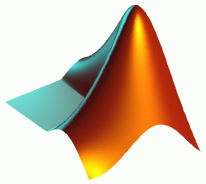


Laboratory of Image Processing

Matrix Manipulation with MatLab

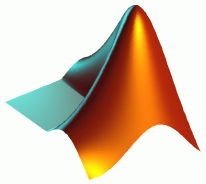
Pier Luigi Mazzeo

pierluigi.mazzeo@cnr.it

Institute of Applied Sciences and Intelligent Systems

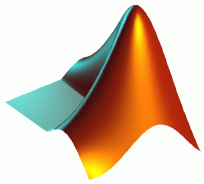
ScienceApp

National Research Council of Italy



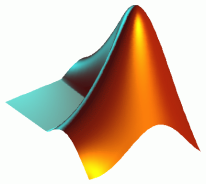
Goals

- Introduce the Notion of Variables & Data Types.
- Master Arrays manipulation
- Learn Arrays Mathematical Operations
- Learn Basic String Manipulations
- Learn to Use Array Editor
- Solving Linear System of Equations



Reference

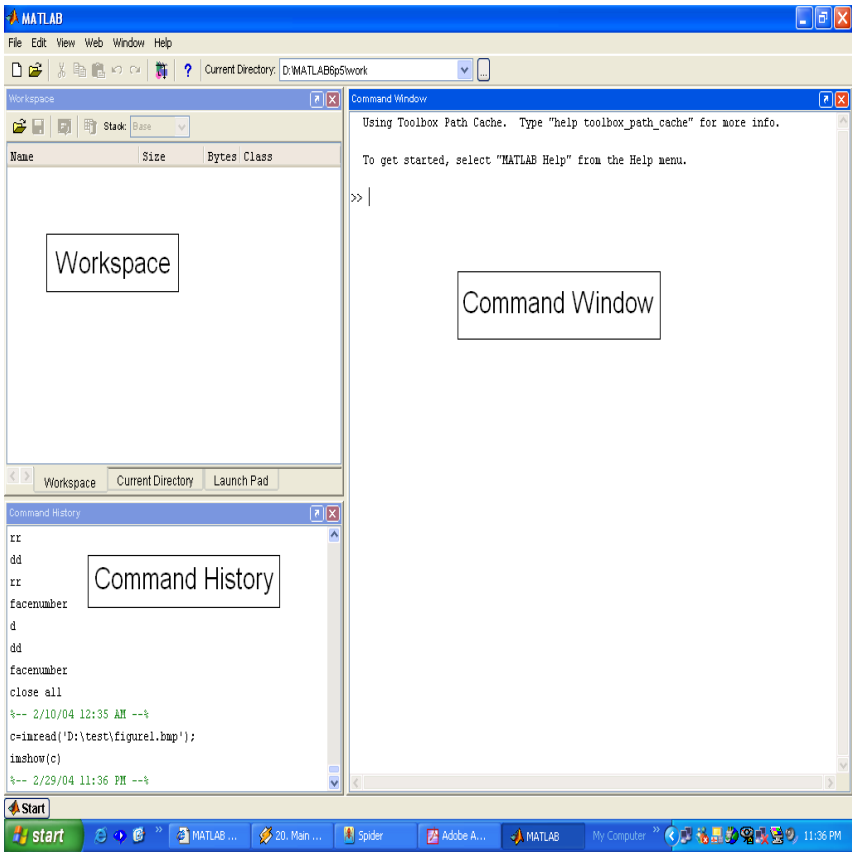
- **An Introduction to Digital Image Processing with MATLAB** by Alasdair MacAndrew. (download from www.ino.it/home/mazzeo/downloads/)
- **Digital Image Processing using MATLAB. 2° Edition** Rafael C. Gonzales, Richard E. Woods, Steven L. Eddins. Gatesmark Publishing.



What is MATLAB?

- MATLAB = Matrix Laboratory
- "MATLAB is a high-level language and interactive environment that enables you to perform computationally intensive tasks faster than with traditional programming languages such as C, C++ and Fortran." (www.mathworks.com)
- MATLAB is an interactive, interpreted language that is designed for fast numerical matrix calculations

The MATLAB Environment



- MATLAB window components:

