MATLAB - Basics

Centro de Informática
Universidade Federal de Pernambuco
Aprendizagem de Máquina – IN1102

Arley Ristar – arrr2@cin.ufpe.br

Based on ES 156 Matlab Presentation - Harvard SEAS

Outline

- Introduction
- Data structure: matrices, vectors and operations
- Programming
  - Functions, scripts
  - Programming control commands
- File I/O
- Basic line plots
- Specialized Graphics
  - Bar graph
  - Pie chart
  - Histogram
- Convolution
- Practice Problems

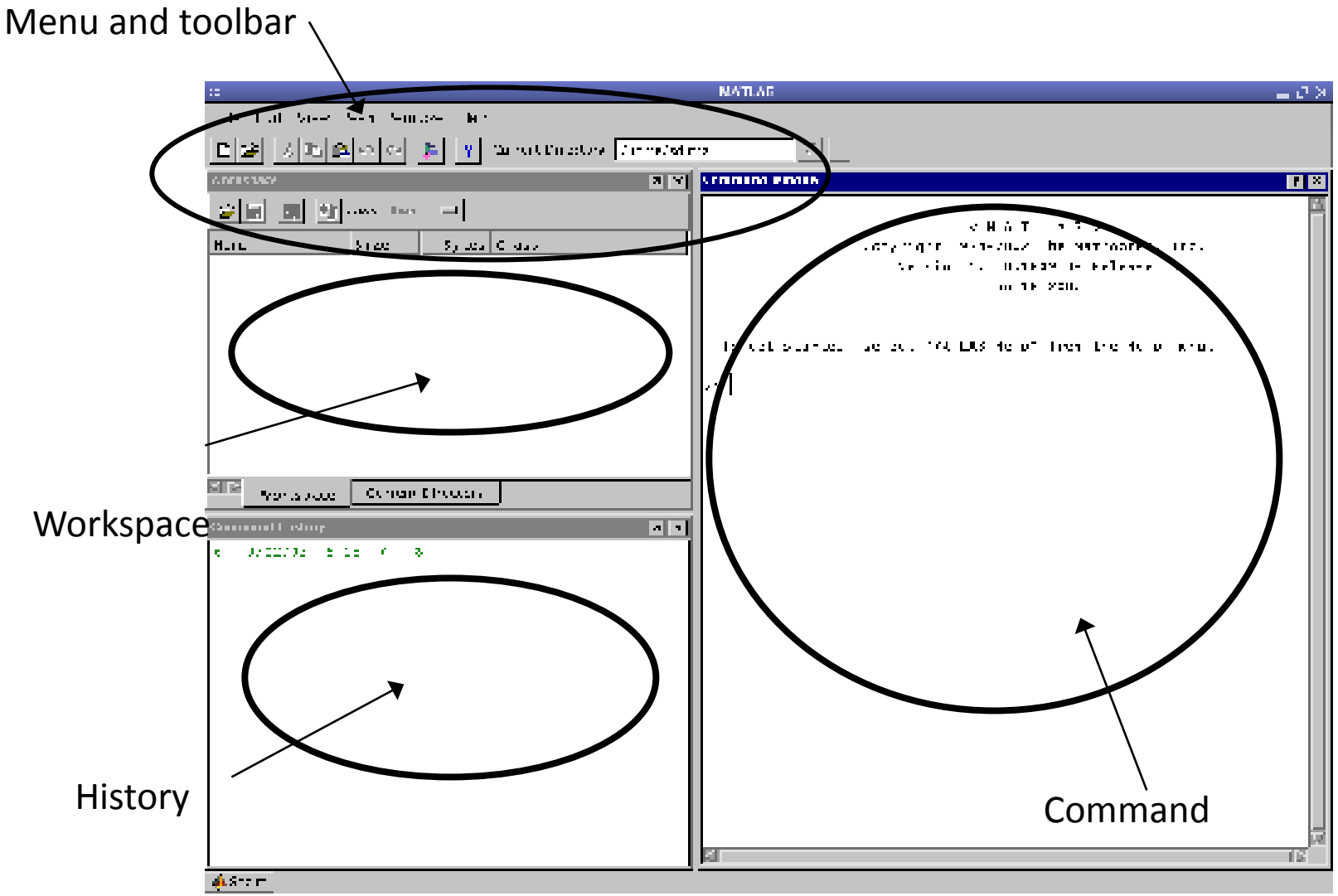
What is MATLAB

- High level language for technical computing
- Stands for **MAT**rix **LAB**oratory
- Everything is a matrix - easy to do linear algebra

The MATLAB System

- Development Environment
- Mathematical Function Library
- MATLAB language
- Application Programming Language

