بسم الله الرحمن الرحيم

MATLAB Tutorial

- MATLAB basics
- Built-in functions
- Graphs
- m files

Semester Teams

- Team 1:Abdulaziz, Ibrahim, Abdulhamid
- Team 2: Fawzi, Yasin, M. Naser
- Team 3: Sameer, Aiman, M. Hassanein

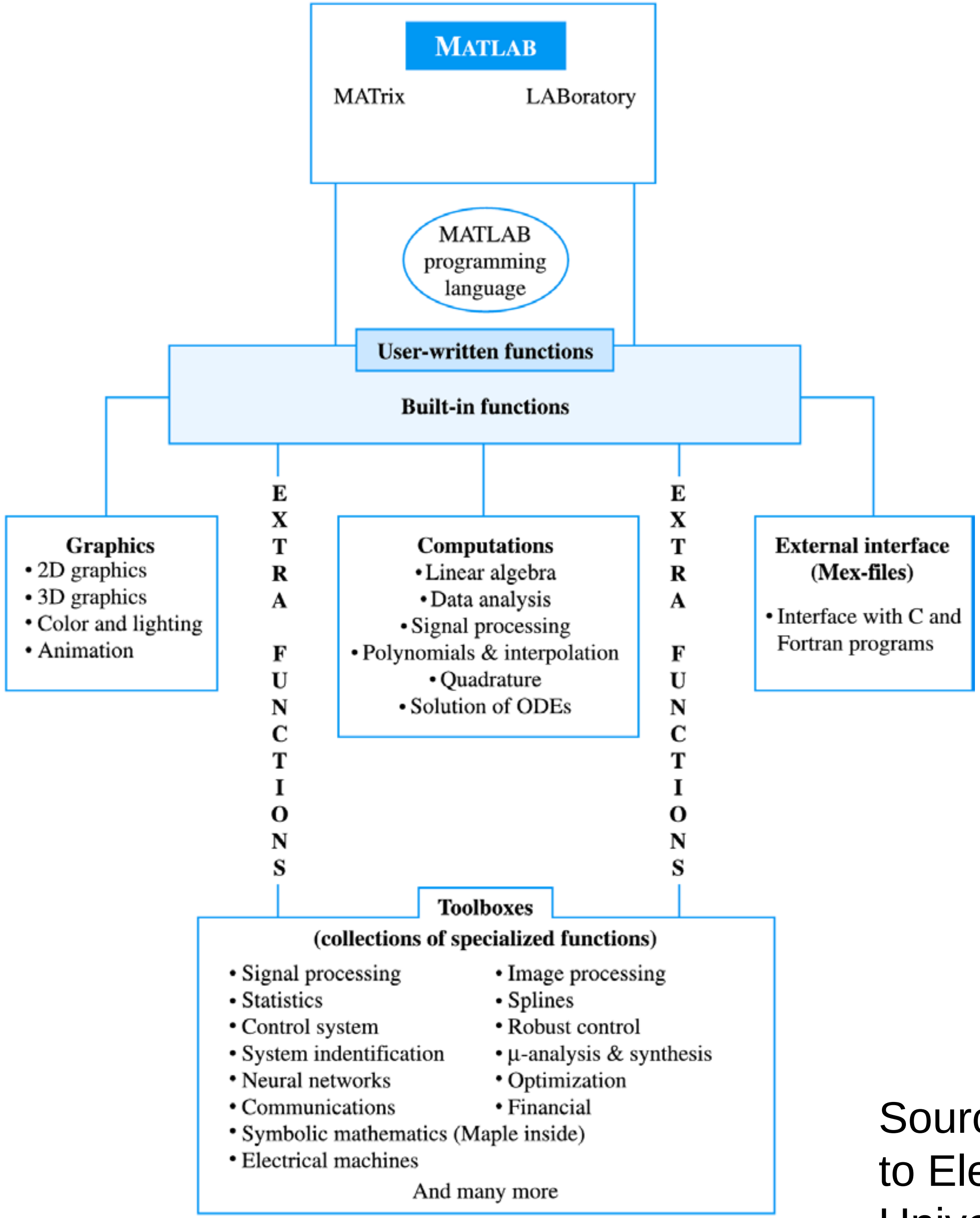


Figure 2.6.1 Schematic diagram of MATLAB's main features.

Source: Sarma MS – Introduction to Electrical Engineering, Oxford University press 2001

Graphs supported

MATLAB supports many types of graph and surface plots:

- **line plots (x vs. y),**
- **filled plots,**
- **bar charts,**
- **pie charts,**
- **parametric plots,**
- **polar plots,**
- **contour plots,**
- **density plots,**
- **log axis plots,**
- **surface plots,**
- **parametric plots in 3 dimensions and spherical plots.**

Toolboxes

- **MATLAB has a number of add-on software modules, called *toolbox* , that perform more specialized computations.**

Signal & Image Processing

Signal Processing- Image Processing Communications - System Identification - Wavelet Filter Design

Control Design

Control System - Fuzzy Logic - Robust Control - µ-Analysis and Synthesis - LMI Control - Model Predictive Control Model-Based Calibration

More than 60 toolboxes!

Numbers

MATLAB recognizes several different kinds of numbers:

- **Integer (like: 12 - 678),**
- **Real (like: 4.607 - 199.34),**
- **Complex (like: 2 + 3i , i=j=sqrt(-1)),**
- **Inf (like: Infinity 2/0),**
- **NaN (like: Not a Number, 0/0).**

All computations in MATLAB are done in double precision, which means about 15 significant figures.

Number Format

The format command in MATLAB is used to control pints numbers.

The number of digits displayed is not related to the accuracy.

To change the format of the display, type

format short e for scientific notation with 5 decimal places,

format long e for scientific notation with 15 significant decimal places and

format bank for placing two significant digits to the right of the decimal.

Numbers and Format

Use help format for more information.

