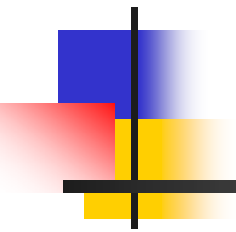
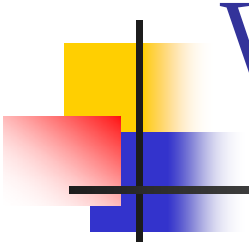


An Introduction to MATLAB

What is MATLAB?

- A high-performance language for technical computing (Mathworks, 1998)
- Typical uses of MATLAB
  - Mathematical computations
  - Algorithmic development
  - Model prototyping (prior to complex model development)
  - Data analysis and exploration of data (visualization)
  - Scientific and engineering graphics for presentation

Why MATLAB?

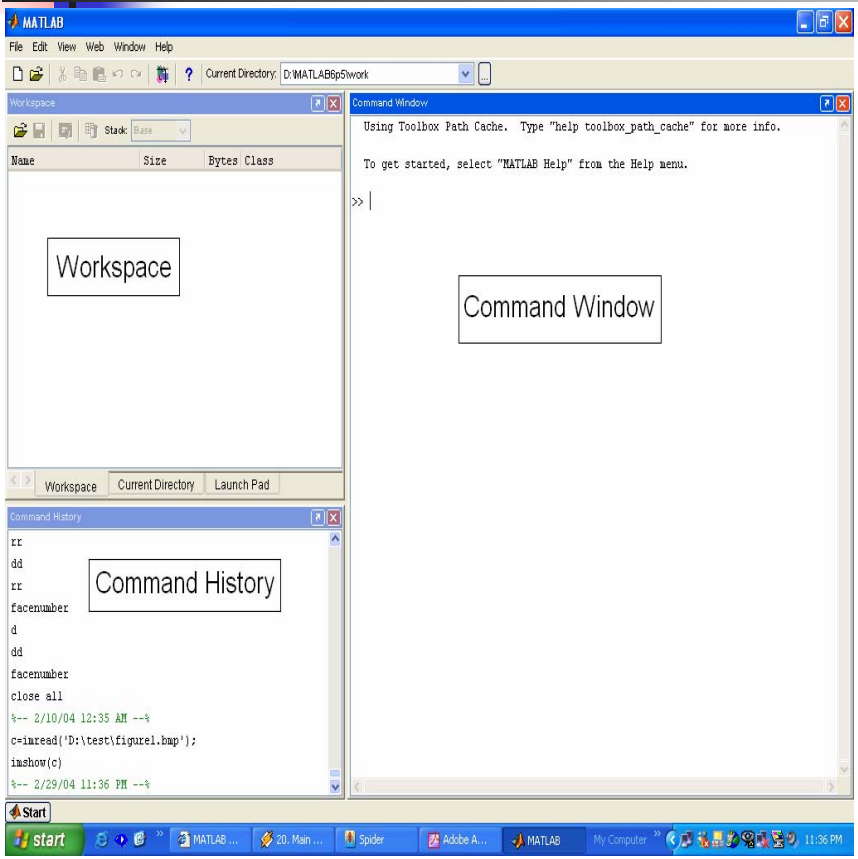


- Because it simplifies the analysis of mathematical models
- It frees you from coding in high-level languages (saves a lot of time - with some computational speed penalties)
- Provides an extensible programming/visualization environment
- Provides professional looking graphs

MATLAB

- MATLAB==MATrix LABoratory
- It is widely used to solve different types of scientific problems.
- The basic data structure is a complex double precision matrix.

The MATLAB Environment



- MATLAB window components:

  Workspace

  > Displays all the defined variables

  Command Window

  > To execute commands in the MATLAB environment

  Command History

  > Displays record of the commands used

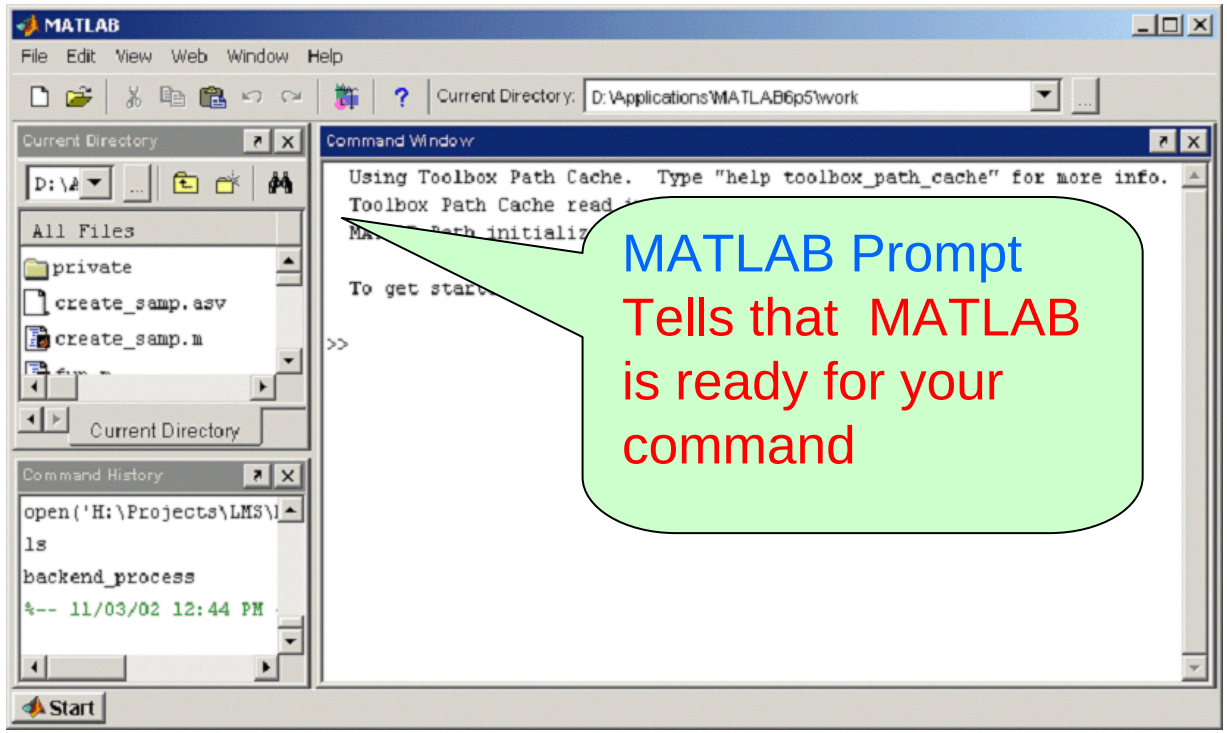
  File Editor Window

  > Define your functions

Run MATLAB

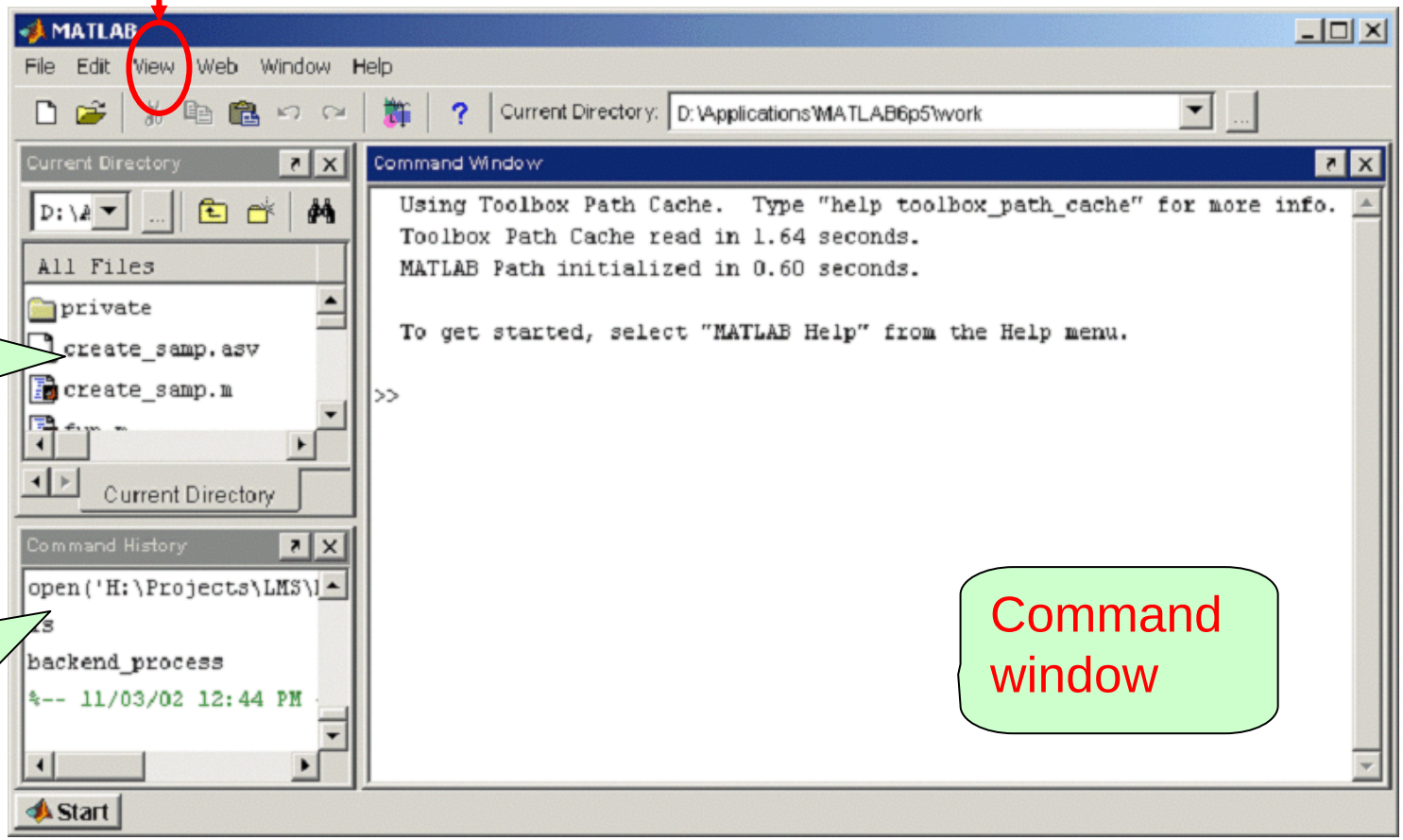
From Start Menu

- Select Programs
- Select MATLAB



MATLAB Layout

1 to 5 different windows can be selected to appear (View)