The MATLAB Desktop



Matrices & Vectors

- All (almost) entities in MATLAB are matrices
- Easy to define:

```
>> A = [16 3; 5 10]
  A =    16     3
         5    10
```

- Use ‘,’ or ‘ ’ to separate row elements -- use ‘;’ to separate rows

Matrices & Vectors - II

- Order of Matrix -
  - m=no. of rows, n=no. of columns

$$m \times n$$

- Vectors - special case
  - n = 1 column vector
  - m = 1 row vector

Creating Vectors and Matrices

- Define

```
>> A = [16 3; 5 10]
   A =    16     3
          5    10
>> B = [3 4 5
   6 7 8]
   B = 3   4   5
       6   7   8
```

- Transpose

Vector :

```
>> a=[1 2 3];
>> a'
      1
      2
      3
```

Matrix:

```
>> A=[1 2; 3 4];
>> A'
ans =
      1     3
      2     4
```

Creating Vectors

Create vector with equally spaced intervals

```
>> x=0:0.5:pi
x =
 0 0.5000 1.0000 1.5000 2.0000 2.5000 3.0000
```

Create vector with *n* equally spaced intervals

```
>> x=linspace(0, pi, 7)
x =
  0 0.5236 1.0472 1.5708 2.0944 2.6180 3.1416
```

Equal spaced intervals in logarithm space

```
>> x=logspace(1,2,7)
x =
  10.0000 14.6780 21.5443 … 68.1292  100.0000
```

Creating Matrices

- `zeros(m, n)`: matrix with all zeros
- `ones(m, n)`: matrix with all ones.
- `eye(m, n)`: the identity matrix
- `rand(m, n)`: uniformly distributed random
- `randn(m, n)`: normally distributed random
- `magic(m)`: square matrix whose elements have the same sum, along the row, column and diagonal.
- `pascal(m)` : Pascal matrix.

Matrix operations

- `^`: exponentiation
- `*`: multiplication
- `/`: division
- `\`: left division. The operation `A\B` is effectively the same as `INV(A)*B`, although left division is calculated differently and is much quicker.
- `+`: addition
- `-`: subtraction

Array Operations

- Evaluated element by element
  - `.'` : array transpose (non-conjugated transpose)
  - `.^` : array power
  - `.*` : array multiplication
  - `./` : array division
- Very different from Matrix operations

```
>> A=[1 2;3 4];
>> B=[5 6;7 8];
>> A*B
    19    22
    43    50
```

But:

```
>> A.*B
     5    12
    21    32
```

Some Built-in functions

- `mean(A)`: mean value of a vector
- `max(A), min (A)`: maximum and minimum.
- `sum(A)`: summation.
- `sort(A)`: sorted vector
- `median(A)`: median value
- `std(A)`: standard deviation.
- `det(A)` : determinant of a square matrix
- `dot(a,b)`: dot product of two vectors
- `Cross(a,b)`: cross product of two vectors
- `Inv(A)`: Inverse of a matrix A

Indexing Matrices

Given the matrix:

A = (n →, m ↕)

0.9501	0.6068	0.4231
0.2311	0.4860	0.2774

Then:

```
A(1,2) = 0.6068
A(3) = 0.6068
A(:,1) = [0.9501
          0.2311 ]
1:m
A(1,2:3)=[0.6068   0.4231]
```

$$\longrightarrow A_{ij}, i = 1...m, j = 1...n$$

$$\longrightarrow index = (i-1)m + j$$

Adding Elements to a Vector or a Matrix

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

>> B=1:2
B=
    1   2
>> B(5)=7;
B=
    1   2   0   0   7
```

```
>> C=[1  2; 3  4]
C=
    1   2
    3   4
>> C(3,:)=[5  6];
C=
    1   2
    3   4
    5   6

>> D=linspace(4,12,3);
>> E=[C  D']
E=
    1   2   4
    3   4   8
    5   6   12
```

Programming in MATLAB: Scripts

- Scripts have no input and output
- List of commands
- Variables are available on completion
- Simple example

```
theta = -pi:0.01:pi;          % Computations
rho  = 2 *sin(5 *theta).^2;
polar(theta,rho)              % Graphics output
```

Programming MATLAB: Functions

Writing an M-File:

```
function f = fact(n)                    definition
% FACT Factorial.                       help line
% FACT(N) returns the factorial of N
% usually denoted by N!
% Put simply, FACT(N) is PROD(1:N).

f = prod(1:n);      % Function body
return
```

Programming in MATLAB – functions II

- Functions have inputs and outputs
- Variables defined in the function isn't defined after function completes evaluating
- Call a function

  [out1, out2,…, outN] = functionname(in1, in2, ..., inN)
- A MATLAB function is usually saved as a *.m file with the filename the same as the function name. When a function is being called, MATLAB looks for the filename.

Programming in MATLAB-subfunctions

- An M-file can contain code for more than one function.
- Additional functions within the file are called subfunctions.
- Subfunctions are only visible to the primary function or to other subfunctions in the same file.

