Matlab

Acknowledgement: many slides in this lecture were downloaded from various sources in the internet

Vad är Matlab?

MATrix LABoratory

is a high-performance language for technical computing.

is a dynamic, interpreted, environment for matrix/vector analysis

1. Interactive system

   Ideal environment for programming and analysing discrete (indexed) signals and systems

2. Programming language

   User can build programs (in .m files or at command line) C/Java-like syntax

Vad är Matlab?

It integrates computation, visualization, and programming in an easy-to-use environment. Typical uses include:

- Math and computation
- Algorithm development
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics

MATLAB is an interactive system whose basic data element is an array *that does not require dimensioning*. This allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar non-interactive language such as C or Fortran.

Vad är Matlab?

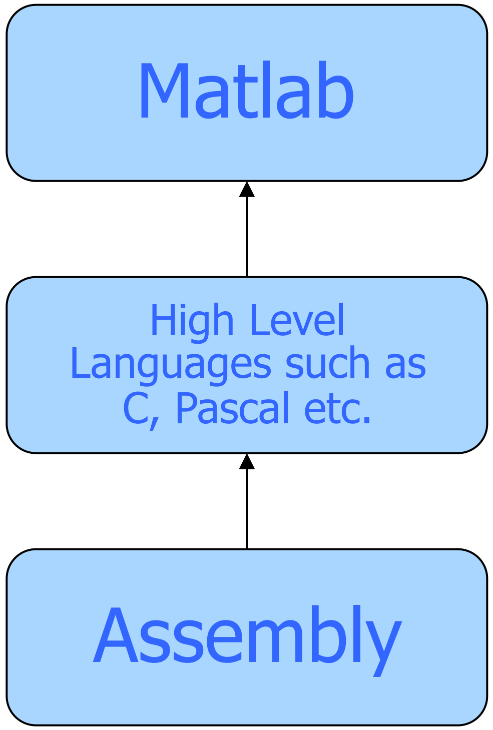
Five parts of Matlab

- The MATLAB language
  - High-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features
- The MATLAB working environment
  - Facilities for managing the variables and importing and exporting data
  - Tools for developing, managing, debugging, and profiling M-files
- Handle Graphics
  - Two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics
  - Graphical User Interface functions
- The MATLAB mathematical function library
- The MATLAB Application Program Interface (API)
  - Allows you to write C and Fortran programs that interact with MATLAB

Vad är Matlab?

Matlab is basically a high level language which has many specialized toolboxes for making things easier for us

How high?



Varför Matlab?

Strengths of MATLAB

- relatively easy to learn
- MATLAB code is optimized to be quick when performing matrix operations
- it may behave like a calculator or as a programming language
- is interpreted, errors are easier to fix
- (Although primarily procedural, MATLAB does have some object-oriented elements)

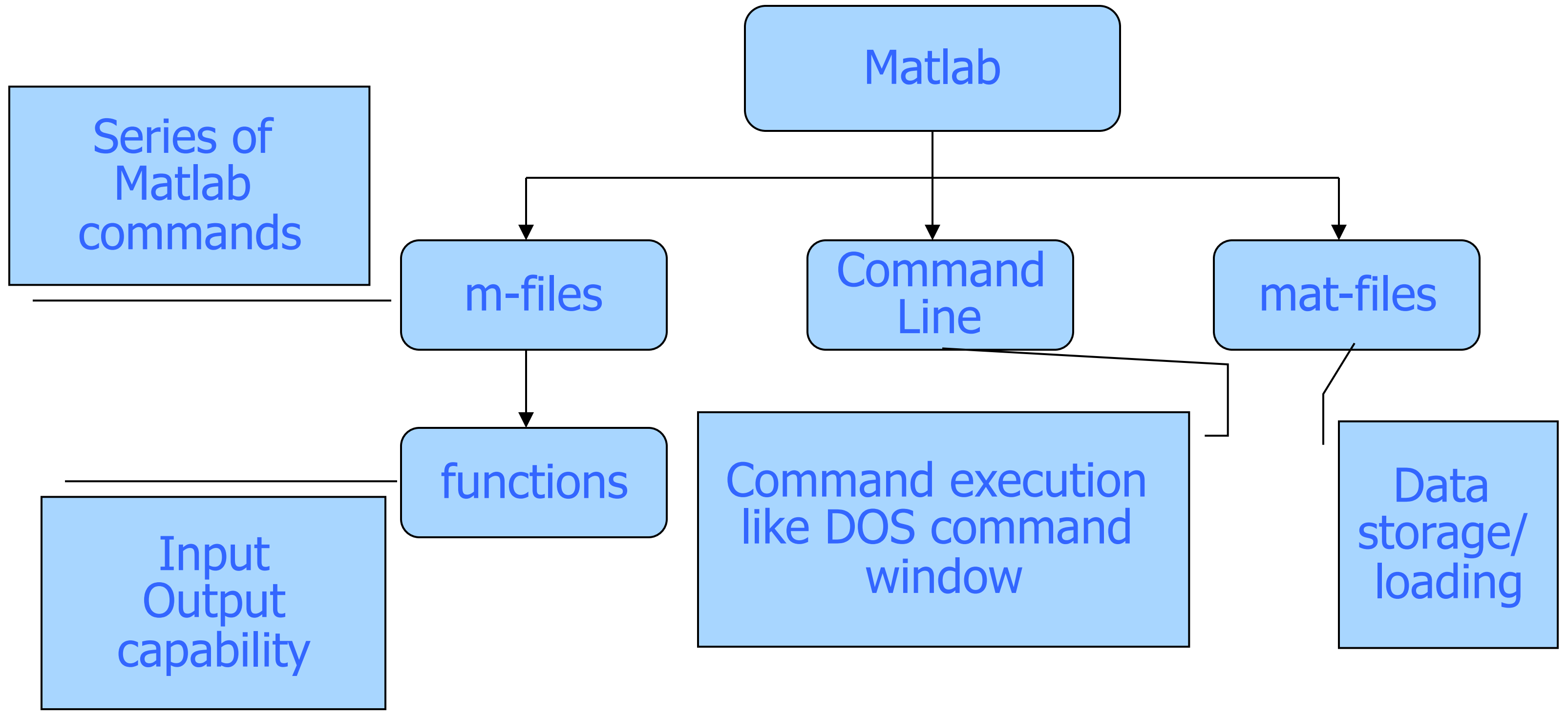
Weaknesses of MATLAB

- MATLAB is NOT a general purpose programming language
- MATLAB is an interpreted language (making it for the most part slower than a compiled language such as C++)
- MATLAB is designed for scientific computation and is not suitable for some things (such as parsing text)

Vad är intressant för oss?

Matlab is too broad for our purposes in this course.

The features we are going to use is:



Variables

- No need for types. i.e.,

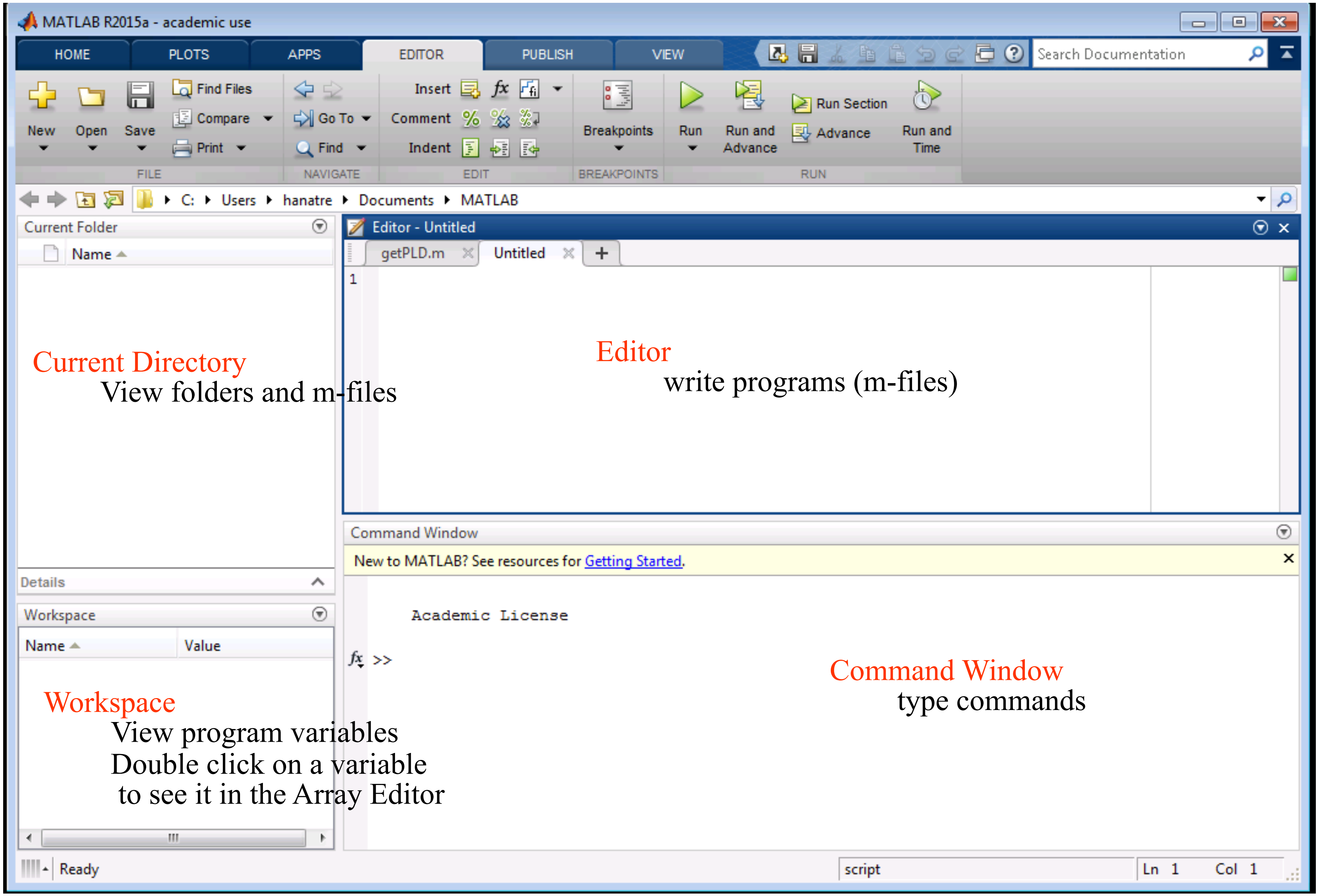
~~int a;~~
~~double b;~~
~~float c;~~

- All variables are created with double precision unless specified and they are matrices.

```
Example:
>>x=5;
>>x1=2;
```

- After these statements, the variables are 1x1 matrices with double precision

Matlab screen



New to MATLAB? See resources for Getting Started.

```
    Academic License

>>
```

Current Directory
View folders and m-files

Editor
write programs (m-files)

Workspace
View program variables
Double click on a variable
to see it in the Array Editor

Command Window
type commands

Matlab Help!

Lots of help available

- Type `help` in the command window or `help operator`. This displays the help associated with the specified operator/function
- Type `lookfor topic` to search for Matlab commands that are related to the specified topic
- Type `helpdesk` in the command window or select help on the pull down menu. This allows you to access several, well-written programming tutorials.

- Google!!!

Learning to program (Matlab) is a “bums on seats” activity.