Command	Example of Output
format short	11.3045 (4-decimal Places)
format short e	1.1304e+01
format long e	1.130452467450893+01
format bank	11.30 (2-decimal places)
format hex	Hexadecimal format
format +	The symbols of +, - and blank are printed

Command Line Help

Help and information on MATLAB can be found in several ways,

- from the command line by using the 'help' topic command
- from the separate Help window found under the Help menu
- from the MATLAB helpdesk stored on disk or on a CD-ROM

Command Line Help - Examples

```
>>help pi
PI 3.14159265358979....
PI = 4*atan(1) = imag(log(-1)) =
3.14159265358979....

>>help sin
SIN Sine.
SIN(X) is the sine of the elements of X.
```

Variables

Variables ans is assigned by MATLAB default.

- For example, typing >>12+2.3*2 or >>12+2.3*2, yields: ans = 16.6000
- >>12+2.3*2;   yields: blank (but the result is saved on the variable "ans"
- (write >>ans see the result of operation which is 16.6000).
- Commas (,) tell MATLAB to display results
- semicolons (;) suppress printing.

Variables

Variables are assigned numerical values by typing the expression directly, for example, typing

- `>>a = 12+2.3*2`
- yields: a = 16.6000
- The answer will not be displayed when a semicolon is put at the end of an expression, for example type
- `>>a = 12+2.3*2;`

Variables

Legal variable names consist of any combination of letters and digits, starting with a letter.

Examples: Ali22B, Cost, X3_f22 and s2Sc6.

- But the variables like:

Ali-22, 5x, 3Cost, &r5, %67 and @xyt56

are not allowed in MATLAB.

Characters in MATLAB is like X='a';

Strings in MATLAB is like mg1='Ali'; or mg2='MATLAB DEMOS';

Variables-Arithmetic Operators

MATLAB utilizes the following arithmetic operators: The following matrix and array operations are available in MATLAB:

+	for	addition
-	for	subtraction
*	for	multiplication
^	for	power
‘	for	transpose
\	for	left division
/	for	right division

Variables

- These matrix operations apply to scalars (1-by-1 matrices) as well.
- Comment statements are preceded by a "**%**".
- The commands **who** and **whose** give the names of the variables that have been defined in the workspace.

Variables

A variable can be assigned using a formula that utilizes these operators and either numbers or previously defined variables.

- For example, since a was defined previously, the following expression is valid
- >>b = 5*a;
- To determine the value of a previously defined quantity, type the quantity by itself:
- >>b
- yields: b = 83.0000

Variables

- If the expression does not fit on one line, we use an ellipsis (three or more periods at the end of the line) and continue on the next line.

-
  ```
  >>c = 1+2+3+...
        5+6+7;
  ```
- There are several predefined variables which can be used at any time, in the same manner as user-defined variables:
- i, sqrt(-1) - j, sqrt(-1) - pi, 3.1416...

Variables

There are also a number of predefined functions that can be used when defining a variable. Some common functions that are used in this workshop are:

abs	magnitude of a number (absolute value for real numbers)
angle	angle of a complex number, in radians
cos	cosine function, assumes argument is in radians
sin	sin function, assumes argument is in radians
exp	exponential function

Variables

- For example, with y defined as above,
- x = abs(y)
- yields: x = 10.8167
- c = angle(y)
- yields: x = 0.9828
- With a = 3 as defined previously,
- x = cos(a)
- yields: c = - 0.9900
- x = exp(a)
- yields: x = - 20.0855
- Note that exp can be used on complex numbers. For example, with y = 2+8i as defined above,
- x = exp(y)
- yields: x = - 1.0751 + 7.3104i
- which can be verified by using Euler's formula:

x = exp(2)cos(8) + j.exp(2)sin(8)

Vectors and Matrices

- **MATLAB is based on matrix and vector algebra; even scalars are treated as 1 by 1 matrices.**
- **Therefore, vector and matrix operations are as simple as common calculator operations.**
- **The number of entries (elements or components) is known as the "length" of the vector.**
- **The entries must be enclosed in square brackets.**

Vectors and Matrices

Vectors can be defined in two ways. The first method is used for arbitrary elements:

- >>v = [1 3 5 sqrt(49)];
- creates a 1x4 vector with elements 1, 3, 5 and 7.
- Note that commas could have been used in place of spaces to separate the elements ([1,3,5,sqrt(49)]).
- Additional elements can be added to the vector:
- >>v(5) = 8;
- yields the vector v = [1 3 5 7 8].
- Previously defined vectors can be used to define a new vector.
- For example, with v defined above
- >>a = [9 10];
- >>b = [v a];
- creates the vector b = [1 3 5 7 8 9 10].

Vectors and Matrices

The second method is used for creating vectors with equally spaced elements:

- `>>t = 0:0.1:10;`
- creates a 1x101 vector with the elements 0, .1, .2, .3,...,10.
- Note that the middle number defines the increment.
- If only two numbers are given, then the increment is set to a default of 1:
- `>>k = 0:10;`
- creates a 1x11 vector with the elements 0, 1, 2, ..., 10.

Vectors and Matrices

Matrices are defined by entering the elements row by row:

- >>A = [2 3 4; 5 -7 6; 10 5 3]
- generates the matrix

$$A = \begin{bmatrix} 2 & 3 & 4 \\ 5 & -7 & 6 \\ 10 & 5 & 3 \end{bmatrix}$$

Vectors and Matrices

There are a number of special matrices that can be defined:

null matrix:	`M = [ ];`
nxm matrix of zeros:	`M = zeros(n,m);`
nxm matrix of ones:	`M = ones(n,m);`
nxn identity matrix:	`M = eye(n);`

Vectors and Matrices

A particular element of a matrix can be assigned:

- `>>M(1,2) = 5;`
- places the number 5 in the first row, second column.
- Operations and functions that were defined for scalars in the previous section can also be used on vectors and matrices.
- For example,
- `>>a = [1 2 3];`
- `>>b = [4 5 6];`
- `>>c = a + b`
- yields: c = 5 7 9
- Functions are applied element by element.
- For example,
- `>>t = 0:10;`
- `>>x = cos(2*t);`
- creates a vector x with elements equal to cos(2t) for t = 0, 1, 2, ..., 10.

Vectors and Matrices

Operations that need to be performed element-by-element can be accomplished by preceding the operation by a ".".

For example, to obtain a vector x that contains the elements of x(t) = tcos(t) at specific points in time, you cannot simply multiply the vector t with the vector cos(t).