```
function [avg, sdv] = mystats(u) % Primary function
% mystats finds mean and standard deviation of a
  vector
avg=mean(u);
sdv = mysdv(u);

function a = mysdv(v) %subfunction
a=sum((u-mean(u)).^2)/(length(u)-1);
a=sqrt(a);
```

Flow control

- Logic control structures:
- Iterative structures:

```
If/elseif/else
switch/case/otherwise
```

```
for
while
```

- Traditional for loop
- Matlab vector for loop

```
for i=1:10
  for j=1:10
    a(i,j)=b(i,j)*c(i,j);
  end
end
```

```
a = b.*c;
```

Try to avoid for loop

Compare the computation time of these two scripts

```
x=1:1e7;
s=sum(x)
```

```
x=1:1e7;
s=0;
for i=1:1e7
    s=s+x(i);
end
s
```

Examples- if/else I

Construct a tri-diagonal matrix A

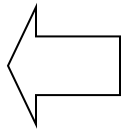
$$\mathbf{A} = \begin{pmatrix} \mathbf{3} & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & \mathbf{3} & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & \mathbf{3} & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & \mathbf{3} & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & \mathbf{3} \end{pmatrix}$$

Example – if/else II

```
a=zeros(5,5);
for i=1:5
    for j=1:5
        if i==j

  a(i,j)=3;
              elseif
  abs(i-j)==1

  a(i,j)=1;
        end
    end
end
```



If we write the code into a *.m file, and save it as mytridiag.m. We can execute this script in the main command window by typing

Mytridiag

Examples- for/while loop

- You have a vector with length N-1, all the elements are distinct and belong to the set {1, 2, ……, N}. That is, one integer is not in the vector. Can you find the missing integer?

- Write a function with one input (the vector) and one output (the missing integer).

Examples-for/while loop

```
function y=findmiss(x)
% x is the input, which
is a N-1 vector with
distinct elements in 1 to
N.
% y is the output, that
is not in x.
indicator=zeros(1,N);
for i=1:N-1
    indicator(x(i))=1;
end
for i=1:N
    if indicator(i)==0
        y=i;
    end
end
```

Save as findmiss.m

```
% call function findmiss
% simulate the input x
x=randperm(N);
id=ceil(rand(1)*N);
z=x(id); % z is the
missing integer
% take z out of x
x=[x(1:id-1),
x(id+1:end)];
y=findmiss(x)
```

Workspace

- Matlab remembers old commands
- **And** variables as well
- Each Function maintains its own scope
- The keyword `clear` removes all variables from workspace
- The keyword `who` lists the variables

File I/O

- Matlab has a native file format to save and load workspaces. Use keywords `load` and `save`.
- In addition MATLAB knows a large number of popular formats. Type "`help fileformats`" for a listing.
- In addition MATLAB supports 'C' style low level file I/O. Type "`help fprintf`" for more information.

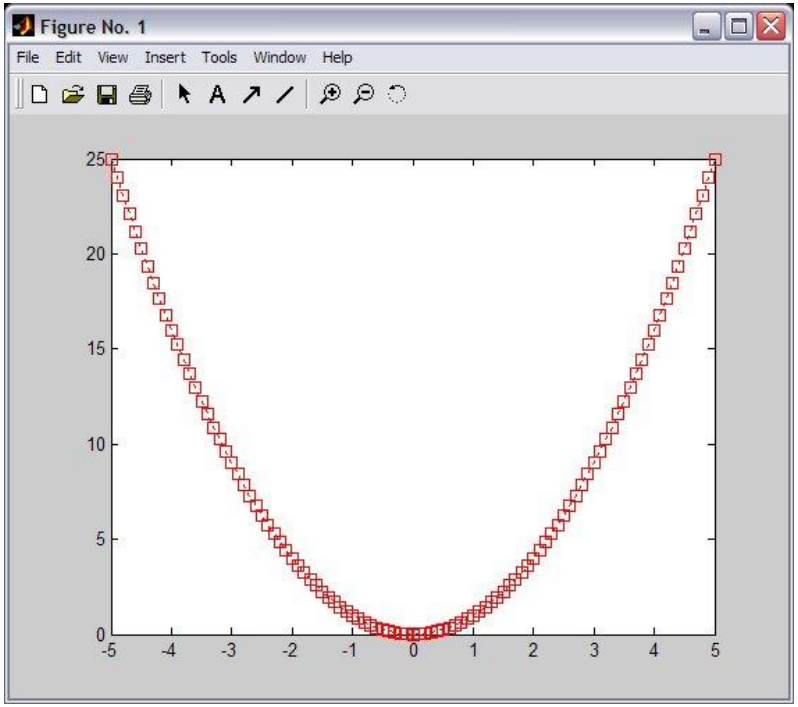
Graphics - 2D Plots