There is no substitute for practice, making mistakes, understanding concepts

Grundläggande Matlab operationer

Grundläggande Matlab operationer

```
>> % This is a comment, it starts with a “%”
>> y = 5*3 + 2^2;          % simple arithmetic
>> x = [1 2 4 5 6];        % create the vector “x”
>> x1 = x.^2;              % square each element in x
>> E = sum(abs(x).^2);     % Calculate signal energy
>> P = E/length(x);        % Calculate a signal power
>> x2 = x(1:3);            % Select first 3 elements in x
>> z = 1+i;                % Create a complex number
>> a = real(z);            % Pick off real part
>> b = imag(z);            % Pick off imaginary part
>> plot(x);                % Plot the vector as a signal
>> t = 0:0.1:100;          % Generate sampled time
>> x3=exp(-t).*cos(t);     % Generate a discrete signal
>> plot(t, x3, ‘x’);       % Plot points
```

Andra Matlab programmering struktur

Nästa förelsning

Loops

```
for i=1:100
    sum = sum+i;
end
```

Goes round the for loop 100 times, starting at i=1 and finishing at i=100

```
i=1;
while i<=100
    sum = sum+i;
    i = i+1;
end
```

Similar, but uses a while loop instead of a for loop

Decisions

```
if i==5
    a = i*2;
else
    a = i*4;
end
```

Executes whichever branch is appropriate depending on test

```
switch i
case 5
    a = i*2;
otherwise
    a = i*4;
end
```

Similar, but uses a switch

Viktigt…

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

The keyword `whos` lists the variables with dimensions etc

Array, Matrix

- a vector `x = [1 2 5 1]`

```
x =
        1    2    5    1
```

- a matrix `x = [1 2 3; 5 1 4; 3 2 -1]`

```
x =
        1      2      3
        5      1      4
        3      2     -1
```

- transpose `y = x’`

```
y =
             1
             2
             5
             1
```

Long Array, Matrix

- `t =1:10`

```
t =
     1    2    3    4   5   6    7   8    9    10
```

- `k =2:-0.5:-1`

```
k =
     2   1.5   1   0.5   0   -0.5   -1
```

- `B = [1:4; 5:8]`

```
x =
     1      2      3     4
     5      6      7     8
```

Generating Vectors from functions

- zeros(M,N) MxN matrix of zeros

```
x = zeros(1,3)
x =
   0      0      0
```

- ones(M,N) MxN matrix of ones

```
x = ones(1,3)
x =
   1      1      1
```

- rand(M,N) MxN matrix of uniformly distributed random numbers on (0,1)

```
x = rand(1,3)
x =
 0.9501  0.2311 0.6068
```

Matrix Index

- The matrix indices begin from 1 (not 0 (as in C))
- The matrix indices must be positive integer

Given:

```
A =

     3     5     3
     6     8     2
     2     7     3
```

```
>> A(6)

ans =

     7
```

```
>> A(3,2)

ans =

     7
```

```
>> A(2,:)

ans =

     6     8     2
```

```
>> A(1:2,2)

ans =

     5
     8
```

A(-2), A(0)

Error: ??? Subscript indices must either be real positive integers or logicals.

A(4,2)
Error: ??? Index exceeds matrix dimensions.

Concatenation of Matrices

-
  ```
  x = [1 2], y = [4 5], z=[ 0 0]

  A = [ x y]

      1   2   4   5

   B = [x ; y]

       1  2
       4  5
  ```

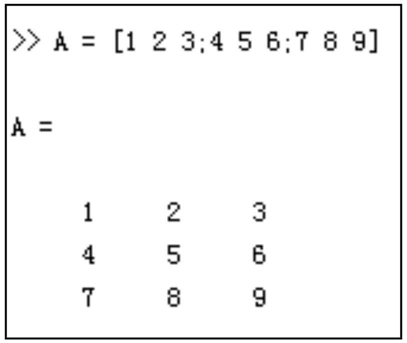
```
        C = [x y ;z]
Error:
??? Error using ==> vertcat CAT arguments dimensions are not consistent.
```

Operators (arithmetic)

+	addition
-	subtraction
*	multiplication
/	division
^	power
‘	complex conjugate transpose
\	inverse (solving systems of linear equations)

Matrices Operations

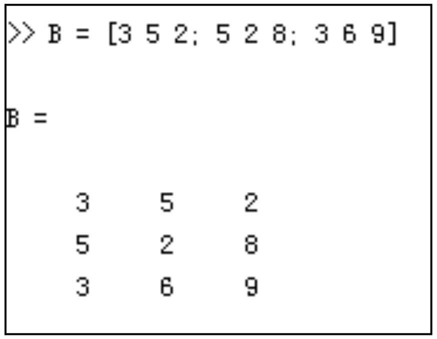
Given A and B:



```
>> A = [1 2 3;4 5 6;7 8 9]

A =

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

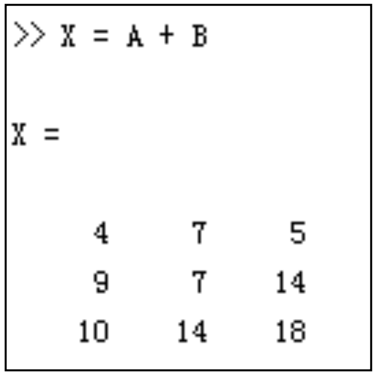


```
>> B = [3 5 2; 5 2 8; 3 6 9]

B =

     3     5     2
     5     2     8
     3     6     9
```

Addition

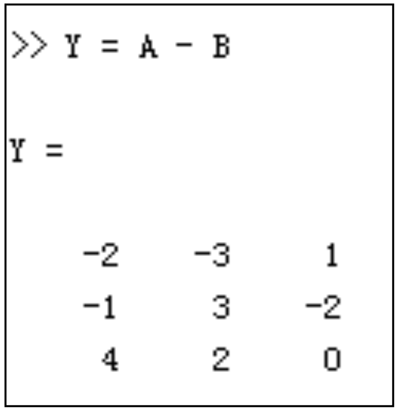


```
>> X = A + B

X =

     4     7     5
     9     7    14
    10    14    18
```

Subtraction

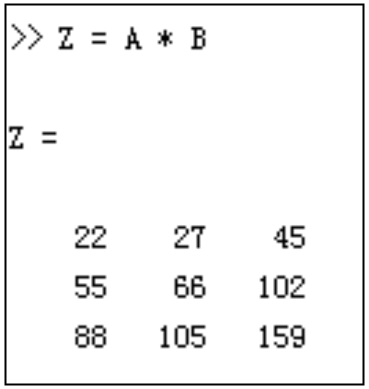


```
>> Y = A - B

Y =

    -2    -3     1
    -1     3    -2
     4     2     0
```

Product

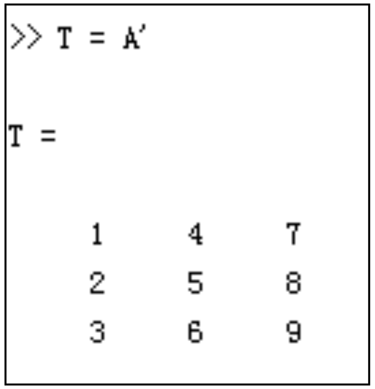


```
>> Z = A * B

Z =

    22    27    45
    55    66   102
    88   105   159
```

Transpose



```
>> T = A'

T =

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

Operators (Element by Element)

.* element-by-element multiplication

./ element-by-element division

.^ element-by-element power

The use of “.” – “Element” Operation

```
A = [1 2 3; 5 1 4; 3 2 1]
   A =
          1    2    3
          5    1    4
          3    2   -1
```

```
x = A(1,:)

x=
     1   2   3
```

```
y = A(:,3)

y=
    3  4  -1
```

```
b = x .* y

b=
     3  8 -3
```

```
c = x . / y

c=
  0.33   0.5   -3
```

```
d = x .^2

d=
     1    4    9
```

```
K= x^2
Erorr:
 ??? Error using ==> mpower  Matrix must be square.
B=x*y
Erorr:
??? Error using ==> mtimes Inner matrix dimensions must agree.
```

- Complex numbers
- real(); imag();abs()

sum()	zeros()
abs()	ones()
sqrt()	rand()
length()	()
size()	()
max()	()
min()	()

whos
who

Hur funkar egentligen sum() …

Basic Task: Plot the function sin(x) between $0 \leq x \leq 4\pi$

- Create an x-array of 100 samples between 0 and 4π.

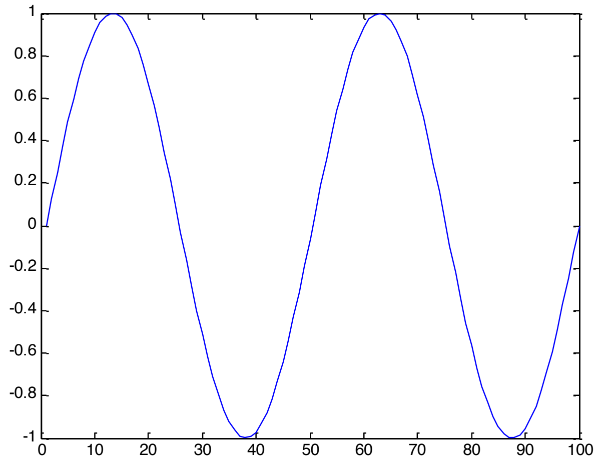
```
>>x=linspace(0,4*pi,100);
```

- Calculate sin(.) of the x-array

```
>>y=sin(x);
```

- Plot the y-array

```
>>plot(y)
```



Plot the function $e^{-x/3}\sin(x)$ between $0 \le x \le 4\pi$

- Create an x-array of 100 samples between 0 and 4π.

```
>>x=linspace(0,4*pi,100);
```

- Calculate sin(.) of the x-array

```
>>y=sin(x);
```

- Calculate $e^{-x/3}$ of the x-array

```
>>y1=exp(-x/3);
```

- Multiply the arrays y and y1

```
>>y2=y*y1;
```

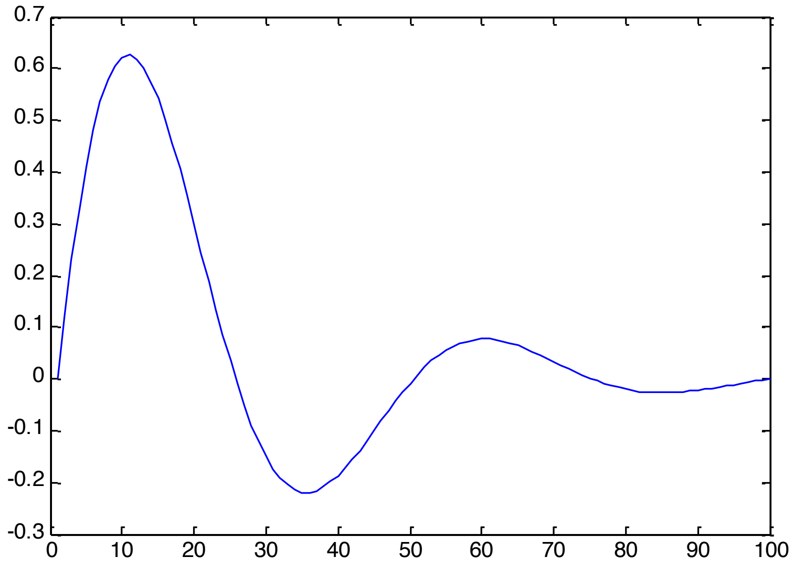
Plot the function $e^{-x/3}\sin(x)$ between $0\leq x\leq 4\pi$

- Multiply the arrays y and y1 correctly

```
>>y2=y.*y1;
```

- Plot the y2-array

```
>>plot(y2)
```



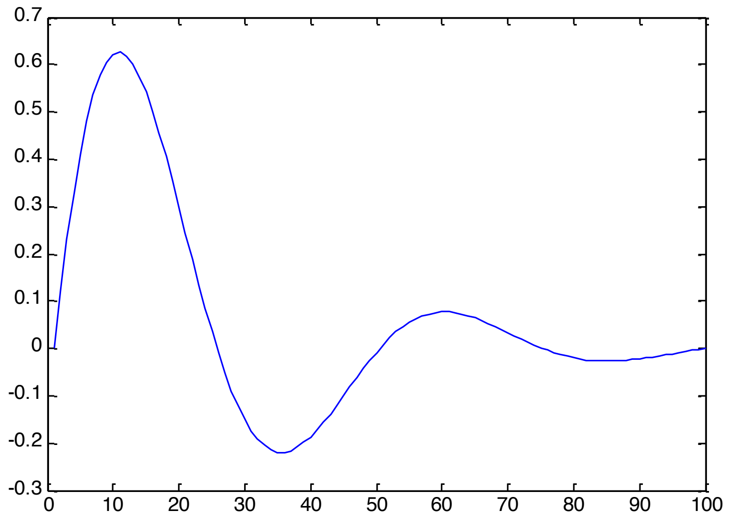
Display Facilities

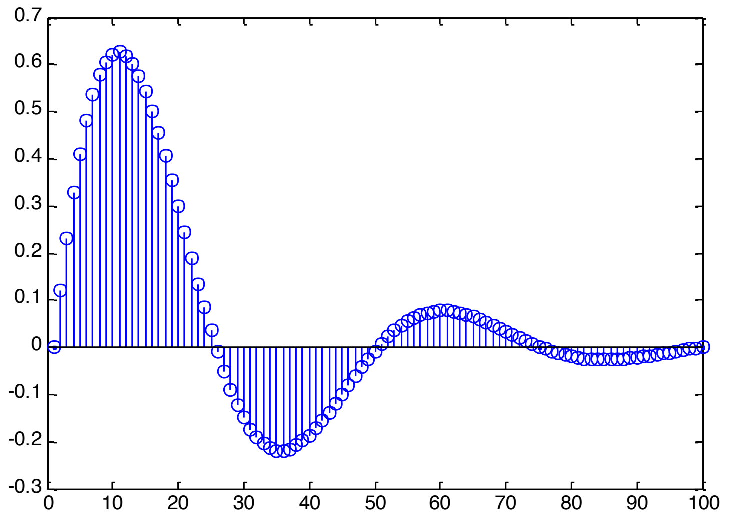
- plot(.)

```
Example:
>>x=linspace(0,4*pi,100);
>>y=sin(x);
>>plot(y)
>>plot(x,y)
```

- stem(.)

```
Example:
>>stem(y)
>>stem(x,y)
```





Display Facilities

- title(.)