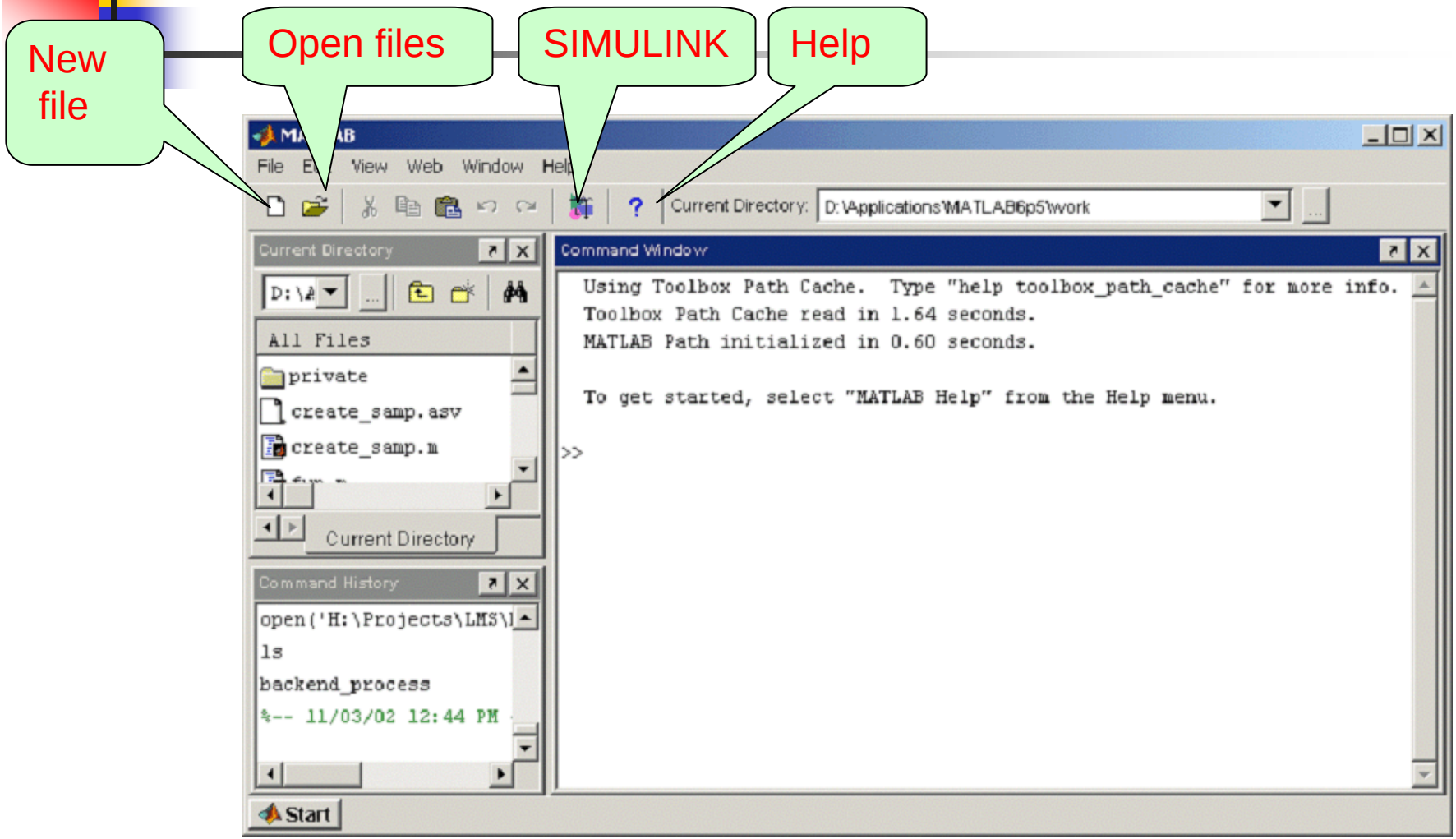


Current directory window

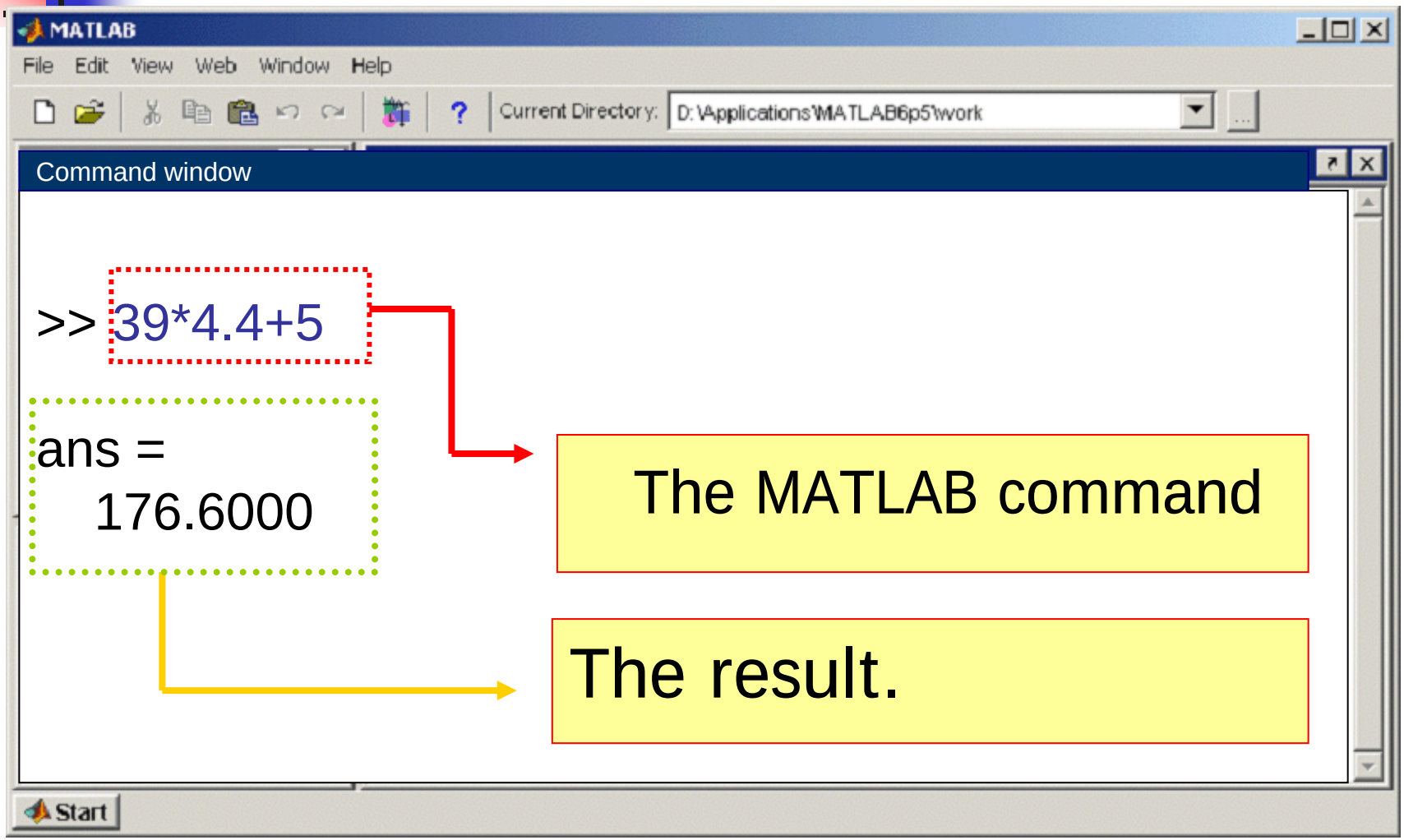
Command History window

Command window

MATLAB shortcuts



MATLAB AS A CALCULATOR

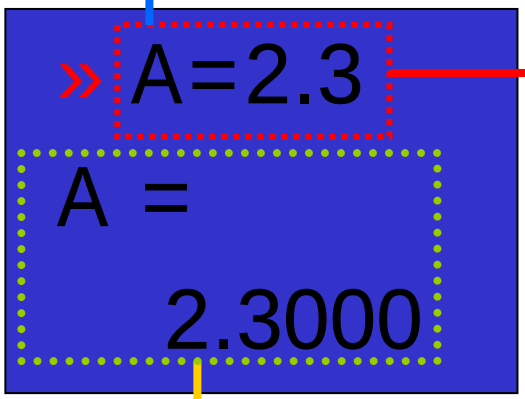


MATLAB

- Variable names:
  - Starts with a letter
  - Up to 31 characters (some use 19 or 21)
  - May contain letters, digits and underscore_
  - Case sensitive (“A” is not the same as “a”)

MATLAB Assignment

- Variable names:
  - Starts with a letter
  - Up to 31 characters (some use 19 or 21)
  - May contain letters, digits and underscore_
  - Case sensitive (“A” is not the same as “a”)



The MATLAB command

This is the result of the MATLAB statement

Scalar Assignment

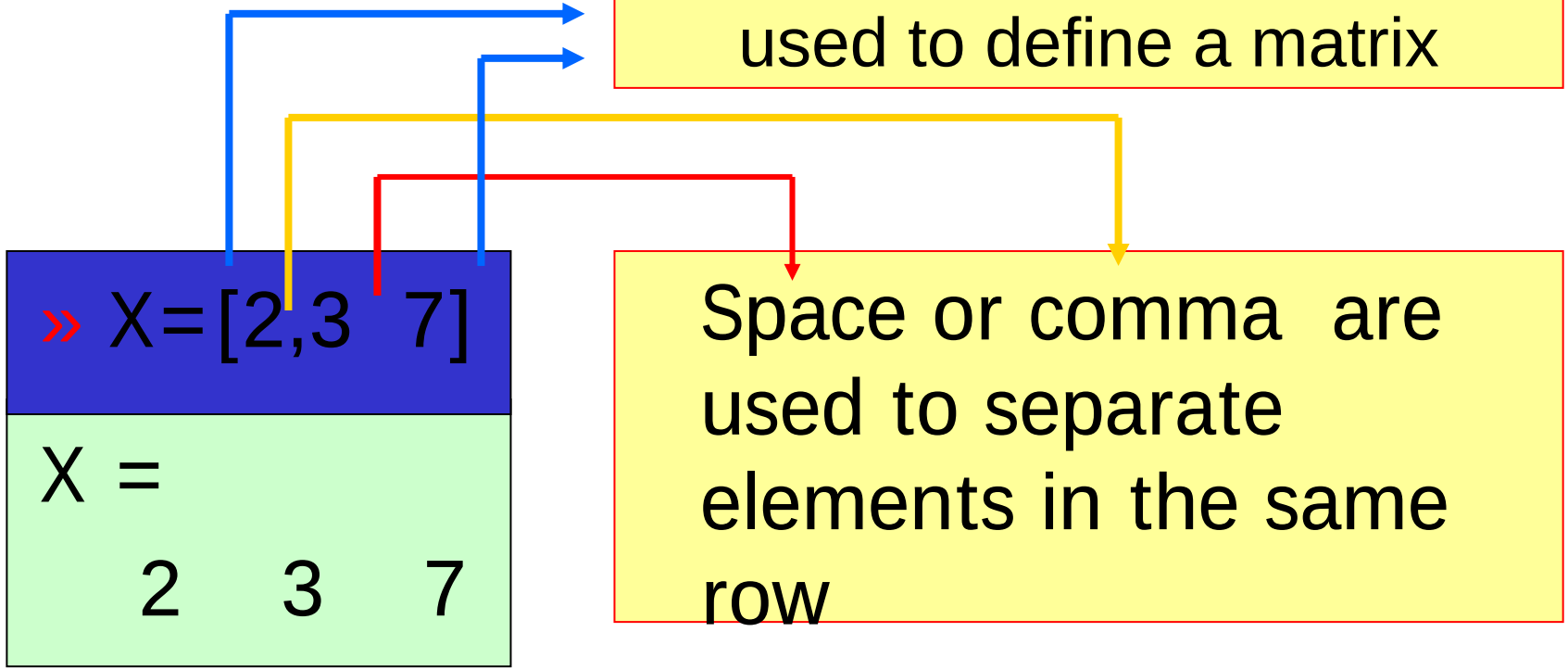
```
» A=2.3
A =
    2.3000
```

this creates a variable “A” and set its value to 2.3

```
» A=[2.3]
A =
    2.3000
```

The square braces [ ] are used to define matrices. We can use them for scalars too.

Row vector


The square braces are used to define a matrix
» X=[2,3 7]
X =
2 3 7
Space or comma are used to separate elements in the same row

Column vector

The square braces are used to define a matrix

```
» X=[2;3 ; 7]

X =
     2
     3
     7
```

semicolon are used to end a row.

You can also use ENTER to end a row

MATLAB Statements

MATLAB Statement	Remarks
C=5.66	C is a scalar
C=[5.66]	An alternative way
X=[3.5 6.3, 33]	X is a 1X3 matrix with elements 3.5 , 6.3 and 33. Commas or space are used to separate the elements in a row
Y=[1 4]	Y is a 2X1 matrix whose elements are 1 and 4.
Y = [1 ; 4]	Semicolon are used to indicate the end of the row.
A=1:5	Equivalent to A=[1 2 3 4 5]

MATLAB Statements

MATLAB Statement	Remarks
V=[2 3 5 3 3 8]	$V=\begin{bmatrix}2 & 3 & 5\\ 3 & 3 & 8\end{bmatrix}$
C=[1:3:11]	C=[1 4 7 10]
Z=4\8 (Z=8/4)	Z=2
X=ones(2,3)	$X=\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$
Y=eye(2)	$Y=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$
W = zeros(2,3)	$W=\begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}$

MATLAB Statements

X=[1 2 5; 3:2:7]	$X = \begin{bmatrix} 1 & 2 & 5 \\ 3 & 5 & 7 \end{bmatrix}$
Y=X.’	$Y = \begin{bmatrix} 1 & 3 \\ 2 & 5 \\ 5 & 7 \end{bmatrix}$
Z=X(2,3)	Z=7
Z=X(1,:)	Z=[1 2 5]
Z=X(:,2)	$Z = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$
Z=X(:,2:3)	$Z = \begin{bmatrix} 2 & 5 \\ 2 & 7 \end{bmatrix}$

MATLAB Statements

Try this:

>>x=randperm(6)	
>>find(x>3)	
>>sort(x)	
>>mean(x)	
>>x.*[1:6]	
>>size(x)	

Operators (arithmetic)

+	addition
-	subtraction
*	multiplication
/	division
^	power
.*	element-by-element mult
./	element-by-element div
.^	element-by-element power
.‘	transpose

```
>>X=[1 0
      0 1];
>>Y=[2 5
      1 3];

X+Y=
X-Y=
X*Y=
X/Y=
X.*Y=
```