```
plot(xdata, ydata, 'marker_style');
```

For example:

```
>> x=-5:0.1:5;
>> sqr=x.^2;
>> pl1=plot(x, sqr, 'r:s');
```

Gives:

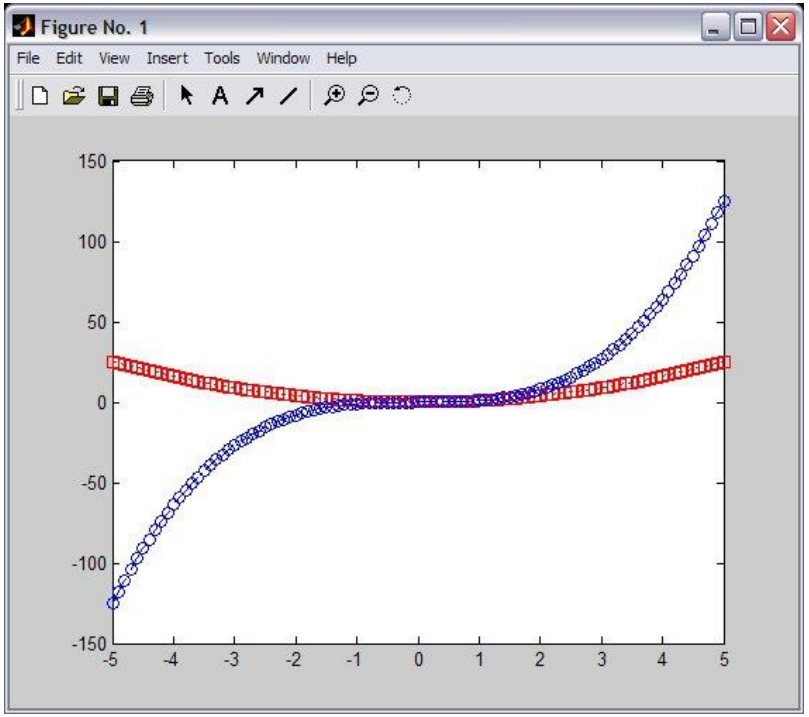


Graphics - Overlay Plots

Use `hold on` for overlaying graphs

So the following: Gives:

```
>> hold on;
>> cub=x.^3;
>> pl2=plot(x, cub,'b-o');
```

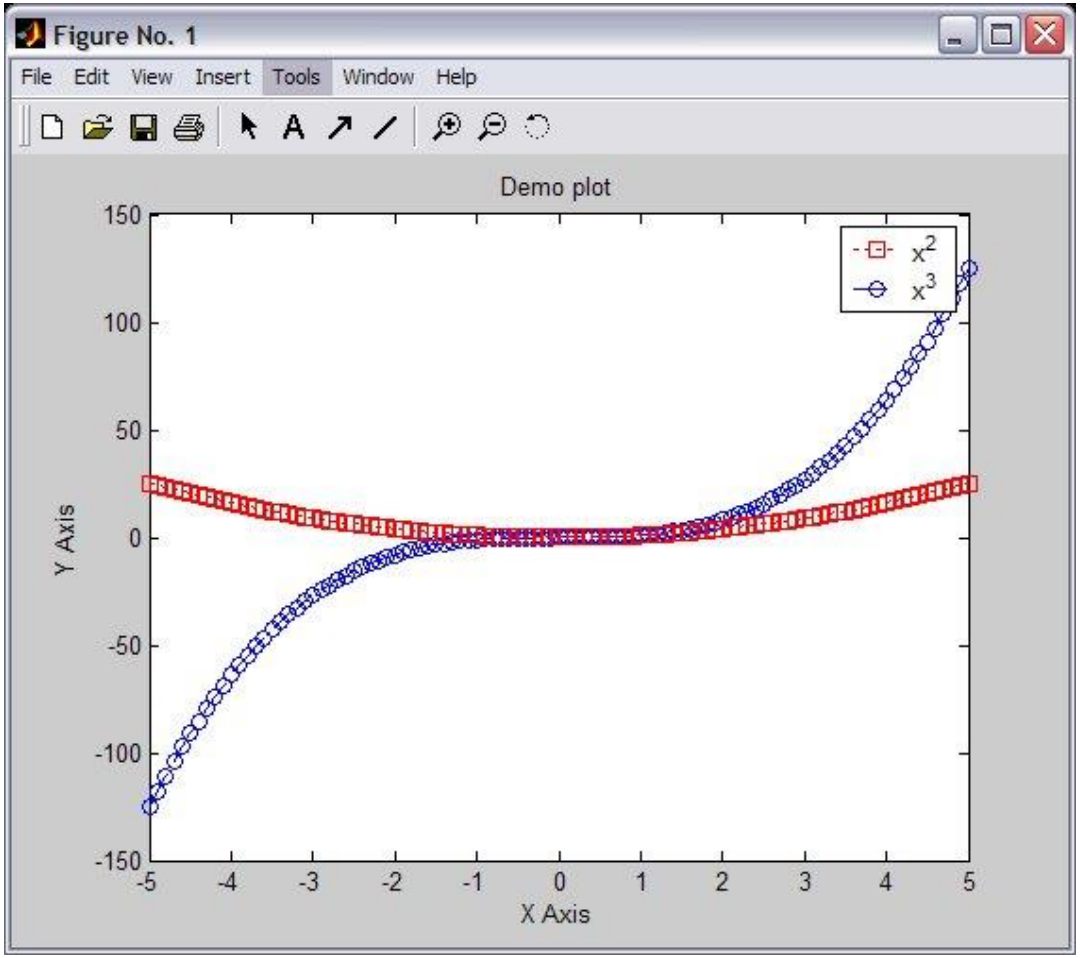


Graphics - Annotation

Use `title, xlabel, ylabel` and `legend` for annotation