Instead you multiply their elements together:

- `>>t = 0:10;`
- `>>x = t.*cos(t);`
- The command length(x) returns the length of a vector x and size(x) returns the dimension of the matrix x.

Built-in Signal Functions

- Trigonometric functions; sin, cos, tan
- Inverse trigonometric functions; asin, acos, atan
- Special mathematical functions; heaviside, square, chirp, sinc, diric, gauspuls, pulstran and rectpuls.
- MATLAB is a "High-Performance Numeric Computation and Visualization Software" package.
- MATLAB is an interactive system whose basic data is a matrix that does not require dimensioning.

Exercise

- Generate and plot functions using the built-in functions discussed previously, i.e.
  - Rectpuls(t) and rectpuls(t,w)
  - Tripuls(t), tripuls(t,w), tripuls(t,w,s)
  - Sinc(t)
  - Diric(t,N) (N = number of equally spaced extrema of the function in the interval 0 to 2*pi, for N odd repeated sinc function)
- Use help when necessary

DIRAC. Delta function

- DIRAC(X) is zero for all X, except X == 0 where it is infinite.
- DIRAC(X) is not a function in the strict sense, but rather a distribution with int(dirac(x-a)*f(x),-inf,inf) = f(a) and
- diff(heaviside(x),x) = dirac(x).

$$\int_{-\infty}^{\infty} \mathrm{g}(t)\delta(t-t_0)\,dt = \mathrm{g}(t_0)$$

Exercise – 1

- Run MATLAB (if not already running)
- Type and execute the following statements
  - >> syms x
  - >> y=sin(2*pi*x);
  - >> d=dirac(x-0.5);
  - >> ys=int(y*d, -inf, inf)
  - ys =
  - 0
- What is sin(2*pi*x) at x=0.5?
  - >> ys=int(y*d, -inf, inf);
  - >> ezplot(x,y)
  - >> d=dirac(x-0.25);
  - >> ys=int(y*d, -inf, inf);
  - >> ys
  - ys =
  - 1
- What is sin(2*pi*x) at x=0.25?

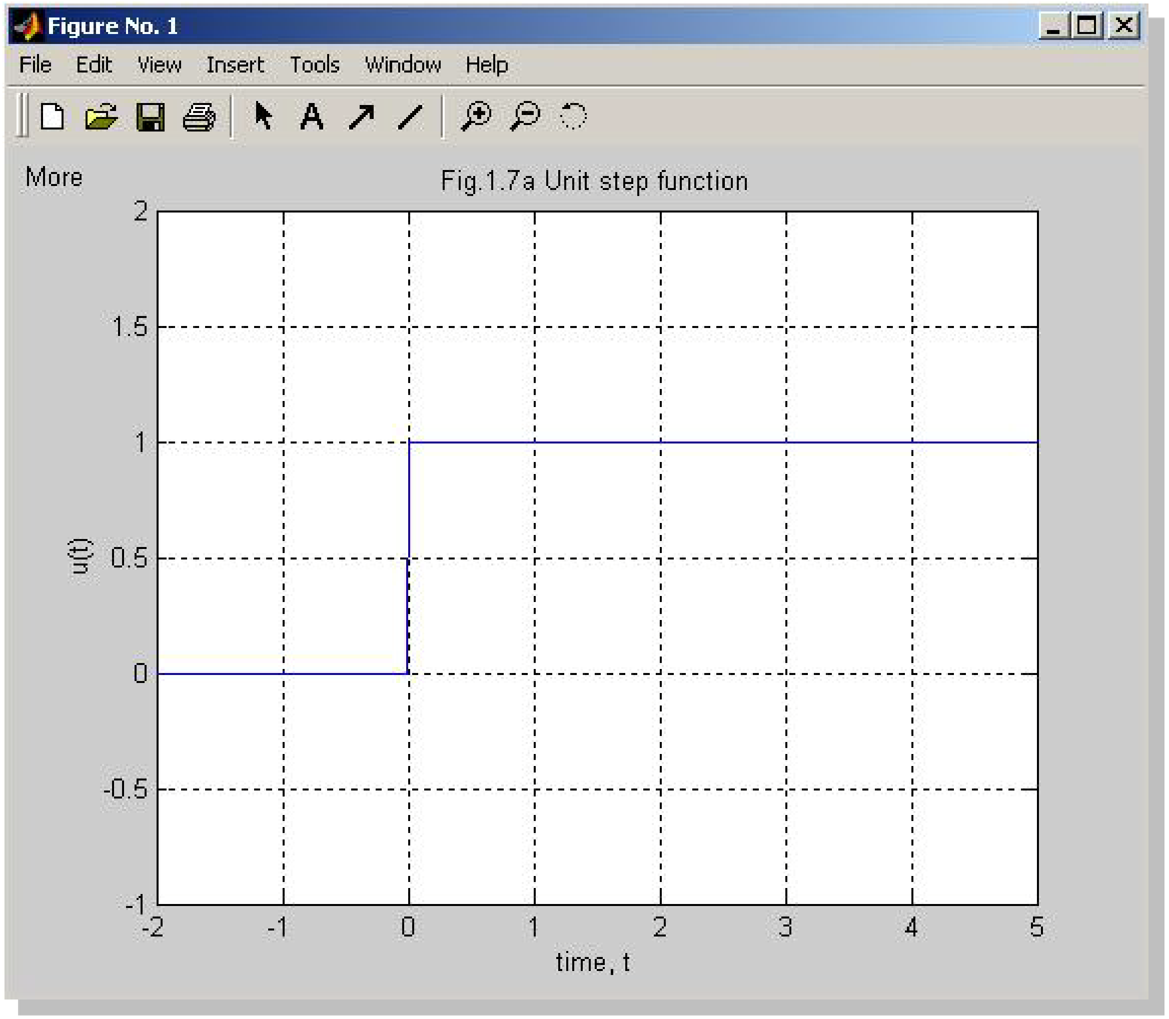
Unit Step Function

Generate and draw the figure shown using the stepfun function

$$Bu(t-t_0)=\begin{cases} B & if \quad t \geq t_0 \\ 0 & if \quad t < t_o. \end{cases}$$