Help

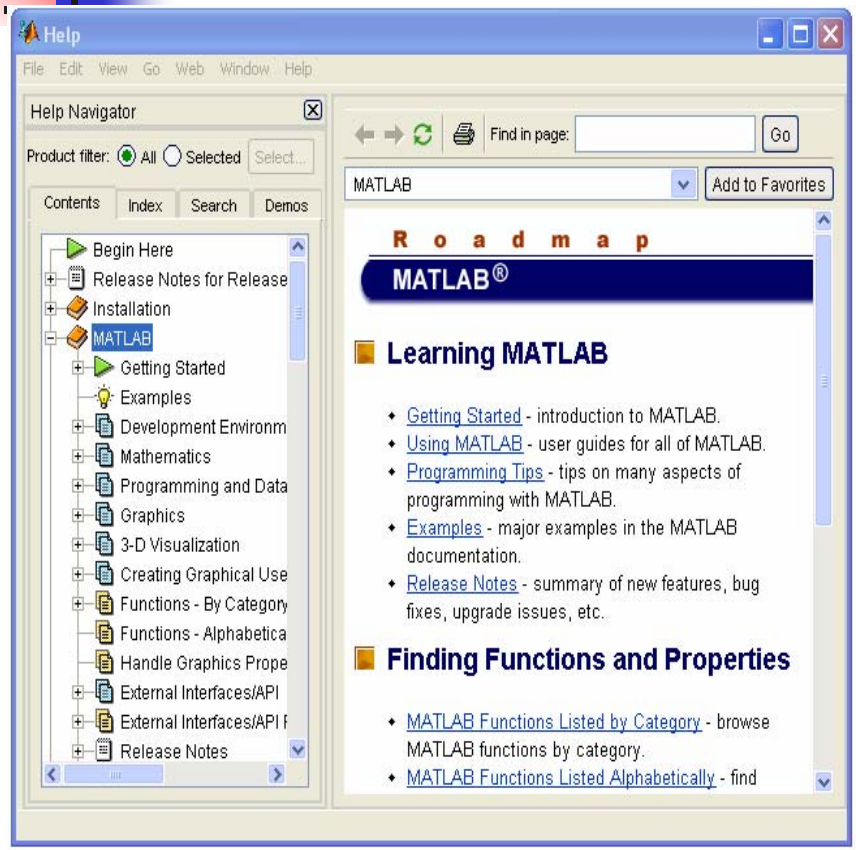
- A good idea is use the help
- help provides information about the available functions and how to use them.
- Try

  help eig

  help inv

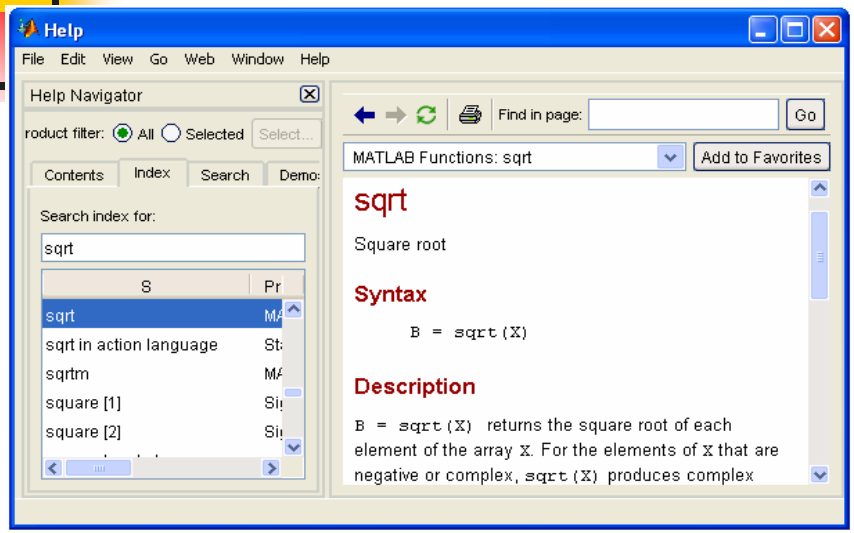
  help roots

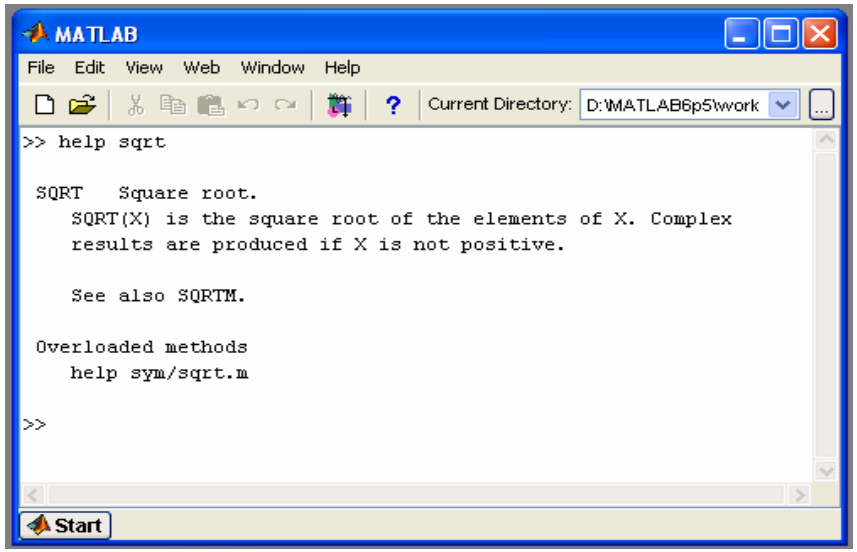
MATLAB Help



- MATLAB Help is an extremely powerful assistance to learning MATLAB
- Help not only contains the theoretical background, but also shows demos for implementation
- MATLAB Help can be opened by using the HELP pull-down menu

MATLAB Help (cont.)





- Any command description can be found by typing the command in the search field
- As shown above, the command to take square root (`sqrt`) is searched
- We can also utilize MATLAB Help from the command window as shown

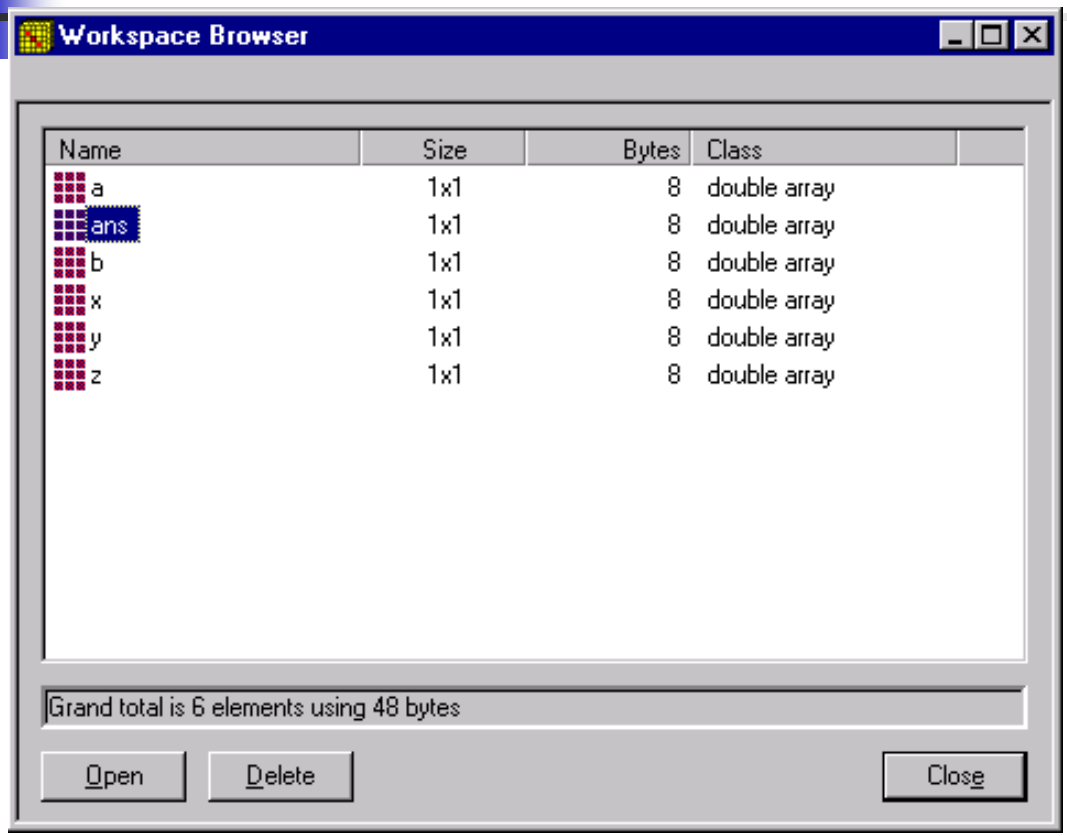
Try these…

- CD / PWD, LS / DIR - navigating directories
- CLEAR - clear workspace variables
- SAVE – save workspace variables to *.mat file
- LOAD – load variables from *.mat file
- WHOS - lists workspace variables and details (size, memory usage, data type)
- WHAT - displays the files within a directory (grouped by type)
- WHICH - identifies the object referenced by given name (function / variable)
- WHY – just for fun

MATLAB Special Variables

ans	Default variable name for results	
pi	Value of π	
eps	Smallest incremental number	
inf	Infinity	
NaN	Not a number	
i & j	basic imaginary unit	
realmax	The largest usable positive real number	
realmin	The smallest usable positive real number	