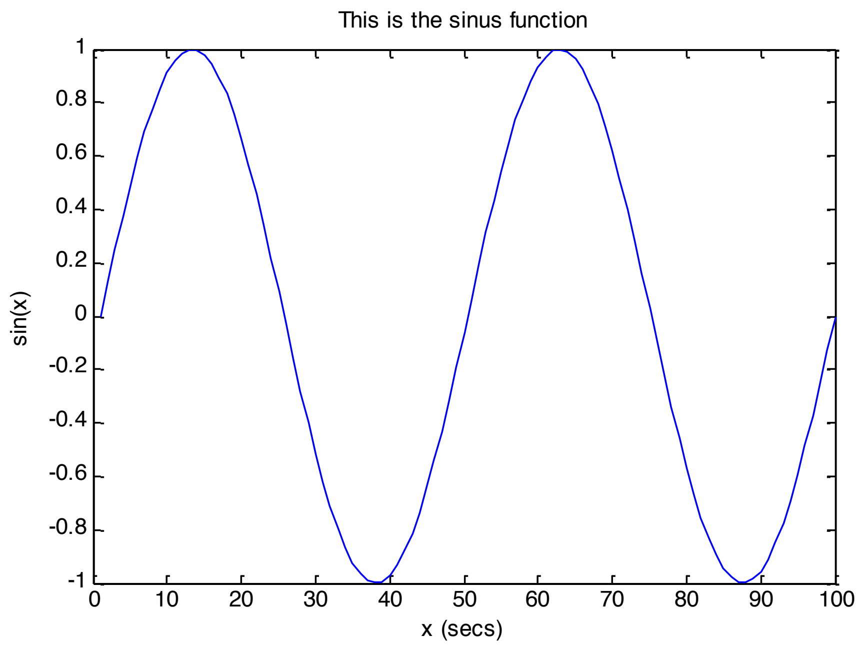
```
>>title( ‘This is the sinus function’ )
```

- xlabel(.)

```
>>xlabel( ‘x (secs)’ )
```

- ylabel(.)

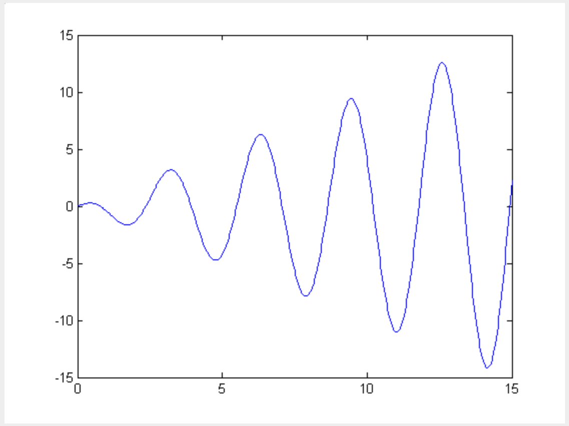
```
>>ylabel( ‘sin(x)’ )
```

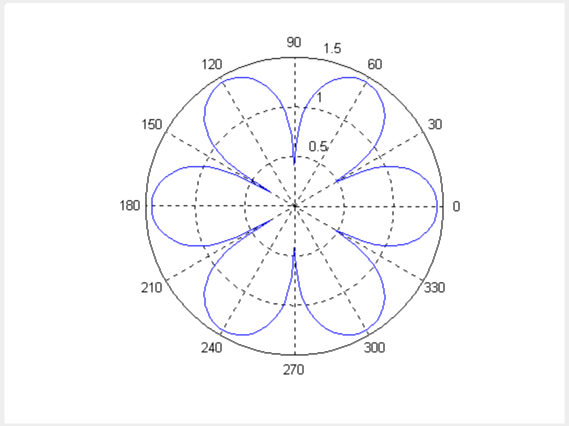


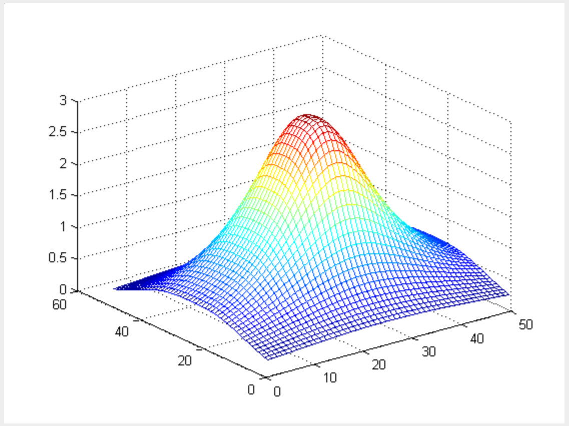
Plotting

MAKING X-Y PLOTS

MATLAB has many functions and commands that can be used to create various types of plots.

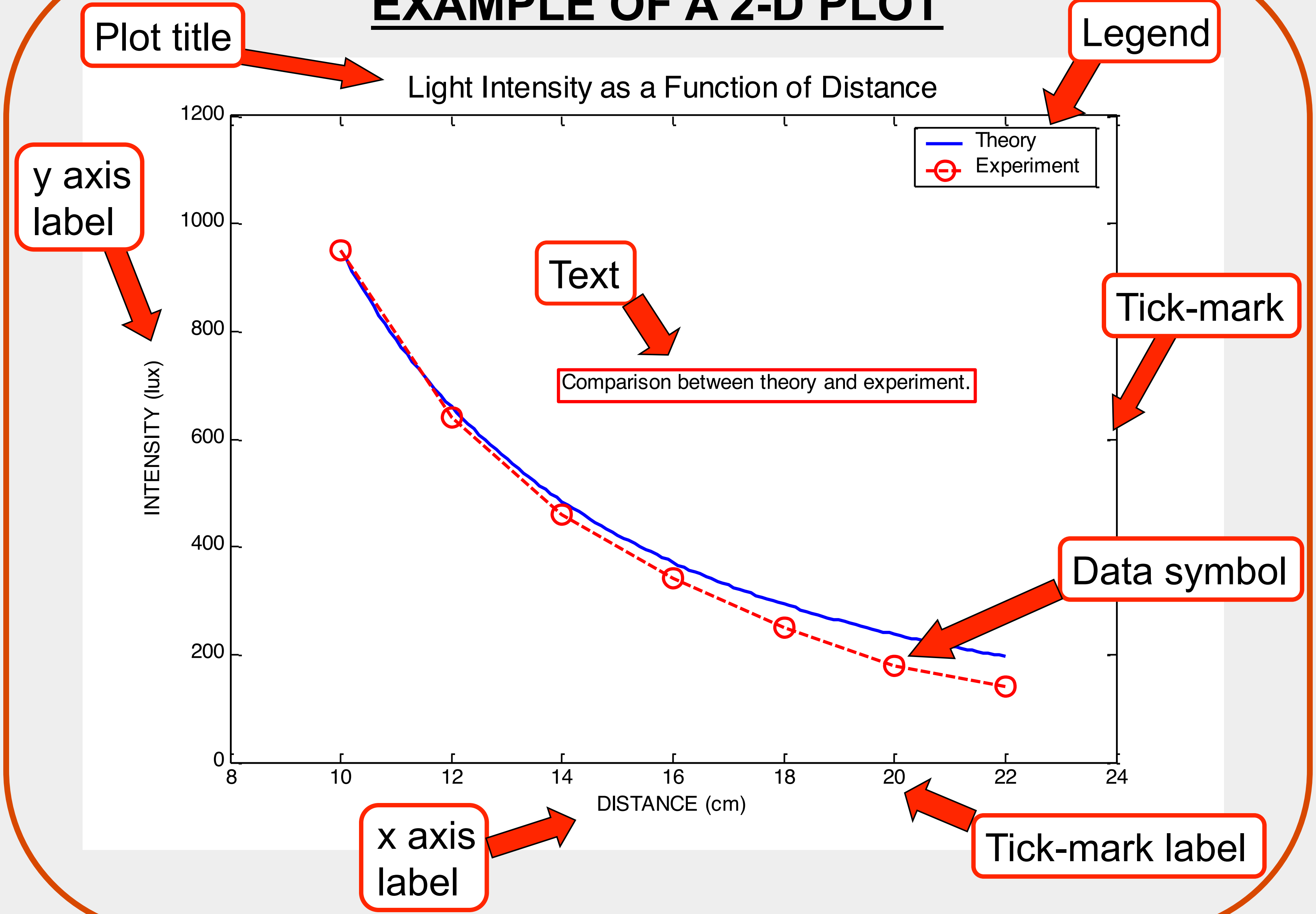






we will only create two dimensional x – y plots.

EXAMPLE OF A 2-D PLOT



TWO-DIMENSIONAL `plot()` COMMAND

The basic 2-D plot command is:

```
plot(x,y)
```

where **`x`** is a vector (one dimensional array), and **`y`** is a vector. Both vectors **must** have the same number of elements.

- The plot command creates a single curve with the **`x`** values on the abscissa (horizontal axis) and the **`y`** values on the ordinate (vertical axis).
- The curve is made from segments of lines that connect the points that are defined by the **`x`** and **`y`** coordinates of the elements in the two vectors.

CREATING THE X AND Y VECTORS

- If data is given, the information is entered as the elements of the vectors **`x`** and **`y`**.

- If the values of **`y`** are determined by a function from the values of **`x`**, than a vector **`x`** is created first, and then the values of **`y`** are calculated for each value of **`x`**. The spacing (difference) between the elements of **`x`** must be such that the plotted curve will show the details of the function.