Generic step function

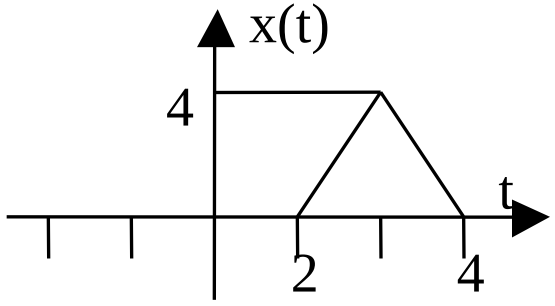
Alternative; ut=heaviside(t-t0)



Step function: Exercise – 2

- Step function can be generated either by Stepfun(t,to) or heaviside(t-to) command; t is the independent variable and to is the starting position of the step
  - >> clear
  - >> t =-2:0.01:2 ;
  - >> s1=stepfun(t,0); s2=0.5*stepfun(t,-0.5); s3=1.5*stepfun(t,0.5);
  - >> h1=heaviside(t); h2=0.5*heaviside(t+0.5); h3=1.5*heaviside(t-0.5);
  - >> s=[s1',s2',s3'];h=[h1',h2',h3'];
  - >> subplot(2,1,1), plot(t,s);subplot(2,1,2), plot(t,h)

Exercise - 3



1. Take a piece of paper and for the function shown at left:
   1. Determine y1(t) = dx(t)/dt and draw y1(t)
   2. Determine y2(t) = integral(x(t)) and draw y2(t)
2. Write down a MATLAB m-file to perform the above operations.
3. Run the m-file, draw the graphs using MATLAB and compare them to your hand drawings.

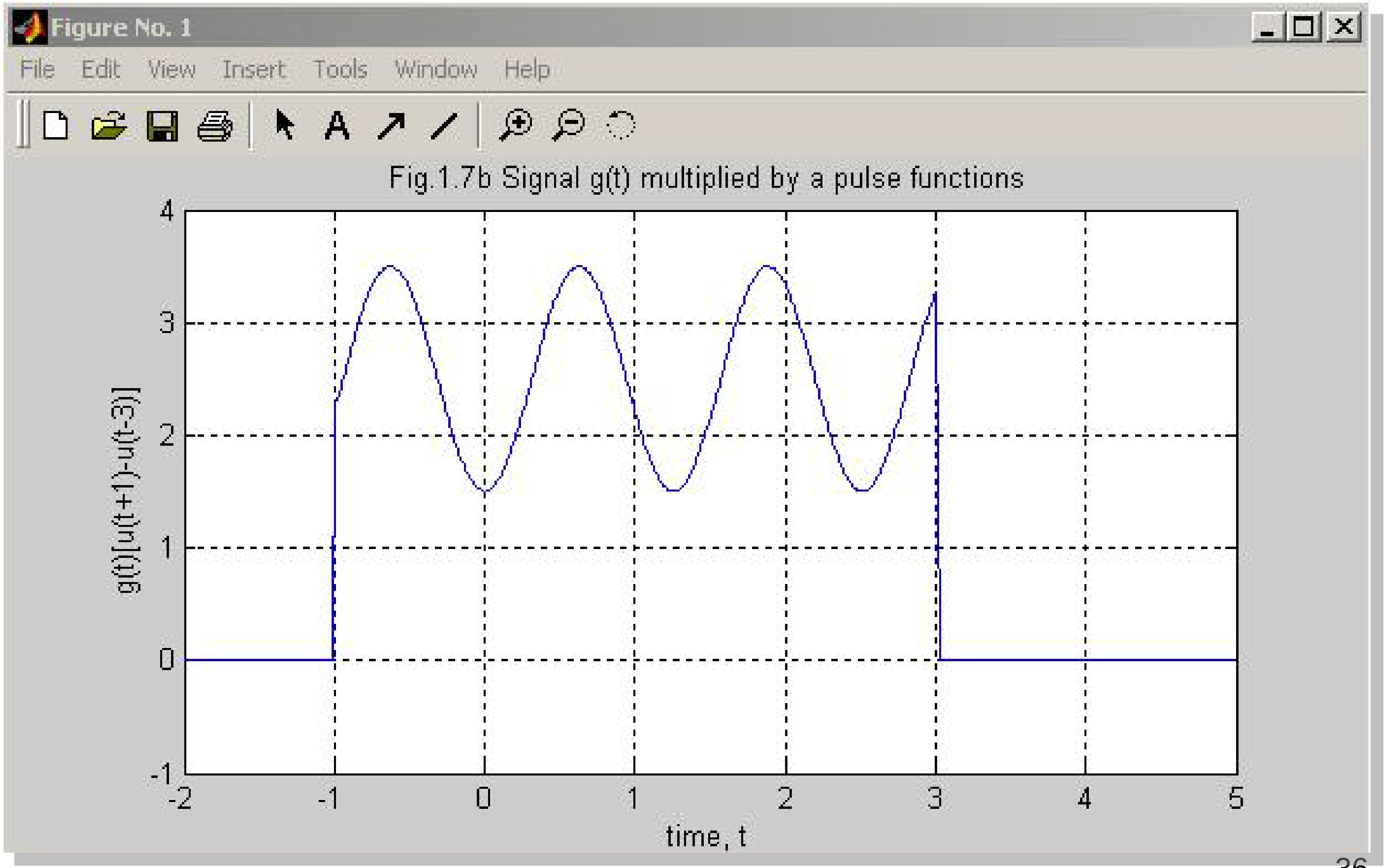
Sinusoidal signals

- Generate, plot and identify critical points for
  - Sin(2*pi*t)
  - Cos(4*pi*t)
  - Sin(2*pi*t)+cos(2*pi*t)
  - Rectpuls(t,2)*sin(4*pi*t)
  - Exp(-2*t)
  - Exp(-2*t)*cos(4*pi*t)

Exercise - 4

- Represent $x(t) = 2\sin(100\pi t) + 3\cos(100\pi t)$ in form of $x(t) = A\cos(\omega t + \theta)$.
- Calculate A, T, f (frequency in Hertz), period and phase shift θ.
- Draw the waveform for x(t) using MATLAB. Indicate all important amplitude and time values.