Workspace

> Displays all the defined variables

Command Window

> To execute commands in the MATLAB environment

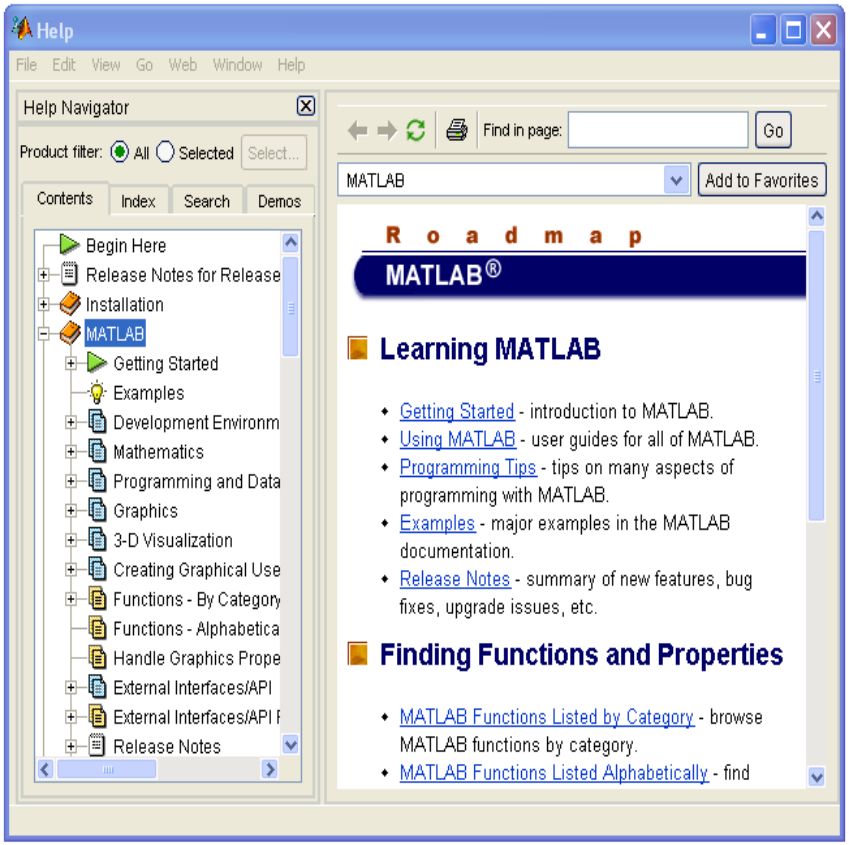
Command History

> Displays record of the commands used

File Editor Window

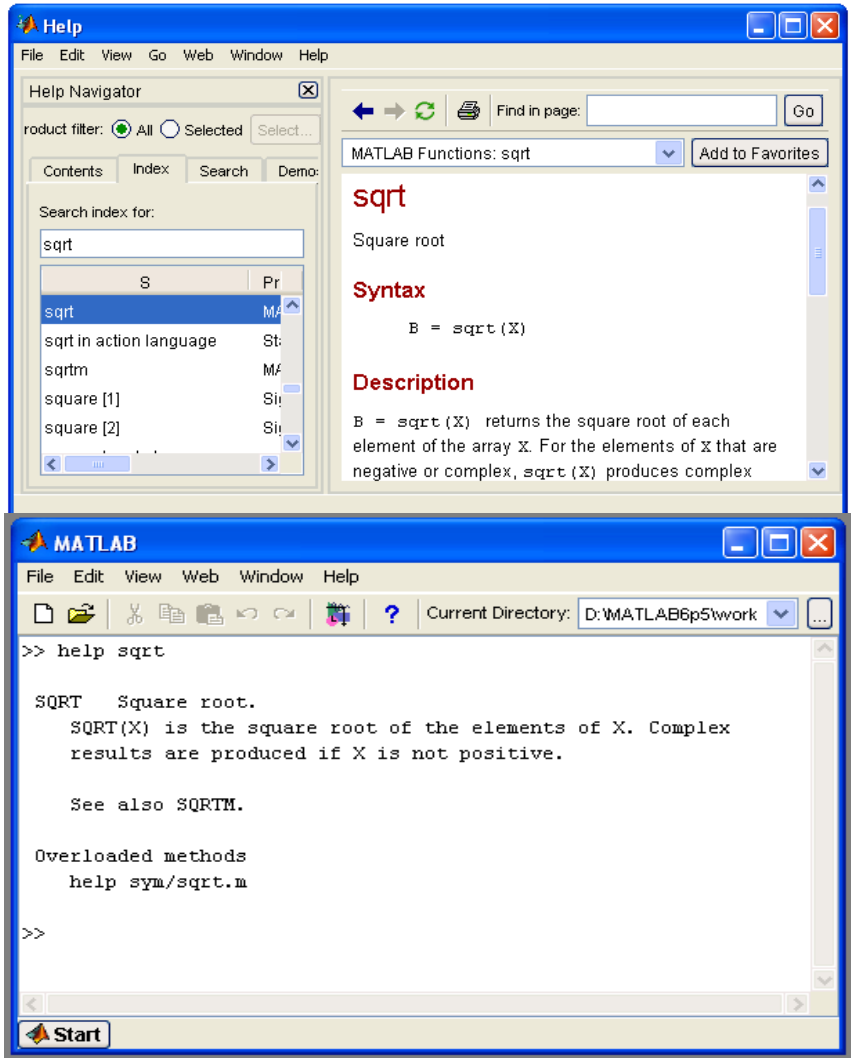
> Define your functions

MATLAB Help

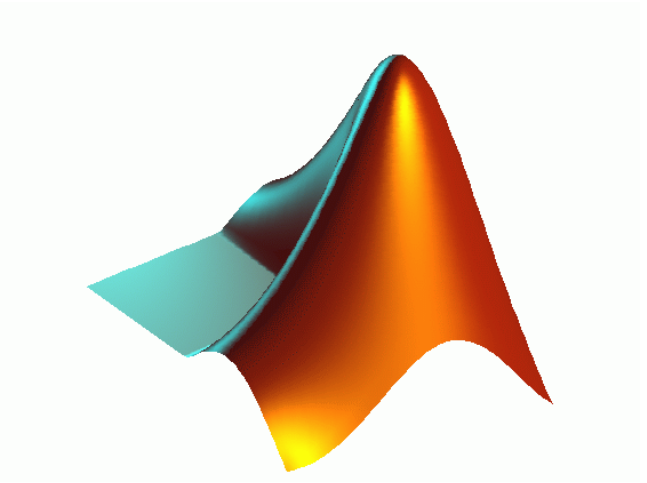


- MATLAB Help is an extremely powerful assistance to learning MATLAB
- Help not only contains the theoretical background, but also shows demos for implementation
- MATLAB Help can be opened by using the HELP pull-down menu