```
>> title('Demo plot');
>> xlabel('X Axis');
>> ylabel('Y Axis');
>> legend([pl1, pl2], 'x^2', 'x^3');
```

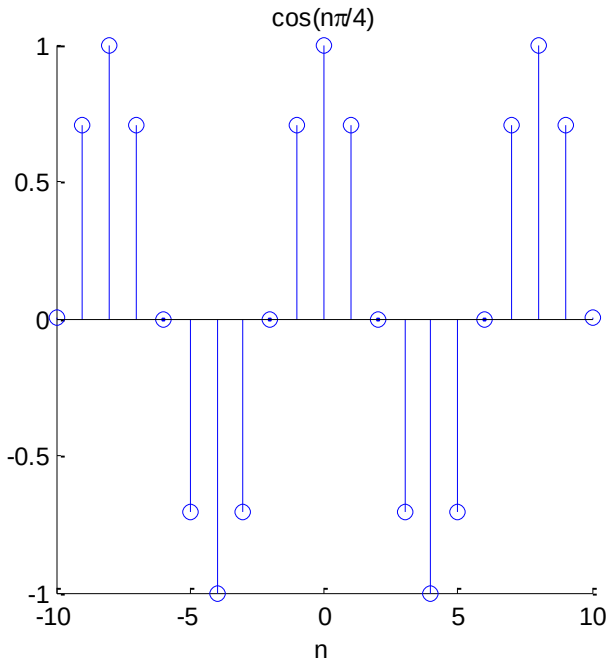
Graphics - Annotation



Graphics-Stem()

- `stem()` is to plot discrete sequence data
- The usage of `stem()` is very similar to `plot()`

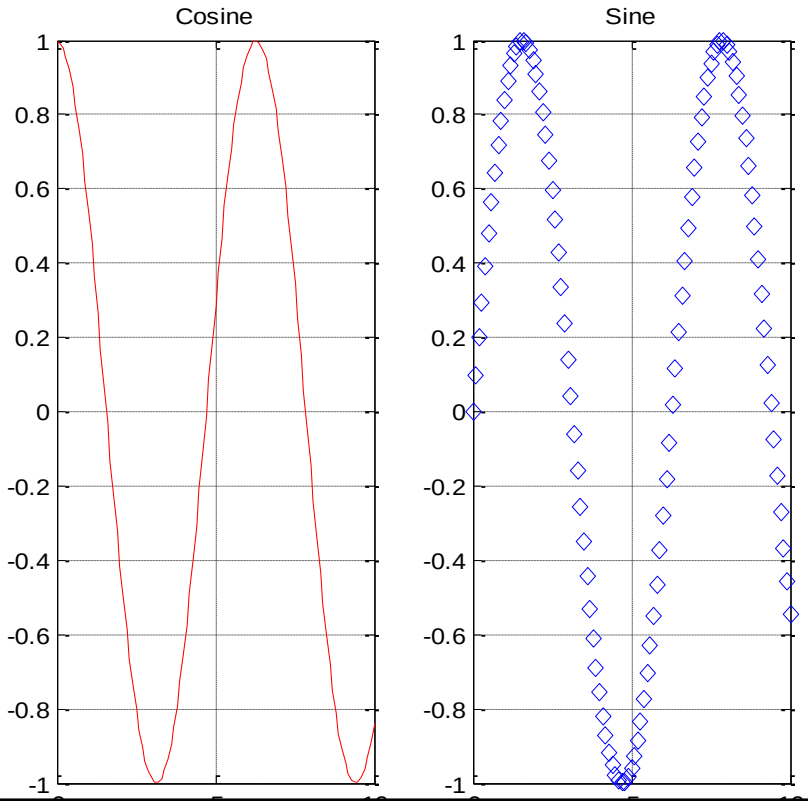
```
>> n=-10:10;
>> f=stem(n,cos(n*pi/4))
>> title('cos(n\pi/4)')
>> xlabel('n')
```



Subplots

- Use subplots to divide a plotting window into several panes.