PLOT OF GIVEN DATA

Given data:

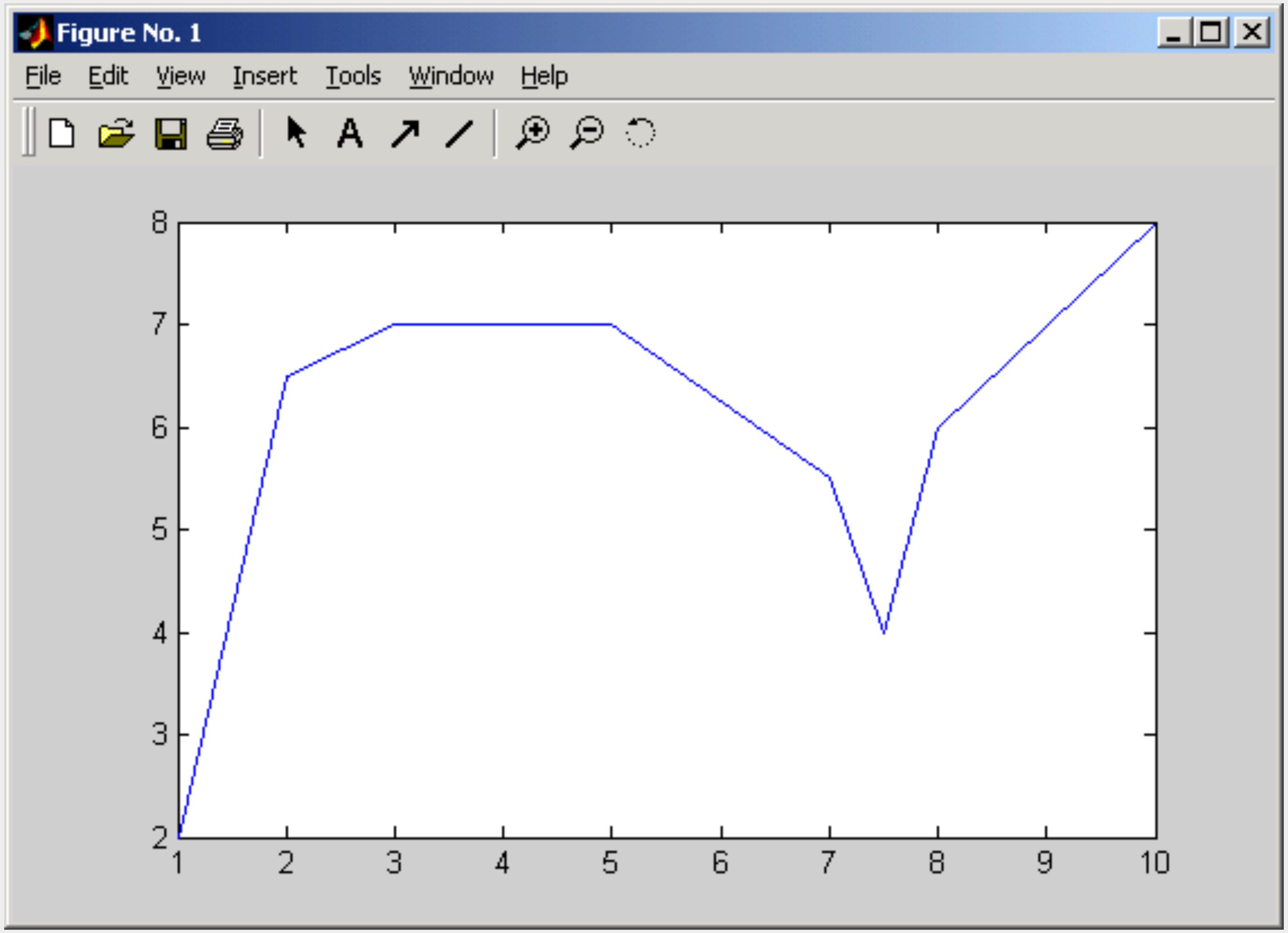
x	1	2	3	5	7	7.5	8	10
y	2	6.5	7	7	5.5	4	6	8

A plot can be created by the commands shown below. This can be done in the Command Window, or by writing and then running a script file.

```
>> x=[1 2 3 5 7 7.5 8 10];
>> y=[2 6.5 7 7 5.5 4 6 8];
>> plot(x,y)
```

Once the plot command is executed, the Figure Window opens with the following plot.

PLOT OF GIVEN DATA



LINE SPECIFIERS IN THE `plot()` COMMAND

Line specifiers can be added in the `plot` command to:

- Specify the style of the line.
- Specify the color of the line.
- Specify the type of the markers (if markers are desired).

```
plot(x,y,'line specifiers')
```

LINE SPECIFIERS IN THE `plot()` COMMAND

```
plot(x,y,‘line specifiers’)
```

Line Style	Specifier	Line Color	Specifier	Marker Type	Specifier
Solid	-	red	r	plus sign	+
dotted	:	green	g	circle	o
dashed	--	blue	b	asterisk	*
dash-dot	-.	Cyan	c	point	.
		magenta	m	square	s
		yellow	y	diamond	d
		black	k		

LINE SPECIFIERS IN THE `plot()` COMMAND

- The specifiers are typed inside the **`plot()`** command as strings.
- Within the string the specifiers can be typed in any order.
- The specifiers are optional. This means that none, one, two, or all the three can be included in a command.

EXAMPLES:

plot(x,y)	A solid blue line connects the points with no markers.
plot(x,y,' r')	A solid red line connects the points with no markers.
plot(x,y,' --y')	A yellow dashed line connects the points.
plot(x,y,' *')	The points are marked with * (no line between the points.)
plot(x,y,' g:d')	A green dotted line connects the points which are marked with diamond markers.

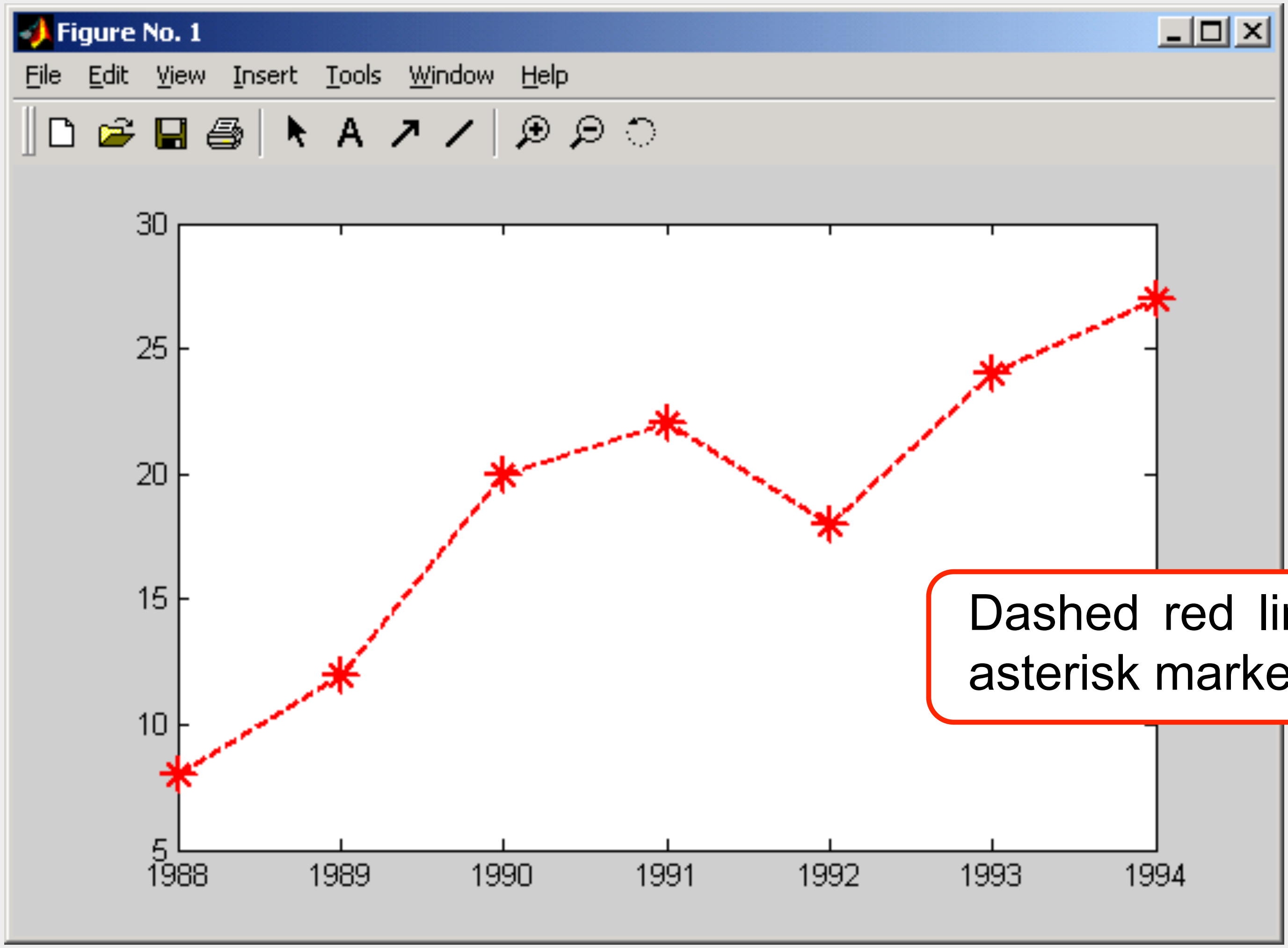
PLOT OF GIVEN DATA USING LINE SPECIFIERS IN THE `plot()` COMMAND

Year	1988	1989	1990	1991	1992	1993	1994
Sales (M)	127	130	136	145	158	178	211

```
>> year = [1988:1:1994];
>> sales = [127, 130, 136, 145, 158, 178, 211];
>> plot(year,sales,'--r*')
```

Line Specifiers: dashed red line and asterisk markers.

PLOT OF GIVEN DATA USING LINE SPECIFIERS IN THE `plot()` COMMAND



Dashed red line and asterisk markers.

CREATING A PLOT OF A FUNCTION

Consider: $y = 3.5^{-0.5x}\cos(6x) \quad \text{for} \quad -2 \le x \le 4$

A script file for plotting the function is:

```
% A script file that creates a plot of
% the function: 3.5^(-0.5x)*cos(6x)
x = [-2:0.01:4];
y = 3.5.^(-0.5*x).*cos(6*x);
plot(x,y)
```

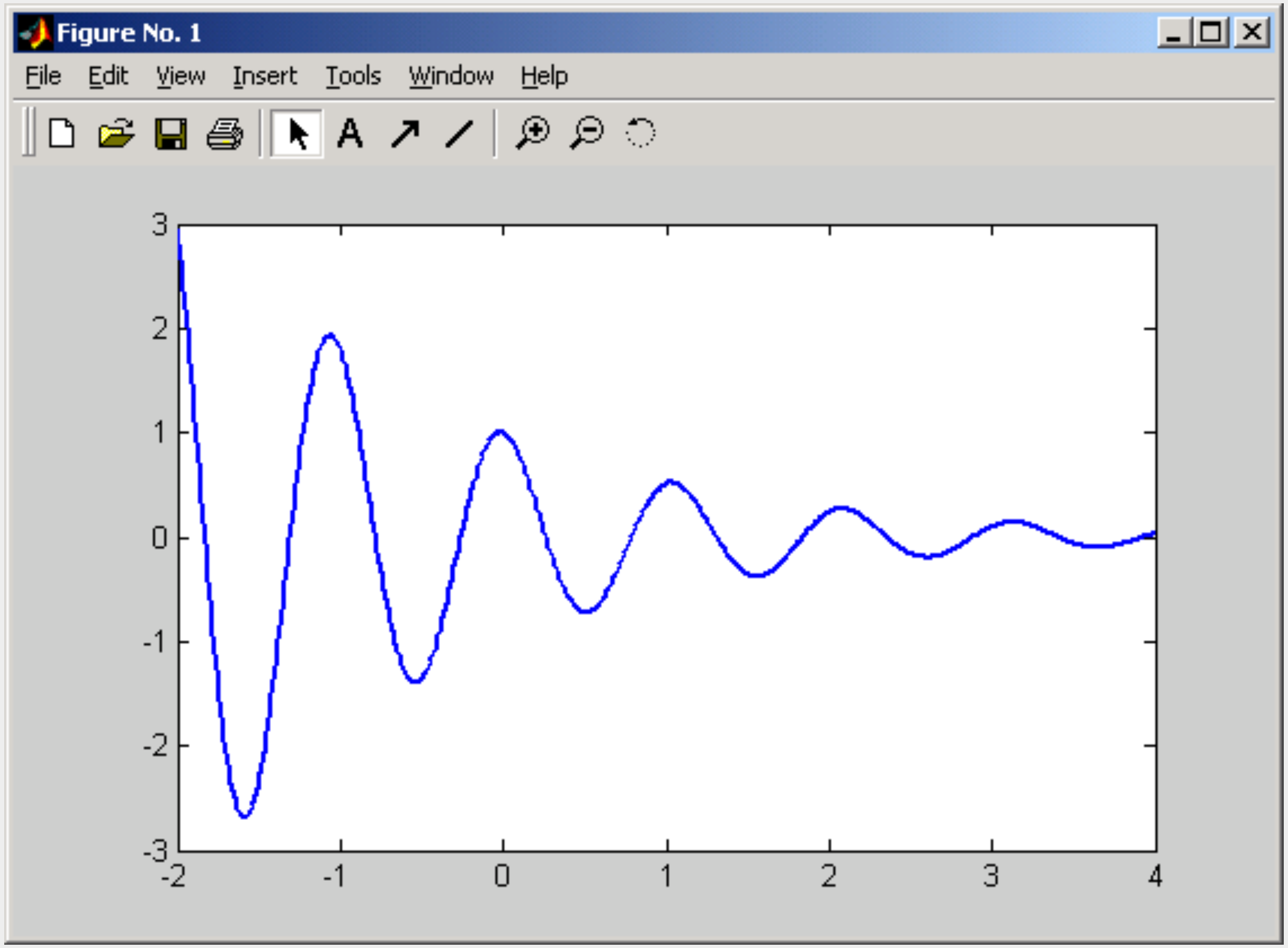
Creating a vector with spacing of 0.01.

