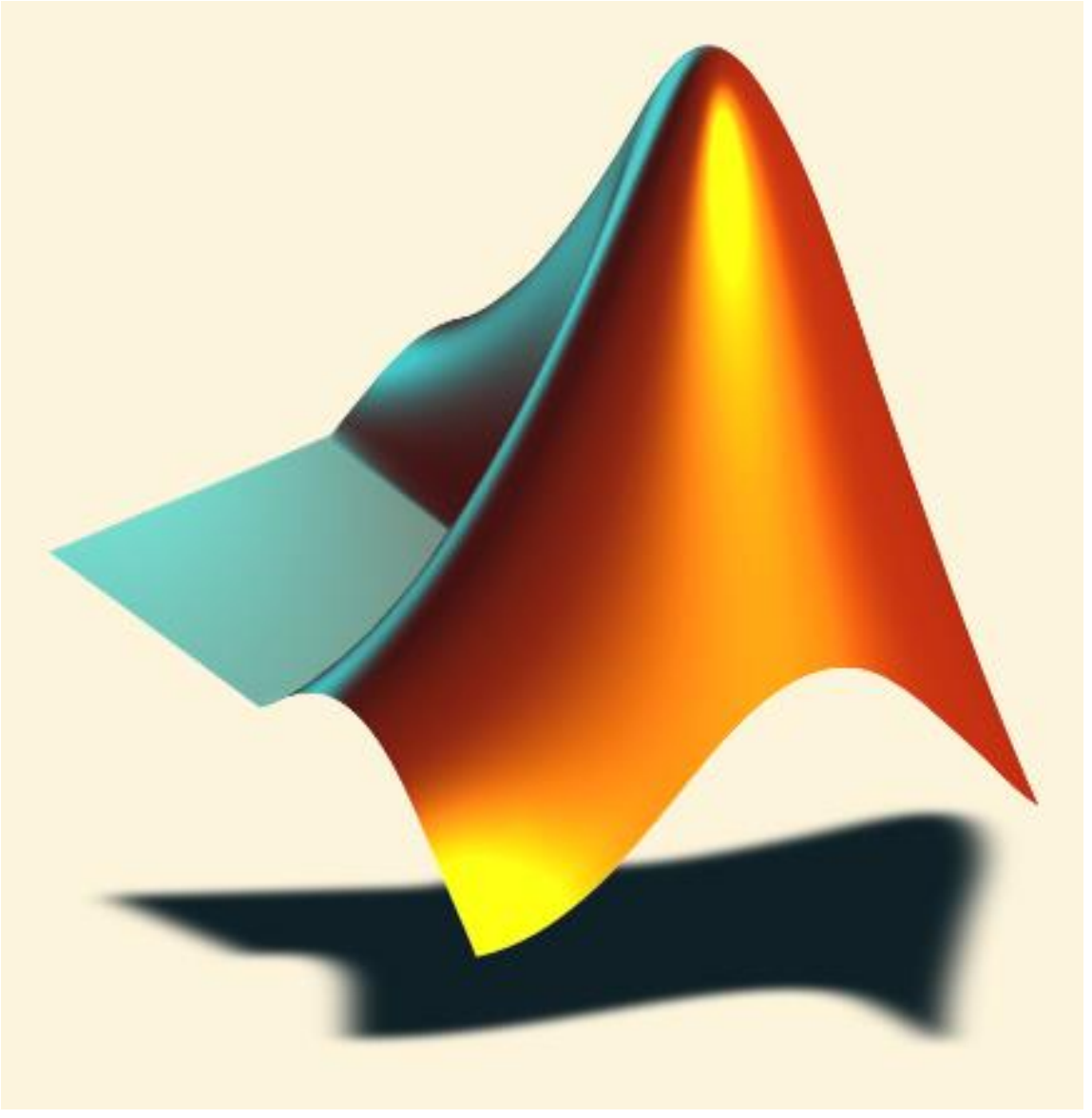
MATLAB



Contents

1. First part of workshop

- What is Matlab?
- MATLAB Components
- MATLAB Desktop
- Matrices
  - Numerical Arrays
  - String Arrays
- Importing and Exporting Data
- Elementary math with MATLAB

Contents

Continued

2. Second part of workshop

- M-file Programming
  - Functions vs. Scripts
  - Variable Type/Scope
  - Debugging MATLAB functions
  - Flow control in MATLAB
- Other Tidbits
    - Function minimization
    - Root finding
    - Solving ODE's
- Graphics Fundamentals
- Data Types → most likely won't have time for it

What is MATLAB?

- Integrated software environment
  - Computation
  - Visualization
  - Easy-to-use environment

- High-level language
  - Data types
  - Functions
  - Control flow statements
  - Input/output
  - Graphics
  - Object-oriented programming capabilities

MATLAB Parts

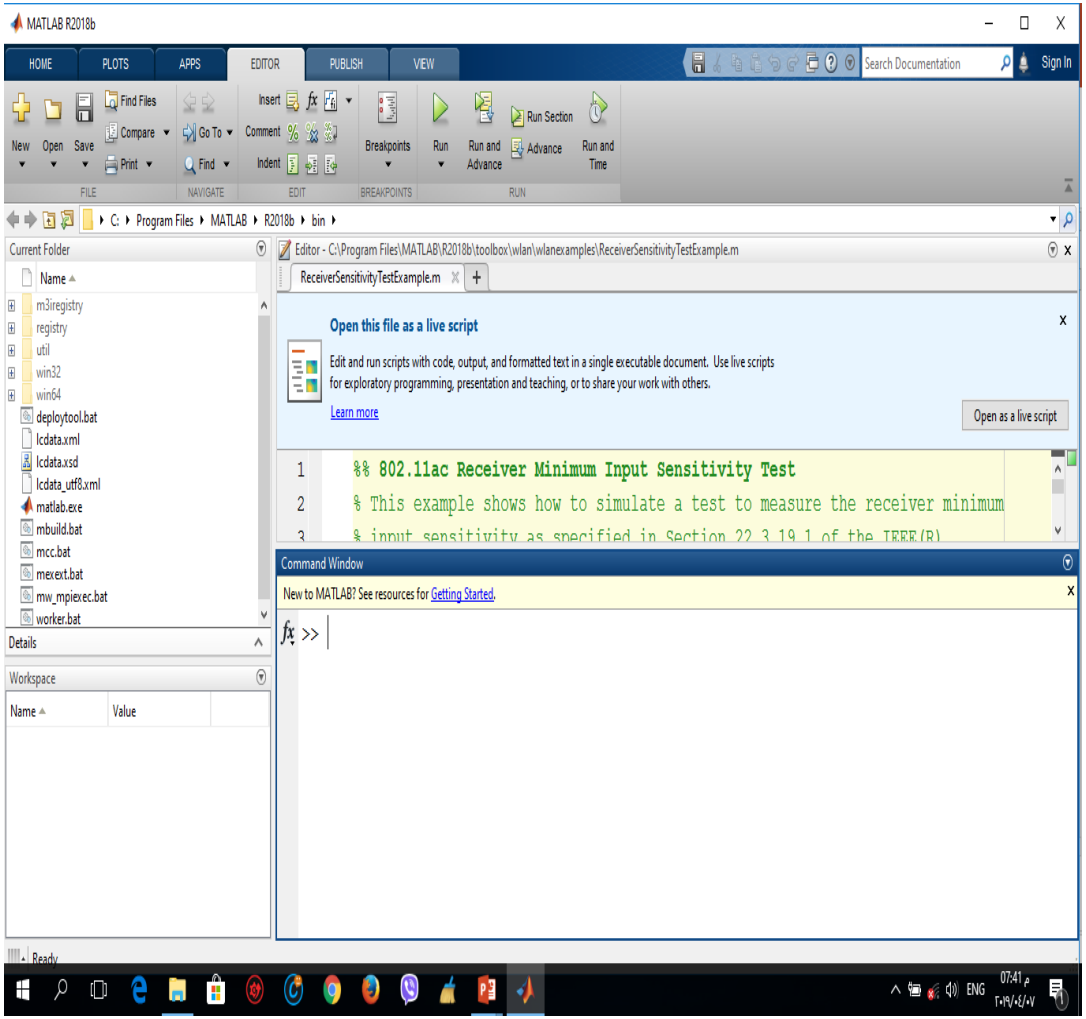
- High Level Development Environment
- Programming Language
- Graphics
- Toolboxes
- Application Program Interface

Toolboxes

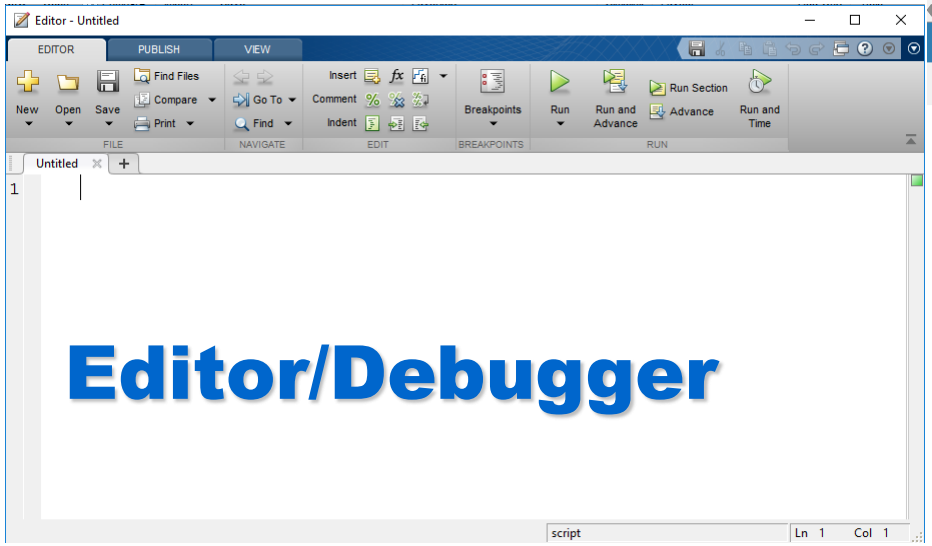
- Collections of functions to solve problems from several application fields.
  - DSP (Digital Signal Processing) Toolbox
  - Image Toolbox
  - Wavelet Toolbox
  - Neural Network Toolbox
  - Fuzzy Logic Toolbox
  - Control Toolbox
  - Multibody Simulation Toolbox
  - And many others…

MATLAB 7x Desktop Tools

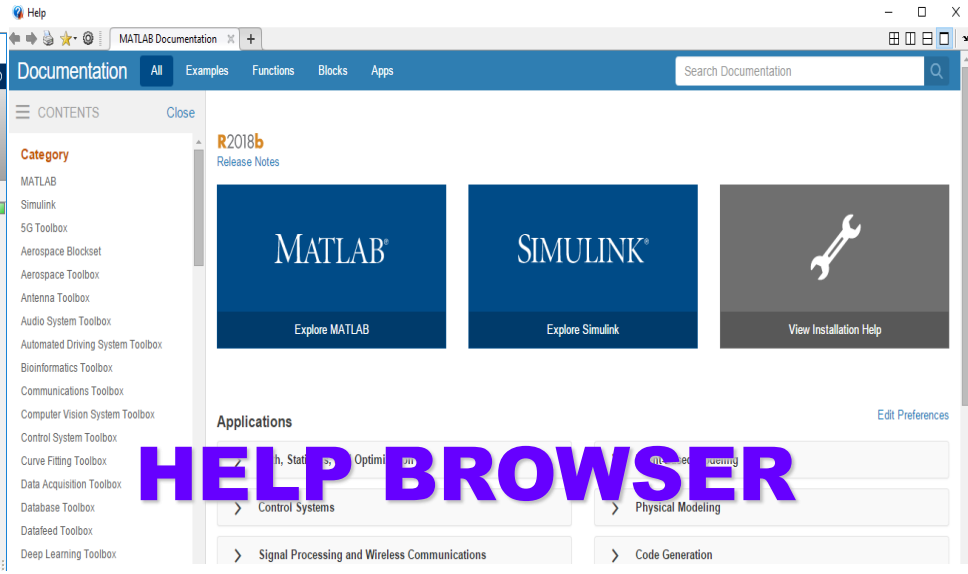
- Command Window
- Command History
- Help Browser
- Workspace Browser
- Editor/Debugger