Generate and plot the function shown

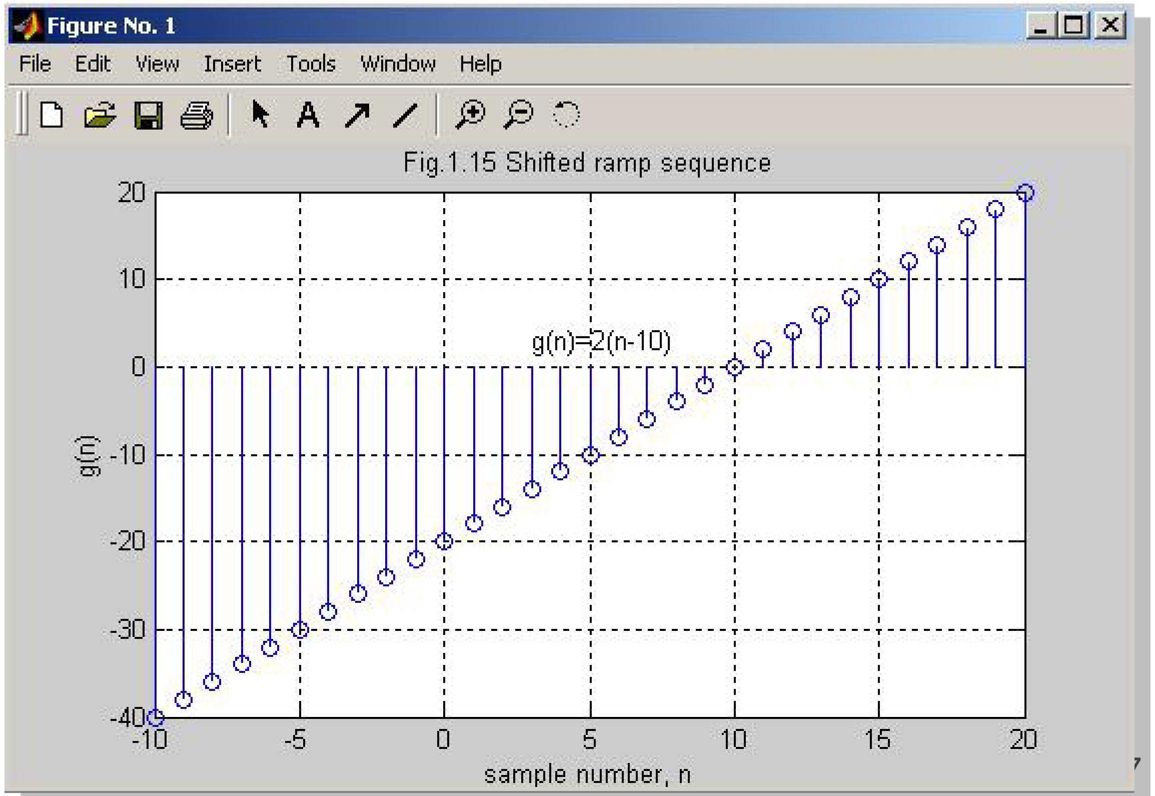


Sequences

1. **Ramp Sequence**
2. A shifted ramp sequence with slop of B is defined by: $g(n) = B(n - n_0)$
3. The unit ramp sequence and shifted ramp sequences
4. **Example:** ***g(t) = 2(n-10).***

MATLAB Code:

```
n=-10:1:20;
f=2*(n-10);
stem(n,f);
```



Real Exponential Sequences

1. Real exponential sequence is defined as: $f(n) = A(a)^n$

Example for A = 10 and a = 0.9, as n goes to infinity the sequence approaches zero and as n goes to minus infinity the sequence approaches plus infinity.

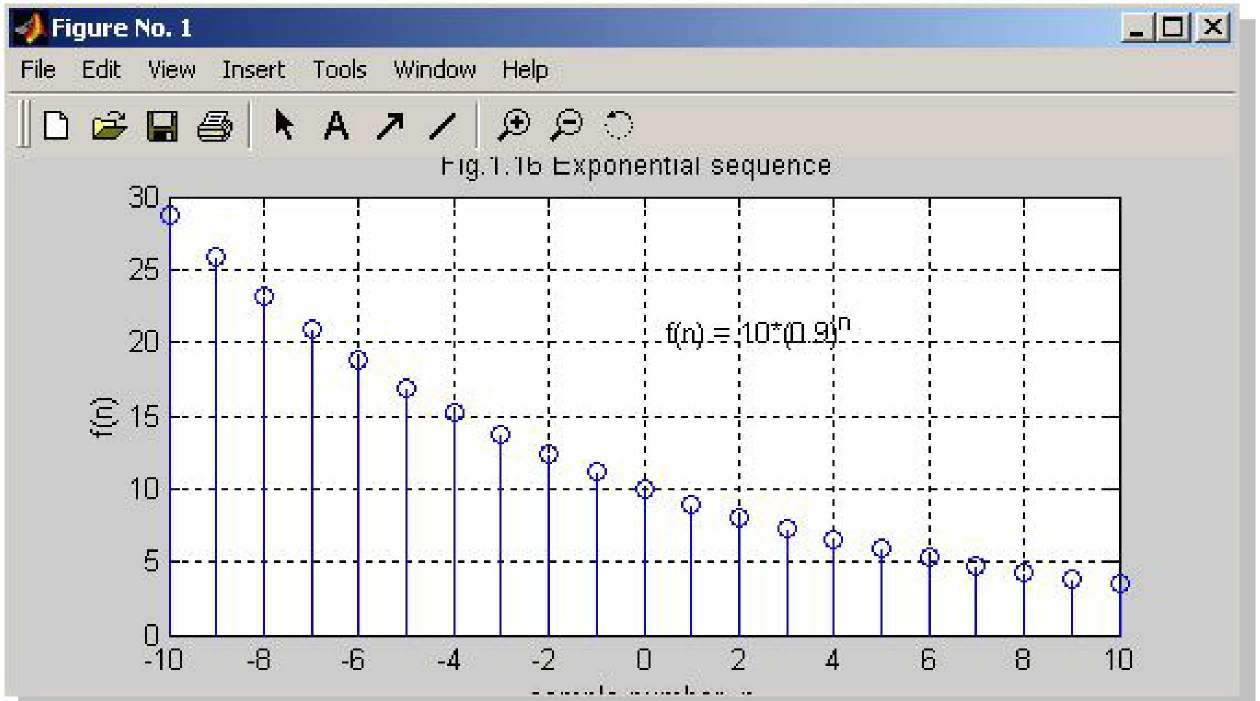
Composite sequence:

