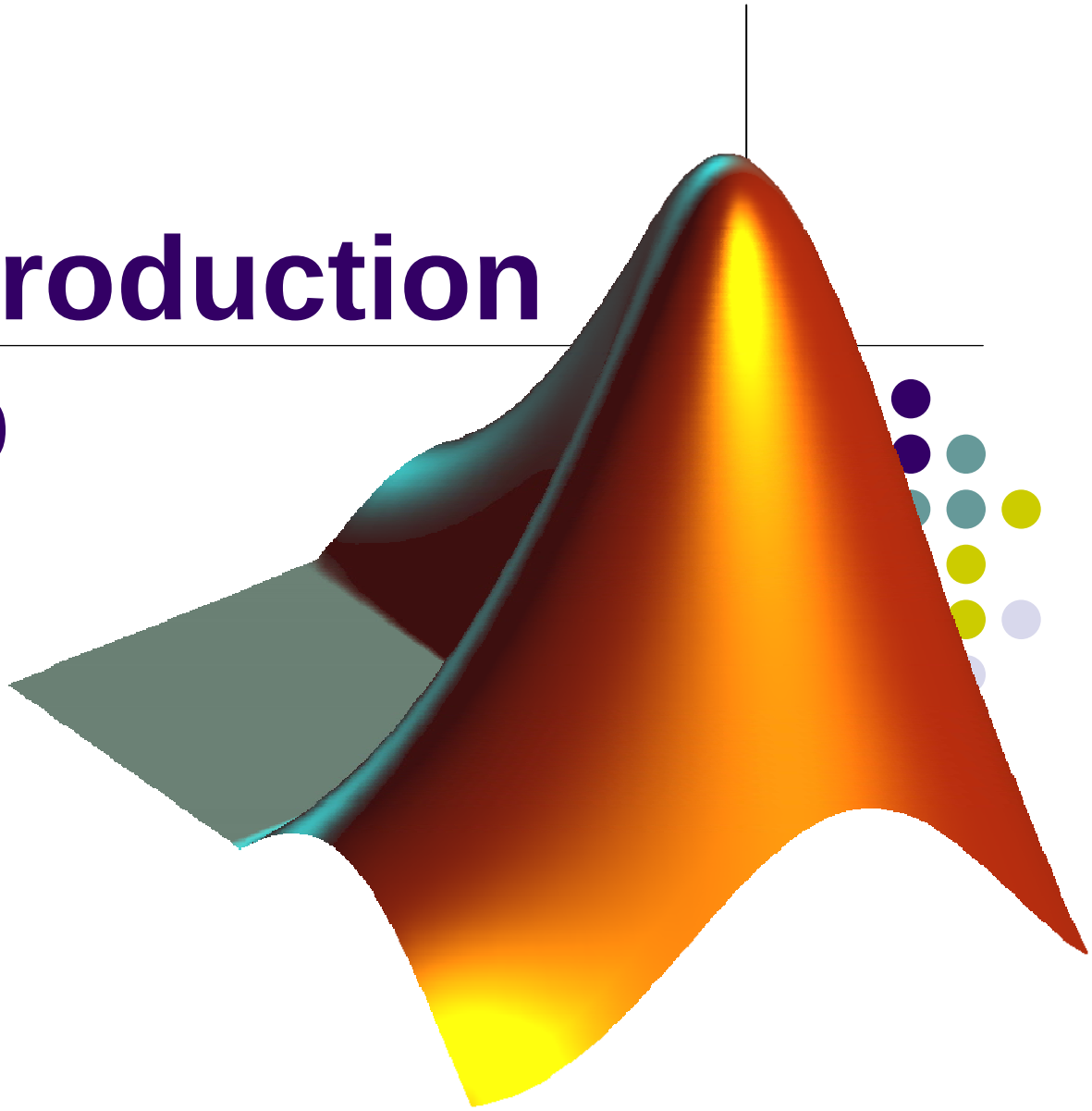
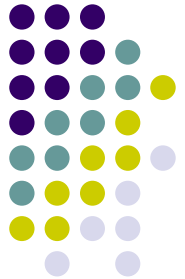
Quick introduction to Matlab



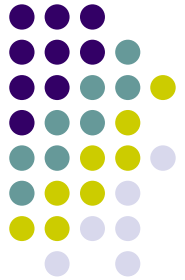
PASCAL Bootcamp in Machine Learning - 2007