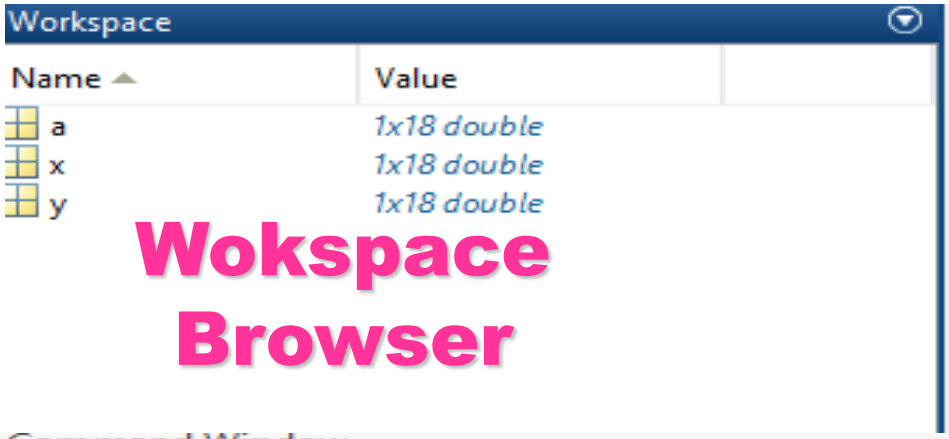
MATLAB 2018b DESKTOP



Editor/Debugger



HELP BROWSER



Wokspace Browser

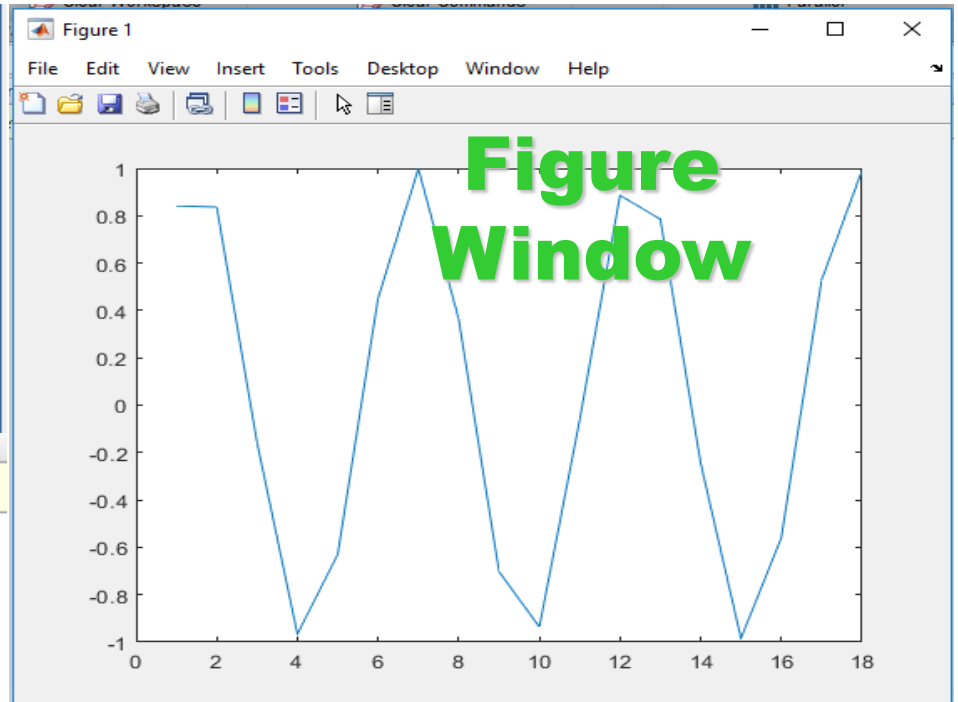
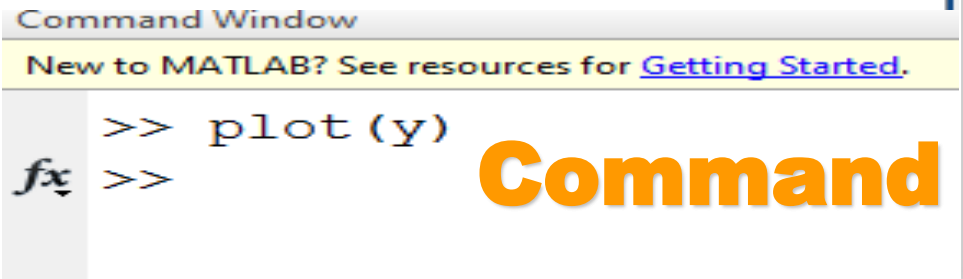


Figure Window



Command

MATLAB – editor (new file)

There are several way to create new file as following:

- Script
- Live script
- Function
- Class
- Figure
- App

Calculations at the Command Line

```
» -5/(4.8+5.32)^2
ans =
   -0.0488
» (3+4i)*(3-4i)
ans =
    25
» 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

A Note about Workspace:
Numbers stored in double-precision floating point format

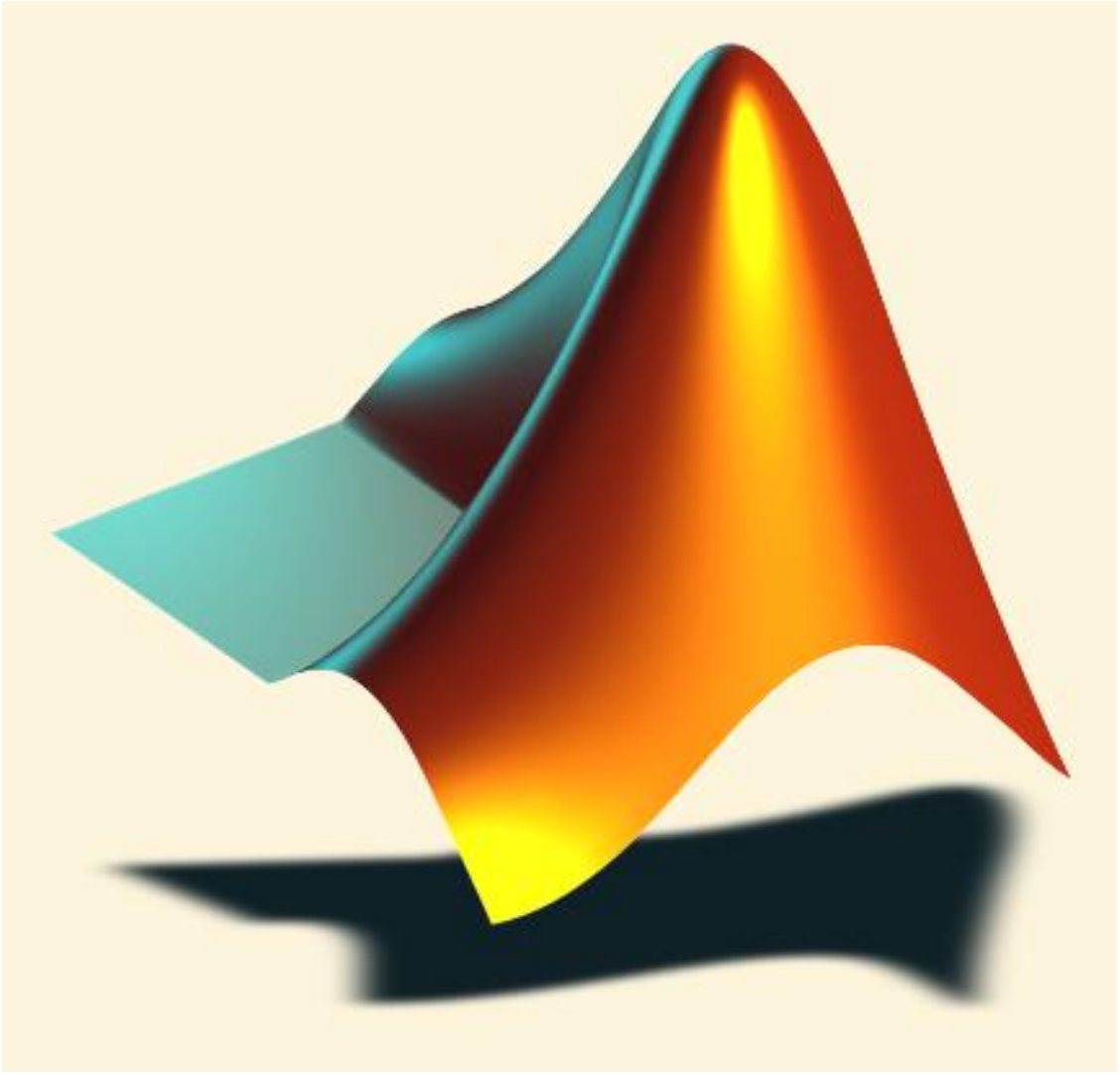
General Functions

- whos: List current variables and their size
- clear: Clear variables and functions from memory
- cd: Change current working directory
- dir: List files in directory
- pwd: Tells you the current directory you work in
- format: Set output format (long, short, etc.)
- diary(filename): Saves all the commands you type in a file in the current directory called filename

Getting help

- *help* command (**>>help**)
- *lookfor* command (**>>lookfor**)
- Help Browser (**>>doc**)
- *helpwin* command (**>>helpwin**)

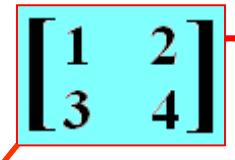
Handling Matrices in Matlab



Matrices

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

Entering Numeric Arrays



NOTE:
1) Row separator
semicolon (;)

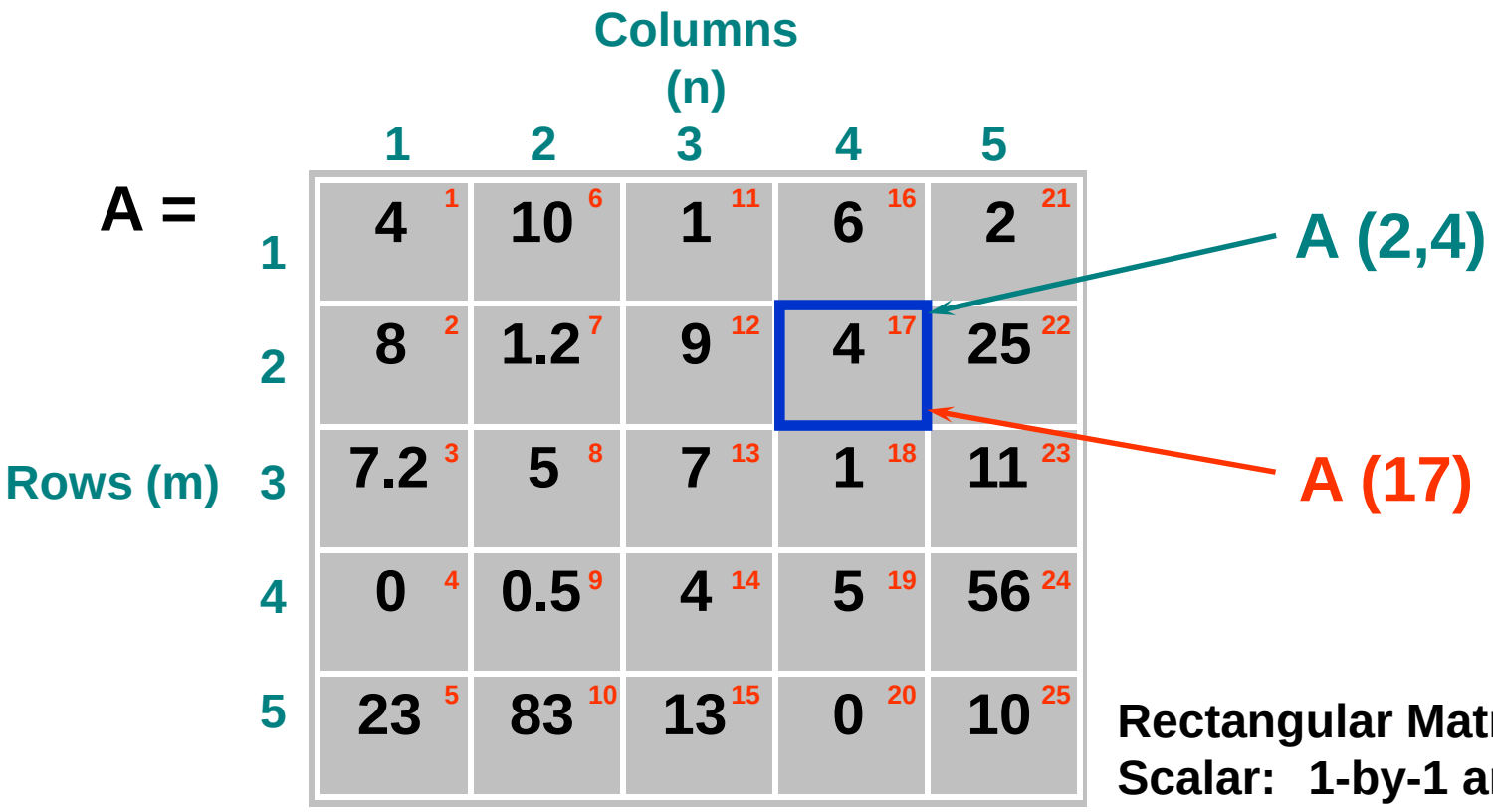
2) Column separator
space OR comma (,)