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



Variables

Variables

- A MATLAB variable is an **identified** piece of data (= the **assigned value**)

1	1	0	0	0	1	0	1

- The value is kept in the memory
- The identifier provides a reference to the value such that your program can:
  - Read it
  - Use it for computation / change its value
  - and save it back to memory

Variable Assignment

```
max_grade = 100;
```

Value

Assignment operator

Variable identifier (name):

- **Letters**
- **Digits**
- **Underscore**

1. Can't start with a number

2. Case sensitive!!!

3. Can't be a keyword

```
clear max_grade;
clear;
clear all;
```

Variable - Hands On

- Exercise 1:
  - Consider the following:

```
x = 50;
disp = 100;
disp(x);
builtin('disp', x);
clear disp;
disp(x);
```

  - What happened?

The disp variable overrides the disp built-in function

Don't override built-in functions!!!

Variable - Hands On (II)

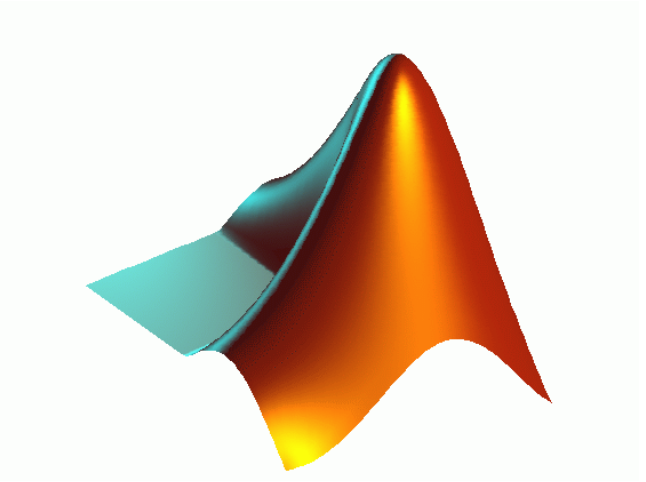
- Exercise 2:
  - Define a variable "grade" and assign the value 80 to it.
  - "Personal Factor" - Now add to the grade its square root (there are several options to solve this…)
  - Define a "classFactor" and assign 10 to it.
  - Define a "finalGrade" variable and assign the value of the factored grade plus the class factor to it.
  - What is the final grade?

Lets look at the workspace window.

Try to change grade value to 100.