Calculating a value of ***y*** for each ***x***.

Once the plot command is executed, the Figure Window opens with the following plot.

A PLOT OF A FUNCTION

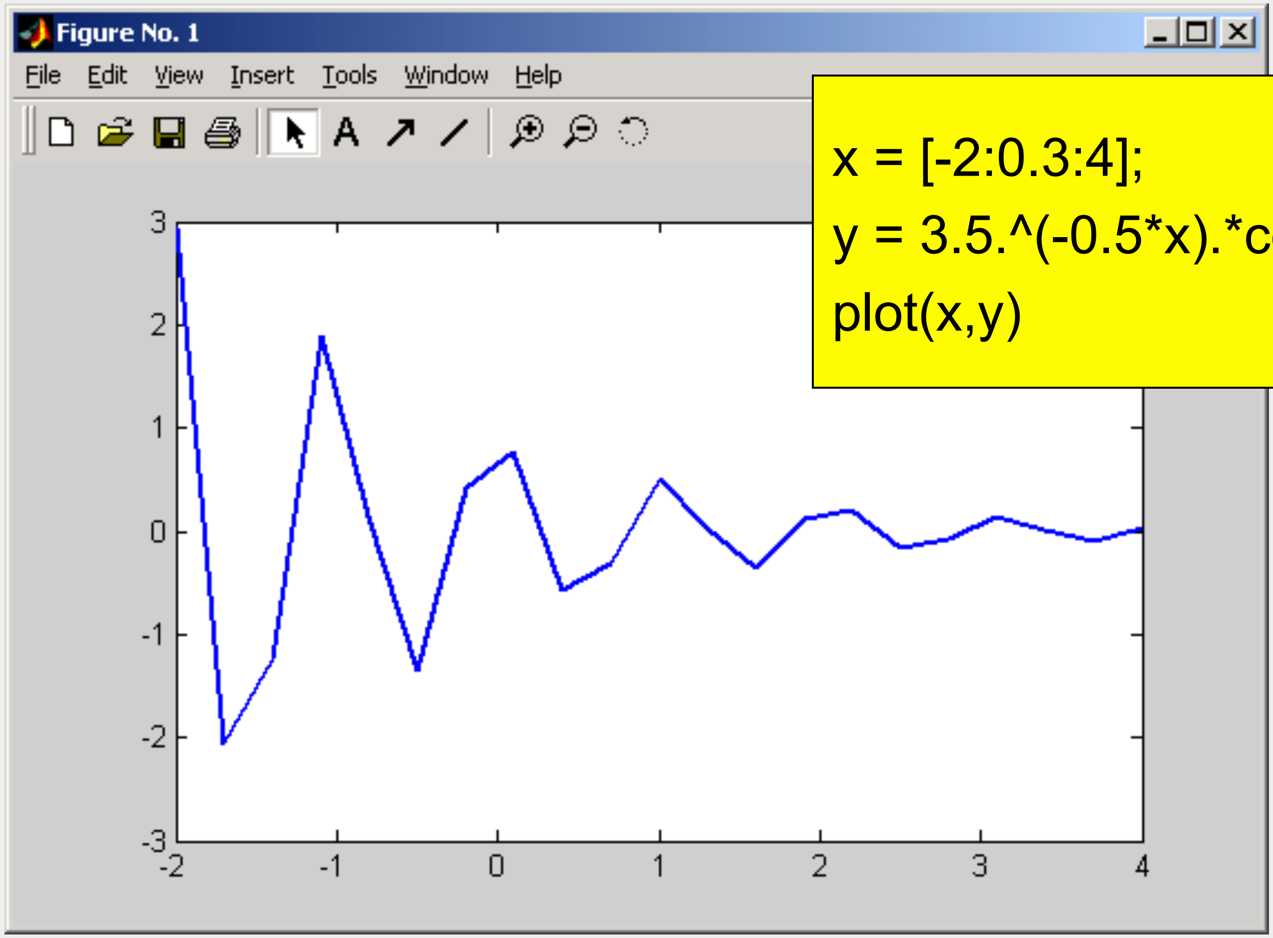
$$y = 3.5^{-0.5x}\cos(6x) \qquad \text{for} \quad -2 \le x \le 4$$



CREATING A PLOT OF A FUNCTION

If the vector **x** is created with large spacing, the graph is not accurate.

Below is the previous plot with spacing of 0.3.



```
x = [-2:0.3:4];
y = 3.5.^(-0.5*x).*cos(6*x);
plot(x,y)
```

THE `fplot` COMMAND

The **`fplot`** command can be used to plot a function with the form: $y = f(x)$

```
fplot('function',limits)
```

- The function is typed in as a string.
- The limits is a vector with the domain of x, and optionally with limits of the y axis:

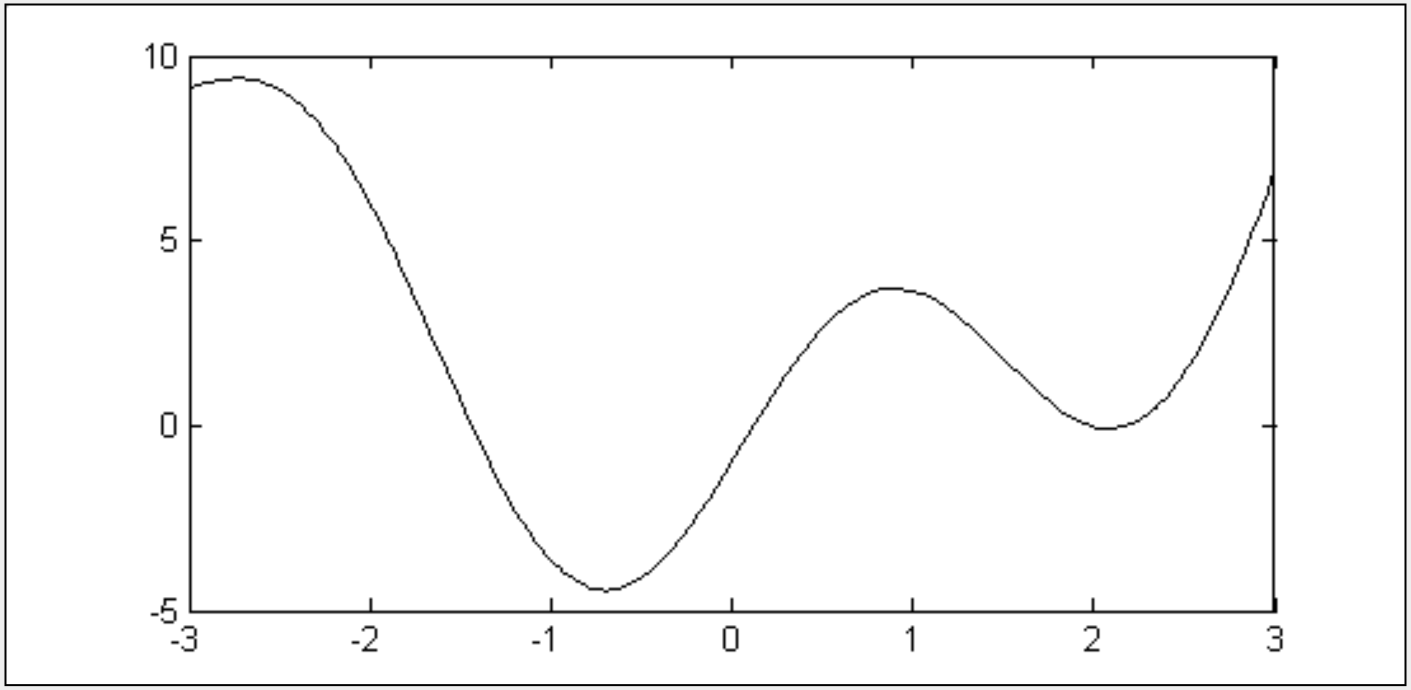
`[xmin,xmax]` or `[xmin,xmax,ymin,ymax]`

- Line specifiers can be added.

PLOT OF A FUNCTION WITH THE `fplot()` COMMAND

A plot of: $y = x^2 + 4\sin(2x) - 1$ for $-3 \le x \le 3$

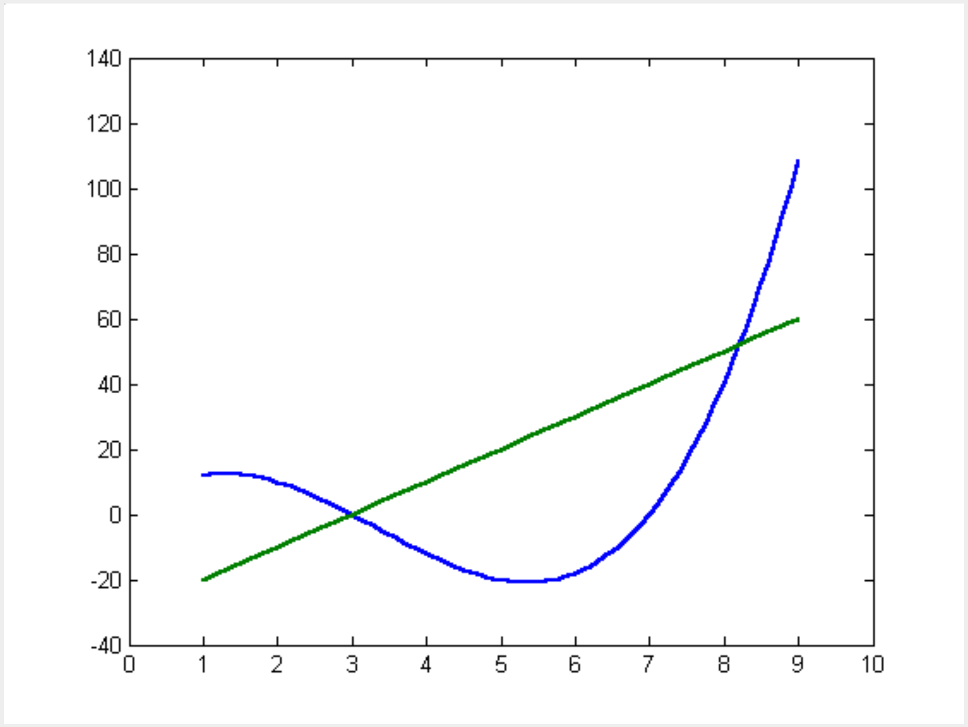
```
>> fplot('x^2 + 4 * sin(2*x) - 1', [-3 3])
```



PLOTTING MULTIPLE GRAPHS IN THE SAME PLOT

Plotting two (or more) graphs in one plot:

1. Using the **`plot`** command.
2. Using the **`hold on`**, **`hold off`** commands.



USING THE `plot()` COMMAND TO PLOT MULTIPLE GRAPHS IN THE SAME PLOT

```
plot(x,y,u,v,t,h)
```

Plots three graphs in the same plot:

`y` versus **`x`**, **`v`** versus **`u`**, and **`h`** versus **`t`**.