```
>> x=0:0.1:10;
>> f=figure;
>> f1=subplot(1,2,1);
>> plot(x,cos(x),'r');
>> grid on;
>> title('Cosine')
>> f2=subplot(1,2,2);
>> plot(x,sin(x),'d');
>> grid on;
>> title('Sine');
```



Save plots

- Use `saveas(h,'filename.ext')` to save a figure to a file.

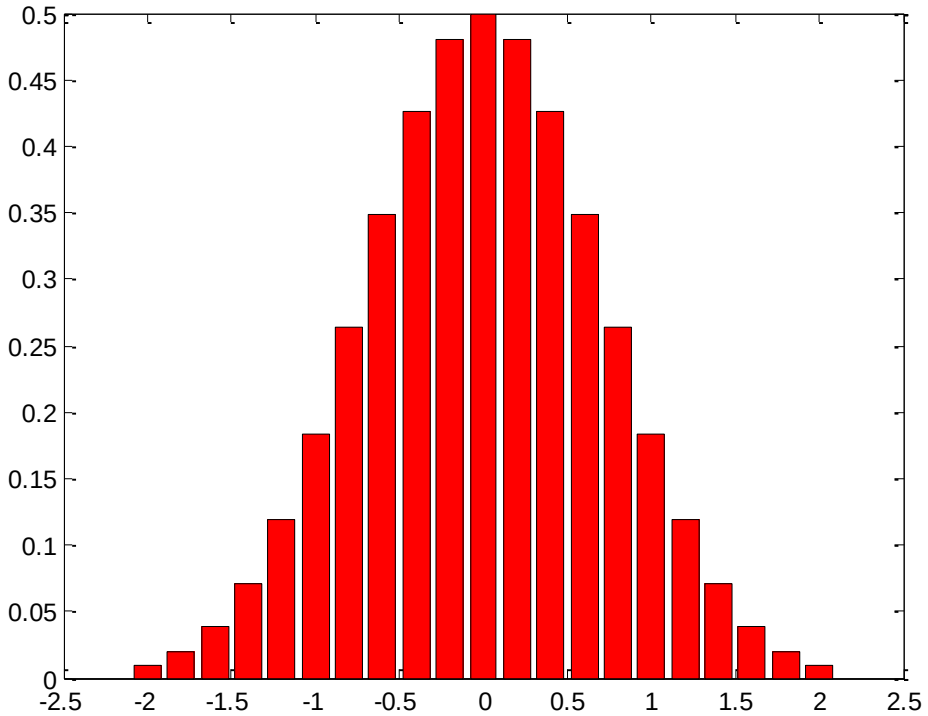
```
>> f=figure;
>> x=-5:0.1:5;
>> h=plot(x,cos(2*x+pi/3));
>> title('Figure 1');
>> xlabel('x');
>> saveas(h,'figure1.fig')
>> saveas(h,'figure1.eps')
```

Useful extension types:

bmp: Windows bitmap
emf: Enhanced metafile
eps: EPS Level 1
fig: MATLAB figure
jpg: JPEG image
m: MATLAB M-file
tif: TIFF image, compressed

Bar Graphs

- bar(Y) draws one bar for each element in Y
- bar(x, Y) draws a bar for each element in Y at locations specified in x, where x is a monotonically increasing vector defining the x-axis intervals for the vertical bars.

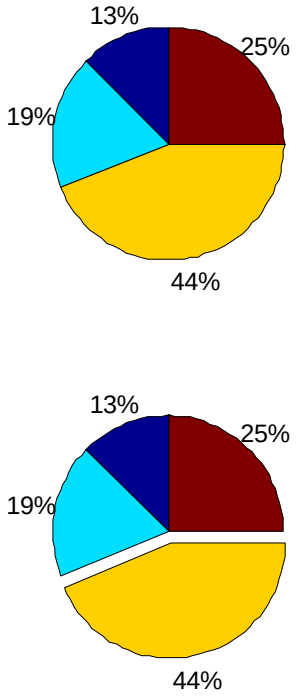