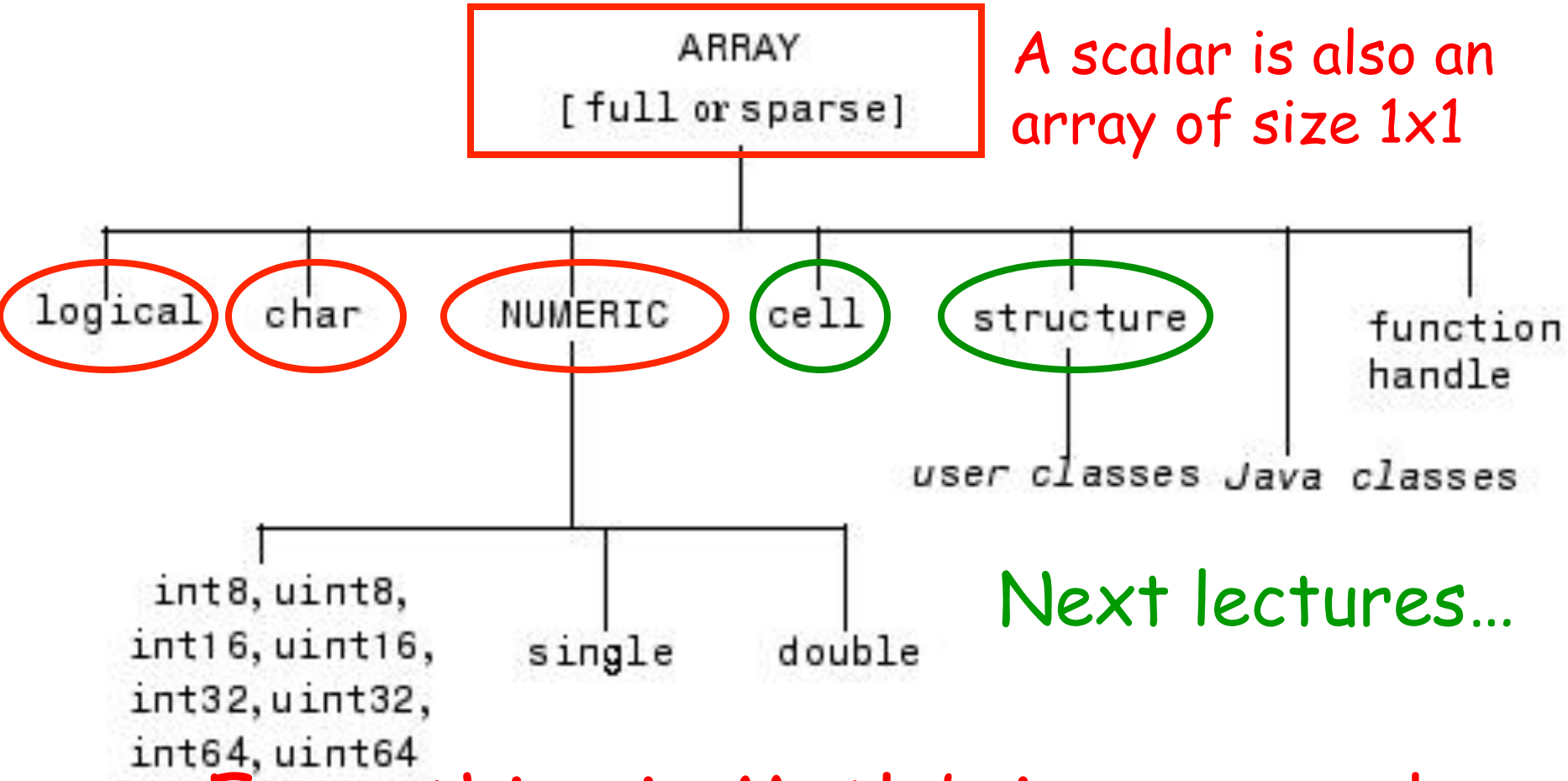
Workspace functions

- which(‘linspace’);
  - locates a function or identifies as variable
- clear x;
- save(‘file_name’ , ‘var1’ , ‘var2’ , …)
  - saves the workspace to a “.mat” file
- load(‘file_name’);
  - loads variables from a “.mat” file into the workspace
- Example:
  - Create some a,b,c variables
  - save(‘tutorial2.mat’, ‘a’, ‘b’,’c’);
  - clear;
  - Load(‘tutorial2.mat’)

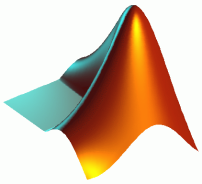


Data Types

Data Types (Variable types)



Everything in Matlab is an array!



Data Types (Variable Types)

Data Type	Example	Description
int8, uint8, int16, uint16, int32, uint32, int64, uint64 **Integer**	`uint16(65000)`	Array of signed (`int`) and unsigned (`uint`) integers. Some integer types require less storage space than `single` or `double`. All integer types except for `int64` and `uint64` can be used in mathematical operations.
single	`single(3 * 10^38)`	Array of single-precision numbers. Requires less storage space than `double`, but has less precision and a smaller range.
double **Default**	`3 * 10^300` `5 + 6i`	Array of double-precision numbers. Two-dimensional arrays can be sparse. The default numeric type in MATLAB.
logical	`magic(4) > 10`	Array of logical values of `1` or `0` to represent `true` and `false` respectively. Two-dimensional arrays can be sparse.
char	`'Hello'`	Array of characters. Strings are represented as vectors of characters. For arrays containing more than one string, it is best to use cell arrays.

Data types - Numeric

- Integer:

  a = int16(100);

  Be careful of memory overflow

  b = int8(5000);

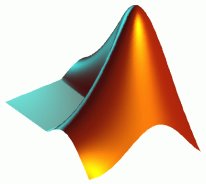
- Real(/Complex):

  x = double(235.5);

  x = single(235.5);

  x = 235;

Data Type	Range of Values	Conversion Function
Signed 8-bit integer	-2^7 to 2^7-1	int8
Signed 16-bit integer	-2^{15} to $2^{15}-1$	int16
Signed 32-bit integer	-2^{31} to $2^{31}-1$	int32
Signed 64-bit integer	-2^{63} to $2^{63}-1$	int64
Unsigned 8-bit integer	0 to 2^8-1	uint8
Unsigned 16-bit integer	0 to $2^{16}-1$	uint16
Unsigned 32-bit integer	0 to $2^{32}-1$	uint32
Unsigned 64-bit integer	0 to $2^{64}-1$	uint64

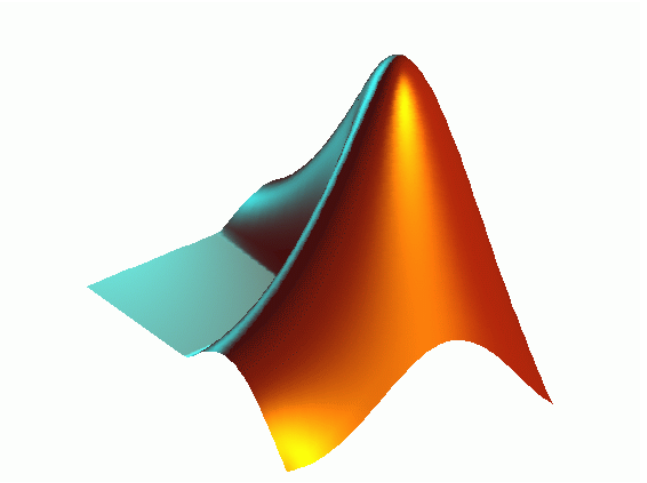


Numeric Data Functions

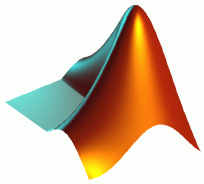
- A summery of numeric functions:
  - Help -> contents -> MATLAB -> Functions -> Mathematics -> Array and Matrices (and also the rest of the items…)
- Notice:
  - isinf, isnan, floor, ceil

Infinite

Is Not A Number



Arrays Manipulation



Arrays and Matrices

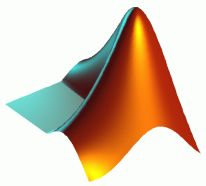
- A Matlab variable is an array.
  - A one dimensional array is named vector
  - A two dimensional array is named matrix
- Creating arrays
  - Which are equal?
    - a = [1 2 3 4];
    - b = [1, 2, 3, 4];
    - c = [1; 2; 3; 4];
    - d =1:1:4;
    - e = (1:4)';

Space / comma – New element in same line

New line / Semicolon – new line

Colon operator – from:step:to. Default step is one.

