The Islamia University of Bahawalpur

**U**niversity **C**ollege of **E**ngineering **&T**echnology **D**epartment of **C**omputer **S**ystem **E**ngineering

|  |  |  |
| --- | --- | --- |
| **LAB MANUAL** | **SIGNALS AND SYSTEMS EE-311** | **4thSemester** |

**LAB EXPERIMENT # 03**

**Performing operations on continuous and discrete time signals in MATLAB**

|  |  |
| --- | --- |
| **Student Name: Ahsan Tariq** | **Roll No: 20-CSE-26** |
| **Lab Instructor Signatures:** | **Date:** |

# OBJECTIVE:

To perform time reversal (flip), time shift (advance & delay),converting signal into odd and even components, time scaling (signal expansion-compression) for continuous-time signals in MATLAB.

1. **SIGNAL TRANSFORMATIONS:**
   1. **Time Reversal (Signal Flipping):**

y(t) = x(−t)is a time-reversed version of x(t), horizontally symmetric with respect to the origin t = 0.

# MATLAB IMPLEMENTATION:

**Code:**

%% Time Reversal (Signal Flip)

% time reversal being applied by if-else loop to introduce decision loops. Moreover with this scheme, we can perform signal flip whether signal defined as column or row vector.

t=-10:0.002:10;% time vector t0=2;% start position of the signal x=double((t-t0>=0) &(t<6)); plot(t,x);

hold on;

[r c]=size(x); if c==1

y=flipud(x);% Returns X with coloumns preserved and rows flipped in up/down direction.

else

y=fliplr(x);% Flip matrix in left/right direction.

end

[r c]=size(t); if c==1

t=-flipud(t); else

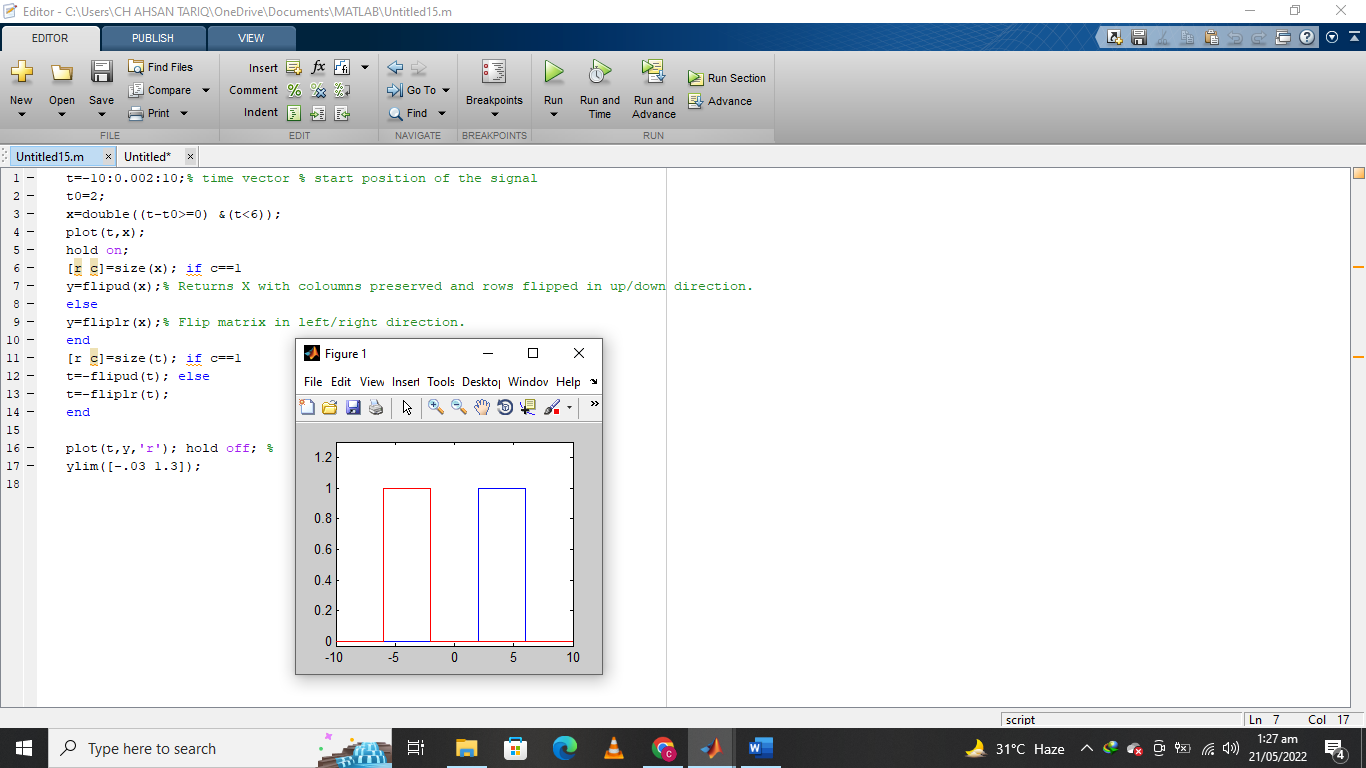
t=-fliplr(t);

end

plot(t,y,'r'); hold off; %

ylim([-.03 1.3]);