```
>> x=-2:0.2:2;
>> y= exp(-x.*x)/2;
>> bar(x,y,'r')
```

Pie Charts

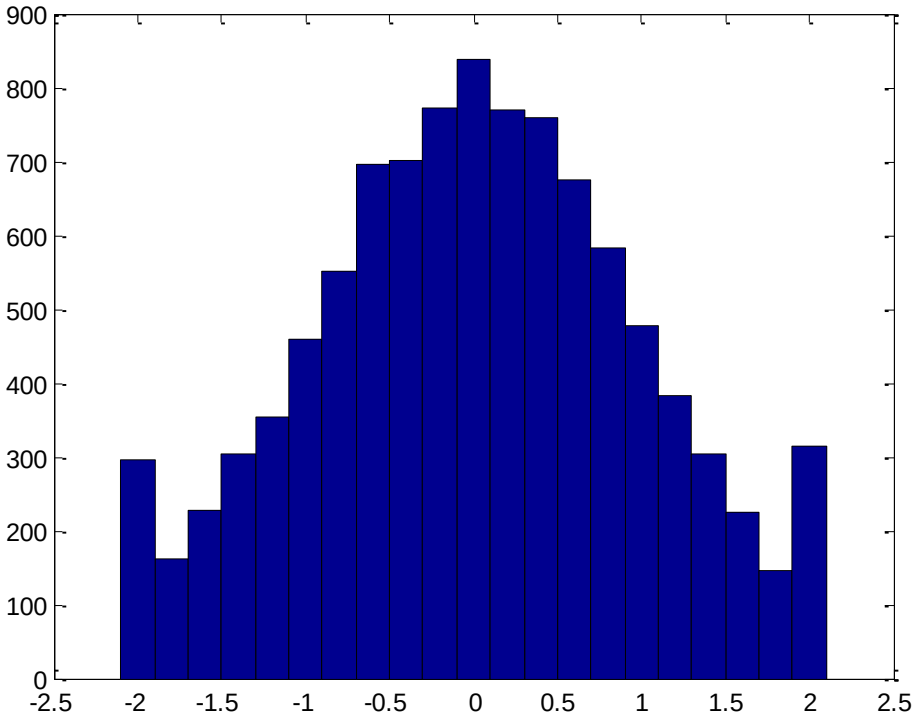
- pie(X) draws a pie chart using the data in X. Each element in X is represented as a slice in the pie chart.
- pie(X,explode) offsets a slice from the pie

```
>> x=[2 3 7 4];
>> subplot(2,1,1);
>> pie(x)
>> subplot(2,1,2);
>> explode=[0 0 1 0]
>> pie(x,explode)
```



Histogram

- `n = hist(Y,nbins)` uses nbins number of equally spaced bins, and returns the number of elements in each container.
- `n = hist(Y,x)` where x is a vector, returns the distribution of Y among length(x) bins with centers specified by x.




```
x = -2:0.2:2;
y =randn(10000,1);
hist(y,x);
```

Convolution

- conv()

  C = CONV(A, B) convolves vectors A and B. The resulting vector is LENGTH(A)+LENGTH(B)-1

Convolution examples

$$x[n] = a^n u[n], a = 1/2$$

- Find y[n] when $h_1[n] = \delta[n]$
- Find y[n] when $h_2[n]$ is a rectangular function

$$h_2[n] = \begin{cases} 1 & -5 \le n \le 5 \\ 0 & \text{otherwise} \end{cases}$$

Convolution – examples

Define x[n]:

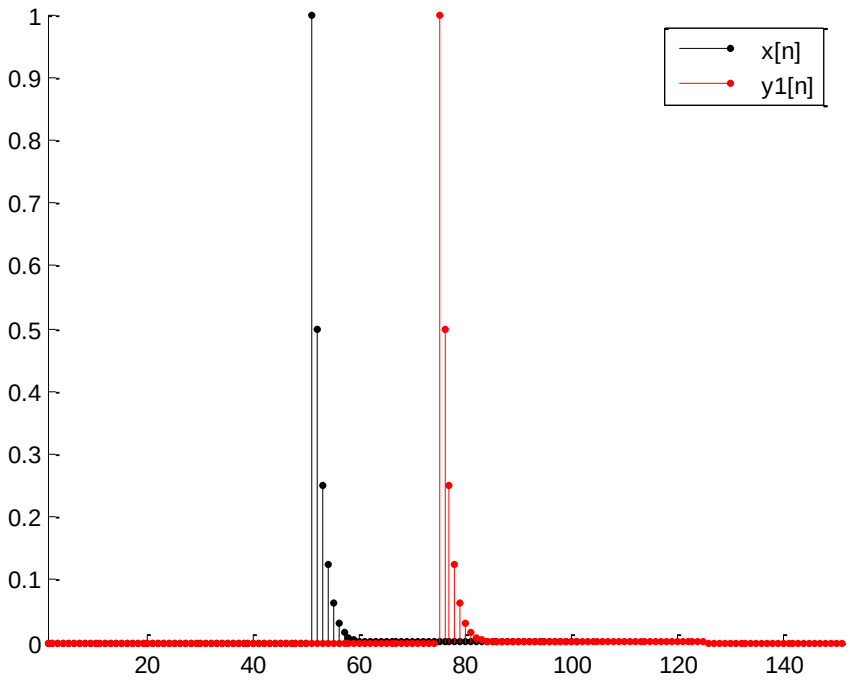
```
n=-50:50;
x1=zeros(1,50);
n2=0:50;
x2=(0.5).^n2;
x=[x1,x2];
```

Define h1[n]