```
» 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 (internally, it is regarded as such)
- In MATLAB, the arrays are always rectangular

The Matrix in MATLAB



Rectangular Matrix:
Scalar: 1-by-1 array
Vector: m-by-1 array
1-by-n array
Matrix: m-by-n array

Entering Numeric Arrays

Scalar expansion

Creating sequences:
colon operator (:)

Utility functions for creating matrices.

```
» w=[1 2;3 4] + 5
w =
     6     7
     8     9
» 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
```

Examples (Entering Numeric Arrays)

- Enter the matrix:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 5 \end{bmatrix}$$

- enter the row vector:

$$\mathbf{B} = \begin{bmatrix} 6 & 7 & 8 & 9 \end{bmatrix}$$

- Enter the column vector :

$$\mathbf{C} = \begin{bmatrix} 6 \\ 7 \\ 8 \\ 9 \end{bmatrix}$$

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
     3     6     4     8
     9    12    12    16
     5    10     6    12
    15    20    18    24
```

Use square brackets [ ]

4*a

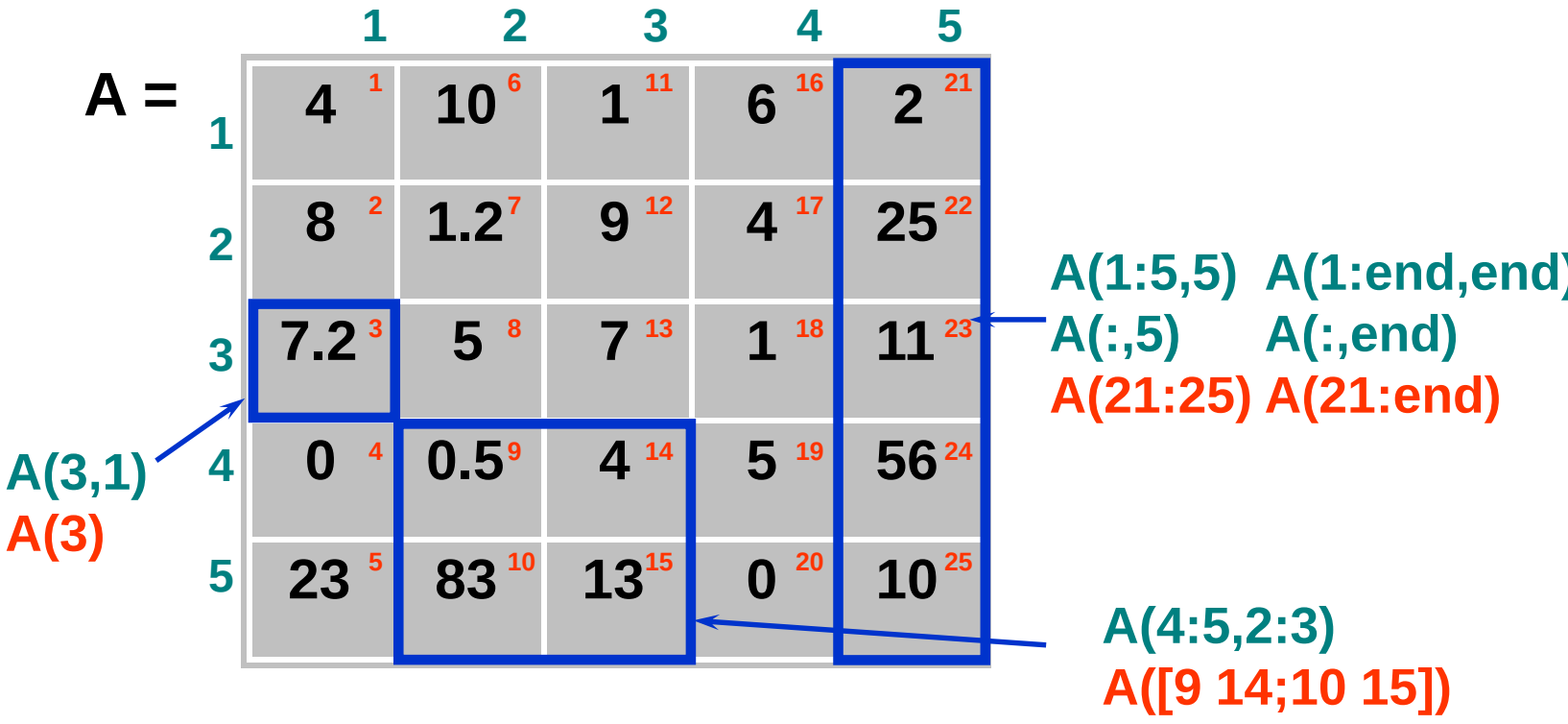
Note:

The resulting matrix must be rectangular

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.
```

Array Subscripting / Indexing



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
```

[2x4]

[4x3]

[2x4]*[4x3] → [2x3]

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)*b(2,4)

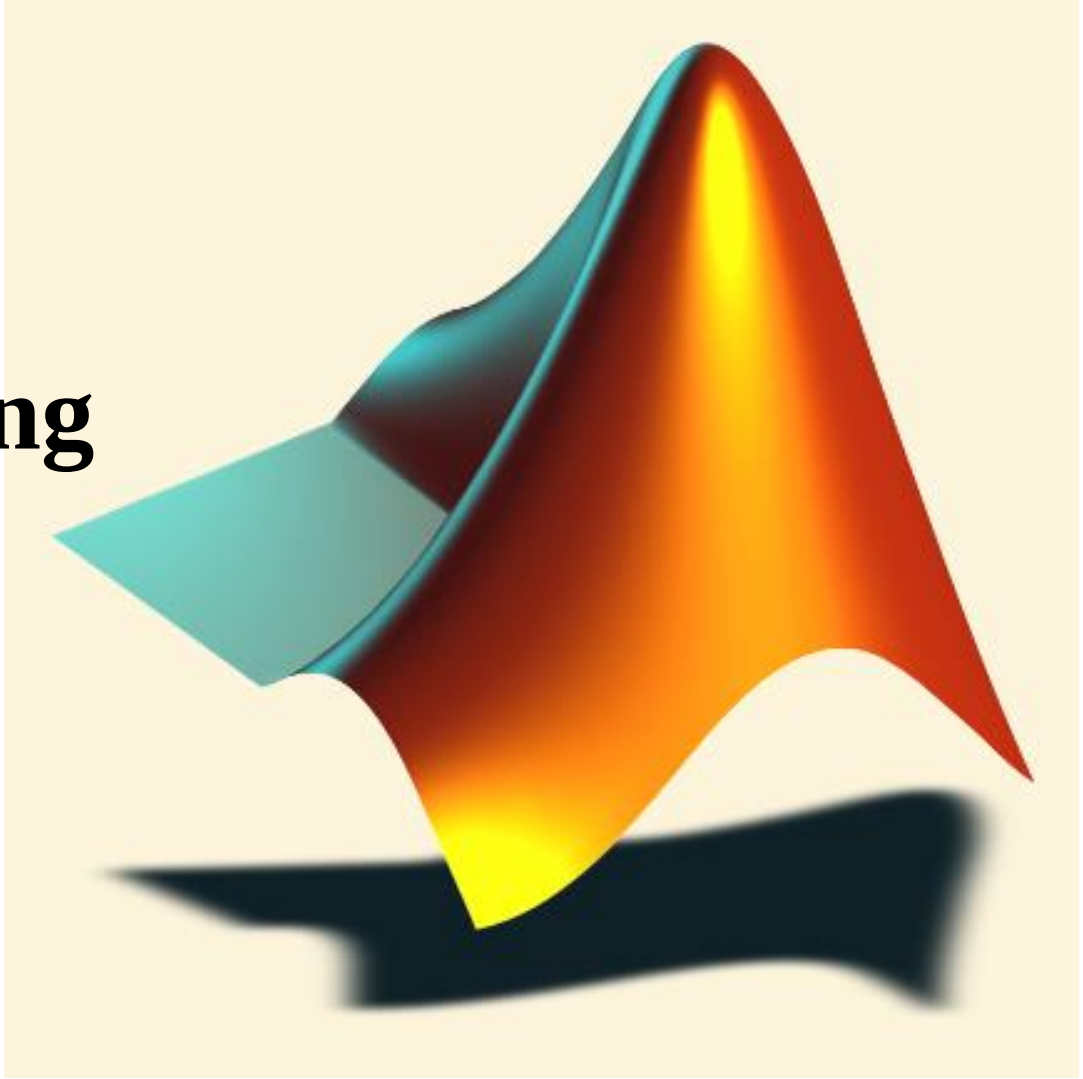
Matrix Manipulation Functions

- 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

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

Character String Manipulation



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)

- Indexing same as for numeric arrays

String Array Concatenation

Using [ ] operator:
Each row must be same length

Row separator:
semicolon (;)

Column separator:
space / comma (,)

```
» 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?
```

1x9 vectors

1x19 vectors

2x19 matrix

For strings of different length:

- STRVCAT
- char

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

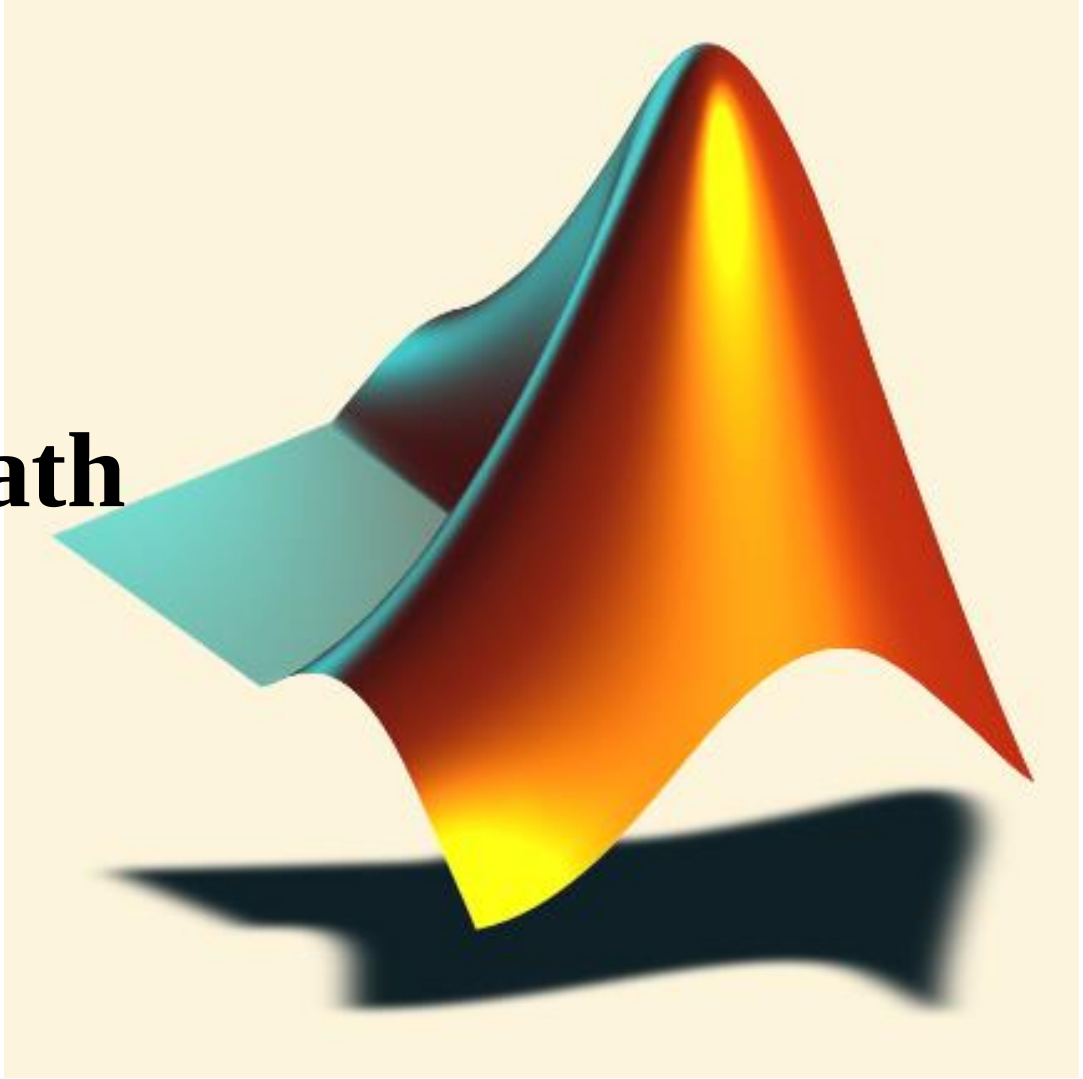
2x19 matrix (zero padded)

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

Elementary Math



Elementary Math

- Logical Operators
- Math Functions
- Polynomial and Interpolation

Logical Operations

= =	equal to
>	greater than
<	less than
>=	Greater or equal
<=	less or equal
~	not
&	and
|	or