# OUTPUT:

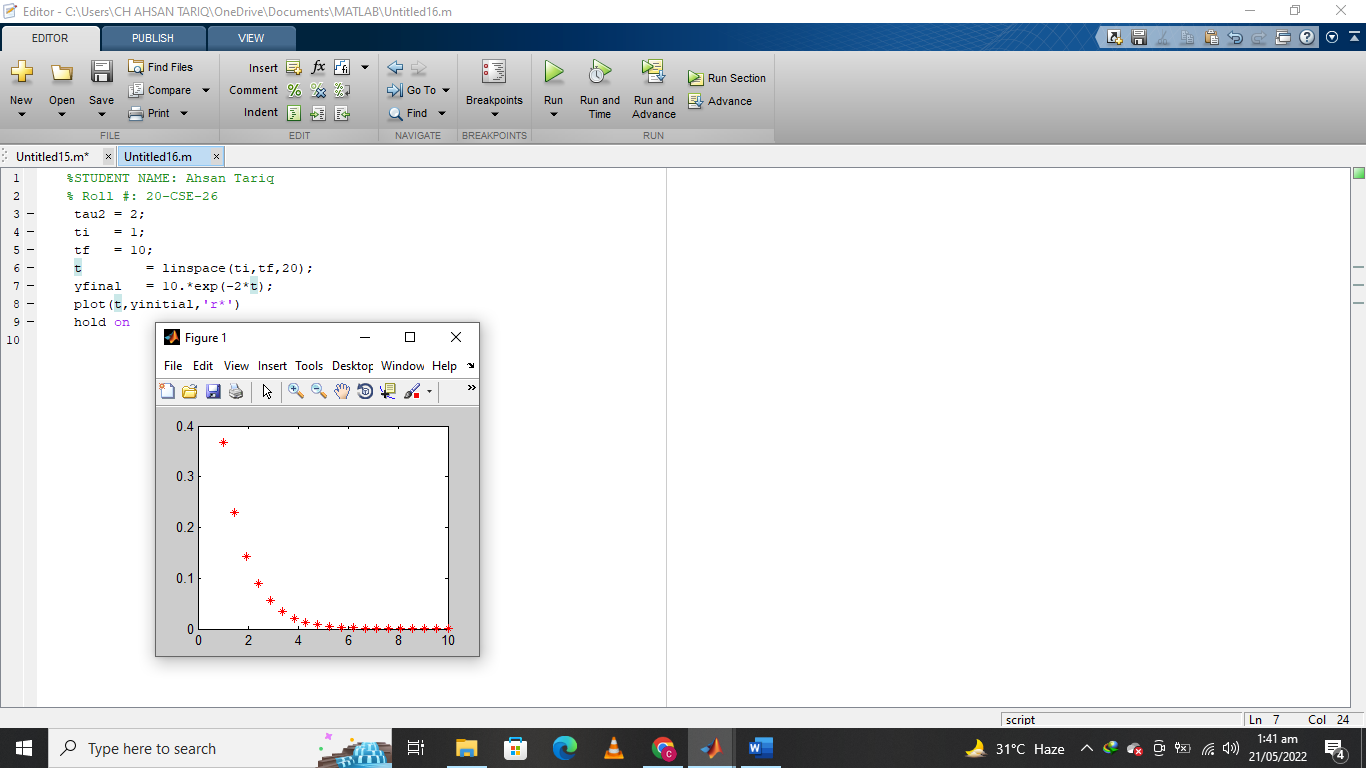


**Task 1:** Modify the above code to create a decaying real exponential function, (𝑡) = 10𝑒−2𝑡, where t= 0-to-10 with 1000 points and then time reversed (flip) this new signal.

# MATLAB CODE:

|  |
| --- |
| tau2 = 2;  ti = 1;  tf = 10;  t = linspace(ti,tf,20);  yfinal = 10.\*exp(-2\*t);  plot(t,yinitial,'r\*')  hold on |

**OUTPUT:**



# Time Shifting:

𝑦(𝑡) = 𝑥(𝑡 − 𝑡𝑜)is either right-shifted if 𝑡𝑜 > 0 , or left-shifted if 𝑡𝑜 < 0 .

%% Shifting the signal t3=-10:0.002:10; t0=0;

x3=double(t3-t0>=0);% convert to double precision. plot(t3,x3);

hold on; k=-2;

t=t3+k; y=x3;