' operator- transpose (near the enter key)



linspace and logspace

- linspace(0,100,51);

Why? Consider 0:0.33:1 or 0:0.08:1

- logspace(1,10,11);

  Same, but logarithmically spaced between 10^1 and 10^10

Drawing the Sine and Cosine Functions

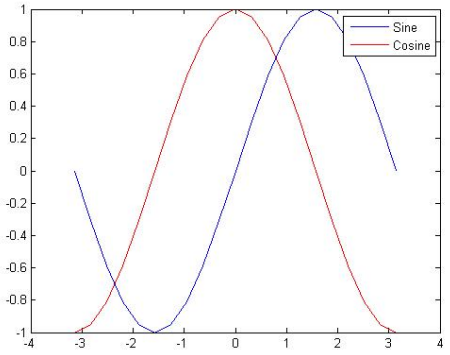
Reminder: linspace

- Use the commands we learned to define x as:

$$[-\pi,-0.9\pi,-0.8\pi,-0.7\pi,-0.6\pi,-0.5\pi,-0.4\pi,-0.3\pi,-0.2\pi,-0.1\pi,0,0.1\pi,0.2\pi,0.3\pi,0.4\pi,0.5\pi,0.6\pi,0.7\pi,0.8\pi,0.9\pi,\pi]$$

- Compute y and z – the sine and cosine function of x respectively.
- Draw a plot of the sine and cosine function of [-π, π] using the following code:
  - hold off; plot(x,y,'b-'); hold on; plot(x,z,'r-'); legend('Sine', 'Cosine');

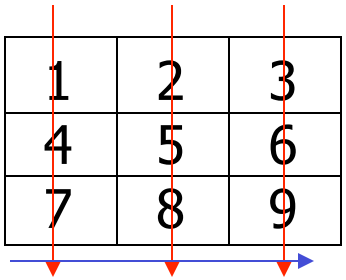
```
x = linspace(-pi,pi, 21);
% another way: x = (-1:0.1:1) * pi;

y = sin(x);
z = cos(x);
```

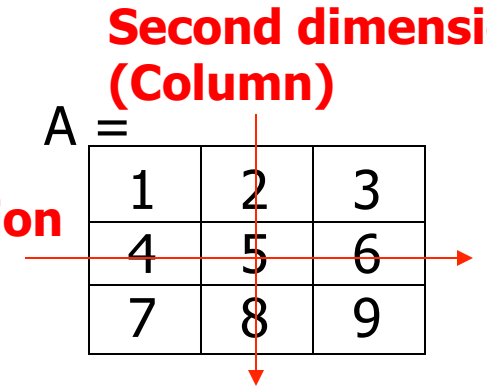


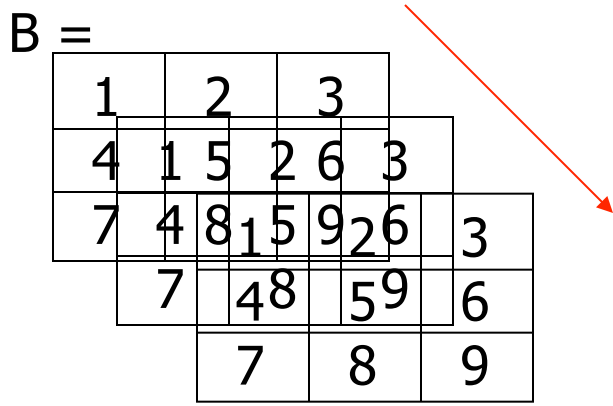
Multidimensional Array Indexing

- A = [1, 2, 3; 4, 5, 6; 7, 8, 9]
- Multi dimensional arrays:
  - B(:, :, 1) = A;
  - B(:, :, 2) = A;
  - B(:, :, 3) = A;
- Getting the matrix size:
  - **size**(A) ans = 3 3
  - See also: **length**(A) **numel**(A)
- Subscripts to single index:
  - **sub2ind**(size(A), 2,3)

    ans = 8
  - A(2,3) <->A(8)







Arrays Manipulation

a = [1, 2; 3, 4]
a’ % or a.’
a(2) a(:)
a(1,end:-1:1)
b = fliplr(a)
c = flipud(a)

Array flattening

The last coordinate in the dimension

Arrays Manipulation– More…

d = 1:2:9
e = 2:2:10
f = [d; e]
g = [d(1:3), e]
reshape(a,1,4)
repmat(e,2,2)

Vertical Array Concatenation

Horizontal Array Concatenation

See also: diag()

Assignment into Array

```
>> A = [1, 2, 3; 4, 5, 6; 7, 8, 9]
A =
     1     2     3
     4     5     6
     7     8     9
>> A(3,4)
??? Index exceeds matrix dimensions.
>> A(4,4) = 10
A =
     1     2     3     0
     4     5     6     0
     7     8     9     0
     0     0     0    10
```

1	2	3	
4	5	6	
7	8	9	**?**

Notice the difference!