```
» 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

Elementary Math Function

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

Elementary Math Function

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

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
- diff, gradient: Differences and Numerical Gradient

Polynomials and Interpolation

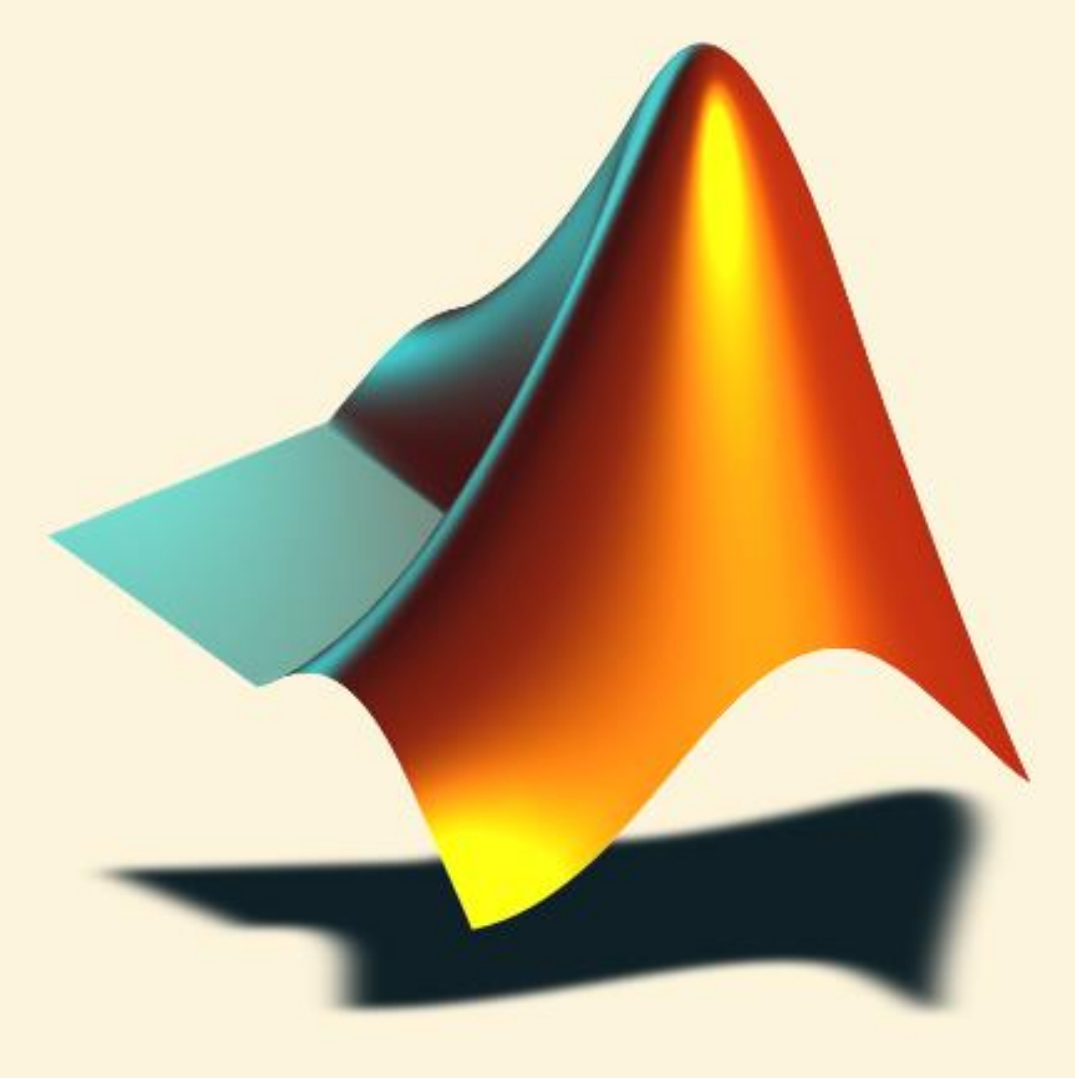
- Polynomials
  - Representing
  - Roots (**>> roots**)
  - Evaluation (**>> polyval**)
  - Derivatives (**>> polyder**)
  - Curve Fitting (**>> polyfit**)
  - Partial Fraction Expansion (>>**residue**)

- Interpolation
  - One-Dimensional (**interp1**)
  - Two-Dimensional (**interp2**)

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
```

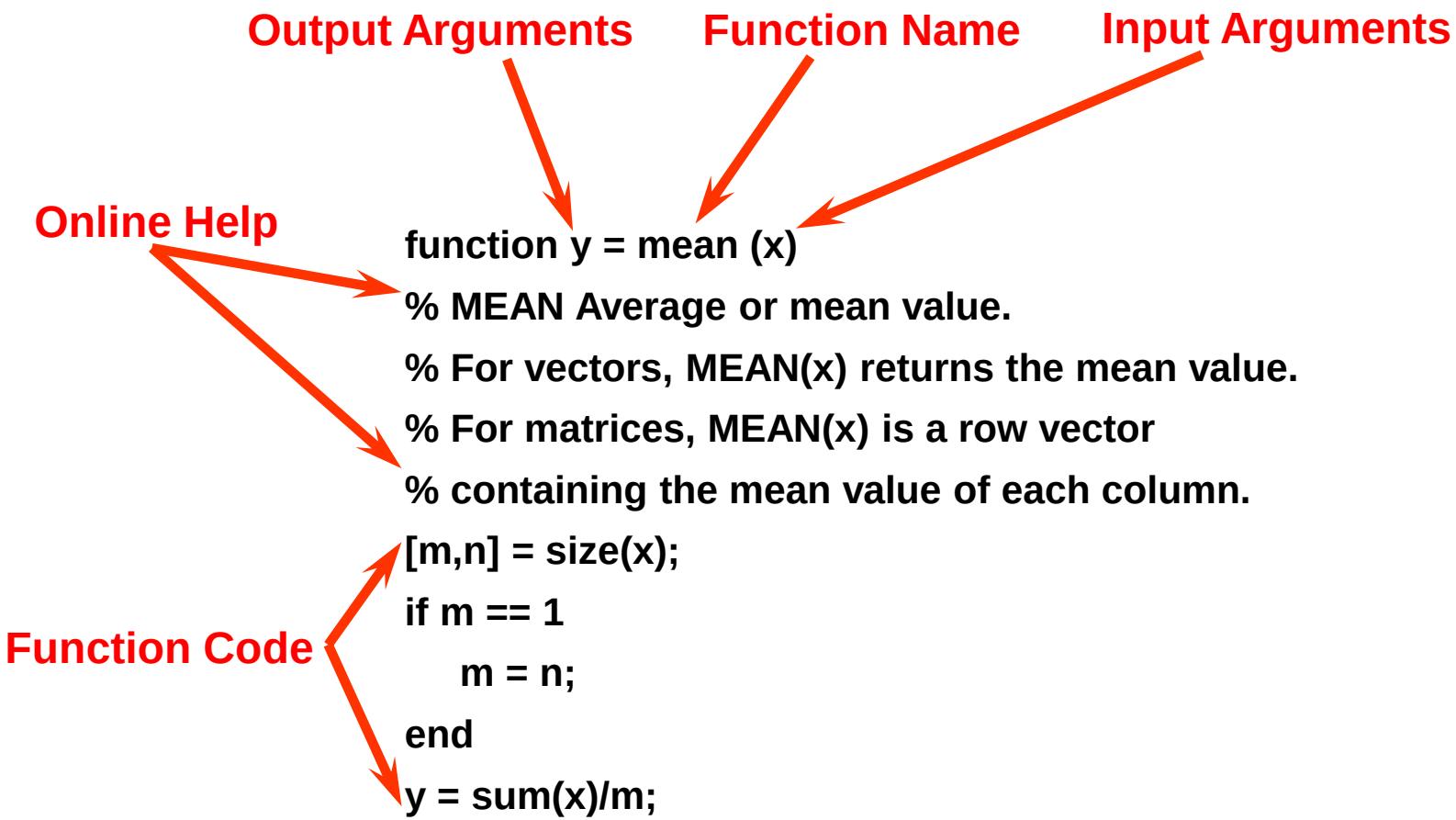
Programming and Application Development



Topics discussed…

- The concept of m-file in MATLAB
- Script versus function files
- The concept of workspace
- Variables in MATLAB
  - Type of a variable
  - Scope of a variable
- Flow control in MATLAB
- The Editor/Debugger

Basic Parts of a Function M-File



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

The concept of *Workspace*

- At any time in a MATLAB session, the code has a workspace associated with it
- The workspace is like a sandbox in which you find yourself at a certain point of executing MATLAB
- Base Workspace: the workspace in which you live when you execute commands from prompt
- Remarks:
  - Each MATLAB function has its own workspace (its own sandbox)
  - A function invoked from a calling function has its own and separate workspace (sandbox)

Variable Types in MATLAB

- Local Variables
  - In general, a variable in MATLAB has ***local*** scope, that is, it's only available in its workspace
  - The variable disappears when the workspace ceases to exist
  - Recall that a script does not define a new workspace – be careful, otherwise you can step on variables defined at the level where the script is invoked
  - Since a function defines its own workspace, a variable defined in a function is local to that function
  - Variables defined outside the function should be passed to function as arguments. ***Furthermore***, the arguments are passed by value
  - Every variable defined in the subroutine, if to be used outside the body of the function should be returned back to the calling workspace

Variable Types in MATLAB

- Global Variables
  - These are variables that are available in multiple workspaces
  - They have to be explicitly declared as being global
  - Not going to expand on this, since using global variables are a bad programming practice

- Note on returning values from a function
  - Since all variables are local and input arguments are passed by value, when returning from a function a variable that is modified in the function will not appear as modified in the calling workspace unless the variable is either global, or declared a return variable for that function

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
```

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
```

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
```