Outline



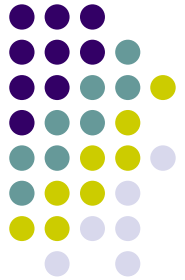
- Matlab introduction
- Matlab elements
  - Types
  - Variables
  - Matrices
- Loading, saving and ploting
- Matlab Programming language
- Scripts and functions

Matlab introduction



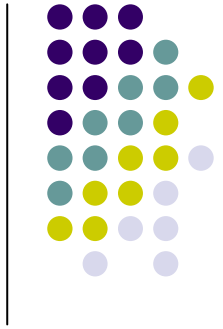
- Matlab is a program for doing numerical computation. It was originally designed for solving linear algebra type problems using matrices. It's name is derived from MATrix LABoratory.
- Matlab is also a programming language that currently is widely used as a platform for developing tools for Machine Learning

Matlab introduction



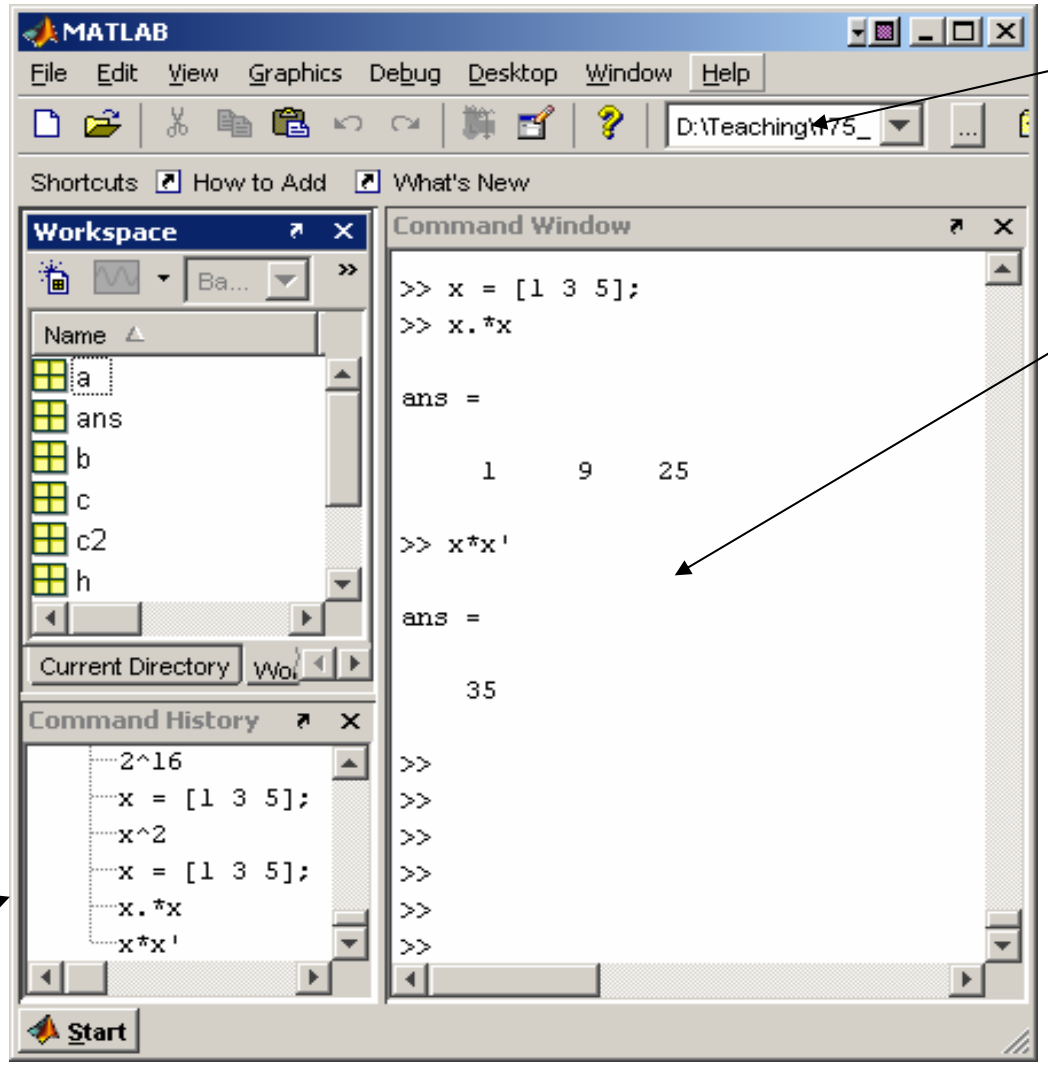
- Why it is useful for prototyping AI projects:
  - large toolbox of numeric/image library functions
  - very useful for displaying, visualizing data
  - high-level: focus on algorithm structure, not on low-level details
  - allows quick prototype development of algorithms