1	2	3	0
4	5	6	0
7	8	9	0
0	0	0	10

Assignment into Array

```
>> A = [1, 2, 3; 4, 5, 6; 7, 8, 9]
A =
     1     2     3
     4     5     6
     7     8     9
>> A(:) = 21:30
??? In an assignment  A(:) = B, the
    number of elements in A and B
    must be the same.

>> B = [21, 22, 23; 24, 25, 26]
B =
    21    22    23
    24    25    26
>> A(1:2,1:3) = B
A =
    21    22    23
    24    25    26
     7     8     9
```

Subscripted assignment dimension must match!

Notice the difference!

1	2	3
4	5	6
7	8	9

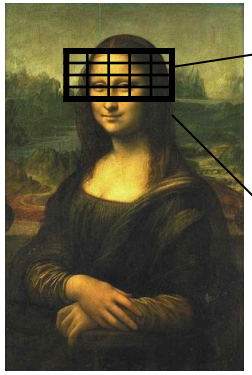
21	22	23
24	25	26

```
>> A(1:2,1:3) = 10
A =
    10    10    10
    10    10    10
     7     8     9
```

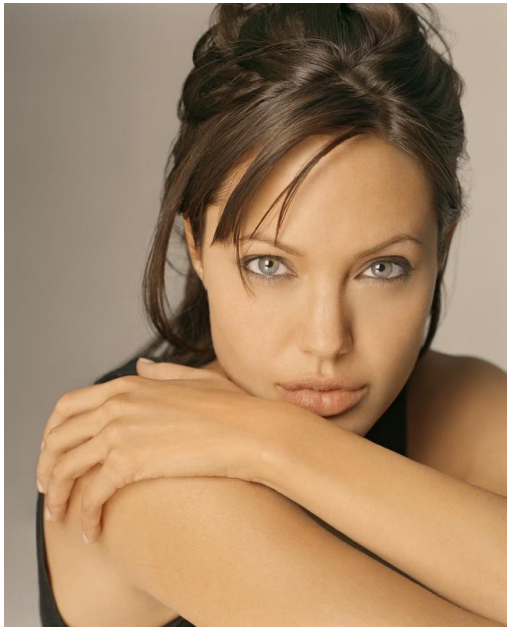
Scalar expansion

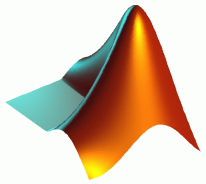
Example - Image Manipulation

- Images are represented by matrices
- Lets use this to visualize matrix manipulation



10	21	10	21
73	21	18	21
10	4	8	21
3	21	10	45
8	21	2	21





Standard Arrays

- zeros(10,10)
- ones(10,10)
- rand(10,10)
- randn(10,10)
- eye(10,10)

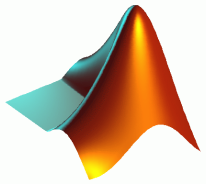
What is it good for?

Allocate memory in advance!

Create a sample.

Identity matrix

```
1   0   0
0   1   0
0   0   1
```



Find & logical Arrays

```
>> A = magic(3)
A =
     8     1     6
     3     5     7
     4     9     2
```

Logical operator

```
>> is_larger_than5 = A > 5
is_larger_than5 =
     1     0     1
     0     0     1
     0     1     0
```

Logical array

```
>> class(is_larger_than5)
ans =
Logical
```

```
>> A(is_larger_than5)
ans =
     8
     9
     6
     7
```

Array can be indexed with logical array!

```
>> index_larger_than5 = find(A>5)
index_larger_than5 =
     1
     6
     7
     8
```

Array of indices

```
>> A(index_larger_than5) = 10 % A(find(A>5))= 10
A =
    10     1    10
     3     5    10
     4    10     2
```

Assignment using indices array

```
>> [r c] = find(A>5)
r =
     1
     3
     1
     2
c =
     1
     2
     3
     3
```

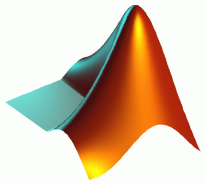
Matlab functions output depends on the # of output parameters!

find can return subscripts

```
>> A(r,c) = 20
% wrong!
A =
    20    20    20
    20    20    20
    20    20    20
```

What is wrong?

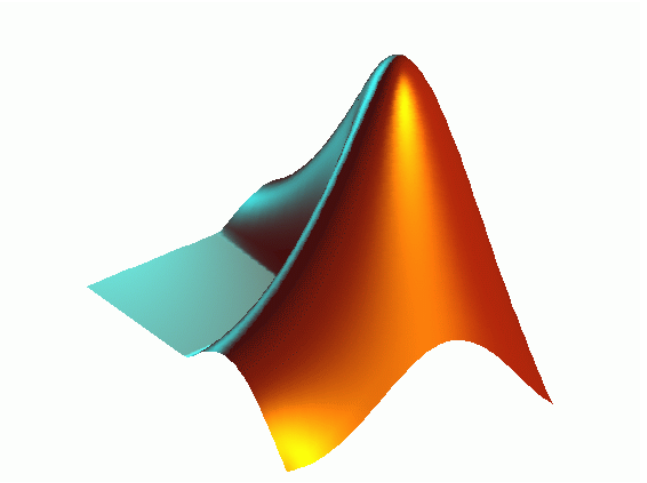
A([1 3 1 2], [1 2 3 3]) = 20



Short Example

- Consider the martix: `A = magic(6);`
  - Try to run `A(1:3,1:3) = ones(7,7);` What did you get?
- Replace all element larger than 20 with the value 100
- Now, make A be only its last three columns
- Then compute the mean of the fourth row.

```
A = magic(6)
A(:,:) = ones(7,7)
??? Subscripted assignment dimension mismatch.

A( find(A > 20) ) = 100;

A = A(:,4:6); % or  A(:,1:3) = [];

mean(A(4,:))
```