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

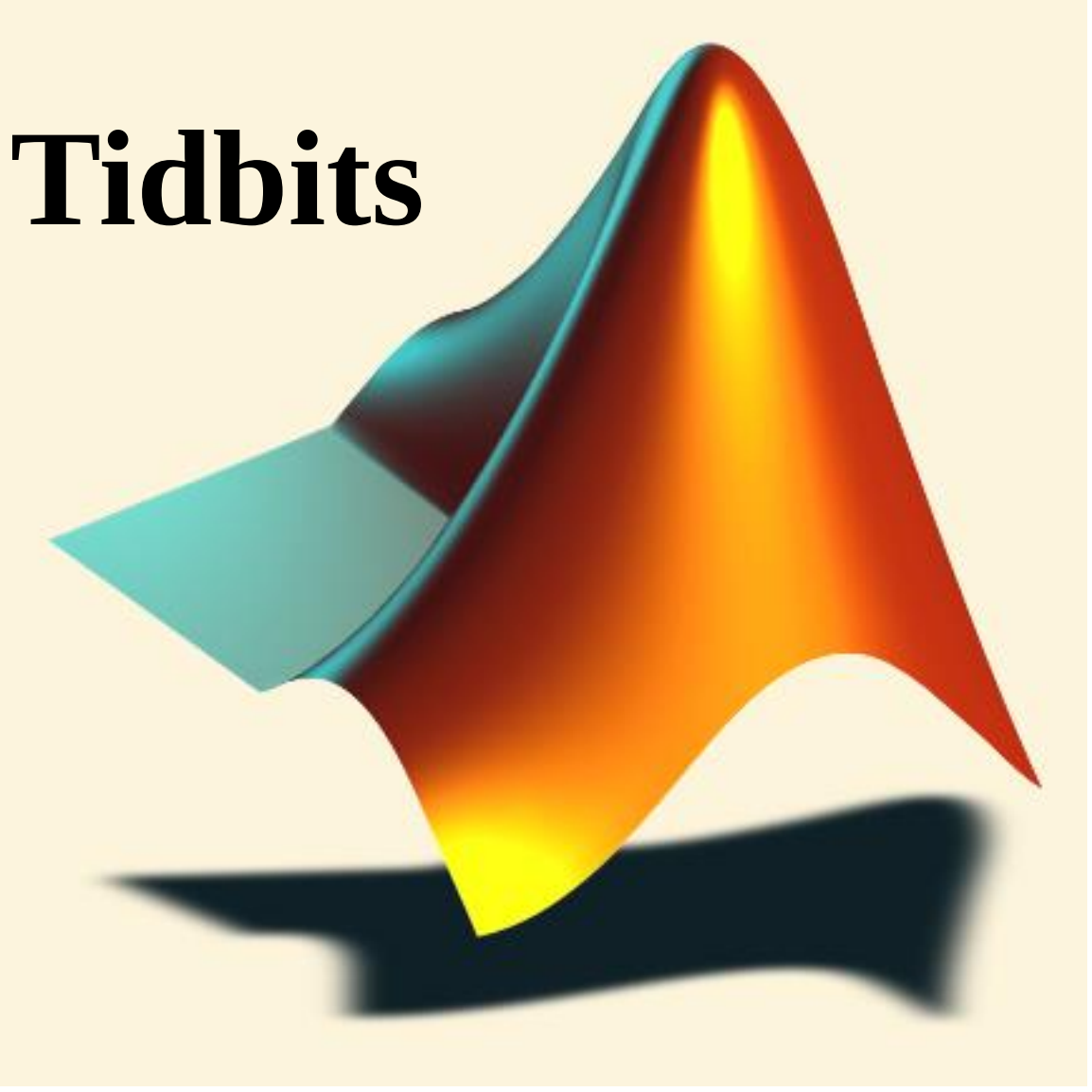
```
» M = dlmread(filename,delimiter,range)
```

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);
```

Other Tidbits



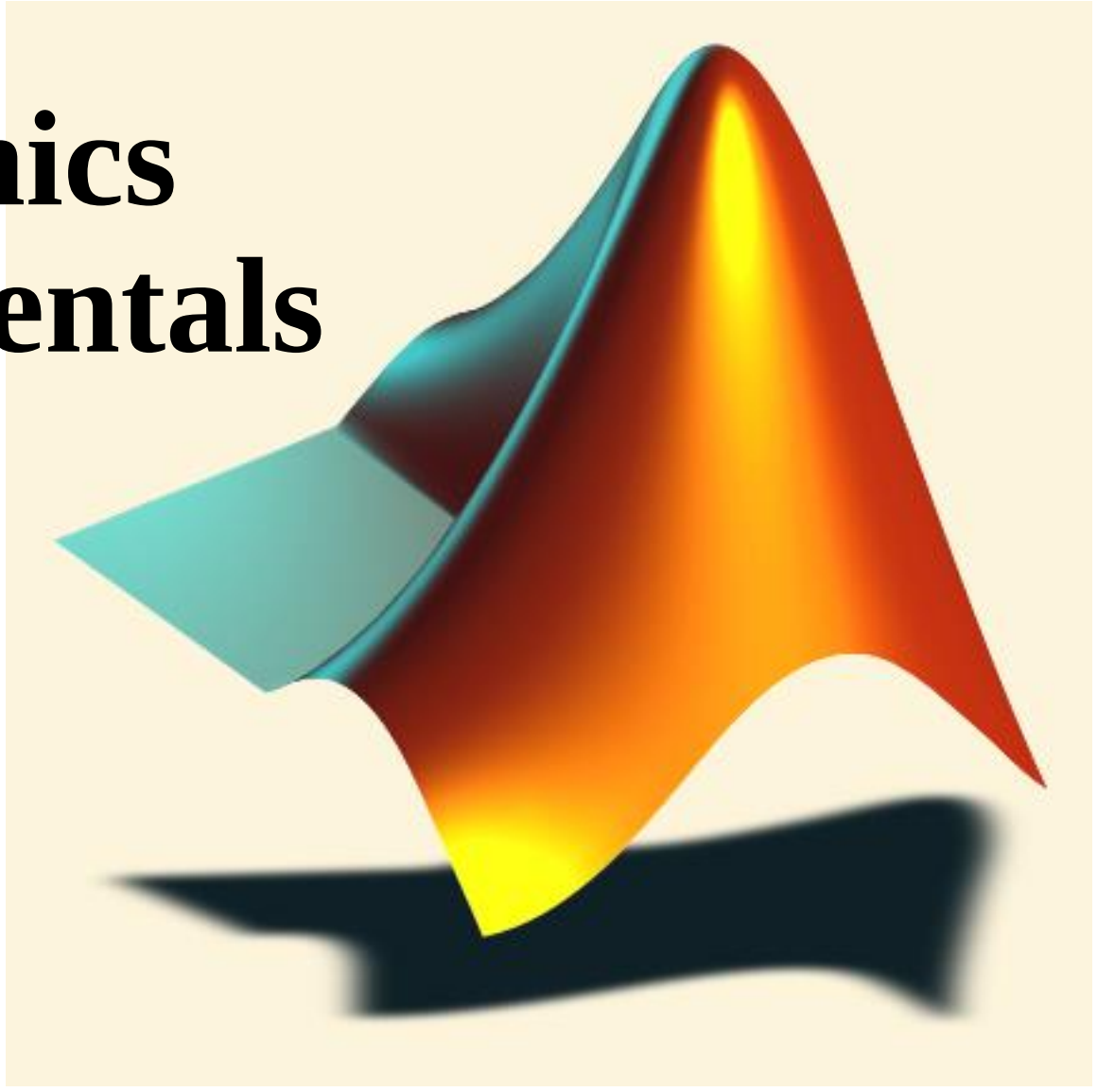
Nonlinear Numerical Functions

- ***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
```

Graphics Fundamentals

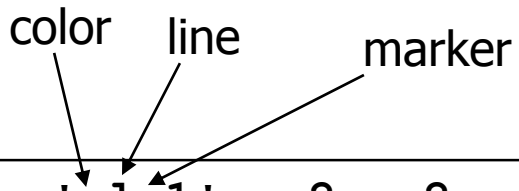


Graphics and Plotting in MATLAB

- 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

2-D Plotting

Syntax:

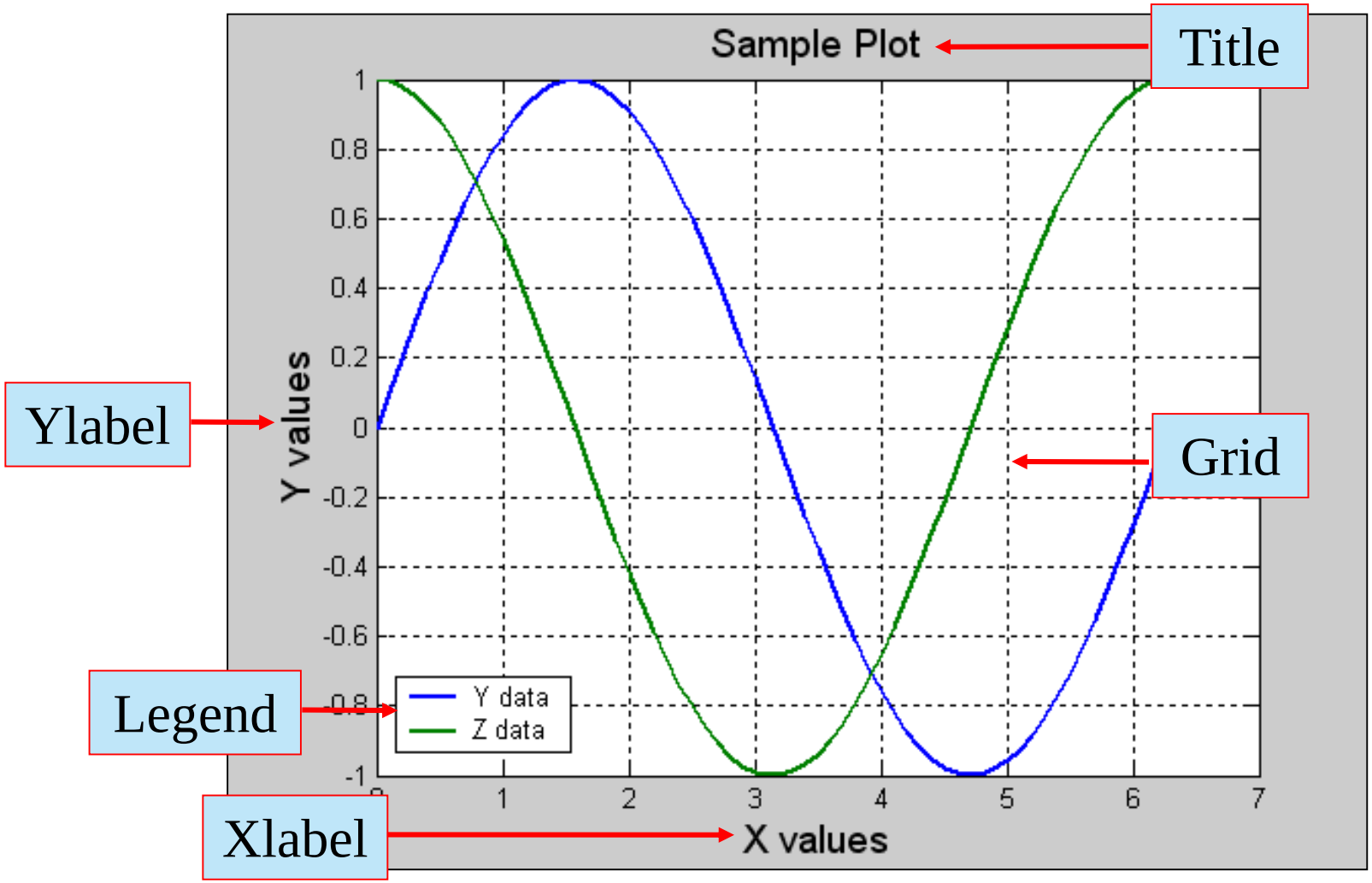



```
plot(x1, y1, 'clm1', x2, y2, 'clm2', ...)
```

Example:

```
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
```

Sample Plot


Sample Plot
Title
Ylabel
Grid
Legend
Xlabel
Y values
X values
Y data
Z data
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
1
2
3
4
5
6
7

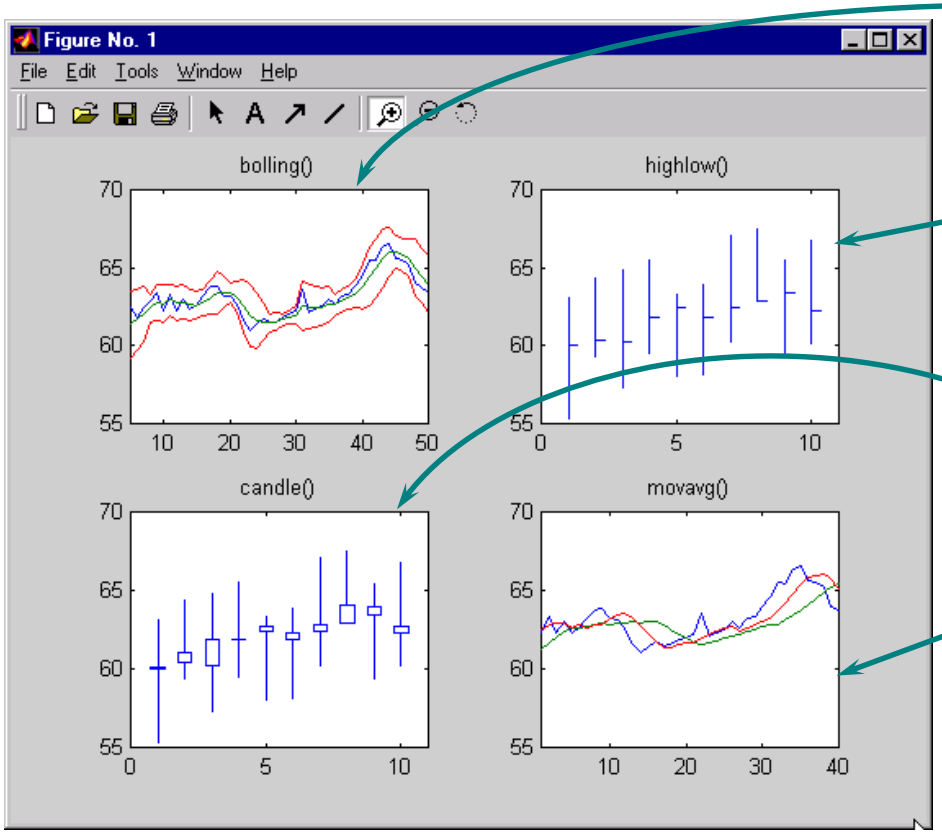
Displaying Multiple Plots

- Nomenclature:
  - *Figure window* – the window in which MATLAB displays plots
  - *Plot* – a region of a window in which a curve (or surface) is displayed