Matlab introduction



- Some other aspects of Matlab
  - Matlab is an interpreter -> not as fast as compiled code
    - Typically quite fast for an interpreted language
    - Often used early in development -> can then convert to C (e.g.,) for speed
  - Can be linked to C/C++, JAVA, SQL, etc
  - Commercial product, but widely used in industry and academia
    - Many algorithms and toolboxes freely available

Opening Matlab



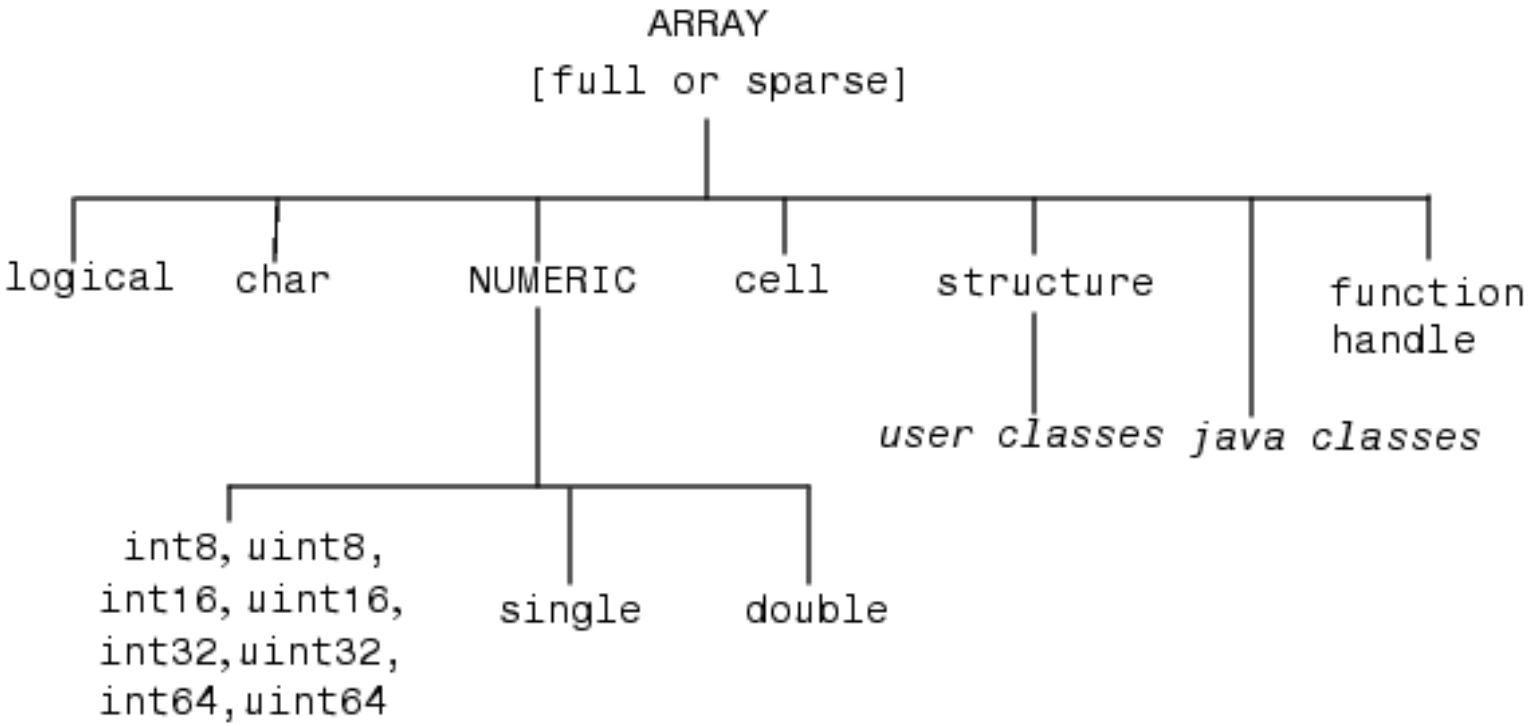
Working Path

Working Memory

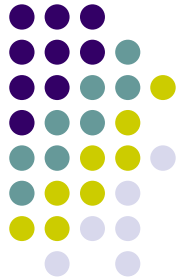
Command Window

Command History

Data Types

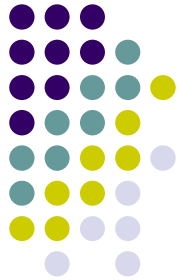

ARRAY
[full or sparse]
logical
char
NUMERIC
cell
structure
function handle
user classes
java classes
int8, uint8,
int16, uint16,
int32, uint32,
int64, uint64
single
double