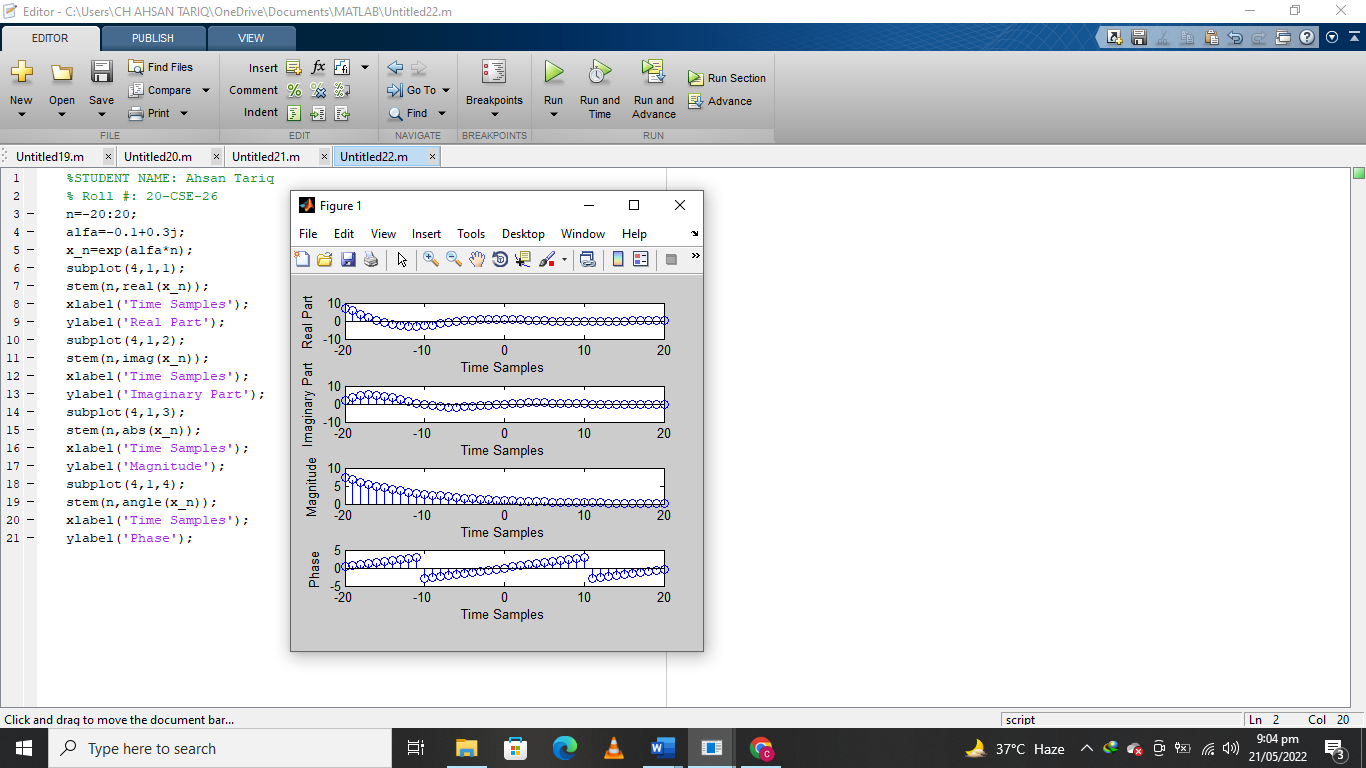
plot(t,y,'r-'); hold off;

ylim([-.03 1.3]);

**OUTPUT:**

|  |
| --- |
|  |

**Task 2:** Generate decaying real exponential function, 𝑥(𝑡) = 10𝑒−2𝑡 , where t= 0-to-10 with 1000 points and then shift it 5 units to right and 3 units to left and show generated and right and left shifted signals on the same figure as subplots.



# Even-Odd Functions:

Even functions and odd functions are [functions](https://en.wikipedia.org/wiki/Function_(mathematics)) which satisfy particular [symmetry](https://en.wikipedia.org/wiki/Symmetry) relations, with respect to taking [additive inverses](https://en.wikipedia.org/wiki/Additive_inverse).

**Even Function:** Let *x*(t) be a [real](https://en.wikipedia.org/wiki/Real_number)-valued function of a real variable t.

Then x(t) is **even** if the following equation holds for all t and *-t* in the domain of x.

𝑥(𝑡) = 𝑥(−𝑡)

**Odd Function:** Again, let x(t) be a [real](https://en.wikipedia.org/wiki/Real_number)-valued function of a real variable t.

Then *x(t)* is **odd** if the following equation holds for all t and *-t* in the domain of x(t).

𝑥(𝑡) = −𝑥(−𝑡)

Any signal can be decomposed into even and odd components xe (t) = 1/ 2 [x(t) + x(−t)]

xo(t) = 1/ 2 [x(t) − x(−t)] .

# Code:

% Converting of a signal into its even-odd components

t3=-10:0.002:10; t0=2;

x3=double(t3-t0>=0); plot(t3,x3);

hold on; negx=fliplr(x3); plot(t3,negx,'r-')

ylim([-.03 1.3]); hold off; figure(2)

subplot(211) evx=(x3+negx)/2; odx=(x3-negx)/2; plot(t3,evx,'b--');