- Three typical ways to display multiple curves in MATLAB (other combinations are possible…)
  - One figure contains one plot that contains multiple curves
    - Requires the use of the command “hold” (see MATLAB help)
  - One figure contains multiple plots, each plot containing one curve
    - Requires the use of the command “subplot”
  - Multiple figures, each containing one or more plots, each containing one or more curves
    - Requires the use of the command “figure” and possibly “subplot”

Subplots

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




```
»subplot(2,2,1);
» ...

»subplot(2,2,2)
» ...

»subplot(2,2,3)
» ...

»subplot(2,2,4)
» ...
```

The "*figure*" Command

- Use if you want to have several figures open for plotting
- The command by itself creates a new figure window and returns its handle

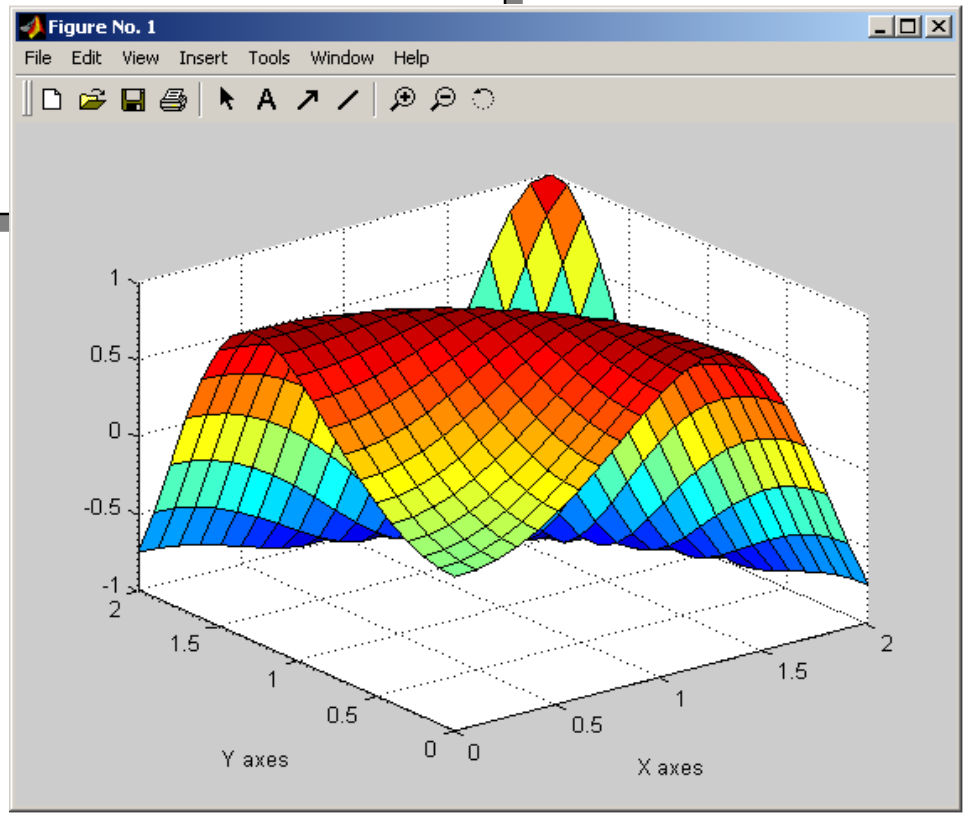
  >> figure
- If you have 20 figures open and want to make figure 9 the default one (this is where the next plot command will display a curve) use

  >> figure(9)

  >> plot(…)
- Use the command *close(9)* if you want to close figure 9 in case you don't need it anymore

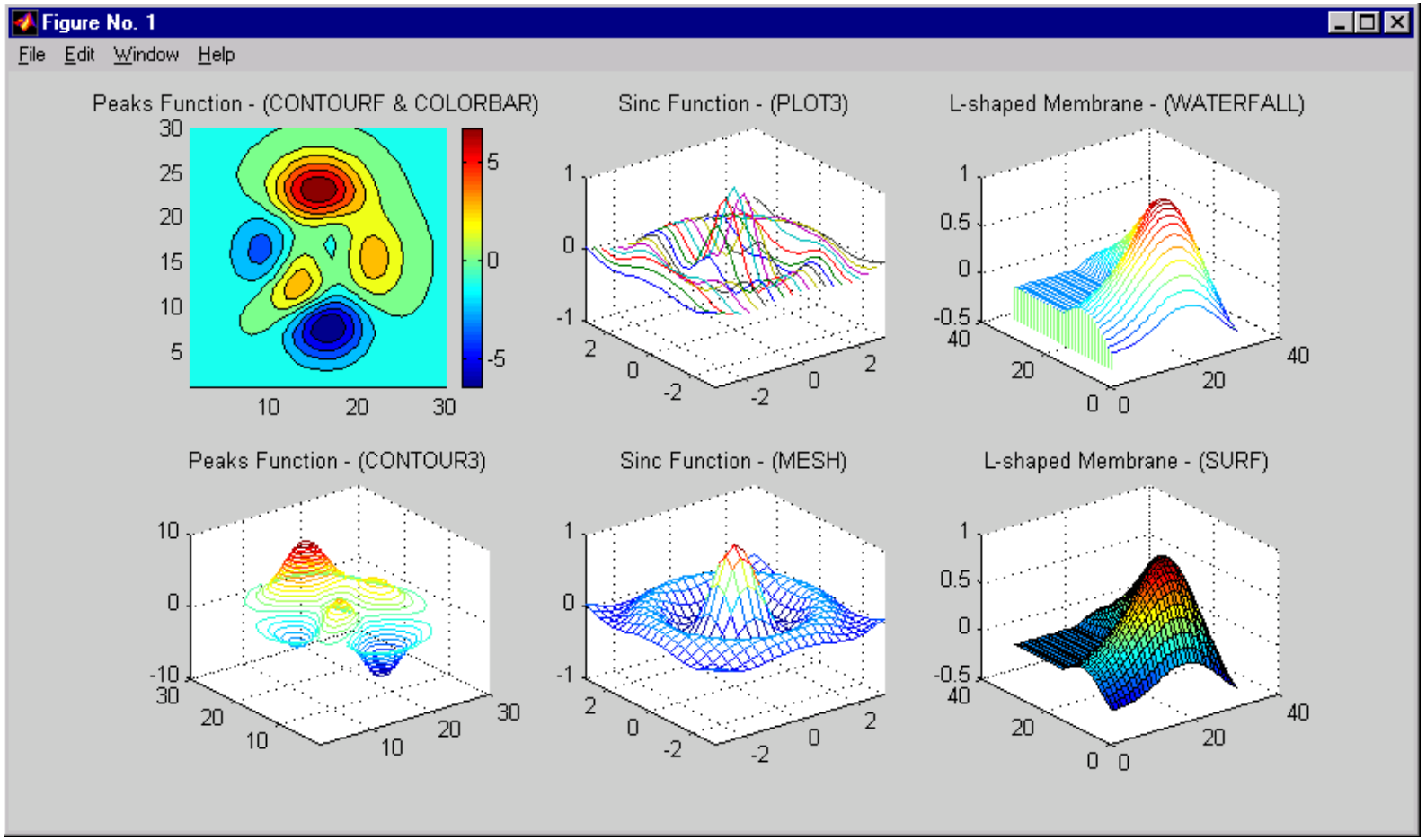
Surface Plot Example

```
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')
```