Variables



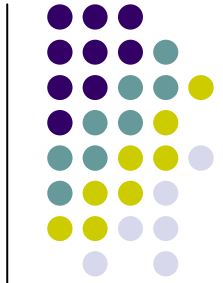
- Have not to be previously declared
- Variable names can contain up to 63 characters
- Variable names must start with a letter followed by letters, digits, and underscores.
- Variable names are case sensitive

Matlab Special Variables



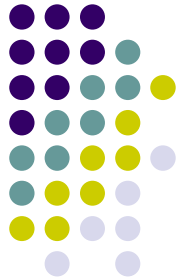
ans	Default variable name for results
pi	Value of π
eps	Smallest incremental number
inf	Infinity
NaN	Not a number e.g. 0/0
realmin	The smallest usable positive real number
realmax	The largest usable positive real number

Matlab Assignment & Operators



Assignment	=	a = b	(assign b to a)
Addition	+	a + b	
Subtraction	-	a - b	
Multiplication	* or .*	a*b or	a.*b
Division	/ or ./	a/b or	a./b
Power	^ or .^	a^b or	a.^b

Matlab Matrices



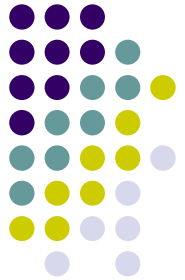
- Matlab treats all variables as matrices. For our purposes a matrix can be thought of as an array, in fact, that is how it is stored.
- Vectors are special forms of matrices and contain only one row OR one column.
- Scalars are matrices with only one row AND one column

Matlab Matrices

- A matrix with only one row is called a row vector. A row vector can be created in Matlab as follows (note the commas):

```
» rowvec = [12 ,  14 , 63]

rowvec =

    12    14    63
```

Matlab Matrices



- A matrix with only one column is called a column vector. A column vector can be created in MATLAB as follows (note the semicolons):

```
» colvec = [13 ; 45 ; -2]

colvec =

    13
    45
    -2
```

Matlab Matrices

- A matrix can be created in Matlab as follows (note the commas AND semicolons):

```
» matrix = [1 , 2 , 3 ; 4 , 5  ,6 ; 7 , 8 , 9]

matrix =

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

Extracting a Sub-Matrix

- A portion of a matrix can be extracted and stored in a smaller matrix by specifying the names of both matrices and the rows and columns to extract. The syntax is:

```
sub_matrix = matrix ( r1 : r2 , c1 : c2 ) ;
```

where r1 and r2 specify the beginning and ending rows and c1 and c2 specify the beginning and ending columns to be extracted to make the new matrix.

Matlab Matrices

- A column vector can be extracted from a matrix. As an example we create a matrix below:

```
» matrix=[1,2,3;4,5,6;7,8,9]

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

- Here we extract column 2 of the matrix and make a column vector:

```
» col_two=matrix( : , 2)

col_two =
     2
     5
     8
```

Matlab Matrices

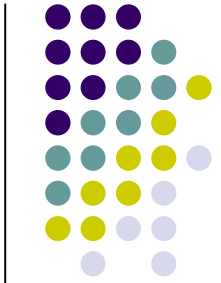
- A row vector can be extracted from a matrix. As an example we create a matrix below:

```
» matrix=[1,2,3;4,5,6;7,8,9]

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

- Here we extract row 2 of the matrix and make a row vector. Note that the 2:2 specifies the second row and the 1:3 specifies which columns of the row.

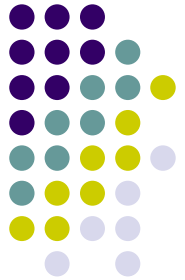
```
» rowvec=matrix(2 : 2 , 1 : 3)