- By default, MATLAB makes the curves in different colors.
- Additional curves can be added.
- The curves can have a specific style by adding specifiers after each pair, for example:

```
plot(x,y,'-b',u,v,'—r',t,h,'g:')
```

USING THE `plot()` COMMAND TO PLOT MULTIPLE GRAPHS IN THE SAME PLOT

Plot of the function, $y = 3x^3 - 26x + 10$ and its first and second derivatives, for $-2 \le x \le 4$, all in the same plot.

```
x = [-2:0.01:4];
y = 3*x.^3-26*x+6;
yd = 9*x.^2-26;
ydd = 18*x;
plot(x,y,'-b',x,yd,'--r',x,ydd,':k')
```

vector **`x`** with the domain of the function.

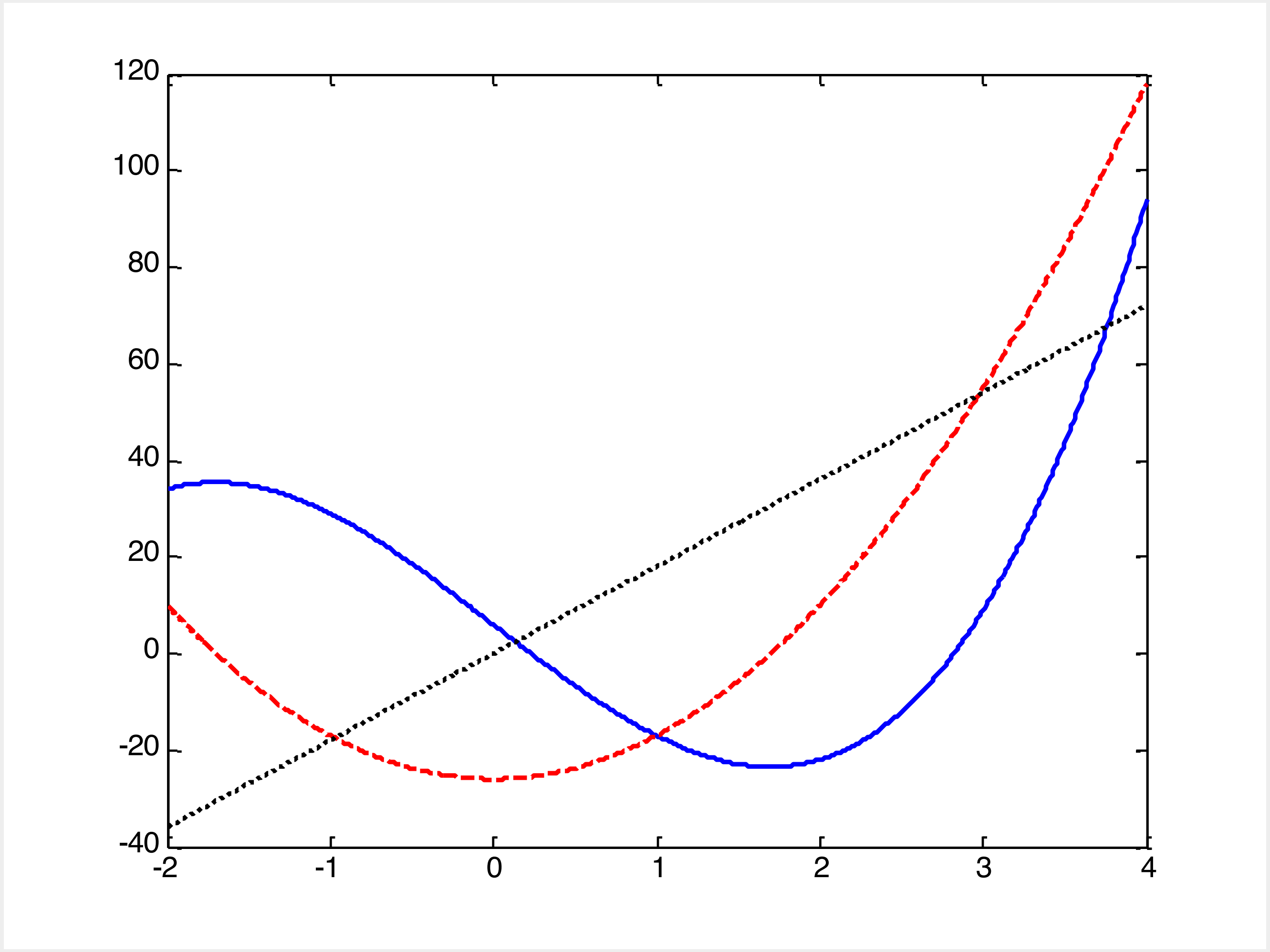
Vector **`y`** with the function value at each **`x`**.

Vector **`yd`** with values of the first derivative.

Vector **`ydd`** with values of the second derivative.

Create three graphs, **`y`** vs. **`x`** (solid blue line), **`yd`** vs. **`x`** (dashed red line), and **`ydd`** vs. **`x`** (dotted black line) in the same figure.

USING THE `plot()` COMMAND TO PLOT MULTIPLE GRAPHS IN THE SAME PLOT



USING THE `hold on`, `hold off`, COMMANDS TO PLOT MULTIPLE GRAPHS IN THE SAME PLOT

`hold on`	Holds the current plot and all axis properties so that subsequent plot commands add to the existing plot.
`hold off`	Returns to the default mode whereby plot commands erase the previous plots and reset all axis properties before drawing new plots.

This method is useful when all the information (vectors) used for the plotting is not available a the same time.

USING THE `hold on`, `hold off`, COMMANDS TO PLOT MULTIPLE GRAPHS IN THE SAME PLOT

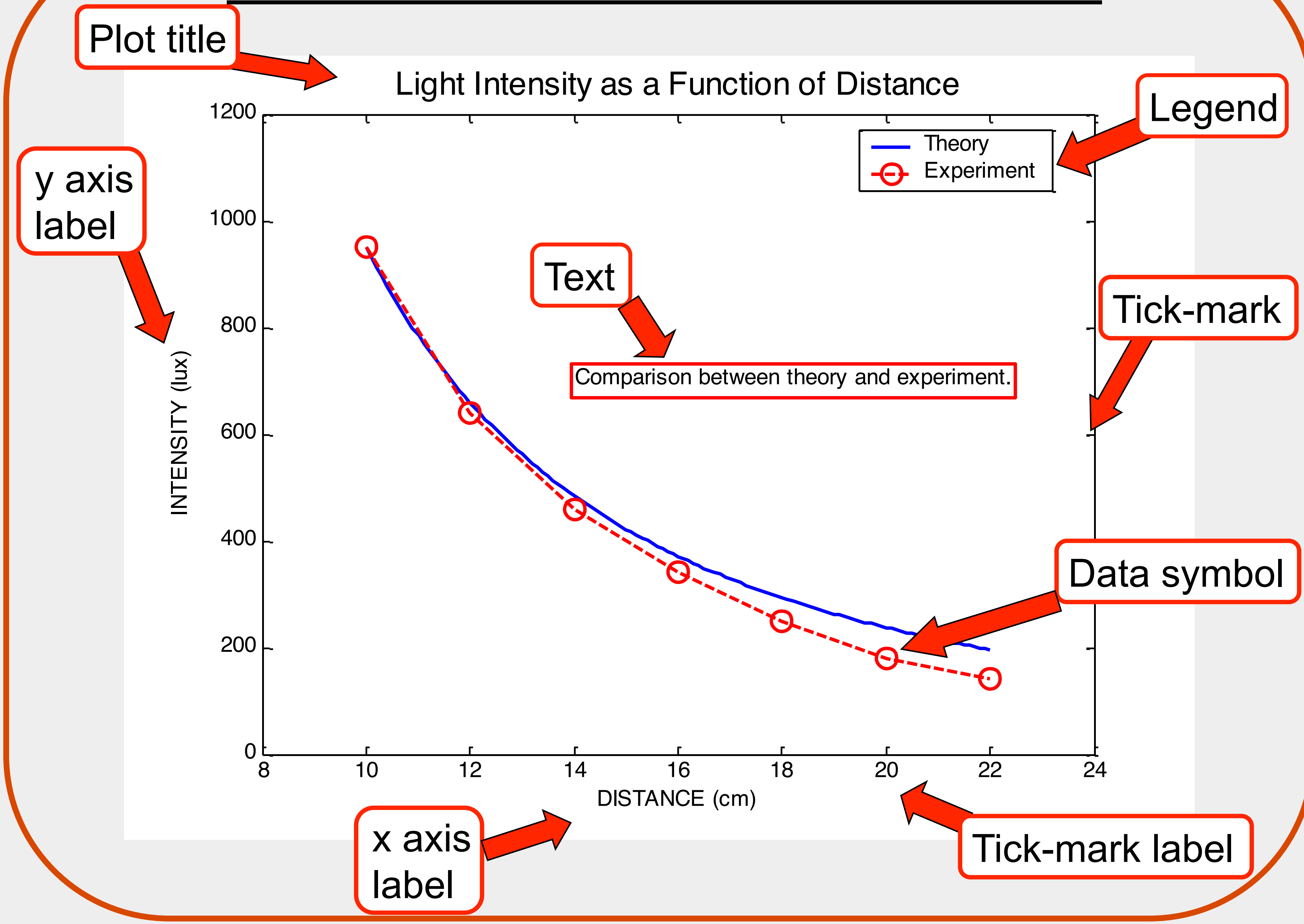
Plot of the function, $y = 3x^3 - 26x + 10$ and its first and second derivatives, for $-2 \le x \le 4$ all in the same plot.

```
x = [-2:0.01:4];
y = 3*x.^3-26*x+6;
yd = 9*x.^2-26;
ydd = 18*x;
plot(x,y,'-b')
hold on
plot(x,yd,'--r')
plot(x,ydd,':k')
hold off
```

First graph is created.

Two more graphs are created.

EXAMPLE OF A FORMATTED 2-D PLOT



FORMATTING PLOTS

A plot can be formatted to have a required appearance.

With formatting you can:

- Add title to the plot.
- Add labels to axes.
- Change range of the axes.
- Add legend.
- Add text blocks.
- Add grid.

FORMATTING PLOTS

There are two methods to format a plot:

1. Formatting commands.

   In this method commands, that make changes or additions to the plot, are entered after the **`plot()`** command. This can be done in the Command Window, or as part of a program in a script file.

2. Formatting the plot interactively in the Figure Window.

   In this method the plot is formatted by clicking on the plot and using the menu to make changes or add details.

FORMATTING COMMANDS

```
title('string')
```

Adds the string as a title at the top of the plot.

```
xlabel('string')
```

Adds the string as a label to the x-axis.

```
ylabel('string')
```

Adds the string as a label to the y-axis.

```
axis([xmin xmax ymin ymax])
```

Sets the minimum and maximum limits of the x- and y-axes.

FORMATTING COMMANDS

```
legend('string1','string2','string3')
```

Creates a legend using the strings to label various curves (when several curves are in one plot). The location of the legend is specified by the mouse.

```
text(x,y,'string')
```

Places the string (text) on the plot at coordinate x,y relative to the plot axes.

```
gtext('string')
```

Places the string (text) on the plot. When the command executes the figure window pops and the text location is clicked with the mouse.

EXAMPLE OF A FORMATTED PLOT

Below is a script file of the formatted light intensity plot (2nd slide). (*Some of the formatting options were not covered in the lectures, but are described in the book*)

```
x=[10:0.1:22];
y=95000./x.^2;
xd=[10:2:22];
yd=[950 640 460 340 250 180 140];
plot(x,y,'-','LineWidth',1.0)
hold on
plot(xd,yd,'ro--','linewidth',1.0,'markersize',10)
hold off
```

Creating vector **x** for plotting the theoretical curve.

Creating vector **y** for plotting the theoretical curve.

Creating a vector with coordinates of data points.

Creating a vector with light intensity from data.