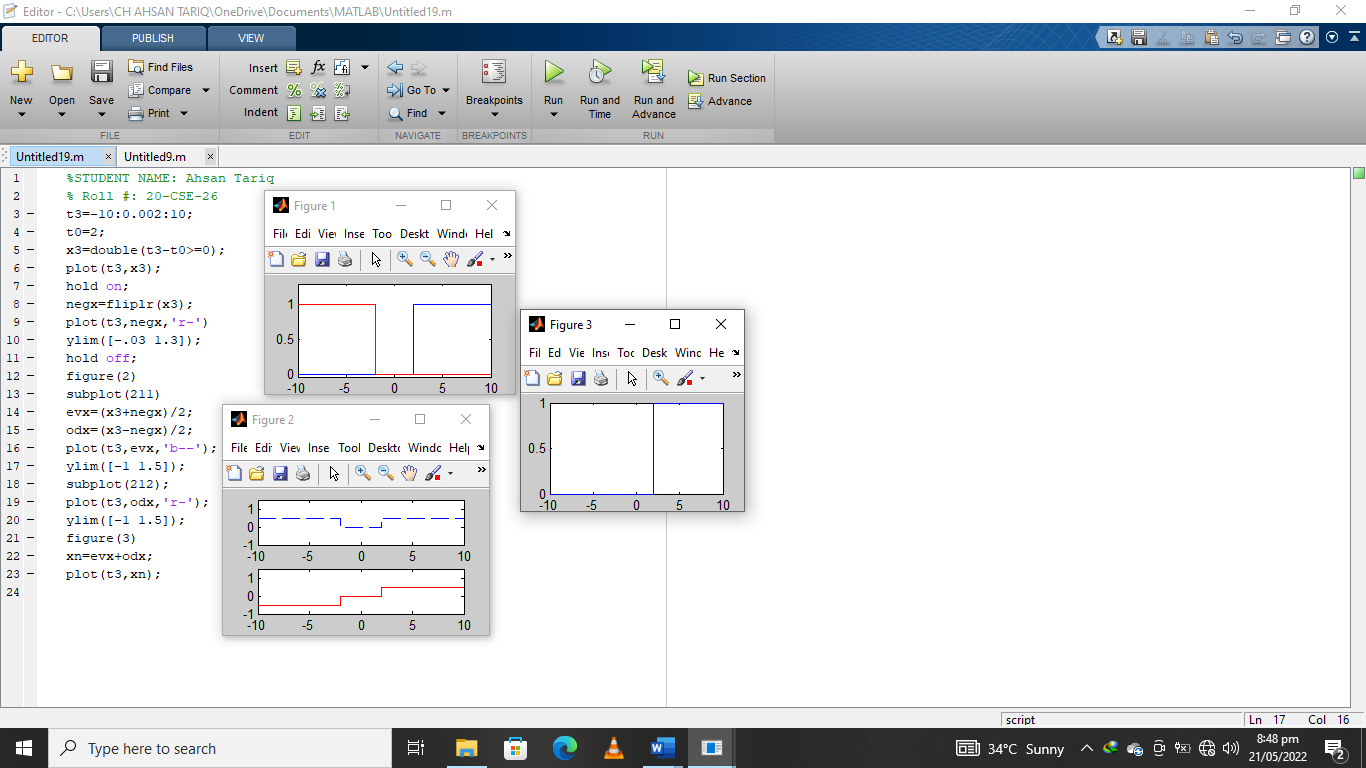
ylim([-1 1.5]);

subplot(212); plot(t3,odx,'r-');

ylim([-1 1.5]);

figure(3) xn=evx+odx; plot(t3,xn);

**Output:**



* 1. **Signal Scaling (Compression and Expansion)**

A signal x(t) is scaled in time by multiplying the time variable by a positive constant b, to produce x(bt). A positive factor of b either expands (0 < b < 1) or compresses (b > 1) the signal in time.

% Inline (EXPR) constructs an inline function object from the Matlab expression contained in string expression.