$$p(n) = A(a)^n u(n)$$

Multiplying point by
Point by the step sequence

MATLAB Code:

```
n=-10:1:10;
f =10*(.9).^n;
stem(n,f);
axis([-10 10 0 30]);
```



Sinusoidal Sequence

1. A sinusoidal sequence may be described as:

$$f(n) = A\cos\left(\frac{2\pi n}{N} + \alpha\right)$$

2. Where ***A*** is positve real number (amplitude), ***N*** is the period, and ***a*** is the p

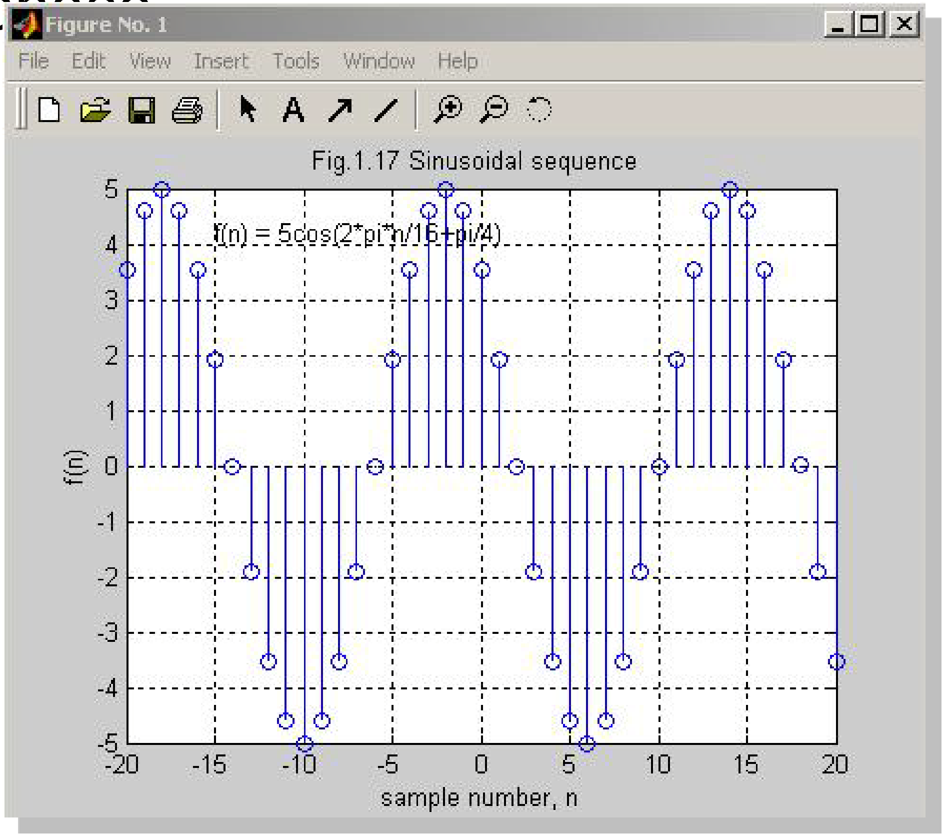
3. **Example:**

4. A = 5, N = 16

5. And $a = \pi / 4$.

6. MATLAB Code:

```
7.   n=-20:1:20;
8.   f=5*[cos(n*pi/8+pi/4)];
9.   stem(n,f);
```



Exponentially Modulated Sinusoidal Sequence

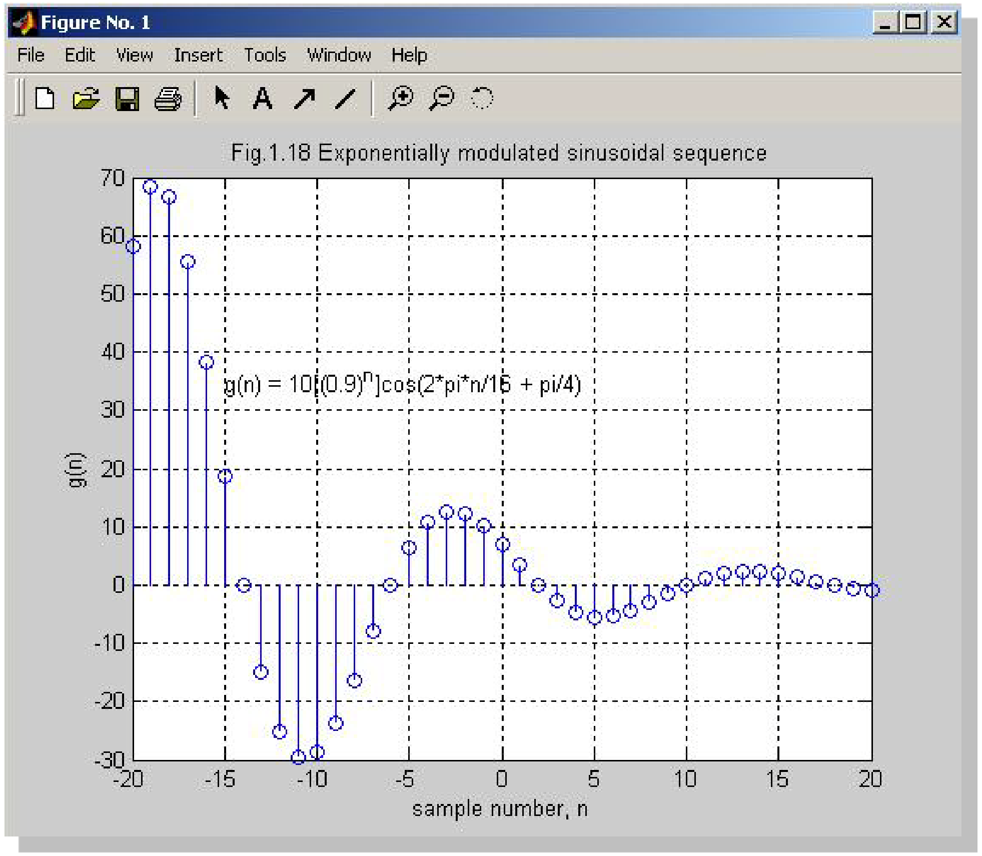
1. By multiplying an exponential sequence by sinusoidal sequence, we obtain an exponentially modulated sequence described by:

$$g(n) = A(a)^n \cos\left(\frac{2\pi n}{N} + \alpha\right)$$

2. **Example:** $\alpha = \pi / 4.$
3. A = 10, N = 16, a = 0.9

1. **MATLAB Code:**

```
n=-20:1:20;
f=10*[0.9 .^n];
g=[cos(2*n*pi/16+pi/4)];
h=f .*g;
stem(n,h);
axis([-20 20 -30 70]);
```