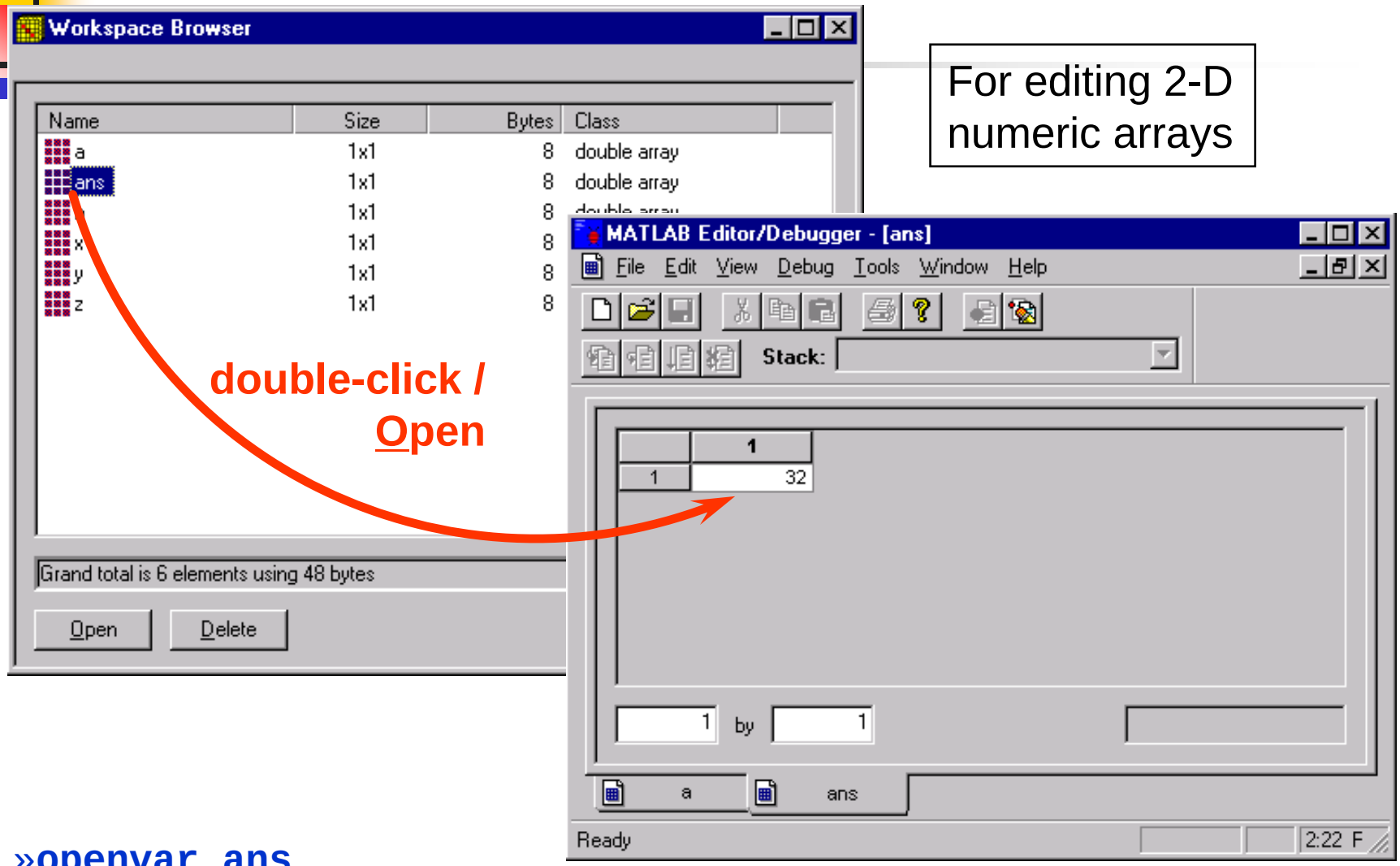
Workspace Browser



Command line variables saved in MATLAB workspace

```
»workspace
```

Array Editor

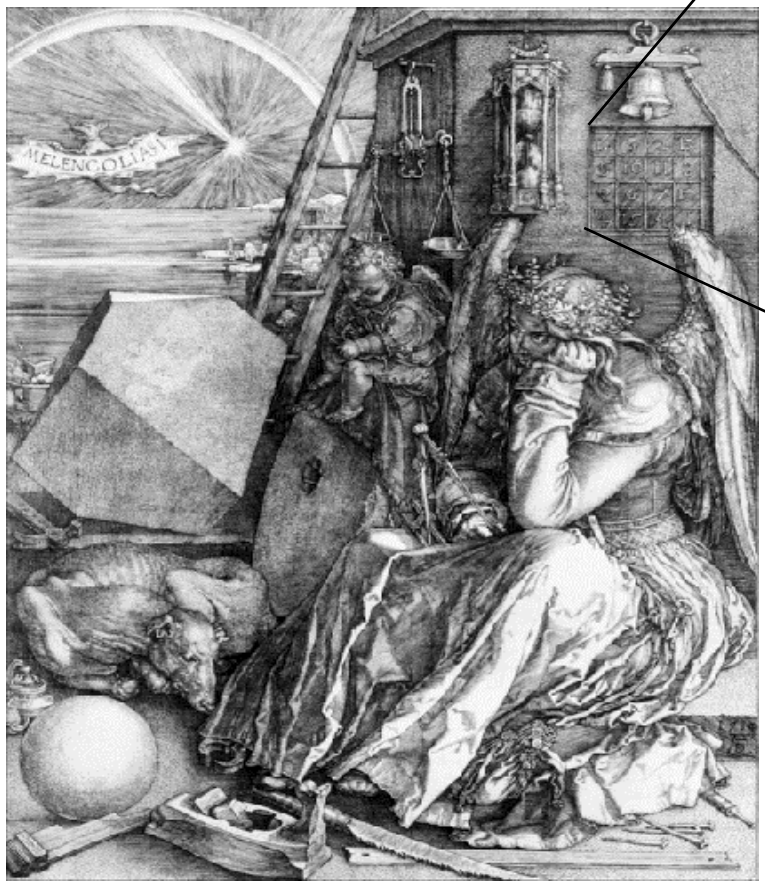


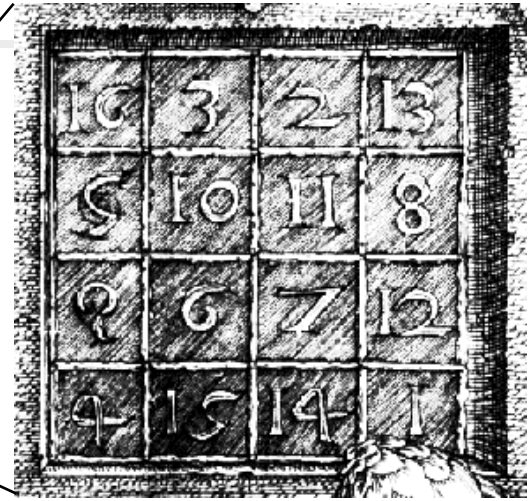
For editing 2-D numeric arrays

»openvar ans

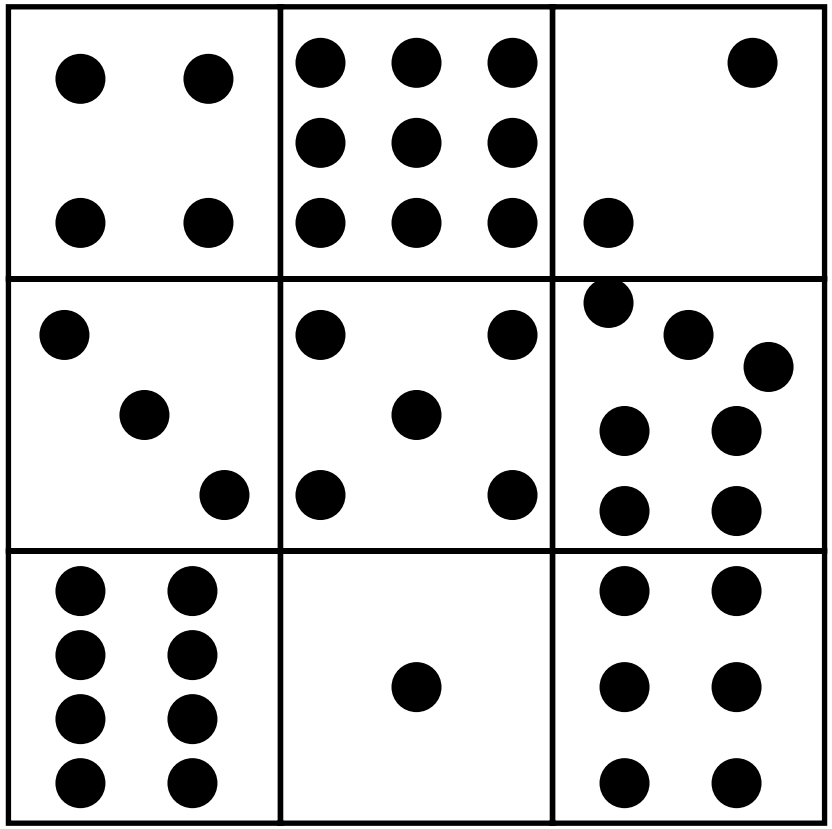
Working with Matrices

```
»load durer
»image(X);
»colormap(map)

»load detail
»image(X);
»colormap(map)

»magic(4)
ans=
  16   3   2  13
   5  10  11   8
   9   6   7  12
   4  15  14   1
```





More about the magic square



```
»magic(3)
ans=
   8   1   6
   3   5   7
   4   9   2
```

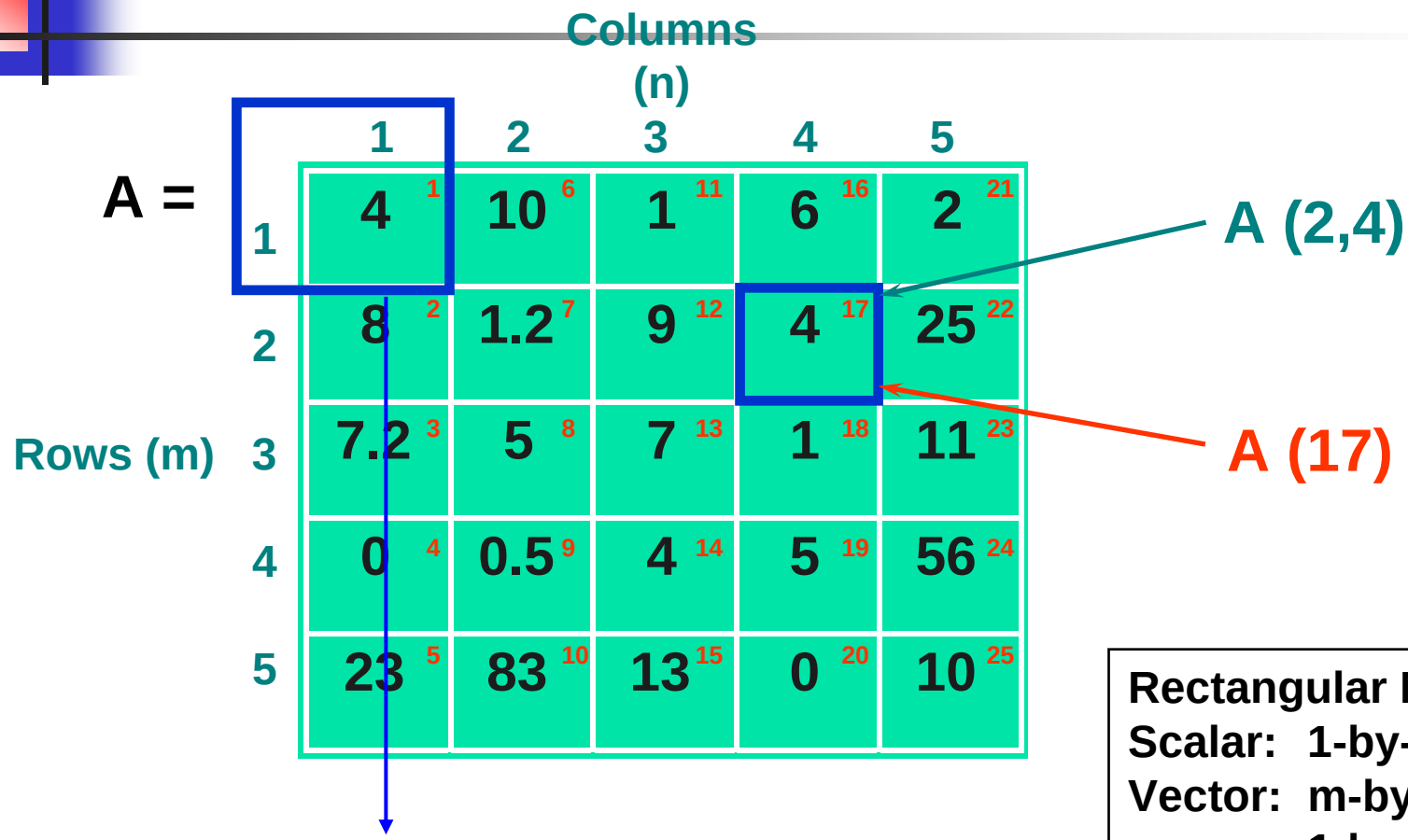
```
>> magic(5)
ans =
  17  24   1   8  15
  23   5   7  14  16
   4   6  13  20  22
  10  12  19  21   3
  11  18  25   2   9
```

More about the magic square

16	05	11	02	16	05	11
03	10	08	13	03	10	08
06	15	01	12	06	15	01
09	04	14	07	09	04	14
16	05	11	02	16	05	11
03	10	08	13	03	10	08
06	15	01	12	06	15	01

Try using matlab programming to find all such structures

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

Matlab index starts from 1, NOT 0

Entering Numeric Arrays

Row separator:
semicolon (;)

Column separator:
space / 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 [ ]

Matrices must be rectangular.
(Set undefined elements to zero)

Any MATLAB expression can be entered as a matrix element

Entering Numeric Arrays - cont.

Scalar expansion

Creating sequences:
colon operator (:)

Utility functions for creating matrices.
(Ref: Utility Commands)

```
» 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
```

Numerical Array Concatenation - [ ]

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

Row separator:
semicolon (;)

Column separator:
space / comma (,)

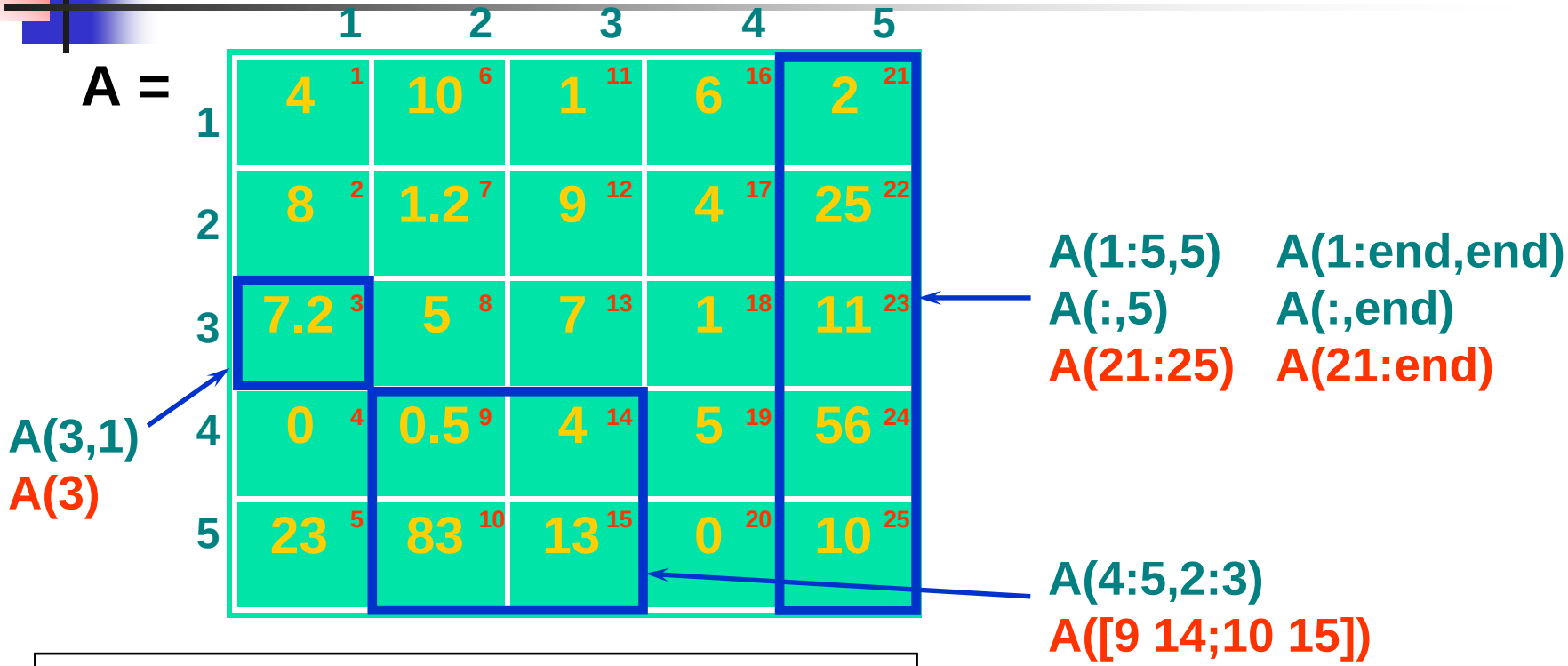
```
» a=[1 2;3 4]
a =
     1     2
     3     4
» cat_a=[a, 2*a; 3*a, 4*a; 5*a, 6*a]
cat_a =
     1     2     2     4
     3     4     6     8
     3     6     4     8
     9    12    12    16
     5    10     6    12
    15    20    18    24
```

Use square brackets [ ]

4*a

The resulting matrix must be rectangular.