# Code:

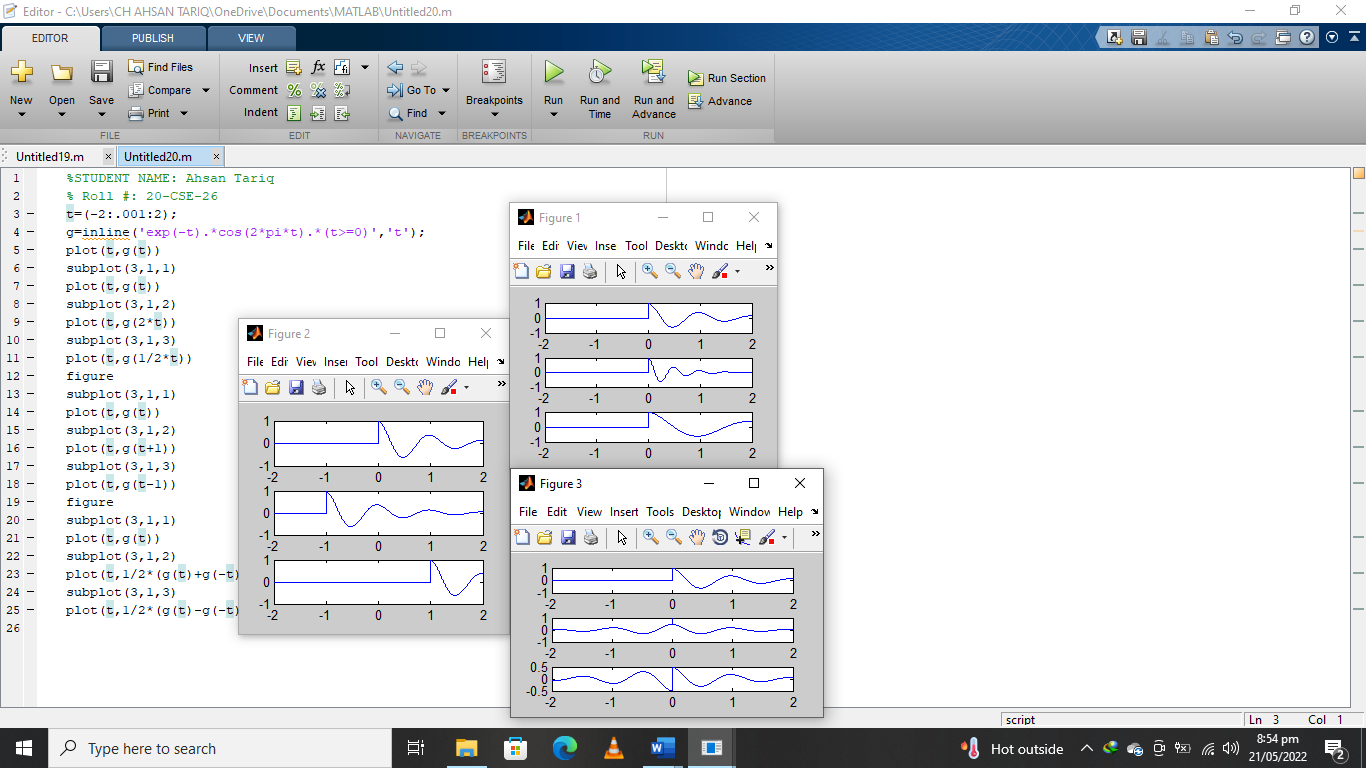
t=(-2:.001:2);

g=inline('exp(-t).\*cos(2\*pi\*t).\*(t>=0)','t'); plot(t,g(t))

subplot(3,1,1) plot(t,g(t)) subplot(3,1,2) plot(t,g(2\*t)) subplot(3,1,3) plot(t,g(1/2\*t)) figure subplot(3,1,1) plot(t,g(t)) subplot(3,1,2) plot(t,g(t+1)) subplot(3,1,3) plot(t,g(t-1)) figure subplot(3,1,1) plot(t,g(t)) subplot(3,1,2)

plot(t,1/2\*(g(t)+g(-t))) subplot(3,1,3) plot(t,1/2\*(g(t)-g(-t)))

**OUTPUT:**



**Task 3:** Generate, (𝑡) = 10𝑒−2𝑡 , where t= 0-to-10 with 1000 points and then expand it 3 times and compress it 2 times. Moreover show generated and time scaled signals on the same figure as subplots.

# Periodic Signals Code:

%% Generation of Periodic Sequence

% x(t)=x(t+T)

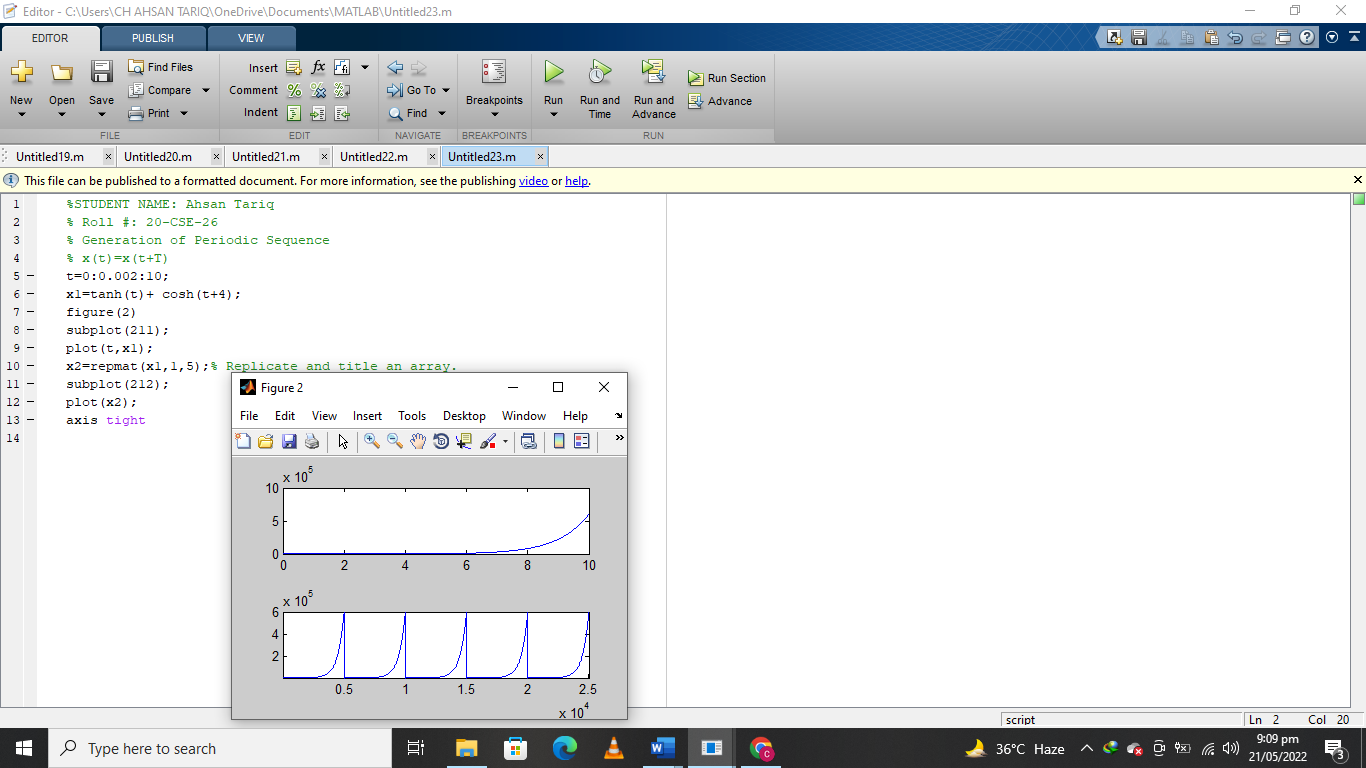
t=0:0.002:10;