rowvec =
     4     5     6
```

Colon Operator



j:k	is the same as [j,j+1,...,k] is empty if j > k
j:i:k	is the same as [j,j+i,j+2i, ..,k] is empty if i > 0 and j > k or if i < 0 and j < k
A(:,j)	is the j-th column of A
A(i,:)	is the i-th row of A
A(:,:)	is the equivalent two-dimensional array. For matrices this is the same as A.
A(j:k)	is A(j), A(j+1),...,A(k)
A(:,j:k)	is A(:,j), A(:,j+1),...,A(:,k)
A(:,:,k)	is the k-th page of three-dimensional array A.
A(i,j,k,:)	is a vector in four-dimensional array A. The vector includes A(i,j,k,1), A(i,j,k,2), A(i,j,k,3), and so on.
A(:)	is all the elements of A, regarded as a single column. On the left side of an assignment statement, A(:) fills A, preserving its shape from before. In this case, the right side must contain the same number of elements as A.

Matlab Matrices



- **Accessing Single Elements of a Matrix**

  A(i,j)

- **Accessing Multiple Elements of a Matrix**

  A(1,4) + A(2,4) + A(3,4) + A(4,4) ➔ sum(A(1:4,4)) or sum(A(:,end))

  The keyword end refers to the *last* row or column.

- **Deleting Rows and Columns**

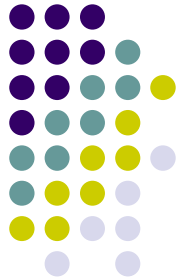
  to delete the second column of X, use

  X(:,2) = []

- **Concatenating Matrices A and B**

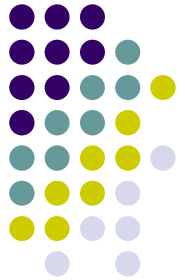
  C=[A;B]

Some matrix functions in Matlab



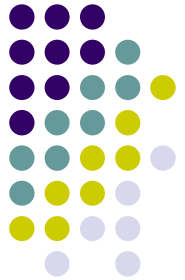
- X = ones(r,c) % Creates matrix full with ones
- X = zeros(r,c) % Creates matrix full with zeros
- A = diag(x) % Creates squared matrix with vector x in diagonal
- [r,c] = size(A) % Return dimensions of matrix A
- + - * / % Standard operations
- .+ .- .* ./ % Wise addition, substraction,…
- v = sum(A) % Vector with sum of columns

Some powerful matrix functions in Matlab



- X = A’ % Transposed matrix
- X = inv(A) % Inverse matrix squared matrix
- X = pinv(A) % Pseudo inverse
- X = chol(A) % Cholesky decomp.
- d = det(A) % Determinant
- [X,D] = eig(A) % Eigenvalues and eigenvectors
- [Q,R] = qr(X) % QR decomposition
- [U,D,V] = svd(A) % singular value decomp.

Sava data in files

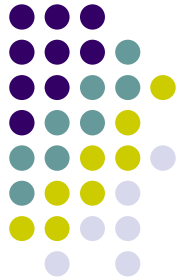


- save myfile VAR1 VAR2 …

  or

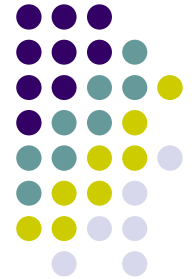
- save(‘myfile’,’VAR1’,’var2’)

Load data from files



- Load
  - load filename
  - load ('filename')
  - load filename.ext
  - load filename -ascii
  - load filename -mat
- File Formats
  - mat -> Binary MAT-file form
  - ascii -> 8-digit ASCII form
  - ascii–tabs Delimit array elements with tabs

Plotting with Matlab



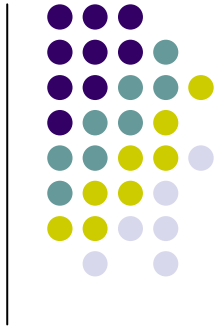
- Matlab has a lot of function for plotting data. The basic one will plot one vector vs. another. The first one will be treated as the abscissa (or x) vector and the second as the ordinate (or y) vector. The vectors have to be the same length.