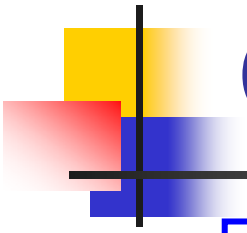
Array Subscripting / Indexing



- Use () parentheses to specify index
- colon operator (:) specifies range / ALL
- [ ] to create matrix of index subscripts
- ‘end’ specifies maximum index value

An Introduction to MATLAB
Lesson 2: M-files

- 关于加机时……
  - 多少同学要加？
  - 加多少机时？
  - 如何办理？

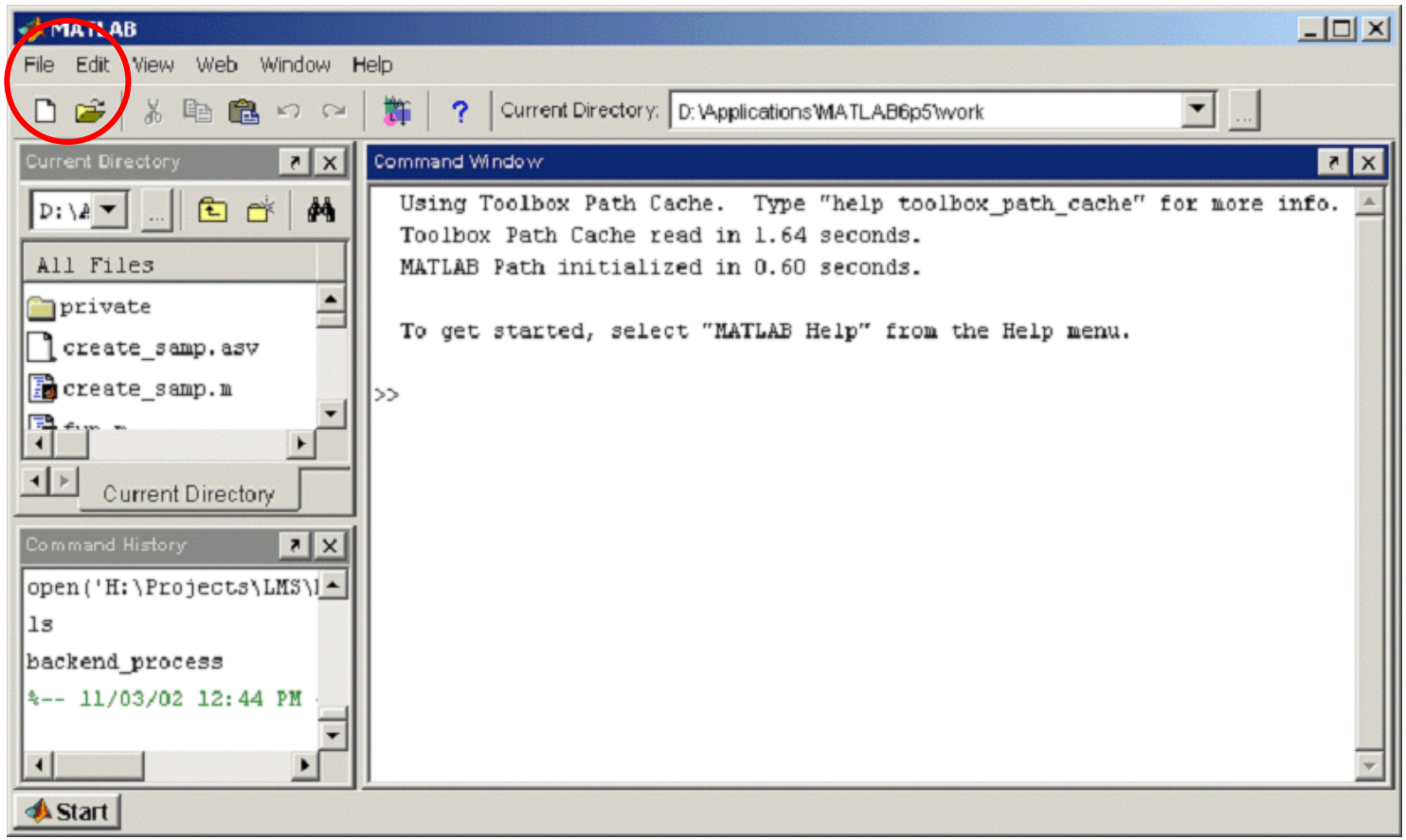


Objectives

- To be able to create MATLAB m-files
- To understands the basics of MATLAB files
- Basic graphics

Creating M-files

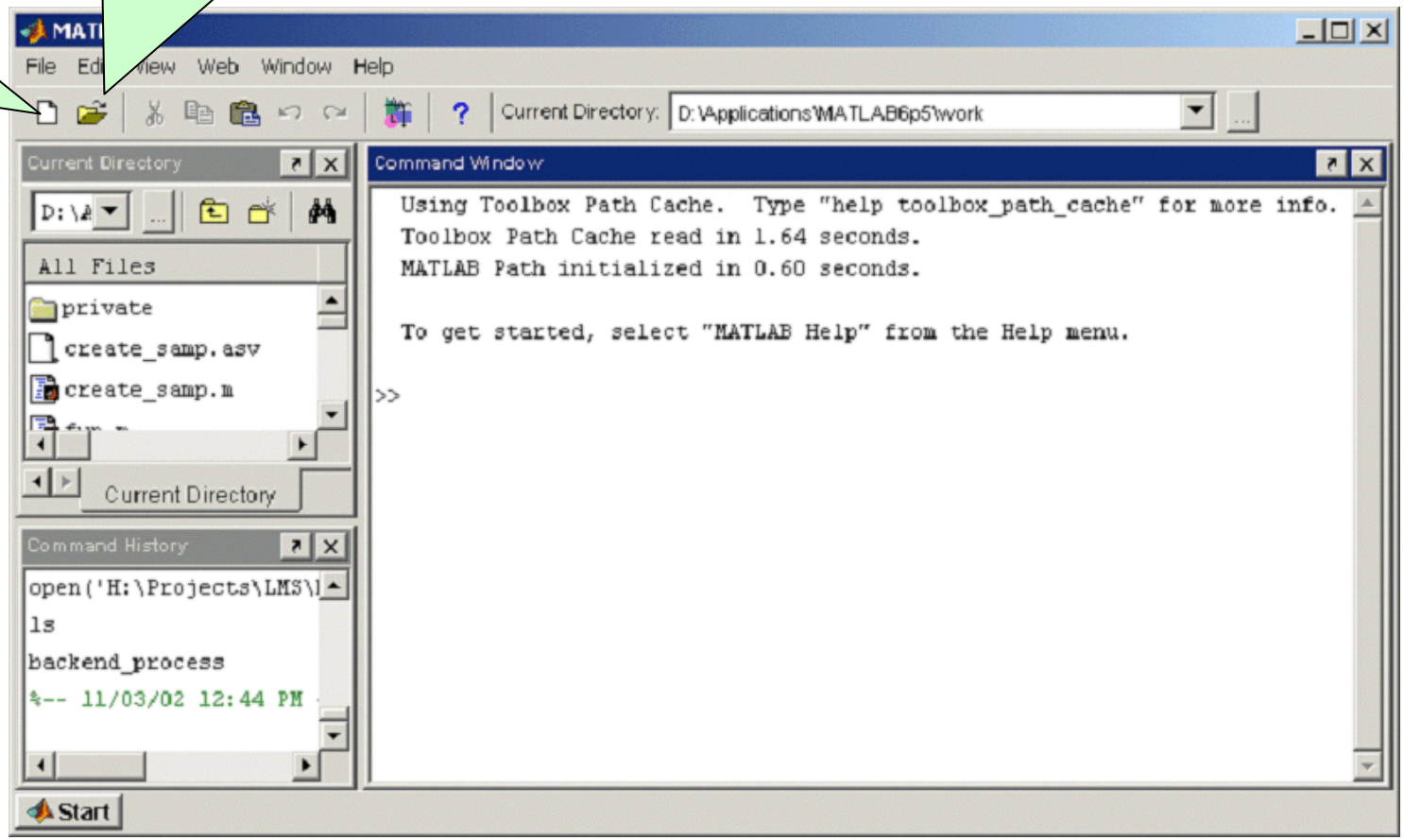
Select FILE → OPEN/NEW → M-files



MATLAB shortcuts

Create a New file

Open an existing files



Script verses functions files

Script files

- List of MATLAB statements
- Variables are global
- Run it by typing the file name

Function files

- Starts with function
- List of MATLAB statements
- Variables are local

Programming in MATLAB

- There are two types of MATLAB programs

script files

```
%  script file
P=[1 3 2];
roots(P);
```

function files

```
%  function  file
function [y]=fun(x)
y=x^2+3*x^2+2;
```

Programming in MATLAB

Script files

- A Script file contains a set of MATLAB command
- Use script file when you have a long sequence of statements to solve a problem
- Run the program by
  - typing its name in the command window
  - from tools in the editor window

Logical Operators

>	Greater than
>=	Greater than or equal
<	Less than
<=	Less than or equal
==	equal
~=	Not equal

Logical Operators

&	AND
|	OR
~	NOT

if (X>6) |(x<3)

If structures

General form:

```
If  condition
    statements
else
    statements
end
```

```
If (x>0)
    sign=1
elseif (x==0)
    sign=0
else
    sign=-1
end
```

for loops

General form:

```
for index=initial: increment:
    limit
    statements
end
```

```
s=0
for i=1:3:11
    s=s+i
end
```

Example

MATLAB program to find the roots of

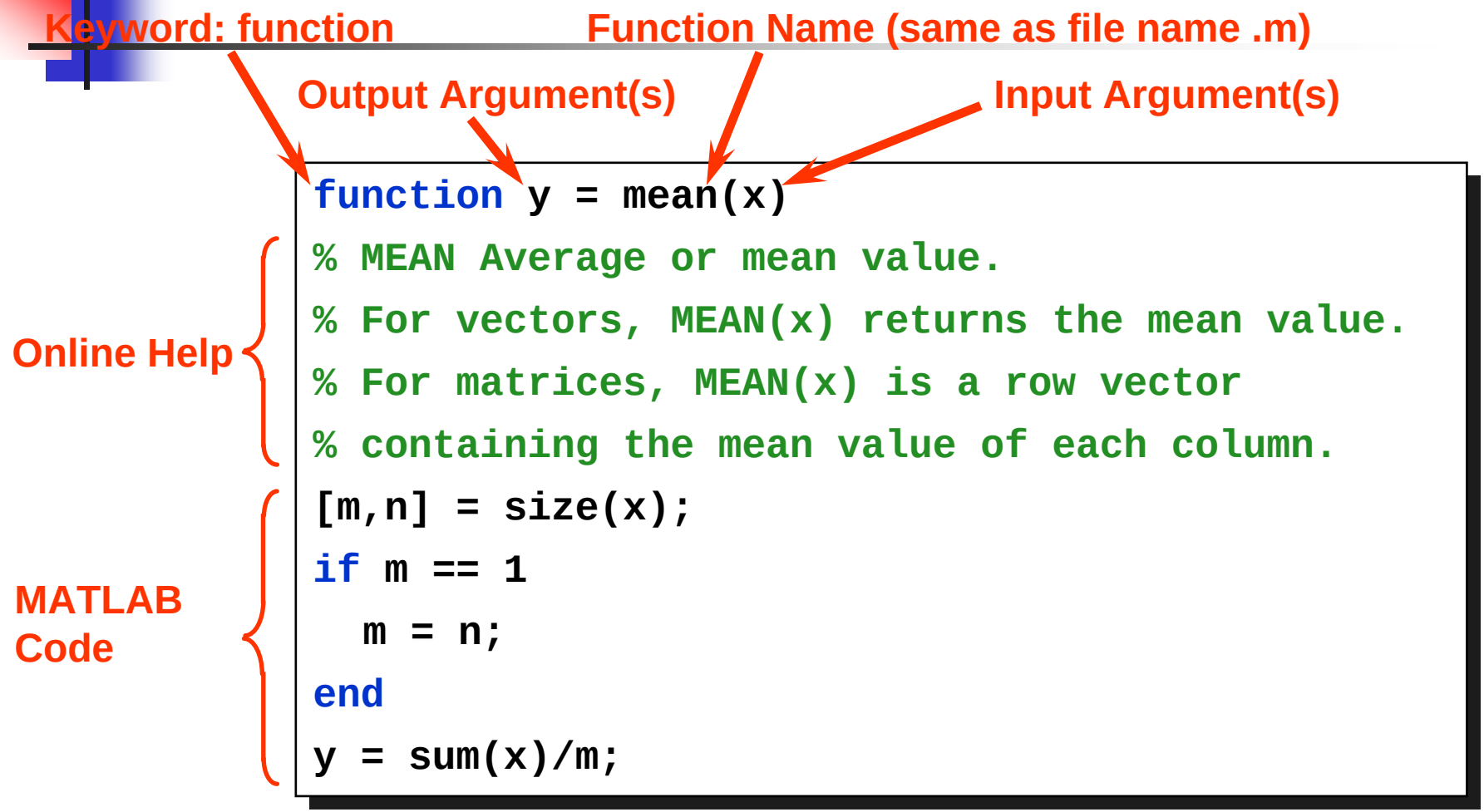
$$f(x) = 2\cos(x) - 1$$

```
% program 1 performs four iterations of
% Newton's Method
X=.7
for i=1:4
X=X – (2*cos(X)-1)/(-2*sin(X))
end
```

Result

```
X =
    1.1111
X =
    1.0483
X =
    1.0472
X =
    1.0472
```

Structure of a Function M-file



Multiple Input & Output Arguments

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

Multiple Input Arguments (,)

Multiple Output Arguments [,]