x1=tanh(t)+ cosh(t+4); figure(2) subplot(211); plot(t,x1);

x2=repmat(x1,1,5);% Replicate and title an array. subplot(212);

plot(x2); axis tight

**OUTPUT:**



# Differentiation

To illustrate how to take derivatives using the Symbolic Math Toolbox, first create a symbolic expression:

**Example 1**

symsx

f =sin(5\*x)

The command **diff** (f)differentiates f with respect to x:

**Output** = 5\*cos(5\*x)

**Example 2:**

Find diff\_f =dx /df

where f = e-axx3bsin(cx) and a, b and c are unspecified constants,

>>clear;

>>syms a b c x;

>>f=exp(-a\*x)\*x^(3\*b)\*sin(c\*x);

>>diff\_f = diff(f,x)

# Output=

diff\_f =

(3\*b\*x^(3\*b-1)\*sin(c\*x))/exp(a\*x)+(c\*x^(3\*b)\*cos(c\*x))/exp(a\*x) (a\*x^(3\*b)\*sin(c\*x))/exp(a\*x)

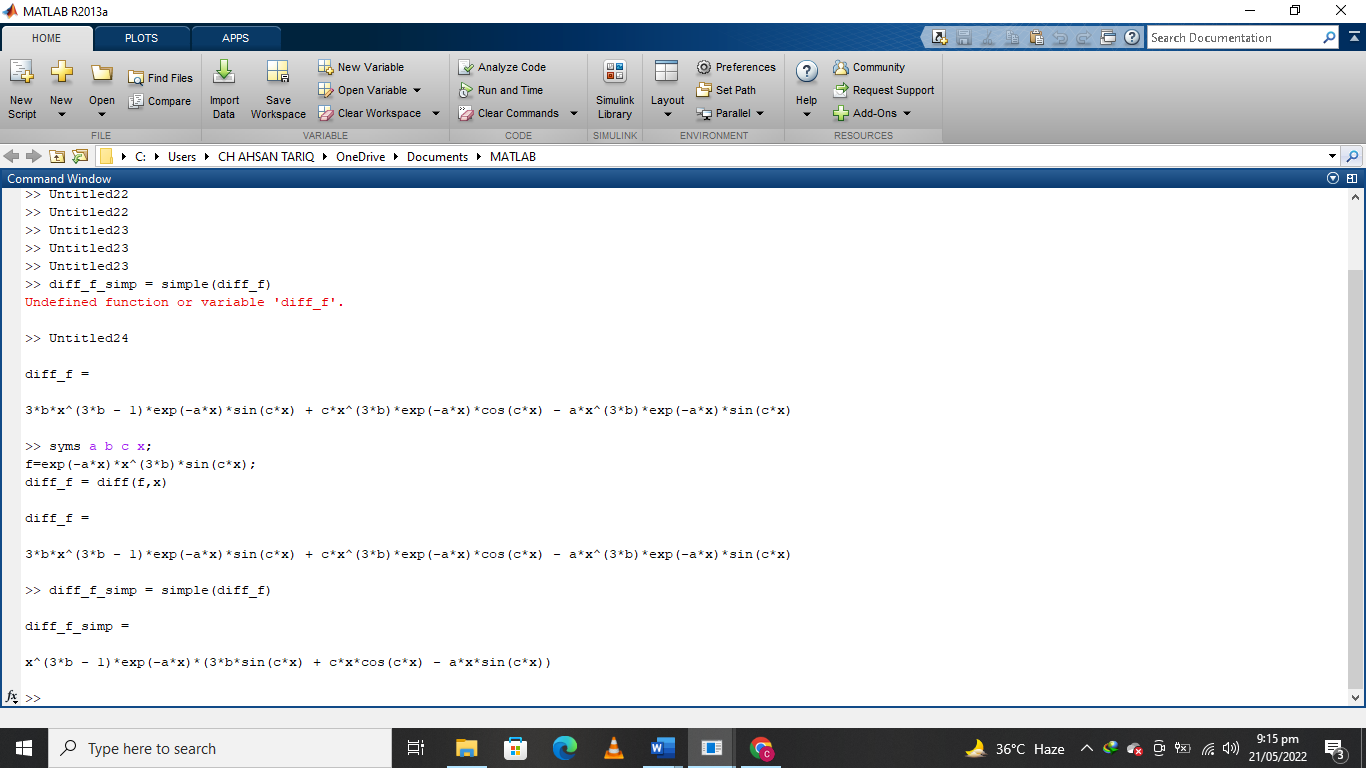
-

simplify it,