EXAMPLE OF A FORMATTED PLOT

Formatting of the light intensity plot (cont.)

```
xlabel('DISTANCE (cm)')

ylabel('INTENSITY (lux)')

title('\fontname{Arial}Light Intensity as a Function of
Distance','FontSize',14)

axis([8 24 0 1200])

text(14,700,'Comparison between theory and
experiment.','EdgeColor','r','LineWidth',2)

legend('Theory','Experiment',0)
```

Labels for the axes.

Title for the plot.

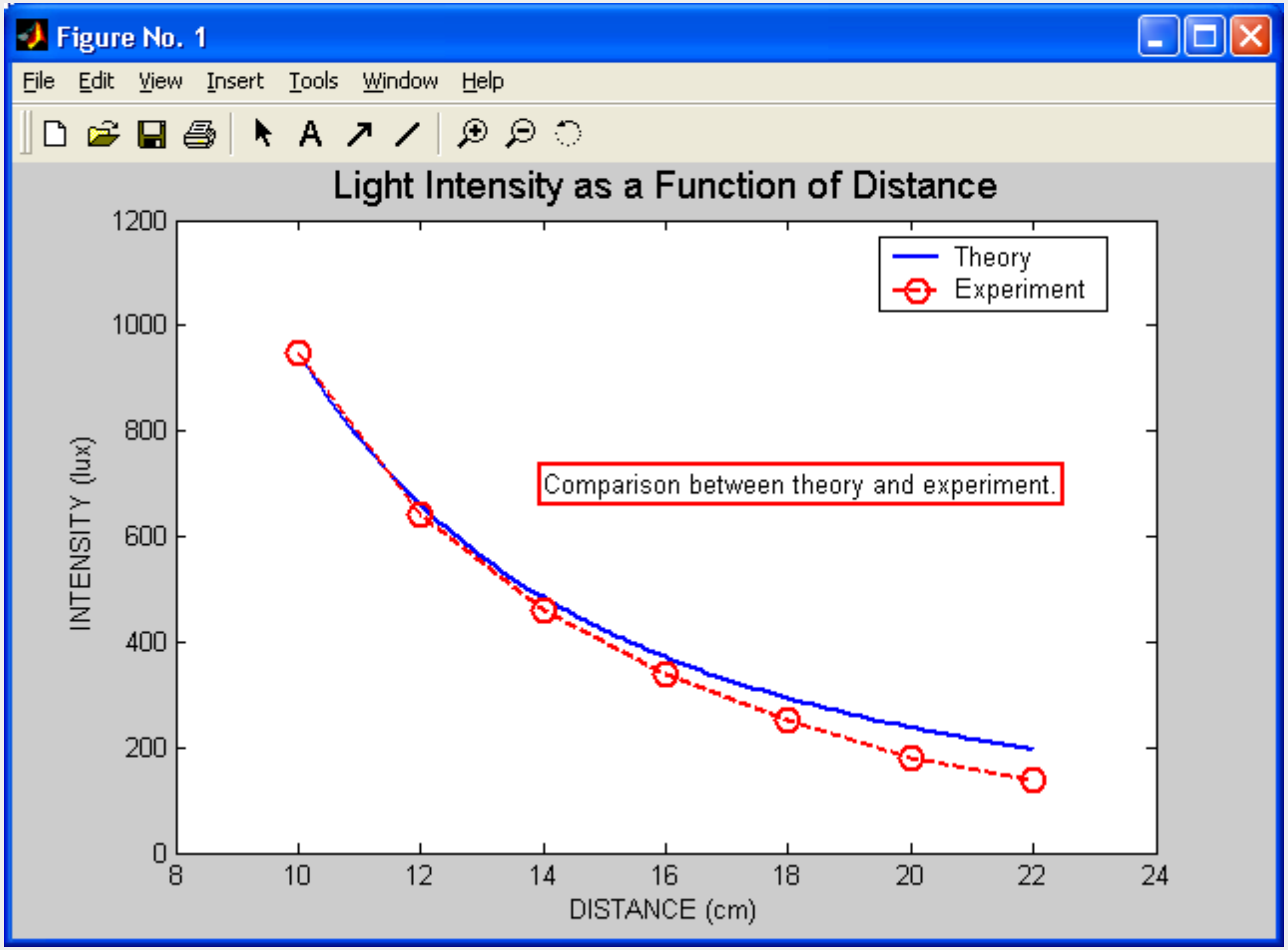
Setting limits of the axes.

Creating text.

Creating a legend.

The plot that is obtained is shown again in the next slide.

EXAMPLE OF A FORMATTED PLOT



FORMATTING A PLOT IN THE FIGURE WINDOW

Once a figure window is open, the figure can be formatted interactively.

Use the insert menu to

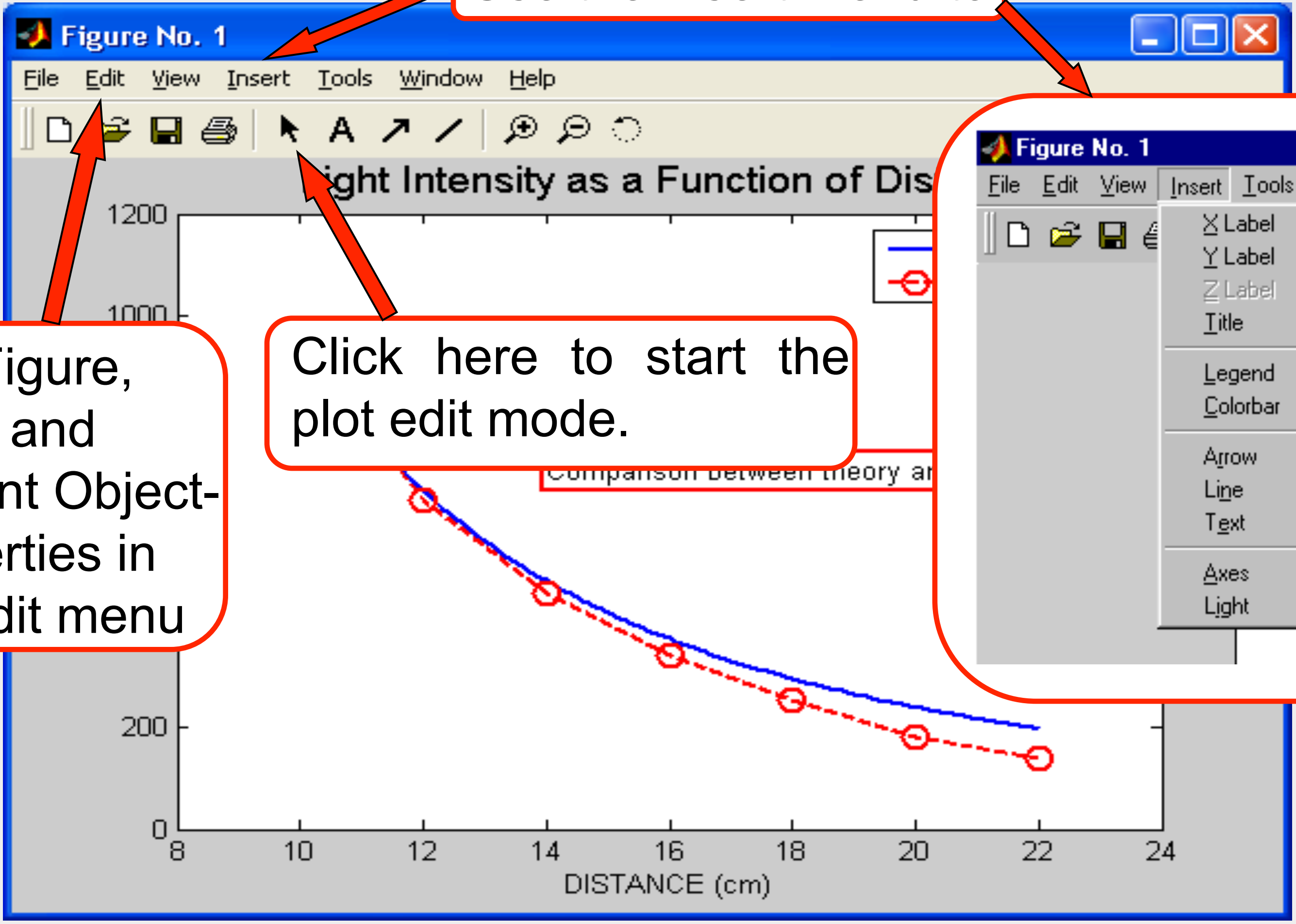
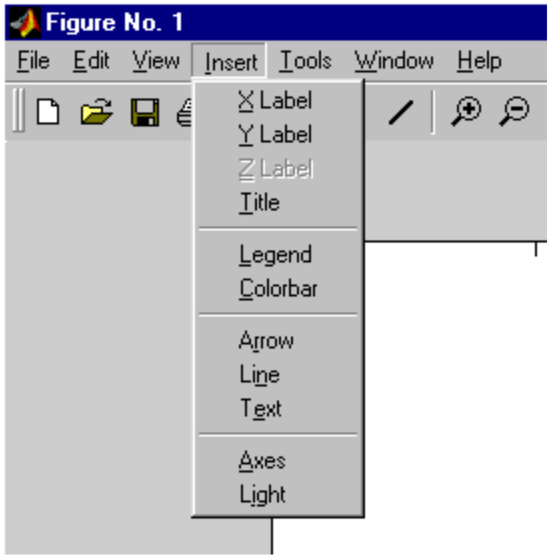


Figure No. 1
File Edit View Insert Tools Window Help
Light Intensity as a Function of Dis
Comparison between theory ar
1200
1000
200
0
8
10
12
14
16
18
20
22
24
DISTANCE (cm)

Use Figure, Axes, and Current Object-Properties in the Edit menu

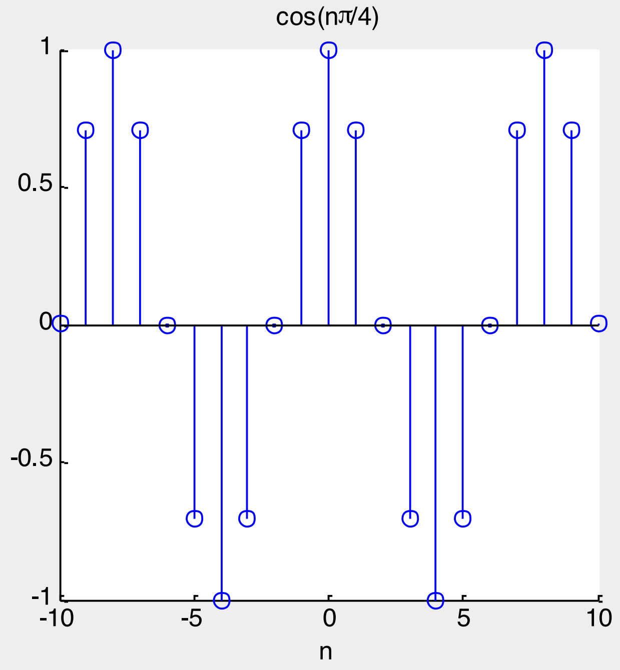
Click here to start the plot edit mode.