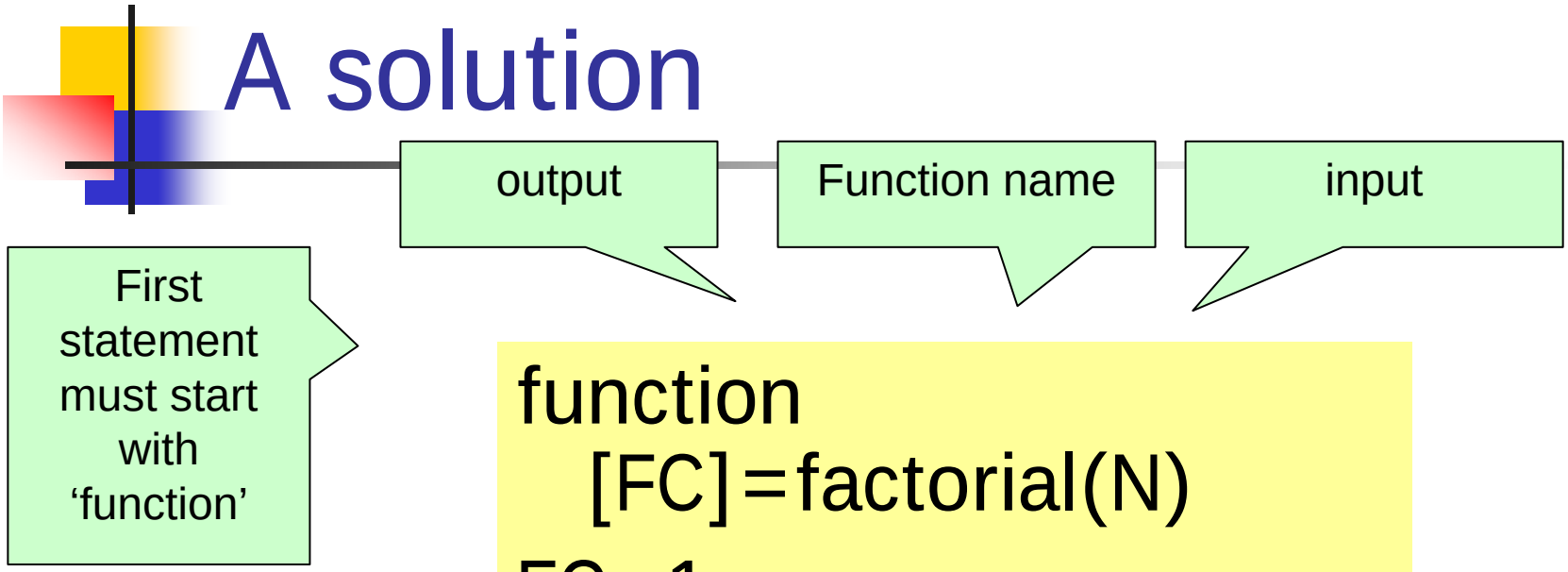
```
function [mean,stdev] = ourstat(x)
% OURSTAT Mean & std. deviation
[m,n] = size(x);
if m == 1
   m = n;
end
mean = sum(x)/m;
stdev = sqrt(sum(x.^2)/m – mean.^2);
```

```
»RANK = ourrank(rand(5),0.1);
»[MEAN,STDEV] = ourstat(1:99);
```

Example 1

- Write a function file to compute the factorial of a number.
- Input: N
- Output :NF
- Function name: factorial

A solution




```
function
   [FC]=factorial(N)
FC=1;
     for  i=1:N
        FC=FC*i;
     end
```

Save the program using 'factorial' as a name

Creating function file

Open an m-file and start typing the file

```
function   [FC]=factorial(N)
FC=1;
    for  i=1:N
       FC=FC*i;
    end
```

- Save the program using ‘factorial’ as a name
- If NOTEPAD is used to create the file use the name ‘factorial.m’
- Save it in directory recognized by MATLAB
- If the directory is not recognized by MATLAB add it to the MATLAB path

A Better one

These comments will be displayed when

'help factorial'

is typed

```
function  [FC]=factorial(N)
% [FC]=factorial(N)
% program to calculate the factorial of a number
% input  N : an integer
% if N is not an integer the program obtains the
% factorial of the integer part of N
% output FC : the factorial of N

%
FC=1;                    % initial value of FC
   for  i=1:N
      FC=FC*i;           % n! =(n-1)!*n
   end
```

Comments are used to explain MATLAB statements

Script file to compute factorial

```
% program to calculate the factorial of a number
% input  N : an integer
% if N is not an integer the program obtains the
% factorial of the integer part of N
% output FC : the factorial of N
%
FC=1;                          % initial value of FC
    for  i=1:N
       FC=FC*i;                % n! =(n-1)!*n
    end
```

Comments are used to explain MATLAB statements

Script file to compute cos

```
% program to calculate an estimate of
   cos(0.2)
% cos(x) ≈1-x^2/2!+x^4/4!
x=0.2
Sum=1
N=2
fact2
Sum=Sum-x^2/FC
N=4
fact2
Sum=Sum+x^4/FC
```

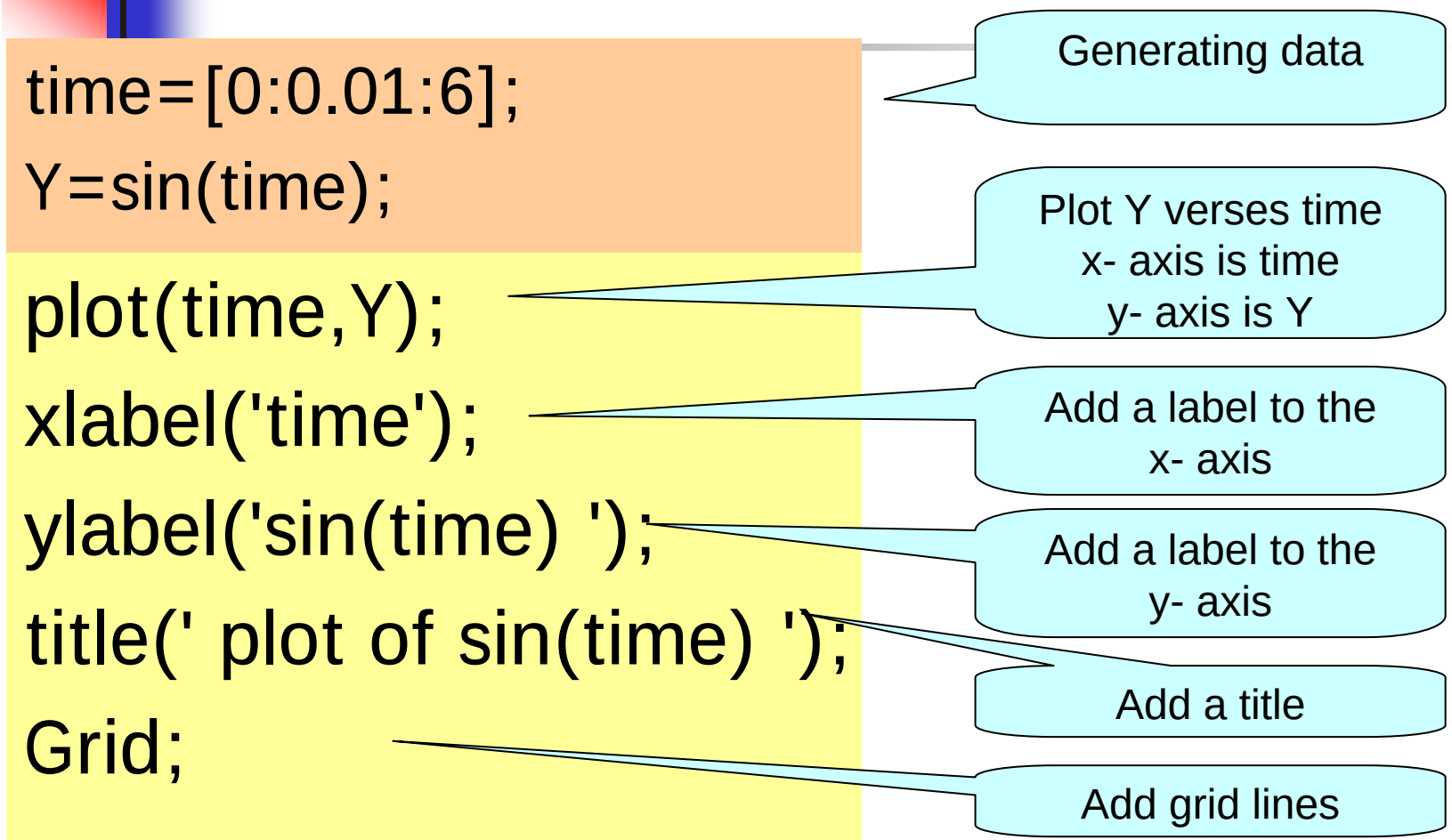
```
% Script file fact2
FC=1;
for  i=1:N
      FC=FC*i;
end
```

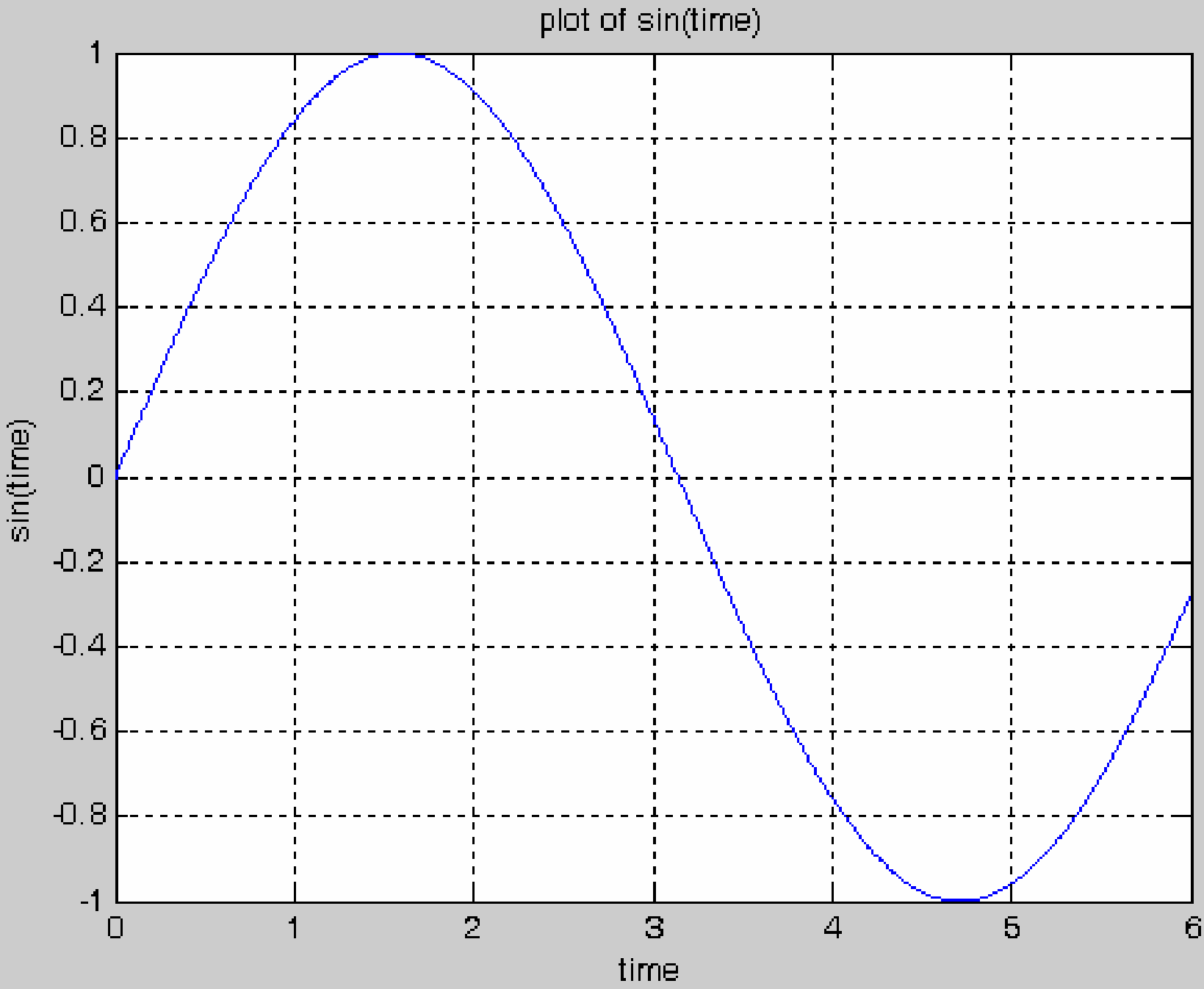
Graphics on MATLAB

- Simple 1D graphics
  - Linear scales
  - Semilog scale
  - Loglog scale
- 2D graphics

Example

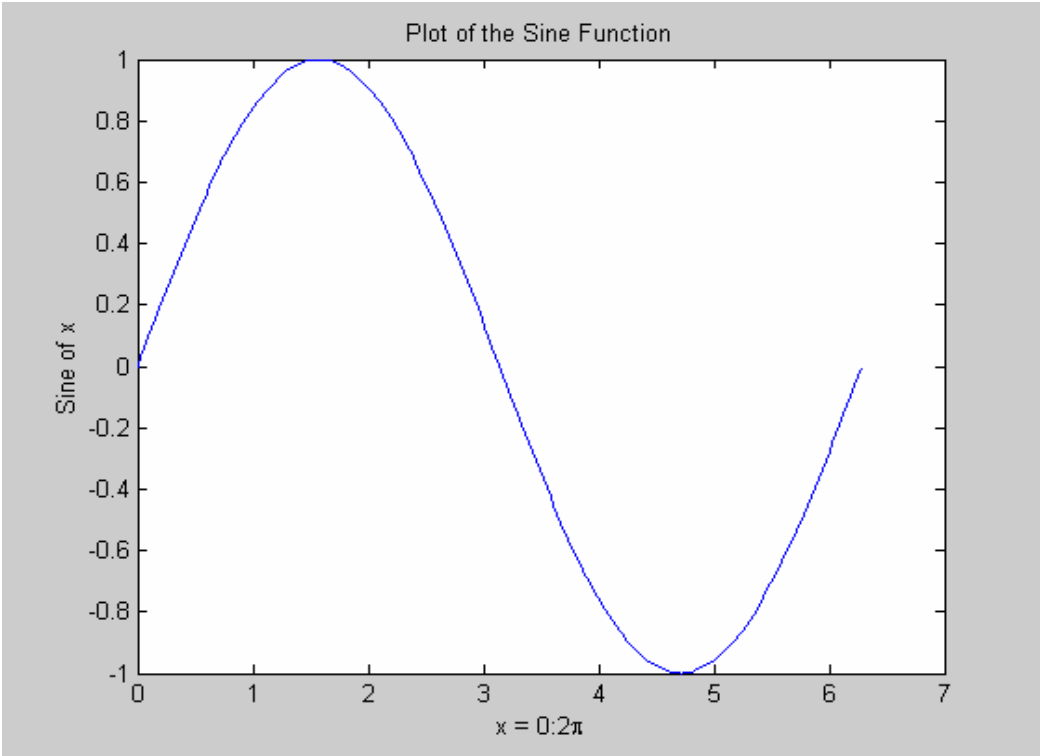
```
time=[0:0.01:6];
Y=sin(time);
plot(time,Y);
xlabel('time');
ylabel('sin(time) ');
title(' plot of sin(time) ');
Grid;
```




plot of sin(time)
sin(time)
time
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6