```
h1=zeros(1,51);
h1(25)=1;
```

Now convolve:

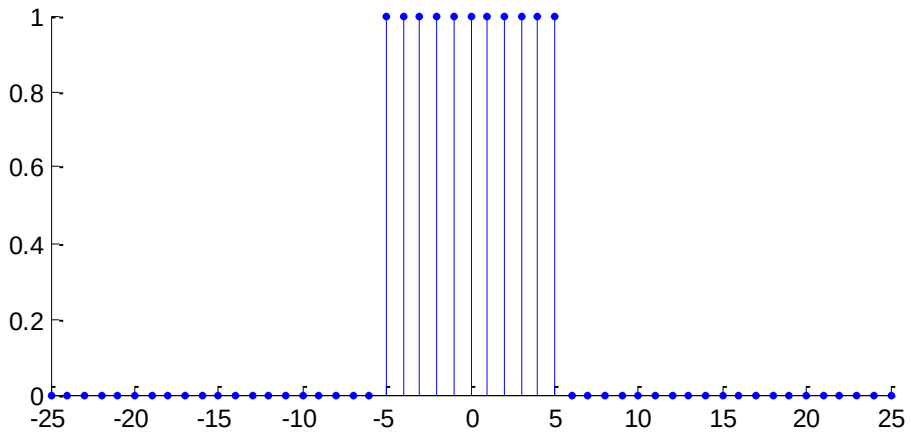
```
y1=conv(x,h1);
figure;stem(x, 'k.');
hold on; stem(y1,'r.');
axis tight;
legend('x[n]','y1[n]')
```



Convolution – examples

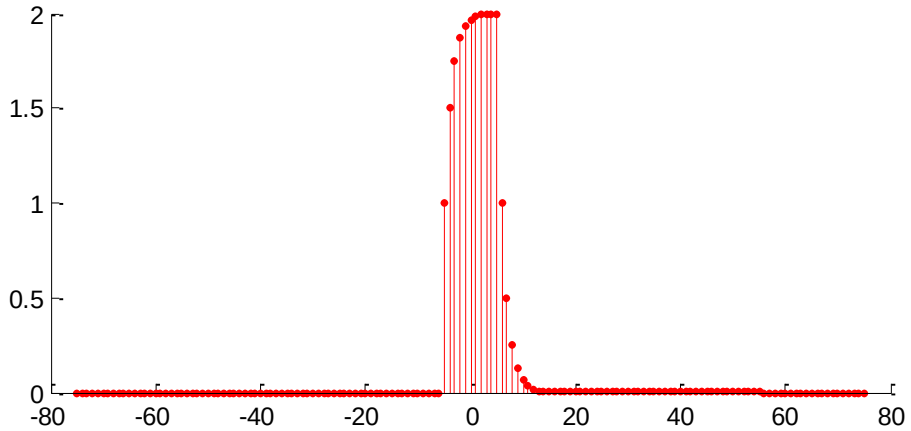
Define h2[n]

```
h2=zeros(1,51);
h2(26-5:26+5)=1;
```



Now convolve:

```
y2=conv(x,h2);
figure; subplot(2,1,1)
stem(-25:25,h2,'.');
subplot(2,1,2);
stem(-25-50:25+50,y2,'r.');
```



Practice Problems

- Plot the following signals in linear scale

$$x(t) = \sin(3t) \qquad -5 < t < 5$$

$$y(t) = e^{2t+3} \qquad 0 < t < 5$$

- Plot the following signals, use log scale for y-axis

$$x(t) = e^{t+2}(2t+1) \quad 0 < t < 10$$

- Plot the real part and imaginary part of the following signal

$$x(t) = e^{0.5t + j(t+\pi/3)} \qquad 0 < t < 10$$

- For the signal in previous question, plot its phase and magnitude

- Use conv() to calculate and plot $y[n] = x[n] * h[n]$

$$x[n] = \begin{cases} 1 & |n| < 5 \\ 0 & \text{otherwise} \end{cases} \qquad h[n] = \begin{cases} 1 & |n| < 5 \\ 0 & \text{otherwise} \end{cases}$$

Questions?

– Arley Ristar – arrr2@cin.ufpe.br