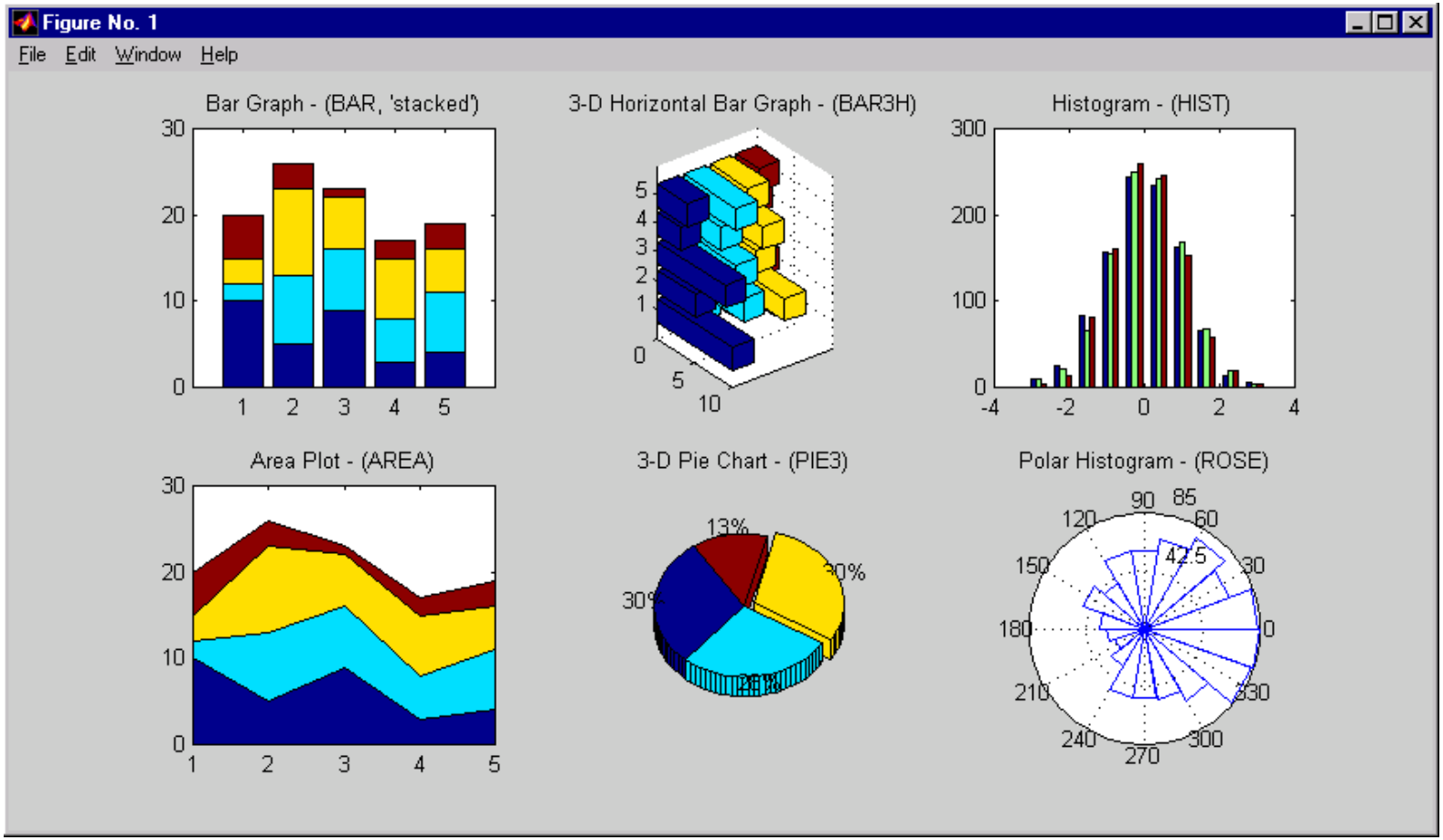
3-D Surface Plotting

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



Specialized Plotting Routines

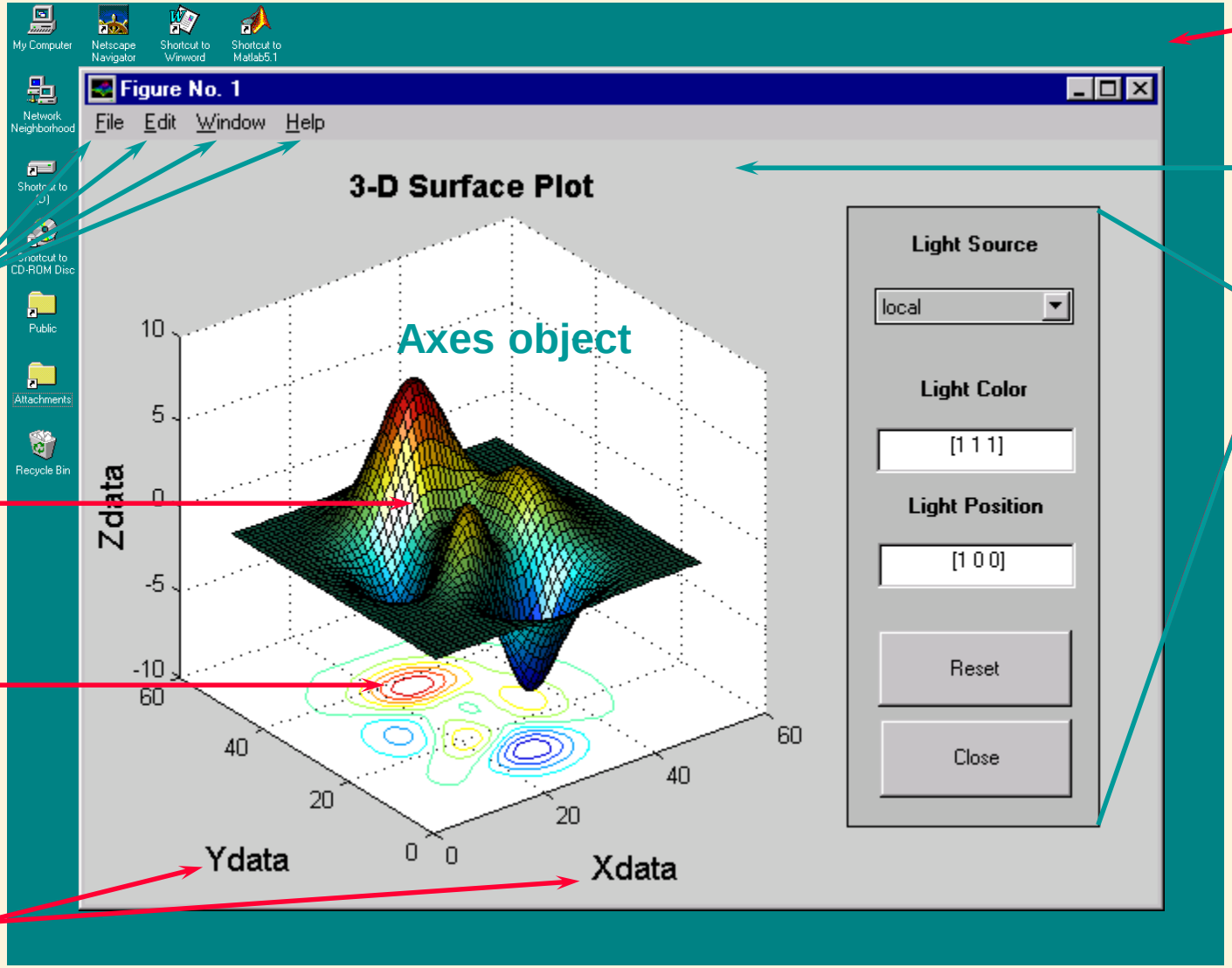
bar-bar3h-hist-area-pie3-rose



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*

Graphics Objects



Obtaining an Object's Handle

1. Upon Creation

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

2. 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**

3. FINDOBJ

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

Default = 0 (root object)

Modifying Object Properties

- Obtaining a list of current properties:

```
get(h_object)
```

- Obtaining a list of settable properties:

```
set(h_object)
```

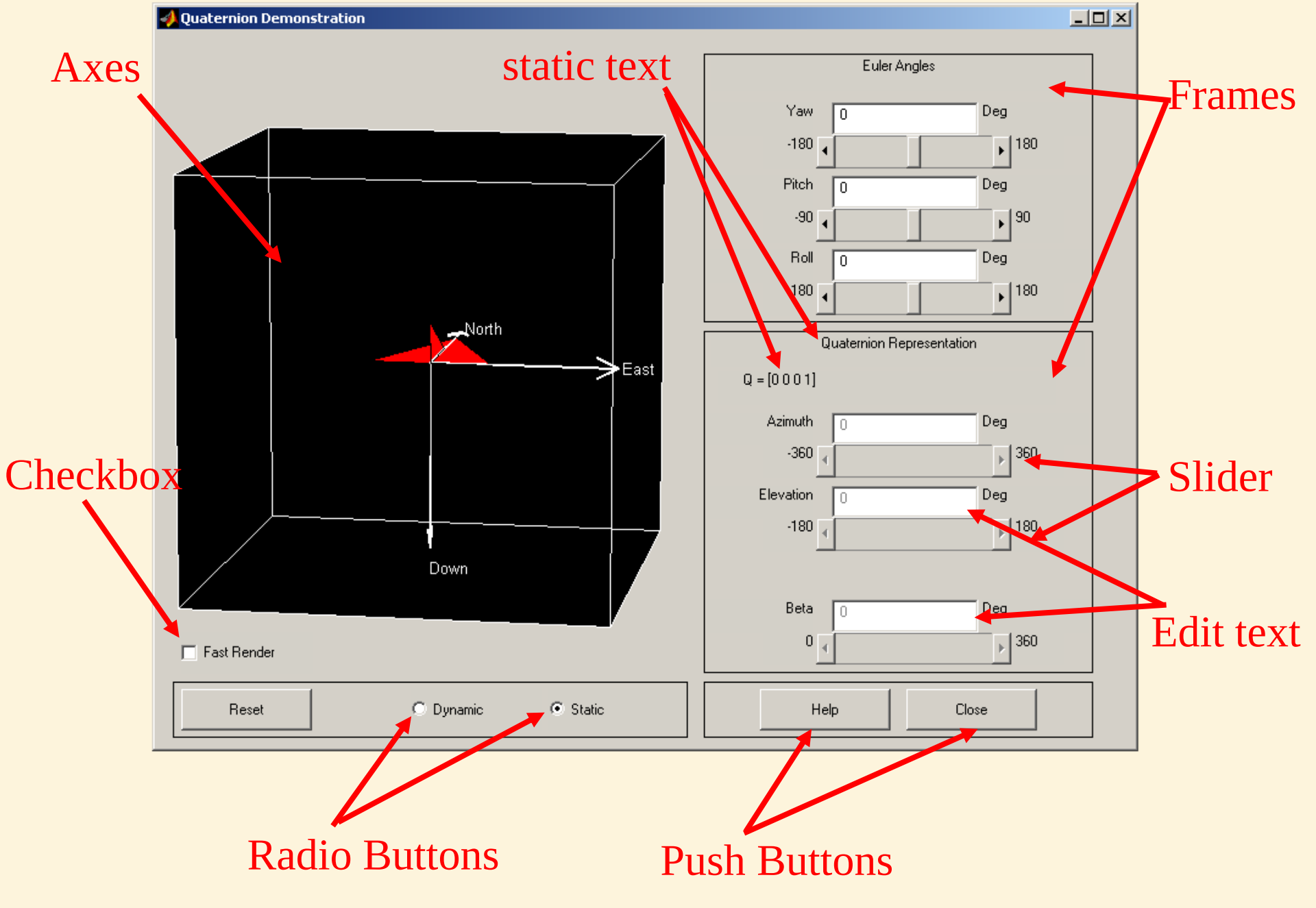
- Modifying an object's properties
  - Using Command Line

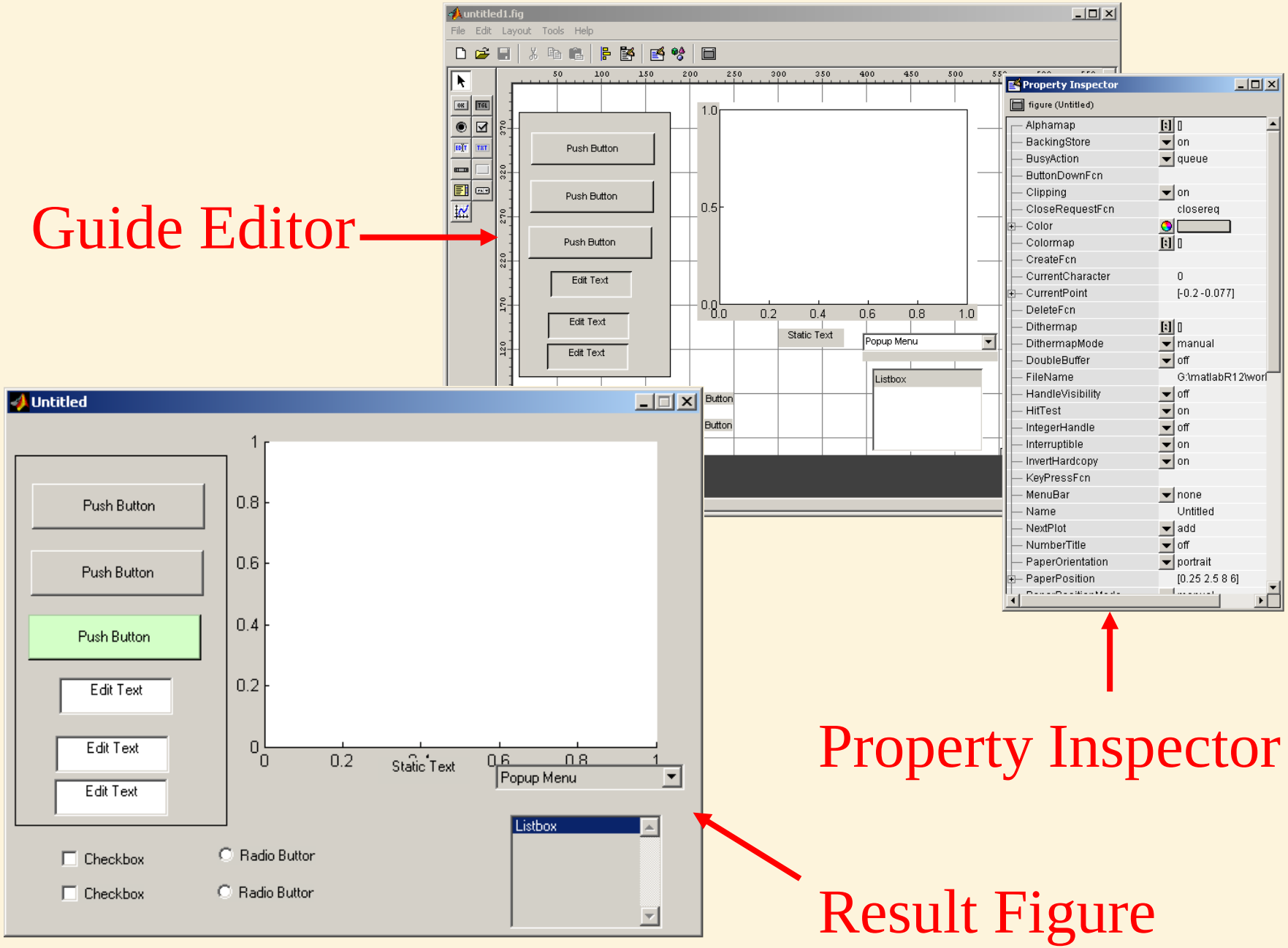
```
set(h_object,'PropertyName','New_Value',...)
```