**>>**diff\_f\_simp = simple(diff\_f)

Output**=**

diff\_f\_simp = (x^(3\*b - 1)\*(3\*b\*sin(c\*x) + c\*x\*cos(c\*x) - a\*x\*sin(c\*x)))/exp(a\*x)



# Integration:

Let f is a symbolic expression, then int(f) returns the indefinite integral or antiderivative of f.

**Indefinite integrals**: For int\_g = ʃ gdx where g = e^-axsin(cx),

>>clear;

>>syms a c x;

>>g=exp(-a\*x)\*sin(c\*x);

>>int\_g = int(g,x)

+ ח

**Definite integrals**: int\_def\_g= ʃ gdx

- -ח

>>clear;

>>syms a c x;

>>g = exp(-a\*x)\*sin(c\*x);

>>int\_def\_g = int(g,x,-pi,pi)

# HOMEWORK PROBLEMS:

1. Decompose the following signal into its even and odd parts

[ ] 1; 𝑛 ≥ 0

𝑥 𝑛 = {

0; 𝑛 < 0

Ans:



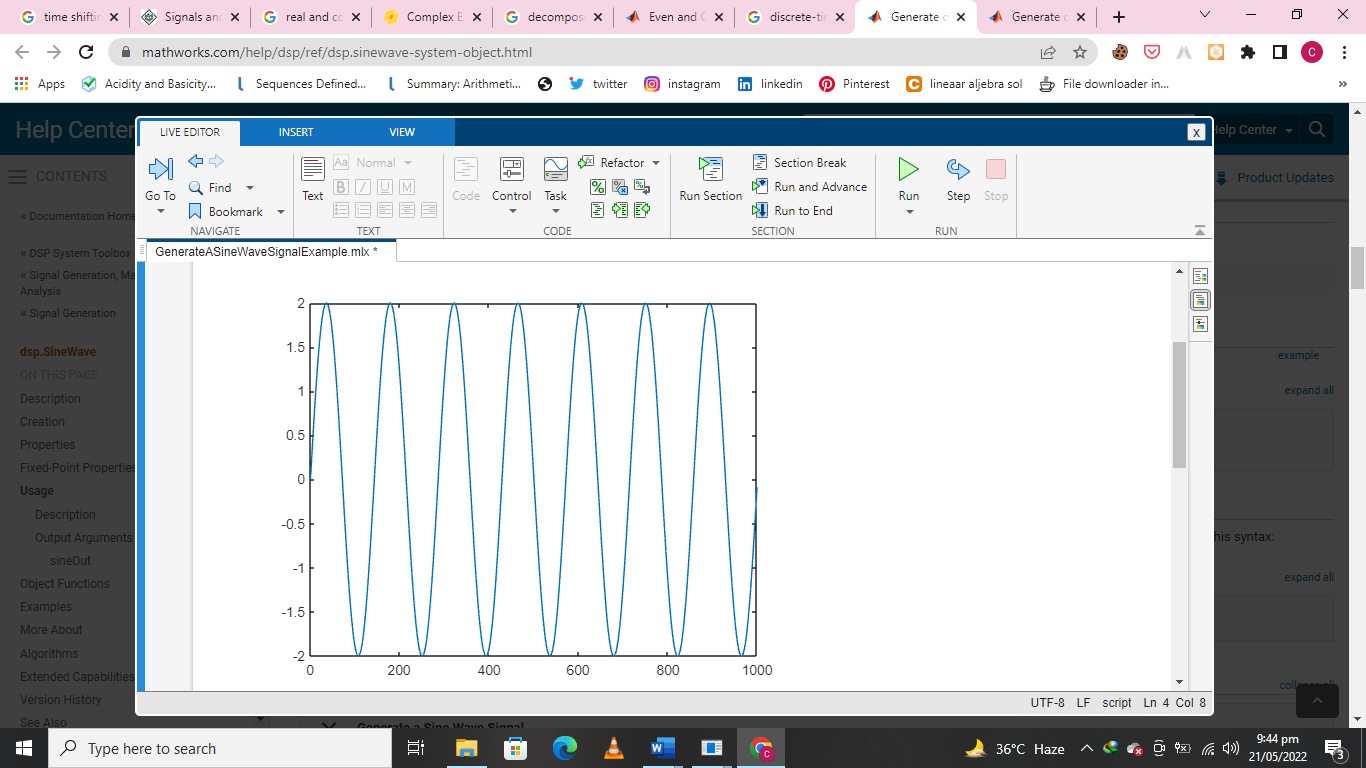
1. Use MATLAB to produce a plot of a discrete-time sinusoid with amplitude 2.0 and a period of T = 7.