```
>> plot (time, dist)        % plotting versus time
```

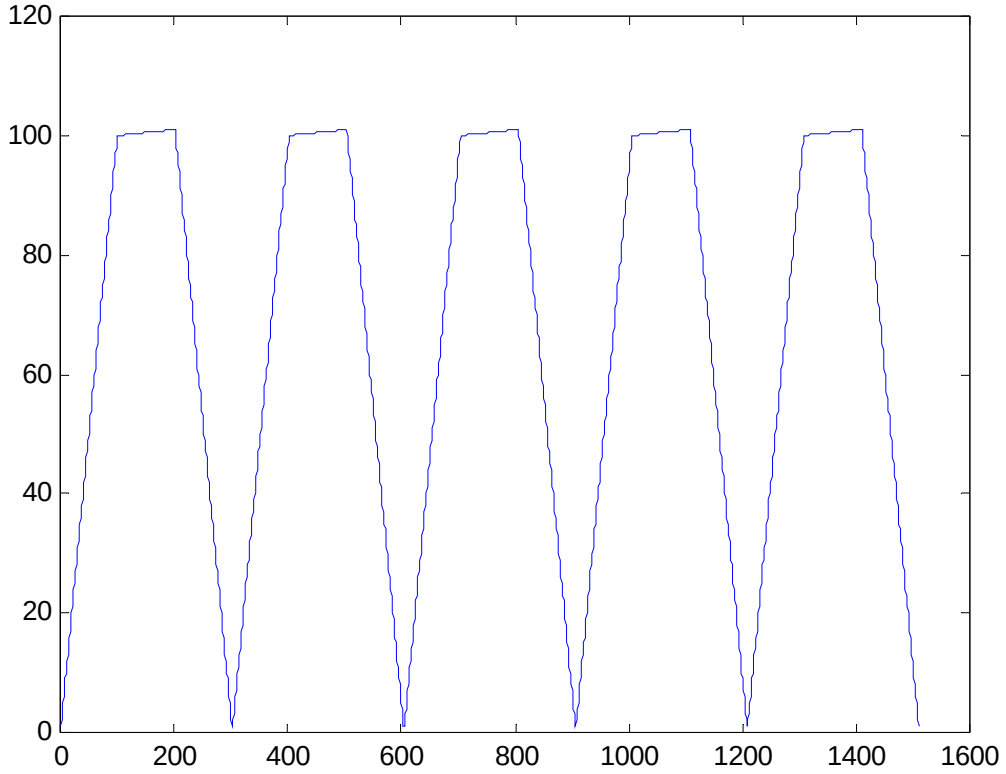
- Matlab will also plot a vector vs. its own index. The index will be treated as the abscissa vector. Given a vector “time” and a vector “dist” we could say:

```
>> plot (dist)              % plotting versus index
```

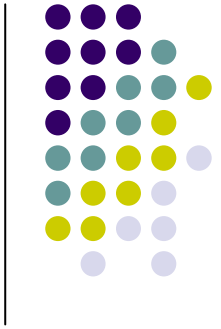
Plotting with Matlab



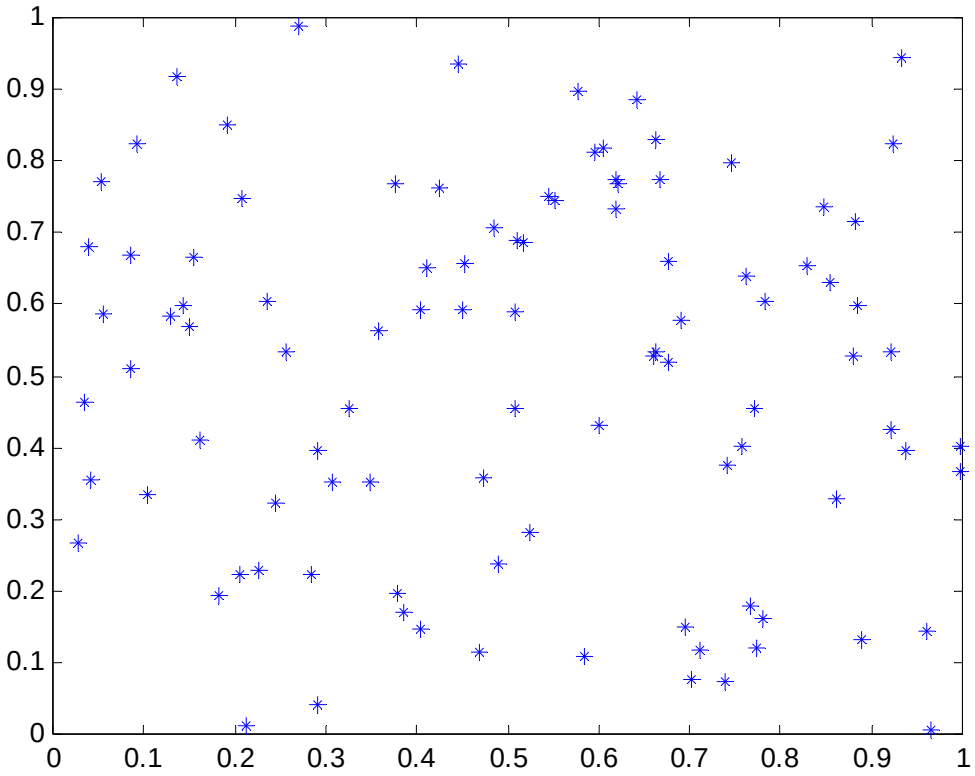
```
» a = 1:100;
» b = 100:0.01:101;
» c = 101:-1:1;
» d = [a b c];
» e = [d d d d d];
» plot(e)
```



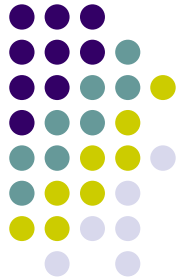
Plotting with Matlab