  - Using Property Editor

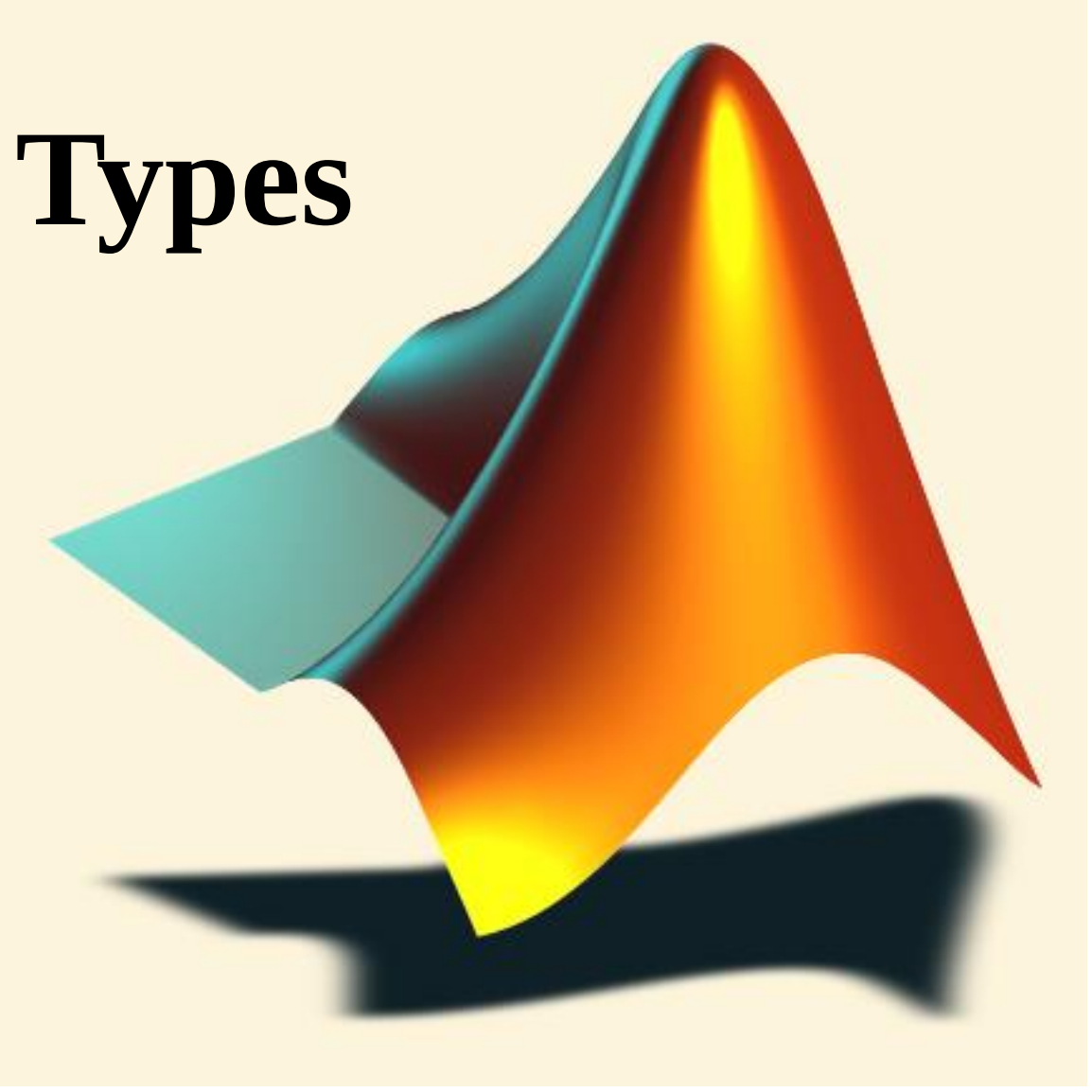
Graphical User Interface

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

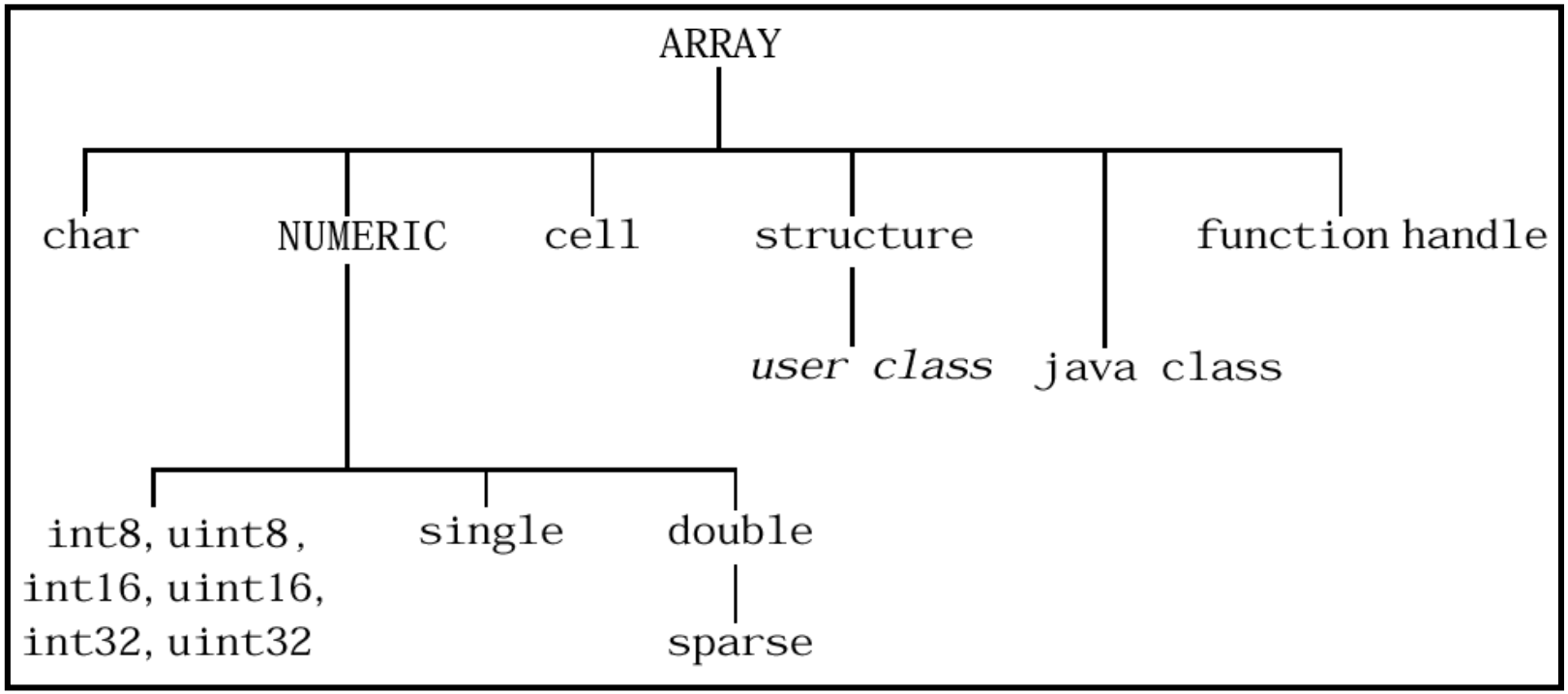

Quaternion Demonstration
Axes
static text
Frames
Euler Angles
Yaw
0
Deg
-180
180
Pitch
0
Deg
-90
90
Roll
0
Deg
180
180
North
East
Down
Quaternion Representation
Q = [0 0 0 1]
Azimuth
0
Deg
-360
360
Elevation
0
Deg
-180
180
Beta
0
Deg
0
360
Checkbox
Slider
Edit text
Fast Render
Reset
Dynamic
Static
Help
Close
Radio Buttons
Push Buttons


Guide Editor
untitled1.fig
File Edit Layout Tools Help
Push Button
Push Button
Push Button
Edit Text
Edit Text
Edit Text
Static Text
Popup Menu
Listbox
Button
Button
Property Inspector
figure (Untitled)
Alphamap
BackingStore
BusyAction
ButtonDownFcn
Clipping
CloseRequestFcn
Color
Colormap
CreateFcn
CurrentCharacter
CurrentPoint
DeleteFcn
Dithermap
DithermapMode
DoubleBuffer
FileName
HandleVisibility
HitTest
IntegerHandle
Interruptible
InvertHardcopy
KeyPressFcn
MenuBar
Name
NextPlot
NumberTitle
PaperOrientation
PaperPosition
Untitled
Push Button
Push Button
Push Button
Edit Text
Edit Text
Edit Text
Static Text
Popup Menu
Listbox
Checkbox
Checkbox
Radio Buttor
Radio Buttor
Property Inspector
Result Figure

Data Types



Data Types



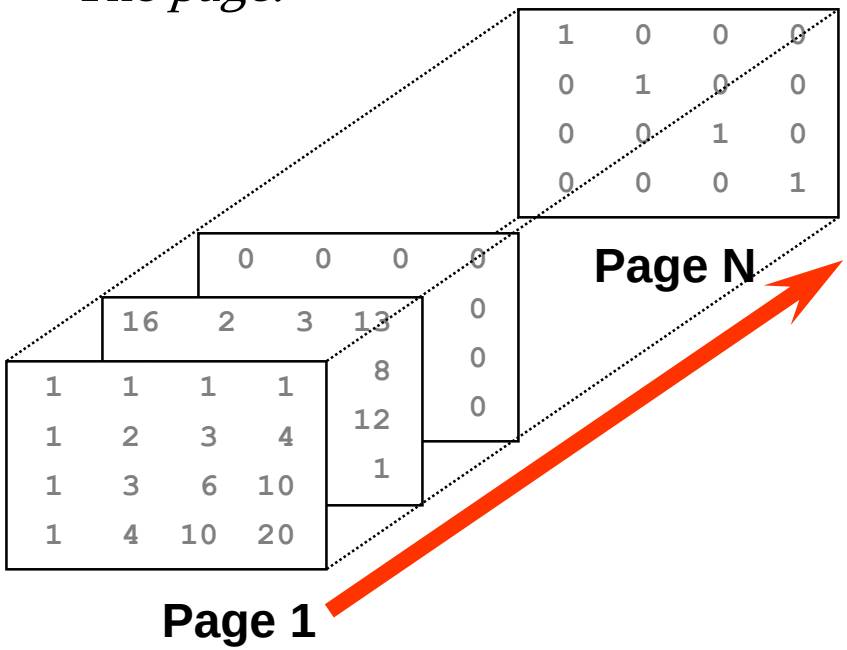
- Numeric Arrays
- Multidimensional Arrays
- Structures and Cell Arrays

Multidimensional Arrays

The first references array dimension 1, the row.

The second references dimension 2, the column.

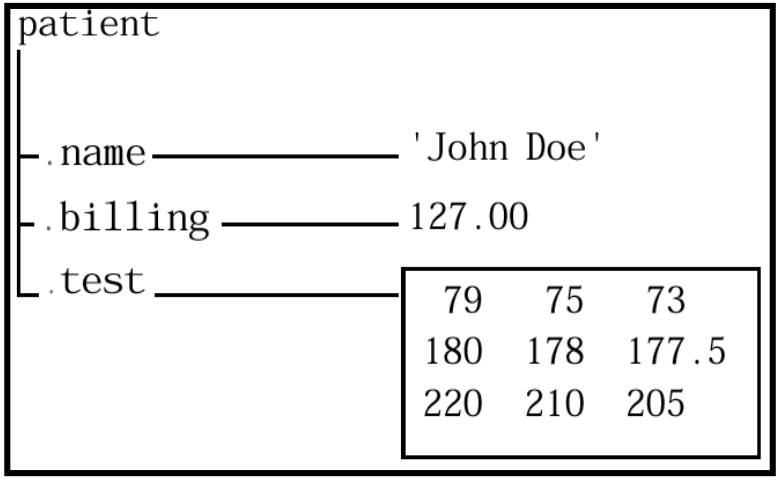
The third references dimension 3, The *page.*




```
» 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));
```

Structures

- Arrays with named data containers called ***fields***.




```
» patient.name='John Doe';
» patient.billing = 127.00;
» patient.test= [79 75 73;
180 178 177.5;
220 210 205];
```

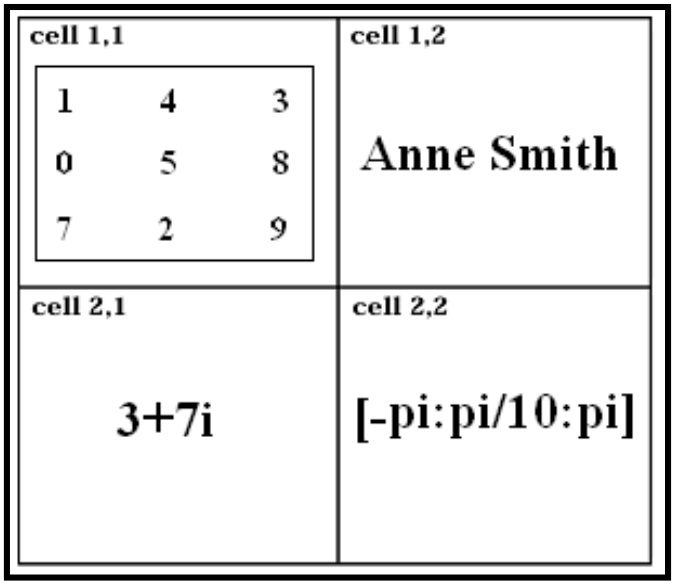
- Also, Build structure arrays using the ***struct*** function.
- Array of *structures*

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

Cell Arrays

- Array for which the elements are *cell*s 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};
```



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

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.