Graphics-Stem()

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

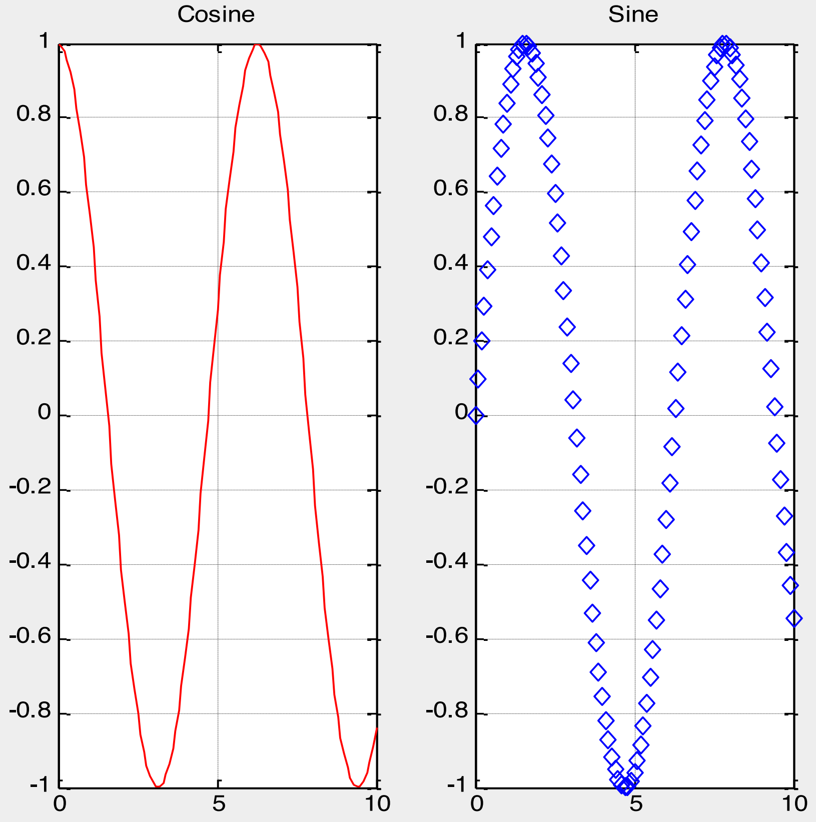
```
>> n=-10:10;
>> f=stem(n,cos(n*pi/4))
>> title('cos(n\pi/4)')
>> xlabel('n')
```



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

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.

Try yourself…

- 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

Matrix Operations 1/5

$$[A]=\begin{bmatrix}0&5&0\\8&3&7\\9&-2&9\end{bmatrix},[B]=\begin{bmatrix}4&6&-2\\7&2&3\\1&3&-4\end{bmatrix},[C]=\begin{bmatrix}-1\\2\\5\end{bmatrix}$$

- Go to Page 422 in the MATLAB Handout

$$[A]+[B]=\begin{bmatrix}0&5&0\\8&3&7\\9&-2&9\end{bmatrix}+\begin{bmatrix}4&6&-2\\7&2&3\\1&3&-4\end{bmatrix}=\begin{bmatrix}0+4&5+6&0+(-2)\\8+7&3+2&7+3\\9+1&-2+3&9+(-4)\end{bmatrix}=\begin{bmatrix}4&11&-2\\15&5&10\\10&1&5\end{bmatrix}$$

$$[B]+[A]=\begin{bmatrix}4&6&-2\\7&2&3\\1&3&-4\end{bmatrix}+\begin{bmatrix}0&5&0\\8&3&7\\9&-2&9\end{bmatrix}=\begin{bmatrix}4+0&6+5&-2+0\\7+8&2+3&3+7\\1+9&3+(-2)&-4+9\end{bmatrix}=\begin{bmatrix}4&11&-2\\15&5&10\\10&1&5\end{bmatrix}$$

$$[A]+[B]=[B]+[A]$$

- Matrix addition:

$$[A]\cdot[B]=\begin{bmatrix}0&5&0\\8&3&7\\9&-2&9\end{bmatrix}\cdot\begin{bmatrix}4&6&-2\\7&2&3\\1&3&-4\end{bmatrix}=\begin{bmatrix}0\cdot4+5\cdot7+0\cdot1&0\cdot6+5\cdot2+0\cdot3&0\cdot(-2)+5\cdot3+0\cdot(-4)\\8\cdot4+3\cdot7+7\cdot1&8\cdot6+3\cdot2+7\cdot3&8\cdot(-2)+3\cdot3+7\cdot(-4)\\9\cdot4+(-2)\cdot7+9\cdot1&9\cdot6+(-2)\cdot2+9\cdot3&9\cdot(-2)+(-2)\cdot3+9\cdot(-4)\end{bmatrix}=\begin{bmatrix}35&10&15\\60&75&-35\\31&77&-60\end{bmatrix}$$

$$[B]\cdot[A]=\begin{bmatrix}4&6&-2\\7&2&3\\1&3&-4\end{bmatrix}\cdot\begin{bmatrix}0&5&0\\8&3&7\\9&-2&9\end{bmatrix}=\begin{bmatrix}4\cdot0+6\cdot8+(-2)\cdot9&4\cdot5+6\cdot3+(-2)\cdot(-2)&4\cdot0+6\cdot7+(-2)\cdot9\\7\cdot0+2\cdot8+3\cdot9&7\cdot5+2\cdot3+3\cdot(-2)&7\cdot0+2\cdot7+3\cdot9\\1\cdot0+3\cdot8+(-4)\cdot9&1\cdot5+3\cdot3+(-4)\cdot(-2)&1\cdot0+3\cdot7+(-4)\cdot9\end{bmatrix}=\begin{bmatrix}30&42&24\\43&35&41\\-12&22&-15\end{bmatrix}$$

$$[A]\cdot[B]\neq[B]\cdot[A]$$

Matrix Operations 2/5

- Matrix multiplication can also be considered as linear equations.

$$[D]=[A]\cdot[C]=\begin{bmatrix}A_{11} & A_{12} & A_{13}\\ A_{21} & A_{22} & A_{23}\\ A_{31} & A_{32} & A_{33}\end{bmatrix}\cdot\begin{bmatrix}C_1\\ C_2\\ C_3\end{bmatrix}=\begin{bmatrix}D_1\\ D_2\\ D_3\end{bmatrix}$$

$$A_{11}\cdot C_1+A_{12}\cdot C_2+A_{13}\cdot C_3=D_1$$

$$A_{21}\cdot C_1+A_{22}\cdot C_2+A_{23}\cdot C_3=D_2$$

$$A_{31}\cdot C_1+A_{32}\cdot C_2+A_{33}\cdot C_3=D_3$$

$$[D]=[A]\cdot[C]=\begin{bmatrix}0 & 5 & 0\\ 8 & 3 & 7\\ 9 & -2 & 9\end{bmatrix}\cdot\begin{bmatrix}-1\\ 2\\ 5\end{bmatrix}=\begin{bmatrix}0\cdot(-1)+5\cdot 2+0\cdot 5\\ 8\cdot(-1)+3\cdot 2+7\cdot 5\\ 9\cdot(-1)+(-2)\cdot 2+9\cdot 5\end{bmatrix}=\begin{bmatrix}10\\ 33\\ 32\end{bmatrix}$$

- The transpose of any matrix switches the column with row

$$[A]^T=\begin{bmatrix}A_{11} & A_{12} & A_{13}\\ A_{21} & A_{22} & A_{23}\\ A_{31} & A_{32} & A_{33}\end{bmatrix}^T=\begin{bmatrix}A_{11} & A_{21} & A_{31}\\ A_{12} & A_{22} & A_{32}\\ A_{13} & A_{23} & A_{33}\end{bmatrix}=\begin{bmatrix}0 & 5 & 0\\ 8 & 3 & 7\\ 9 & -2 & 9\end{bmatrix}^T=\begin{bmatrix}0 & 8 & 9\\ 5 & 3 & -2\\ 0 & 7 & 9\end{bmatrix}$$

- The determinant a matrix

$$|A|=\left\|\begin{bmatrix}A_{11} & A_{12} & A_{13}\\ A_{21} & A_{22} & A_{23}\\ A_{31} & A_{32} & A_{33}\end{bmatrix}\right\|=A_{11}\cdot(A_{22}\cdot A_{33}-A_{32}\cdot A_{23})-A_{12}\cdot(A_{21}\cdot A_{33}-A_{31}\cdot A_{23})+A_{13}\cdot(A_{21}\cdot A_{32}-A_{31}\cdot A_{22})$$

$$|A|=\left\|\begin{bmatrix}0 & 5 & 0\\ 8 & 3 & 7\\ 9 & -2 & 9\end{bmatrix}\right\|=0\cdot(3\cdot 9-(-2)\cdot 7)-5\cdot(8\cdot 9-9\cdot 7)+0\cdot(8\cdot(-2)-9\cdot 3)=-45$$

Matrix Operations 3/5

- The inverse of the A matrix:
  - Find the determinant of the matrix
  - Find the transpose of the matrix
  - Find the cofactors of the matrix
  - Insert the cofactors into the matrix

$$[A]^{-1} = \begin{bmatrix} A_{11}^{-1} = \frac{1}{|A|}\cdot\begin{vmatrix} 3 & -2 \\ 7 & 9 \end{vmatrix} = \frac{41}{-45} & A_{12}^{-1} = -1\cdot\frac{1}{|A|}\cdot\begin{vmatrix} 5 & -2 \\ 0 & 9 \end{vmatrix} = \frac{-45}{45} & A_{13}^{-1} = \frac{1}{|A|}\cdot\begin{vmatrix} 5 & 3 \\ 0 & 7 \end{vmatrix} = \frac{35}{45} \\ A_{21}^{-1} = -1\cdot\frac{1}{|A|}\cdot\begin{vmatrix} 8 & 9 \\ 7 & 9 \end{vmatrix} = \frac{-9}{-45} & A_{22}^{-1} = \frac{1}{|A|}\cdot\begin{vmatrix} 0 & 9 \\ 0 & 9 \end{vmatrix} = 0 & A_{23}^{-1} = -1\cdot\frac{1}{|A|}\cdot\begin{vmatrix} 0 & 8 \\ 0 & 7 \end{vmatrix} = 0 \\ A_{31}^{-1} = \frac{1}{|A|}\cdot\begin{vmatrix} 8 & 9 \\ 3 & -2 \end{vmatrix} = \frac{-43}{-45} & A_{32}^{-1} = -1\cdot\frac{1}{|A|}\cdot\begin{vmatrix} 0 & 9 \\ 5 & -2 \end{vmatrix} = \frac{45}{45} & A_{33}^{-1} = \frac{1}{|A|}\cdot\begin{vmatrix} 0 & 8 \\ 5 & 3 \end{vmatrix} = \frac{-40}{} \end{bmatrix}$$

$$[A]^{-1} = \begin{bmatrix} -0.9111 & 1 & -0.7778 \\ 0.2 & 0 & 0 \\ 0.9556 & -1 & 0.8889 \end{bmatrix}$$

Matrix Operations 4/5

- To create a matrix inside MATLAB:
  - Brackets "[]"
  - Space or comma "," , indicates a new column
  - A semicolon ";" indicates a new row
- Follow the directions inside example 15.4 (pg. 422-423)
  - Add Matrices (A+B)
  - Subtract Matrices (A-B)
  - Multiply Matrices (A*B)
  - Find the determinant of a Matrix (det(A))
- Follow the directions inside example 15.5 (pg. 424)
  - Solve a set of linear equations
  - Gauss elimination (A\B)
  - Inverse of a matrix (A^{-1}=inv(A))
  - A\B=inv(A)*B

Matrix Operations 5/5

- Element by element operation:

$$[A].\cdot[B]=\begin{bmatrix}A_{11} & A_{12} & A_{13}\\ A_{21} & A_{22} & A_{23}\\ A_{31} & A_{32} & A_{33}\end{bmatrix}.\cdot\begin{bmatrix}B_{11} & B_{12} & B_{13}\\ B_{21} & B_{22} & B_{23}\\ B_{31} & B_{32} & B_{33}\end{bmatrix}=\begin{bmatrix}A_{11}\cdot B_{11} & A_{12}\cdot B_{12} & A_{13}\cdot B_{13}\\ A_{21}\cdot B_{21} & A_{22}\cdot B_{22} & A_{23}\cdot B_{23}\\ A_{31}\cdot B_{31} & A_{32}\cdot B_{32} & A_{33}\cdot B_{33}\end{bmatrix}$$

$$[A].\cdot[B]=\begin{bmatrix}0 & 5 & 0\\ 8 & 3 & 7\\ 9 & -2 & 9\end{bmatrix}.\cdot\begin{bmatrix}4 & 6 & -2\\ 7 & 2 & 3\\ 1 & 3 & -4\end{bmatrix}=\begin{bmatrix}0\cdot 4 & 5\cdot 6 & 0\cdot(-2)\\ 8\cdot 7 & 3\cdot 2 & 7\cdot 3\\ 9\cdot 1 & -2\cdot 3 & 9\cdot(-4)\end{bmatrix}=\begin{bmatrix}0 & 30 & 0\\ 56 & 6 & 21\\ 9 & -6 & -36\end{bmatrix}$$

- In the MATLAB command window type: a.*b