptive Simpson quadrature

```

function y = myfun(x)
y = 1./(x.^3-2*x-5);
    
```

58

Nonlinear Numerical Functions

- fzero* finds a zero of a single variable function

```
[x, fval] = fzero(fun, x0, options)
```

– fun is *inline* function or *m-function*

- fminbnd* minimize a single variable function on a fixed interval. $x_1 < x < x_2$

```
[x, fval] = fminbnd(fun, x1, x2, options)
```

- fminsearch* minimize a several variable function

```
[x, fval] = fminsearch(fun, x0, options)
```

- Use *optimset* to determine *options* parameter.

```
options = optimset('param1', value1, ...)
```

59

Ordinary Differential Equations

- An explicit ODE with initial value:

$$y' = f(t, y)$$

$$y(t_0) = y_0$$

- Using *ode45* for non-stiff functions and *ode23t* for stiff functions.

```
[t, y] = solver(odefun, tspan, y0, options)
```

```
function dydt = odefun(t, y)
Initialvalue
[initialtime finaltime]
```

- Use *odeset* to define options parameter

60

ODE Example

$$y_1'' - (1 - y_1^2)y_1' + y_1 = 0$$

```
function dydt=myfunc(t,y)
dydt=zeros(2,1);
dydt(1)=y(2);
dydt(2)=(1-y(1)^2)*y(2)-y(1);
```

$$y_1' = y_2$$

$$y_2' = (1 - y_1^2)y_2 - y_1$$

```
>> [t,y]=ode45('myfunc',[0 20],[2;0])
```

Note:
Help on `odeset` to set options for more accuracy and other useful utilities like drawing results during solving.

61

Handle Graphics

- Graphics in MATLAB consist of *objects*:
 - *root, figure, axes, image, line, patch, rectangle, surface, text, light*
- Creating Objects
- Setting Object Properties Upon Creation
- Obtaining an Object's Handles
- Knowing Object Properties
- Modifying Object Properties
 - Using *Command Line*
 - Using *Property Editor*

62

Graphics Objects

63

Obtaining an Object's Handle

- Upon Creation

```
h_line = plot(x_data, y_data, ...)
```

- Utility Functions

0 - root object handle

gcf - current figure handle

gca - current axis handle

gco - current object handle

What is the current object?

- Last object created
- OR
- Last object clicked

- FINDOBJ

```
h_obj = findobj(h_parent, 'PropertyName', 'Value', ...)
```

Default = 0 (root object)

64

Modifying Object Properties

- Obtaining a list of current properties:

```
get(h_object)
```

- Obtaining a list of settable properties:

```
set(h_object)
```

- Modifying an object's properties

- Using *Command Line*

```
set(h_object, 'PropertyName', 'New_Value', ...)
```

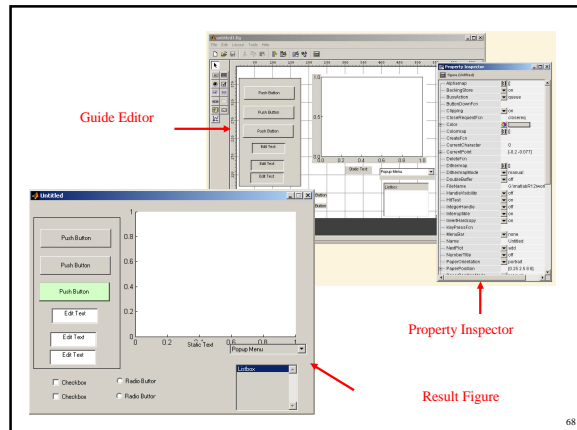
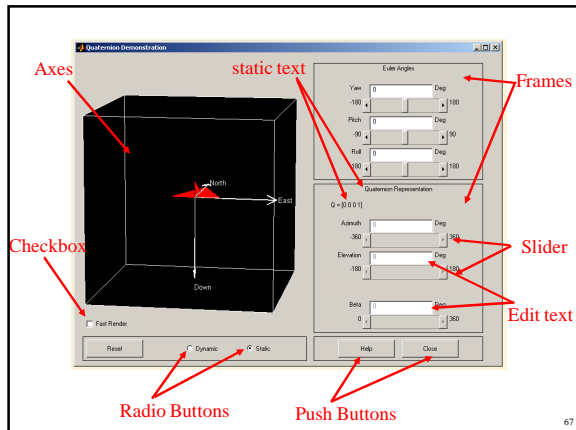
- Using *Property Editor*

65

Graphical User Interface

- What is GUI?
- What is *figure* and *.fig file?
- Using *guide* command
- GUI controls
- GUI menus

66



Conclusion

- Matlab is a language of technical computing.
- Matlab, a high performance software, a high-level language
- Matlab supports GUI, API, and ...
- Matlab Toolboxes best fits different applications
- Matlab ...

Getting more help

- Contact <http://www.mathworks.com/support>
 - You can find more help and FAQ about mathworks products on this page.
- Contact comp.soft-sys.matlab Newsgroup
 - Using Google Groups Page to Access this page
 - <http://groups.google.com/>