MATLAB Basics

Plotting Elementary Functions:

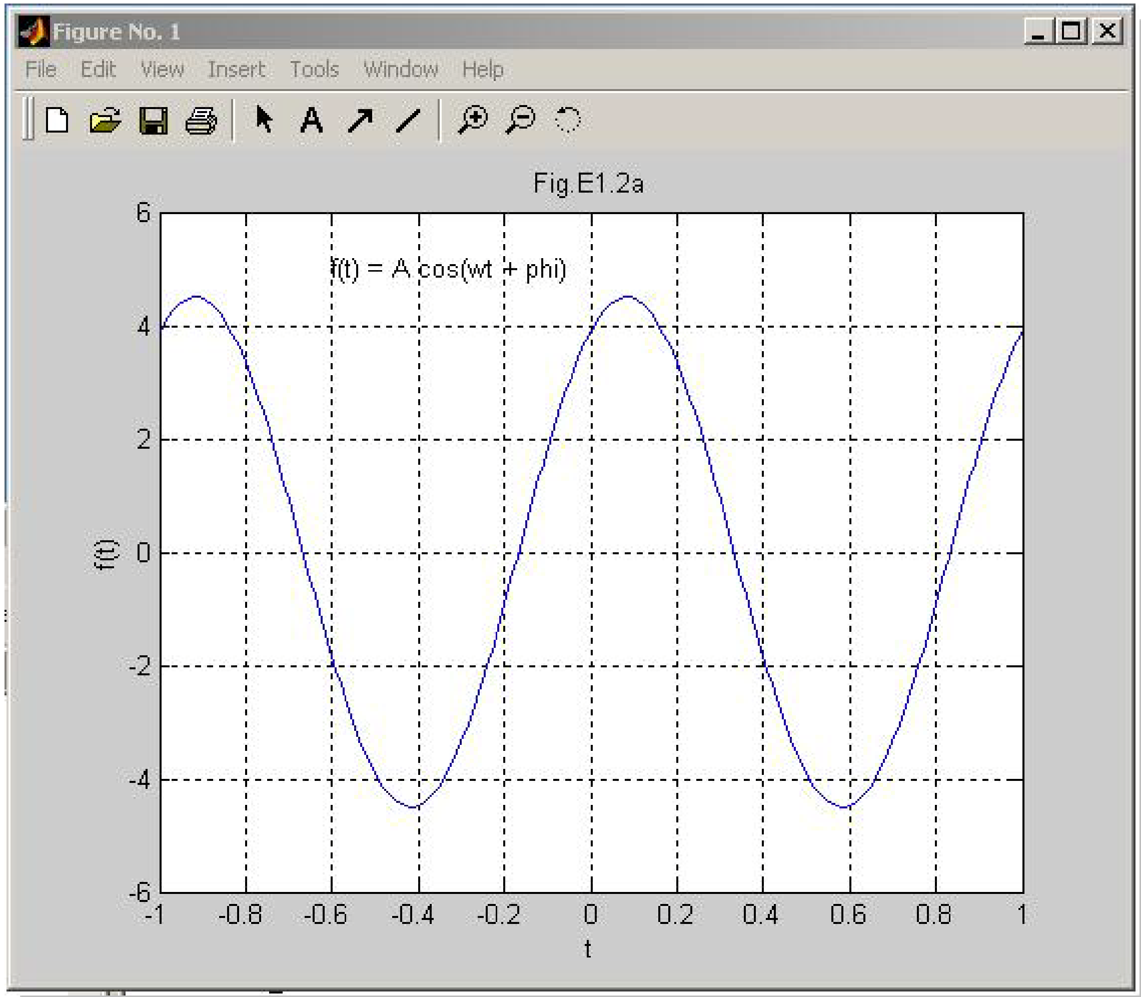
```
>>%Example
>>t=-1:0.01:1;
>>f=4.5*cos(2*pi*t - pi/6);
>>%The following statements plot the sequence and label the plot
>>plot(t,f),title('Fig.E1.2a');
>>axis([-1,1,-6,6]);
>>xlabel('t');
>>ylabel('f(t)');
>>text(-0.6,5,'f(t) = A cos(wt + phi)');
>>grid;
```

Plot on next page

MATLAB Basics

Plotting Elementary Functions:

```
>>%Example E1_2a
```



MATLAB Basics

Plotting Elementary Functions:

PLOT(X,Y)	plots vector Y versus vector X.
TITLE('text')	adds text at the top of the current plot.
XLABEL('text')	adds text beside the X-axis on the current axis.
YLABEL('text')	adds text beside the Y-axis on the current axis.
GRID,	by itself, toggles the major grid lines of the current axes.

GTEXT('string') displays the graph window, puts up a cross-hair, and waits for a mouse button or keyboard key to be pressed.

SUBPLOT(m,n,p), or SUBPLOT(mnp), breaks the Figure window into an m-by-n matrix of small axes.

stemDiscrete sequence or "stem" plot. STEM(Y) plots the data sequence Y as stems from the x axis terminated with circles for the data value.

SEMILOGX(...) is the same as PLOT(...), except a logarithmic (base 10) scale is used for the X-axis.

SEMILOGY(...) is the same as PLOT(...), except a logarithmic (base 10) scale is used for the Y-axis..

MATLAB Basics

Plotting Elementary Functions:

By default, the axes are auto-scaled.

This can be overridden by the command axis. If c = [xmin,xmax,ymin,ymax] is a 4-element vector, then axis(*c*) sets the axis scaling to the prescribed limits.