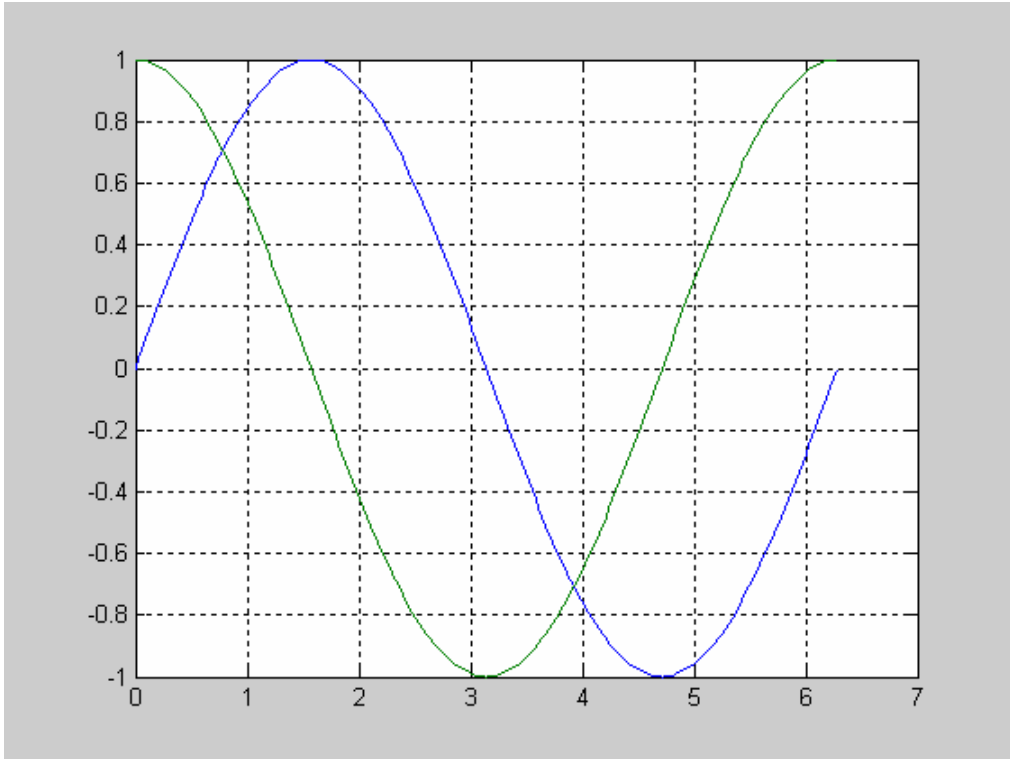
Matlab Graphics

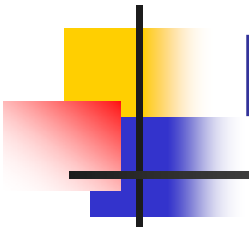
```
x = 0:pi/100:2*pi;
y = sin(x);
plot(x,y);
xlabel(‘x = 0:2\pi’);
ylabel('Sine of x');
title('Plot of the Sine Function');
```



Multiple Graphs

```
t = 0:pi/100:2*pi;
y1=sin(t);
y2=sin(t+pi/2);
plot(t,y1,t,y2);
grid on;
```

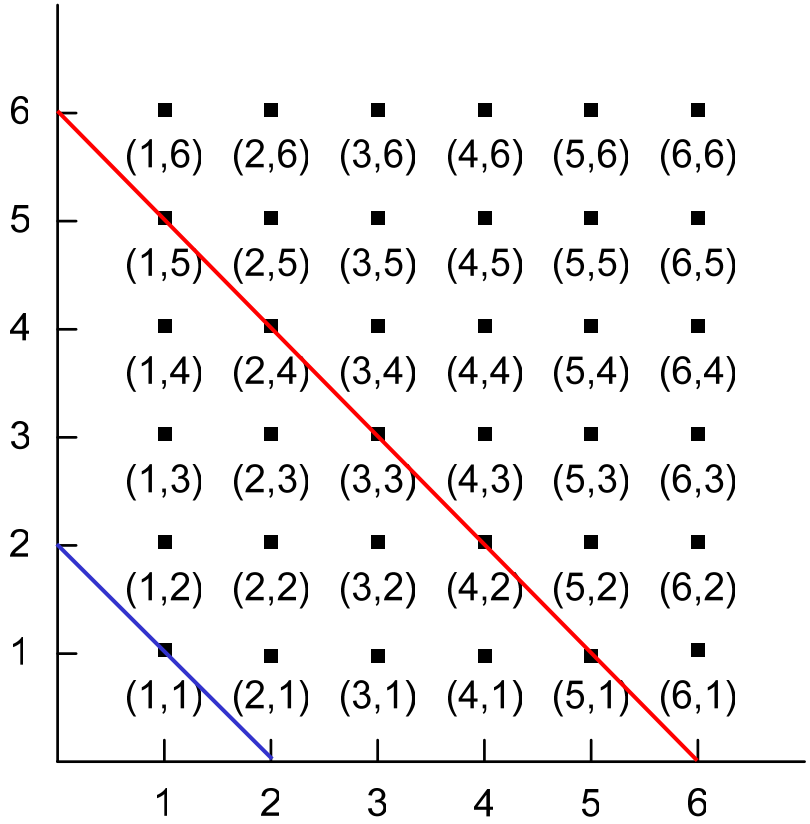




More

Sum of Two Dices

- Simulate 10000 observations of the sum of two fair dices



Sum of Two Dices

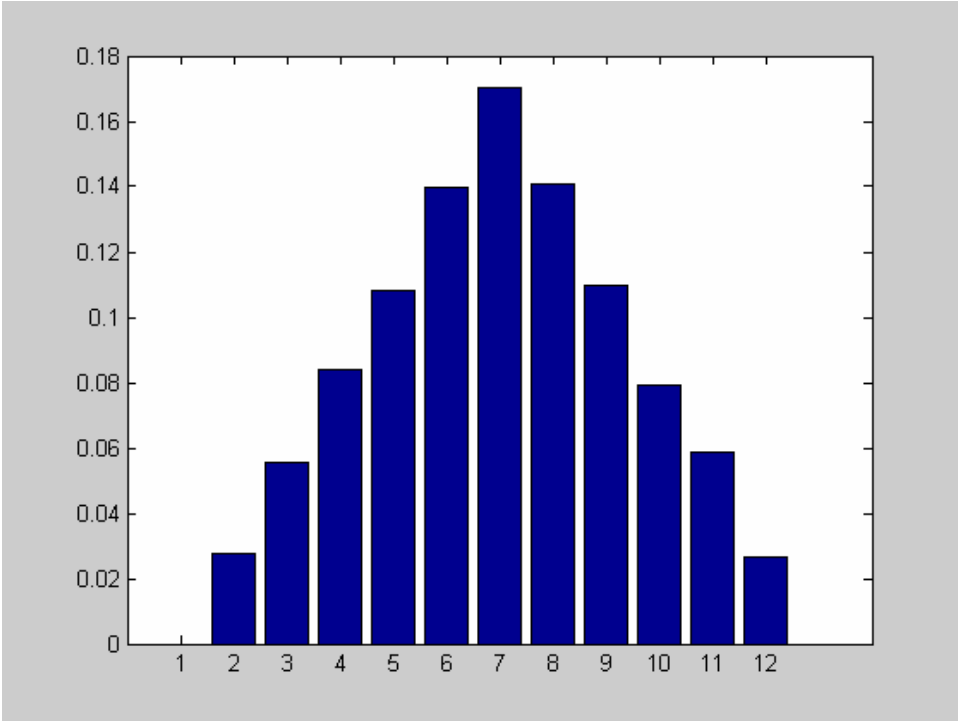
- Simulate 10000 observations of the sum of two fair dices

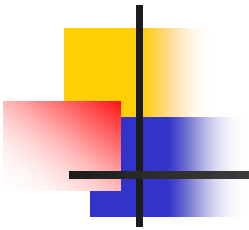
```
x1=floor(6*rand(10000,1)+1);
x2=floor(6*rand(10000,1)+1);
y=x1+x2;
```

sum(y== 2)/10000	ans = 0.0275	p[2]=0.0278
sum(y== 3)/10000	ans = 0.0554	p[3]=0.0556
sum(y== 4)/10000	ans = 0.0841	p[4]=0.0833
sum(y== 5)/10000	ans = 0.1082	p[5]=0.1111
sum(y== 6)/10000	ans = 0.1397	p[6]=0.1389
sum(y== 7)/10000	ans = 0.1705	p[7]=0.1667
sum(y== 8)/10000	ans = 0.1407	p[8]=0.1389
sum(y== 9)/10000	ans = 0.1095	p[9]=0.1111
sum(y==10)/10000	ans = 0.0794	p[10]=0.0833
sum(y==11)/10000	ans = 0.0585	p[11]=0.0556
sum(y==12)/10000	ans = 0.0265	p[12]=0.0278

Sum of Two Dices

```
for i=2:12
    z(i)=sum(y==i)/10000;
end
bar(z)
```





Introduction to MATLAB and image processing

MATLAB and images

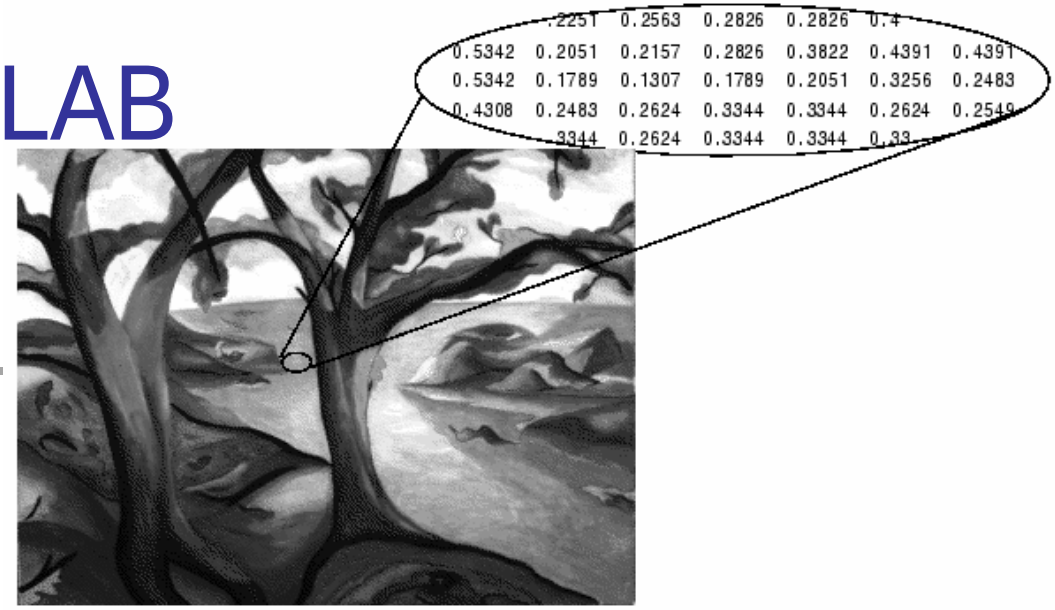
- The help in MATLAB is very good, use it!
- An image in MATLAB is treated as a matrix
- Every pixel is a matrix element
- All the operators in MATLAB defined on matrices can be used on images: +, -, *, /, ^, sqrt, sin, cos etc.