```
» x = rand(1,100);
» y = rand(1,100);
» plot(x,y,'*')
```



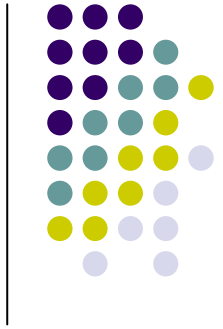
Plotting with Matlab



- There are commands in Matlab to "annotate" a plot to put on axis labels, titles, and legends. For example:

```
>> % To put a label on the axes we would use:
>> xlabel ('X-axis label')
>> ylabel ('Y-axis label')

>> % To put a title on the plot, we would use:
>> title ('Title of my plot')
```

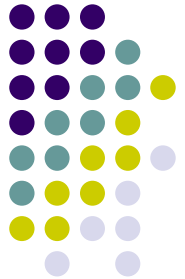
Plotting with Matlab



- Vectors may be extracted from matrices. Normally, we wish to plot one column vs. another. If we have a matrix "mydata" with two columns, we can obtain the columns as a vectors with the assignments as follows:

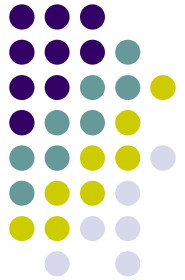
```
>> first_vector = mydata ( : , 1) ;          % First column
>> second_vector = mydata ( : , 2) ;         % Second one
>>% and we can plot the data
>> plot ( first_vector , second_vector )
```

Matlab programming language



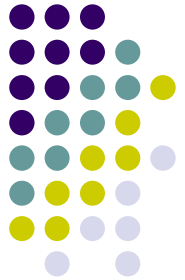
- Elements of Matlab as a programming language:
  - Expressions
  - Flow Control blocks
    - Conditional
    - Iterations
  - Scripts
  - Functions

Expressions: Matlab Relational Operators



- MATLAB supports six relational operators.
  - Less Than <
  - Less Than or Equal <=
  - Greater Than >
  - Greater Than or Equal >=
  - Equal To ==
  - Not Equal To ~=

Expressions: Matlab Logical Operators



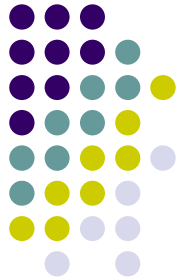
- MATLAB supports three logical operators.
  - not ~ % highest precedence
  - and & % equal precedence with or
  - or | % equal precedence with and

Expressions: Matlab Logical Functions

- MATLAB also supports some logical functions.

any (x)	returns 1 if any element of x is nonzero
all (x)	returns 1 if all elements of x are nonzero
isnan (x)	returns 1 at each NaN in x
isinf (x)	returns 1 at each infinity in x
finite (x)	returns 1 at each finite value in x

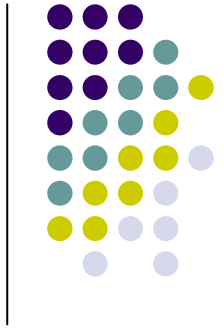
Matlab Conditional Structures



```
if expression cond.
   sentences
elseif expr. cond.
   sentences
else
   sentences
end
```

```
a = input('valor1? ');
b = input('valor2? ');
if a == b,
   fprintf('a is equal to b\n');
elseif a > 0 && b > 0
   fprintf('both positive\n');
else
   fprintf('other case\n');
end
```

Matlab Iteration Structures (I)