Arrays mathematical operations

Array arithmetic

- Element by element:
  - A+B
  - A-B
  - -A
  - A.*B
  - A./B
  - A.\B
  - .^
- Matrices dimensions must match
- + * /\ by scalar are element-wise

Search help for : “Arithmetic Operators“

- C = A*B :
  - A #columns == B #rows

$$C(i,j) = \sum_{k=1}^{n} A(i,k)B(k,j)$$

- A^p

Try to run the following:

```
x = 1:3;
y = x';

x .* y'
x*y
y*x
```

"Grades" Exercise Revisited

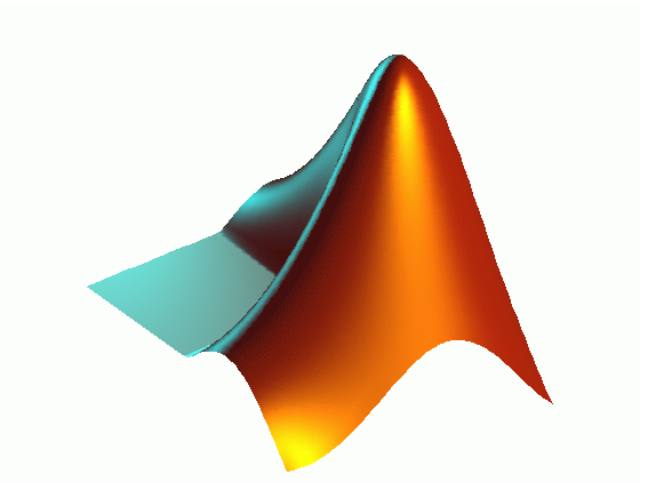
Consider the following:

Grade	80	77	70	60	65
Participation in class	1.10	1.05	1.07	0.99	1.10

- We want to compute the final grade as follows:
  - Multiply each grade by the class participation score
  - Then add 10 points to all grades?

Answer:

```
>> grades = [80 77 70 60 65];
>> participation_in_class   = [1.10, 1.05, 1.07, 0.99, 1.10];
>> final_grades = (grades .* participation_in_class) + 10
final_grades =
   98.0000   90.8500   84.9000   69.4000   81.5000
```



String Manipulations

Strings as Char Arrays

- Strings are represented as Char array
- Assigning a string:

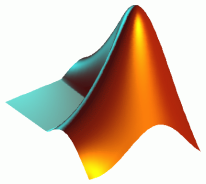
  a= ‘I am a string’

```
>> a = 'It is easy ';
>> b = 'to concatenate strings!';
>> [a, b]
ans =
It is easy to concatenate strings!
```

Char arrays can be concatenated like numerical arrays

Lets try it:

s= ‘I am a string’	
s(1:4)	
[s([1:9,6,10]) '!']	
	ans = gnirts a ma I



Working with Strings – Generating Output

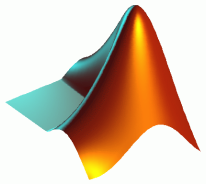
```
>> c = num2str(3.14157)
c = 3.1416

>> class(c)
ans = char

>> d = str2num(c)
d = 3.1416

>> class(d)
ans = double

>> ['pie value is: ' num2str(3.14157)]
ans = pie value is: 3.1416
```

Tip

num2str

str2num

Are very useful for output string generation



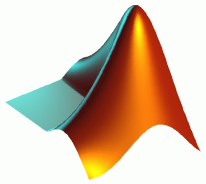
Working with strings – char matrices

```
>> str_mat = ['one  '
              'two  '
              'three']

str_mat =
one
two
three

>> class(str_mat)
ans = char

>> size(str_mat)
ans = 3     5
```

```
>> str_mat = ...
char('one','two','three')
str_mat =
one
two
Three
```

Concatenate vertically!

Rows length must be equal!
(a square matrix)



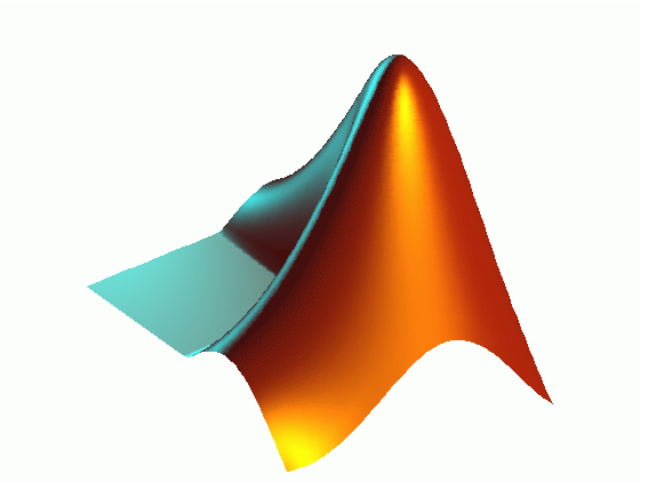
Working with Strings – String Comparison

Compare the strings

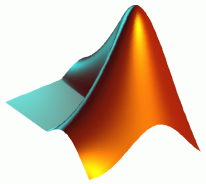
```
>> strcmp('abcd','abc')
ans =
     0
>> strncmp('abcd','abc',3)
ans =
     1
```