By itself, axis freezes the current scaling for subsequent graphs; entering axis again returns to auto-scaling.

The command axis('square') ensures that the same scale is used on both axes.

For more information's on axis see help axis. .

MATLAB Basics

Plotting Elementary Functions:

- >>%Example 1.2
- >>t=-0.5:0.01:3;
- >>t0=0
- >>u=stepfun(t,t0)
- >>gprime=3.17*exp(-1.3*t).*cos(10.8*t + 1.15).*u;

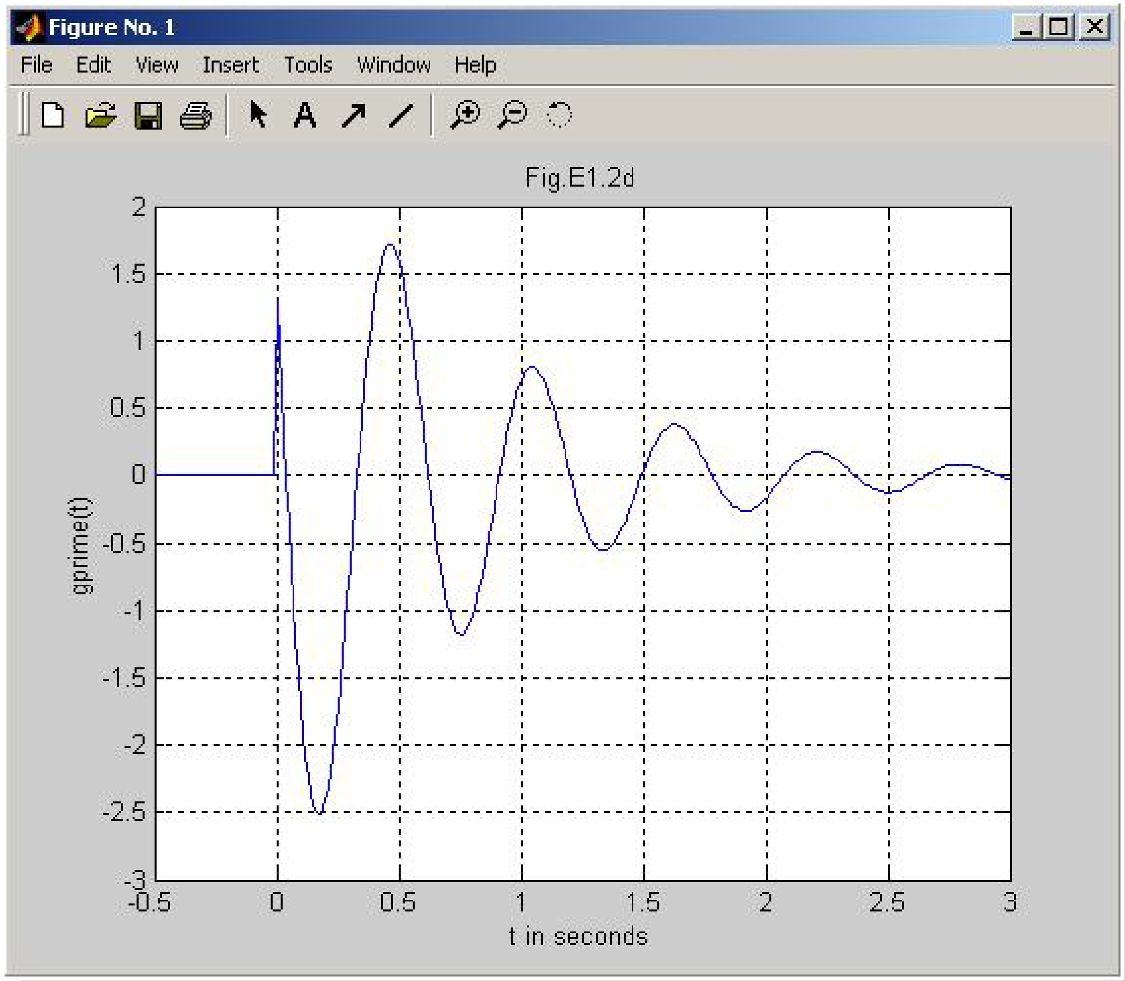
% NOTE the use of the .* operator. The terms 3.17*exp(-1.3*t),
% cos(10.8*t + 1.15), and u are all vectors. We want the
% components of these vectors to be multiplied by the corresponding
% components of the other vectors, hence the need to use .* rather than *.
% The following statements plot the sequence and label the plot

- >>plot(t,gprime);
- >>axis([-.5,3,-3,2]);
- >>title('Fig.E1.2d');
- >>xlabel('t in seconds');
- >>ylabel('gprime(t)');
- >>text(-0.6,5,'f(t) = A cos(wt + phi)');
- >>grid;

MATLAB Basics

Plotting
Elementary
Functions:

>>%Example E1.2



MATLAB Basics

Plotting Elementary Functions:

Two ways to make multiple plots on a single graph are illustrated by

- >>t = 0:.01:2*pi;
- >>y1 = sin(t); y2=sin(2*t); y3=sin(4*t)
- >>plot(t,y1,y2,y3)
- and by forming a matrix Y containing the functional values as columns
- >>t = 0:.01:2*pi;
- >>y = [sin(t)', sin(2*t)', sin(4*t)']
- >>plot(t,y)
- Another way is with the hold command. The command hold freezes the current graphics screen so that subsequent plots are superimposed on it. Entering hold again releases the "hold". The commands hold on and hold off are also available.
- One can override the default linotypes and point types. For example,
- >>t = 0:.01:2*pi;
- >>y1 = sin(t); y2=sin(2*t); y3=sin(4*t)
- >>plot(t,y1,'--',y2,':',y3,'+')

MATLAB Basics

Plotting Elementary Functions:

Colors		Line Styles	
y	yellow	.	point
M	magenta	o	circle
C	cyan	x	x-mark
R	red	+	plus
G	green	-	solid
B	blue	*	star
W	white	:	dotted
K	black	-.	Dashdot
		--	dashed

More mark types are; square(s), diamond(d), up-triangle(v), down-triangle(^), left-triangle(<), right-triangle(>), pentagram(p), hexagram(h)

See also help plot for more line and mark color.