```
for variable = expr
   sentence;
    ...
   sentence;
end
```

```
M = rand(4,4); suma = 0;
for i = 1:4
  for j = 1:4
    suma = suma + M(i,j);
  end
end
fprintf('sum = %d\n',suma);
```

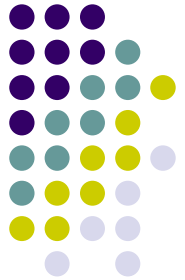
```
M = rand(10,10); suma = 0;
for i = {2,5:8}        % files 2, 5, 6, 7 i 8
  for j = {1:5,8:9}    % rows 1, 2, 3, 4, 5, 8, 9
    suma = suma + M(i,j);
  end
end
fprintf('sum = %d\n',suma);
```

Matlab Iteration Structures (II)

```
while expr
   sentence;
    ...
   sentence;
end
```

```
M = rand(4,4);
i = 1; j = 1; suma = 0;

while i <= 4
  while j <= 4
    suma = suma + M(i,j);
    j = j+1;
  end
  i = i+1;
end

fprintf('suma = %f\n',suma);
```

(Optimizing code: vectorization)



- Loops should be avoided when possible:

```
for ind = 1:10000
    b(ind)=sin(ind/10)
end
```

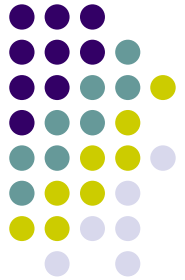
Alternatives:

```
x=0.1:0.1:1000;
b=sin(x);
```

```
x=1:10000;
b=sin(x/10);
```

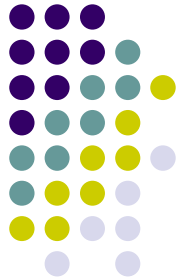
Most of the loops can be avoided!!!

M-files



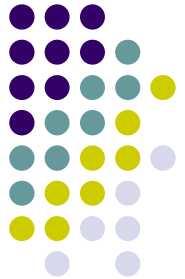
- Text files containing Matlab programs. Can be called form the command line or from other M-files
- Present "**.m**" extension
- Two kind of M-files:
  - Scripts
  - Functions

M-files: Scripts



- Without input arguments, they do not return any value.

M-files: Script Example



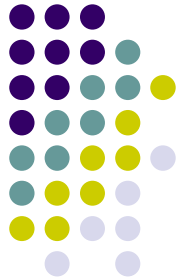
1) >> **edit** estadistica.m
2) Write into the editor:

```
x = [4 3 2 10 -1];
n = length(x);
suma1 = 0; suma2 = 0;
for i=1:n
   suma1 = suma1 + x(i);
   suma2 = suma2 + x(i)*x(i);
end
promig = suma1/n;
desvia = sqrt(suma2/n – promig*promig);
```

3) Save the file
4) >> run estadistica
5) >> promig, desvia

promig = 3.6000
desvia = 3.6111

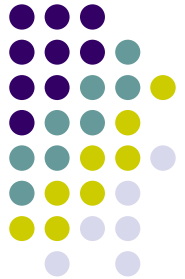
M-files: Functions



- With parameters and returning values
- Only visible variables defined inside the function or parameters
- Usually one file for each function defined
- Structure:

```
function [out1, out2, ..., outN] = name-function (par1, par2, ..., parM)
    sentence;
    ....
    sentence;
end
```

M-files: Functions Example



1) >> **edit** festadistica.m
2) Write into the editor:

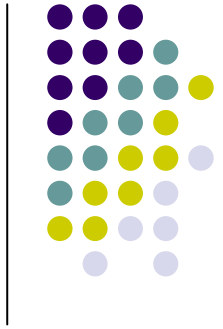
```
function [promig,desvia] = festadistica(x)
n = length(x);
[suma1,suma2] = sumes(x,n);
promig = suma1/n;
desvia = sqrt(suma2/n – promig*promig);
end
```

3) Save the file
4) >> **edit** sumes.m
5) Write into the editor:

```
function [sy1,sy2] = sumes(y,m)
sy1 = 0; sy2 = 0;
for i=1:m
  sy1 = sy1 + y(i); % suma yi
  sy2 = sy2 + y(i)*y(i); % suma yi^2
end
end
```

6) Save the file
7) >> [p,d] = festadistica([4 3 2 10 -1])
p = 3.6000
d = 3.6111

Help



- Within Matlab
  - Type **help** at the Matlab prompt or **help** followed by a function name for help on a specific function
- Online
  - Online documentation for Matlab at the MathWorks website
    - http://www.mathworks.com/access/helpdesk/help/techdoc/matlab.html
  - There are also numerous tutorials online that are easily found with a web search.