Compare the first “n” characters of the two strings

Check also: strfind



Array Editor

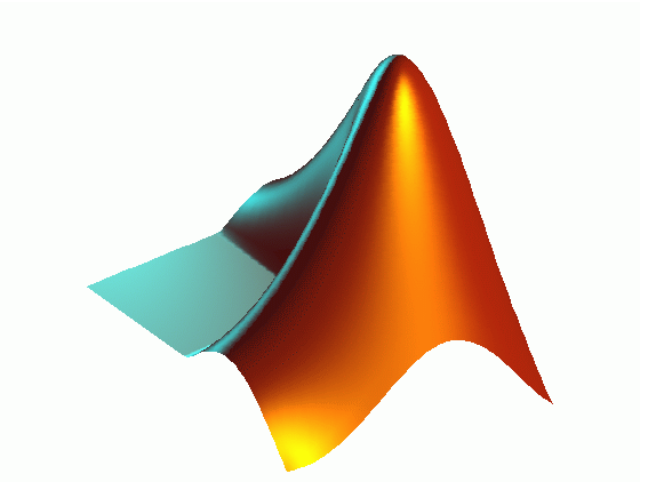


Array Editor and Array Display

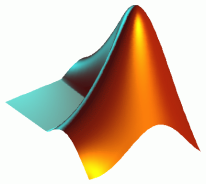
- Display:
  - Omitting the “;”
  - disp(‘hello world’);
  - [‘Today we added’ num2str()]
  - Example:

```
three = 2;
['One plus one is:' num2str(three)]
```

- Array Editor:
  - From: work space, “open selection”
  - Displays variables
  - Edit variables value
  - Copy-paste from Excel (grades example)



Solving Linear System of Equations



Epilogue - Solving Linear Systems of Equations

- Say we want to solve the following system:

$$\begin{cases} a_{11}x_1 & a_{12}x_2 & a_{13}x_3 = b_1 \\ a_{21}x_1 & a_{22}x_2 & a_{23}x_3 = b_2 \\ a_{31}x_1 & a_{32}x_2 & a_{33}x_3 = b_3 \end{cases}$$

Ax=b

- A = [1 1 1 ; 1 2 3 ; 1 3 6];
- b = [3; 1; 4];
- x = A\b;
- Solution: x = 10 -12 5
- More: search help “solving linear equations”

Epilogue - Solving Linear Systems of Equations

Solve the following problem:

$1x+2y+3z = 366$

$4x+5y+6z = 804$

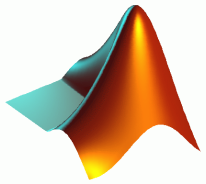
$7x+8y = 351$

Answer:

```
A = [1 2 3
     4 5 6
     7 8 0];
b = [366; 804; 351];

x = A\b
```

Result:

```
x =

   25.0000
   22.0000
   99.0000
```



Summary

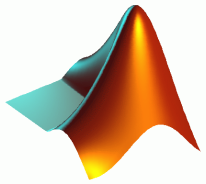
- Introduce the Notion of Variables & Data Types.
- Master Arrays manipulation
- Learn Arrays Mathematical Operations
- Learn Simple String Manipulations
- Learn to Use Array Editor
- Solving Linear System of Equations

Exercises

- Write a row vector, called “v1” with the following five numbers: 12, 23, 54, 8, 6.
- Add 10 to each element of "v1" and store it at "v2".
- Visualize a two-dimension graphic of "v2".
- Create the following matrix of 4x4 elements and call it M:

```
1     4     22     7
9     2     3      11
49    55    6      3
24    7     9      12
```

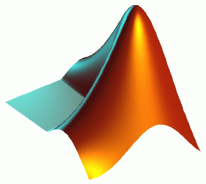
- Obtain the transposed matrix of M and call it Mt.



Exercises

- Obtain the inverse of M and call it Mi.
- Proof that the product of a matrix by its inverse is the identity matrix.
- Indicate the contents of the "ans" variable.
- Visualize a 3D graphic of the M matrix in wire as well as a surface (type: >>mesh(M), >> surf(M) – ">>" is the prompt symbol–).
- Explain what does each one of the demo expressions:

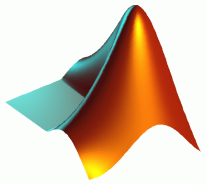
```
x=0:0.02:1;
hndl=plot(x,humps(x));
set(hndl,'Color','cyan')
```



Exercises

- Obtain the inverse of M and call it Mi.
- Proof that the product of a matrix by its inverse is the identity matrix.
- Indicate the contents of the "ans" variable.
- Visualize a 3D graphic of the M matrix in wire as well as a surface (type: >>mesh(M), >> surf(M) – ">>" is the prompt symbol–).
- Explain what does each one of the demo expressions:

```
x=0:0.02:1;
hndl=plot(x,humps(x));
set(hndl,'Color','cyan')
```



Exercises

- What is stored in *hndl*?
- What sentence must be written to change to green the color line?
- What sentence must be written to change the line width to 2?
- What sentence must be written to change the line symbol to "o"?
- What sentence must be written to change the line symbol size to 12?
- Check and indicate what do the next expressions:

```
A=[5 7 8; 0 1 9; 4 3 6]
A(:,:,2)=[1 0 4;3 5 6; 9 8 7]
```

- Next, check the following expressions, write the returned value and indicate what are the differences among them:

```
A(1,2) A(1,2,1) A(1,2,2)
```

- Next, how do you access to the element in the third row, second column in the second dimension?