Images in MATLAB

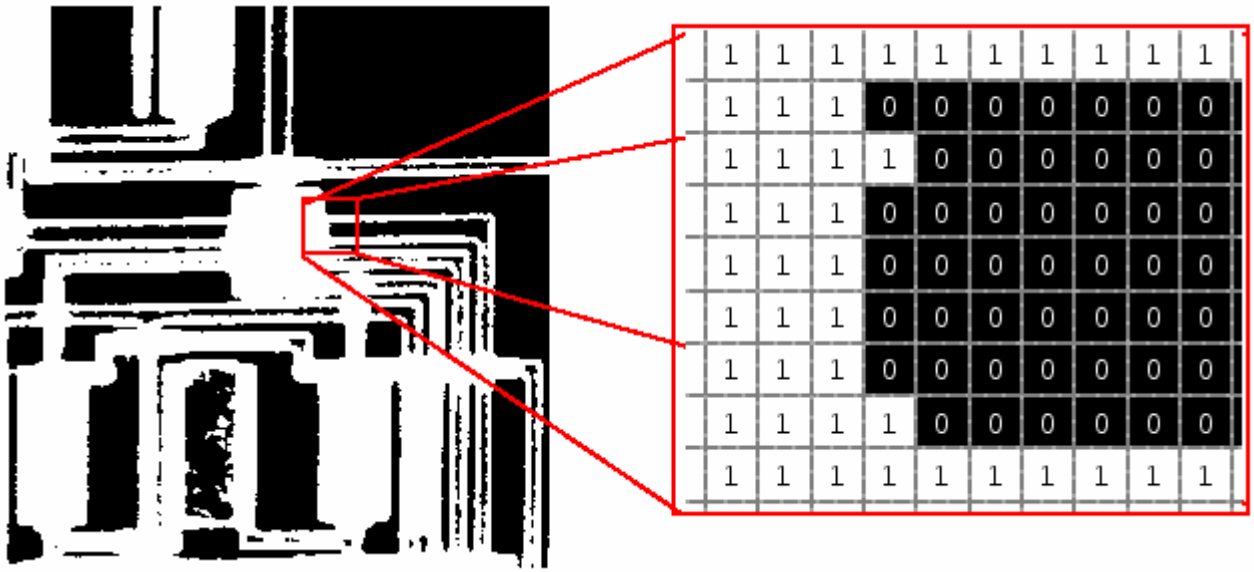
- MATLAB can import/export several image formats
  - BMP (Microsoft Windows Bitmap)
  - GIF (Graphics Interchange Files)
  - HDF (Hierarchical Data Format)
  - JPEG (Joint Photographic Experts Group)
  - PCX (Paintbrush)
  - PNG (Portable Network Graphics)
  - TIFF (Tagged Image File Format)
  - XWD (X Window Dump)
  - MATLAB can also load raw-data or other types of image data

- Data types in MATLAB
  - Double (64-bit double-precision floating point)
  - Single (32-bit single-precision floating point)
  - Int32 (32-bit signed integer)
  - Int16 (16-bit signed integer)
  - Int8 (8-bit signed integer)
  - Uint32 (32-bit unsigned integer)
  - Uint16 (16-bit unsigned integer)
  - Uint8 (8-bit unsigned integer)

Images in MATLAB



- Binary images : {0,1}
- Intensity images : uint8, double, etc.
- RGB images : m-by-n-by-3
- Indexed images : m-by-3 color map



Images in MATLAB

- Binary images : {0,1}
- Intensity images : uint8, double etc.
- RGB images : m-by-n-by-3
- Indexed images : m-by-3 color map
- Multidimensional images m-by-n-by-p (p is the number of layers)

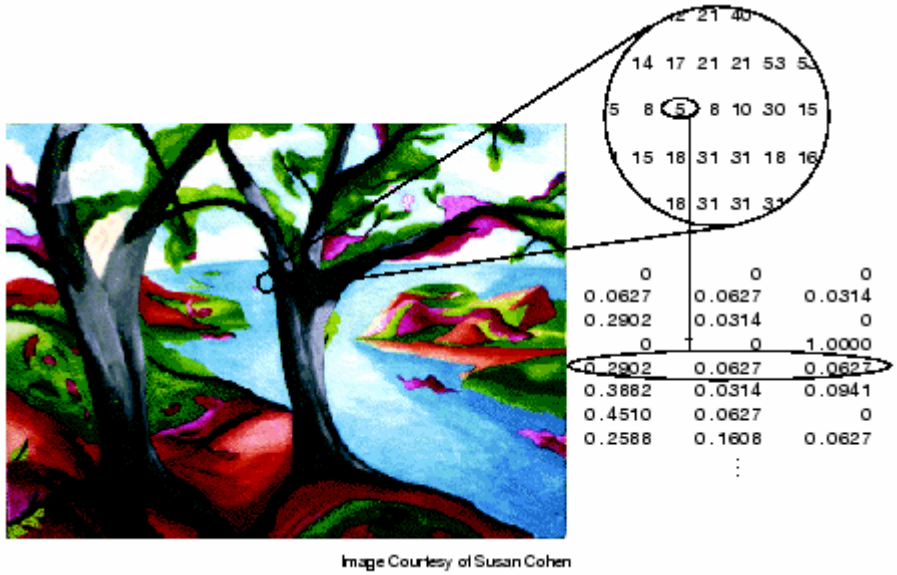


Image Courtesy of Susan Cohen

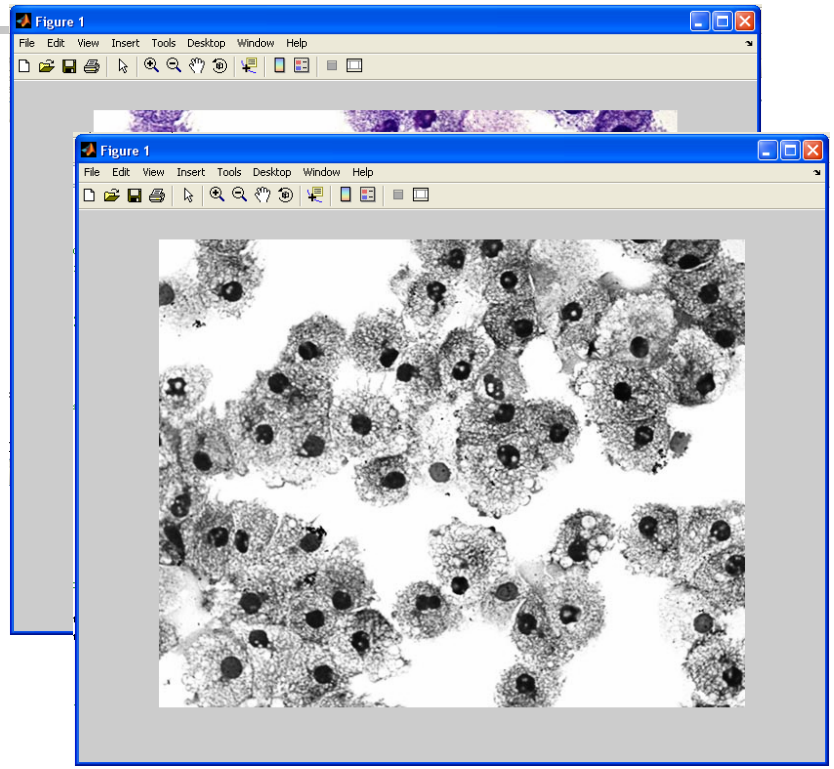
Image import and export

- Read and write images in Matlab

```
>> I=imread('cells.jpg');
>> imshow(I)
>> size(I)
ans =   479   600     3      (RGB image)
>> Igrey=rgb2gray(I);
>> imshow(Igrey)
>> imwrite(Igrey, 'cell_gray.tif', 'tiff')
```

Alternatives to imshow

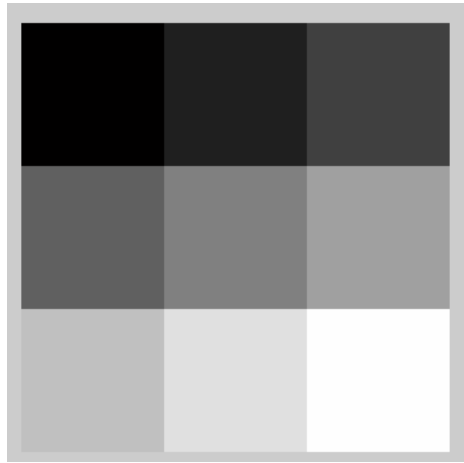
```
>>imagesc(I)
>>imtool(I)
>>image(I)
```



Images and Matrices

- How to build a matrix (or image)?

```
>> A = [ 1 2 3; 4 5 6; 7 8 9 ];
 A =  1   2   3
      4   5   6
      7   8   9
>> B = zeros(3,3)
 B =  0   0   0
      0   0   0
      0   0   0
>> C = ones(3,3)
C =   1   1   1
      1   1   1
      1   1   1
```

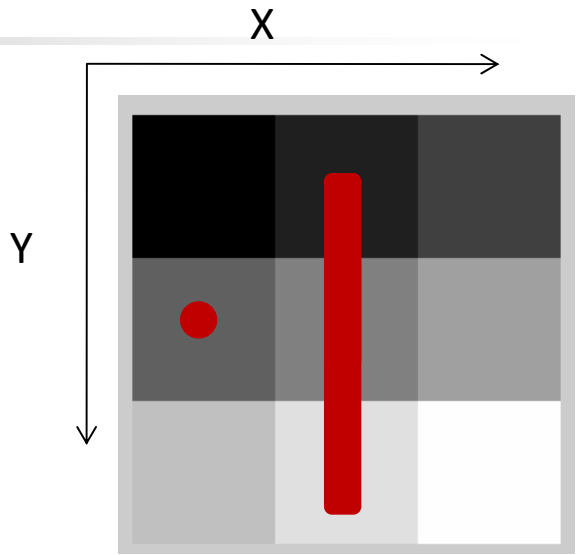


```
>>imshow(A)
```

(imshow(A,[]) to get automatic pixel range)

Images and Matrices

- Accesing image elements (row, column)
  >> A(2,1)
  ans = 4
- : can be used to extract a whole column or row
  >> A(:,2)
  ans =
  2
  5
  8
- or a part of a column or row
  >> A(1:2,2)
  ans =
  2
  5



A =

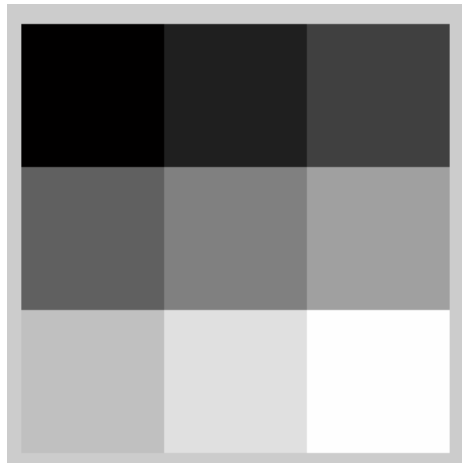
1	2	3
4	5	6
7	8	9

Image Arithmetic

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

- Arithmetic operations such as addition, subtraction, multiplication and division can be applied to images in MATLAB
  - +, -, *, / performs matrix operations

```
>> A+A
ans =     2     4     6
          8    10    12
         14    16    18
>> A*A
ans =    30    36    42
         66    81    96
        102   126   150
```



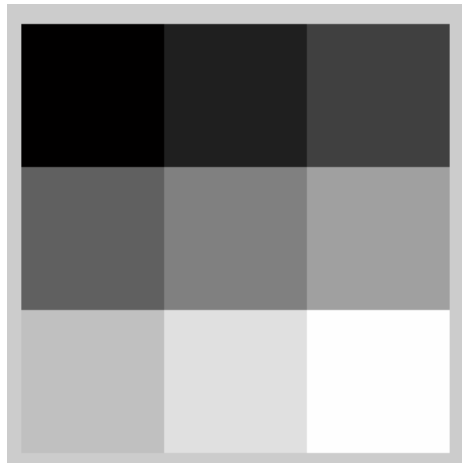
- To perform an elementwise operation use . (.*, ./, .*, .^ etc)

```
>> A.*A
ans =     1     4     9
         16    25    36
         49    64    81
```

Logical Conditions

- equal (==) , less than and greater than (< and >), not equal (~=) and not (~)
- find('condition') - Returns indexes of A's elements that satisfies the condition.

```
>> [row col]=find(A==7)
row =   3
col =    1
>> [row col]=find(A>7)
row =   3
          3
col =    2
          3
>> Indx=find(A<5)
Indx =  1
          2
          4
          7
```



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

Flow Control

- Flow control in MATLAB

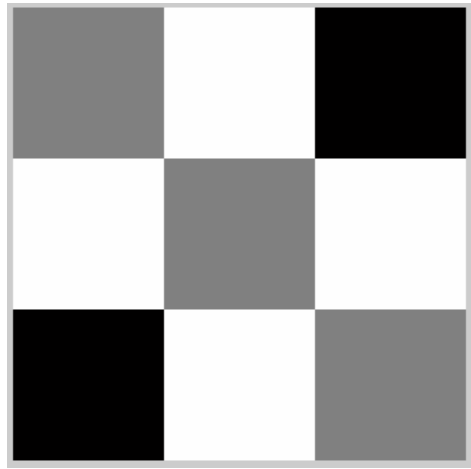
  - if, else and elseif statements

```
(row=1,2,3   col=1,2,3)
if row==col
        A(row, col)=1;
elseif abs(row-col)==1
        A(row, col)=2;
else
        A(row, col)=0;
end
```

Flow Control

- Flow control in MATLAB
  - for loops
    - if, else and elseif statements

```
for row=1:3
        for col=1:3
                if row==col
                        A(row, col)=1;
                elseif abs(row-col)==1
                        A(row, col)=2;
                else
                        A(row, col)=0;
                end
        end
end
```



```
A =

     1     2     0
     2     1     2
     0     2     1
```

Matlab image processing —— a first step

```
clear, close all;                    %clear the workspace, close all active figures
I = imread('pout.tif');              %read the image ‘pout.tif’ into the workspace
imshow(I);                           %display the image
Whos;                                %check the workspace
figure, imhist(I);                   %show the histogram of the image
I2 = histeq(I);                      %histogram equalization
figure, imshow(I2);                  %display the new image I2
figure, imhist(I2);                  % show the equalized histogram
imwrite (I2, 'pout2.png');           %save I2 as ‘pou2.png’
imfinfo('pout2.png');                %display png image information
```

Homework II

- Continue you exploration with Matlab image processing toolbox.
- Write your own code to compute the total area of the white objects in the following picture circles.png.
- Submit your code and result. Compare your result with matlab function bwarea()