sine1 = dsp.SineWave(2,10);

sine1.SamplesPerFrame = 1000;

y = sine1();

plot(y)



The Islamia University of Bahawalpur

**U**niversity **C**ollege of **E**ngineering **&T**echnology **D**epartment of **C**omputer **S**ystem **E**ngineering

|  |  |  |
| --- | --- | --- |
| **LAB MANUAL** | **SIGNALS AND SYSTEMS EE-311** | **4thSemester** |

**LAB EXPERIMENT # 04**

**Differentiation and Integration using symbolic functions in MATLAB**

|  |  |
| --- | --- |
| **Student Name: Ahsan Tariq** | **Roll No: 20-CSE-26** |
| **Lab Instructor Signatures:** | **Date:** |

## OBJECTIVE:

Calculation of Differentiation and Integration in Symbolic way and also calculate energy and power of a function + Convolution sum.

# Differentiation:

**Syntax**: diff(F)

diff([F](https://www.mathworks.com/help/symbolic/diff.html#inputarg_F)) differentiates F with respect to the variable determined by symvar(F,1)

To illustrate how to take derivatives using the Symbolic Math Toolbox, first create a symbolic expression:

**Example**: Find the derivative of the function Sin5x = ?

symsx;

f =sin(5\*x) df= diff(f,x)

**Output:**

5\*cos(5\*x)

**Example**: Find the derivative of the function sin(x^2).?

symsx y;

f = sin(x^2)\*y^2; df = diff(f,y)

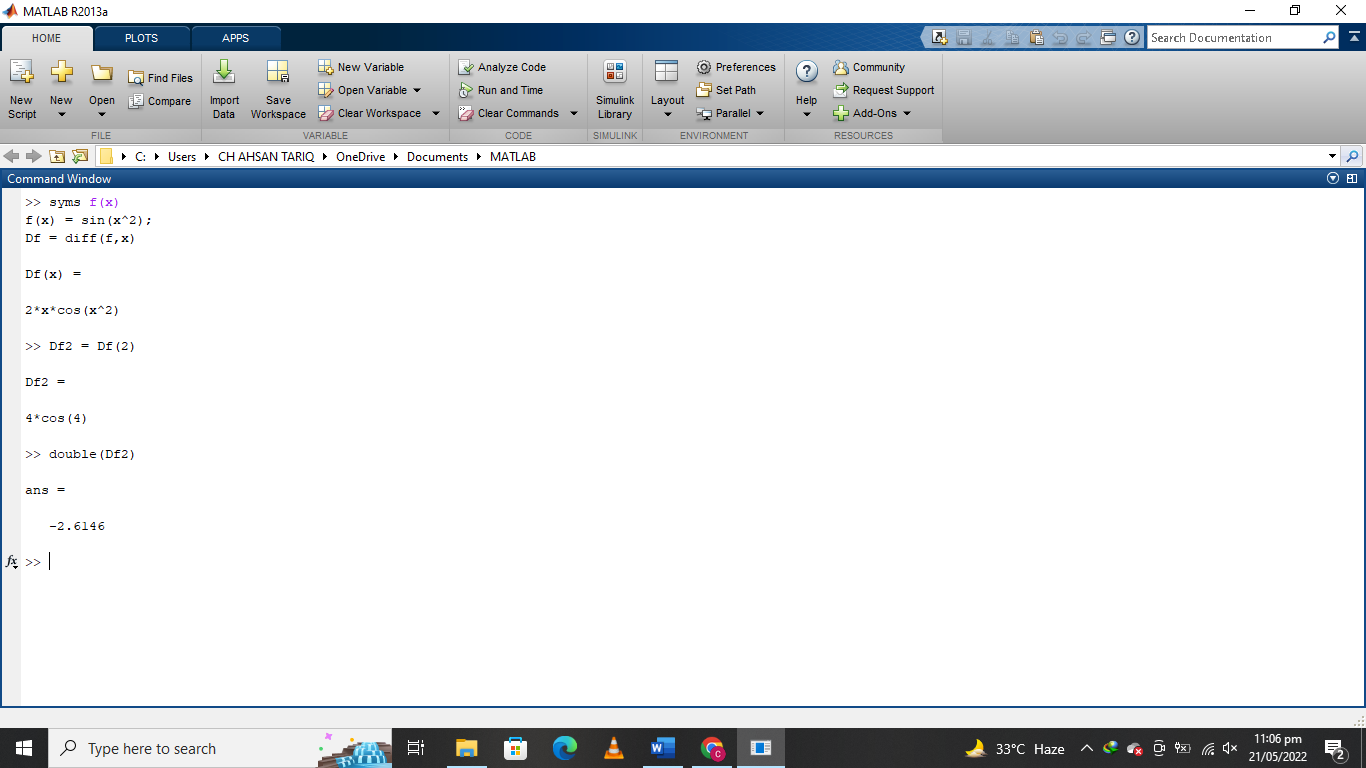
**Output=**

df(x) =2\*y\*sin(x^2)

Find the value of the derivative at x = 2. Convert the value to double.

df2 = subs(f, {y}, {3});

## output=



**Example**: Find diff\_f =dx /df

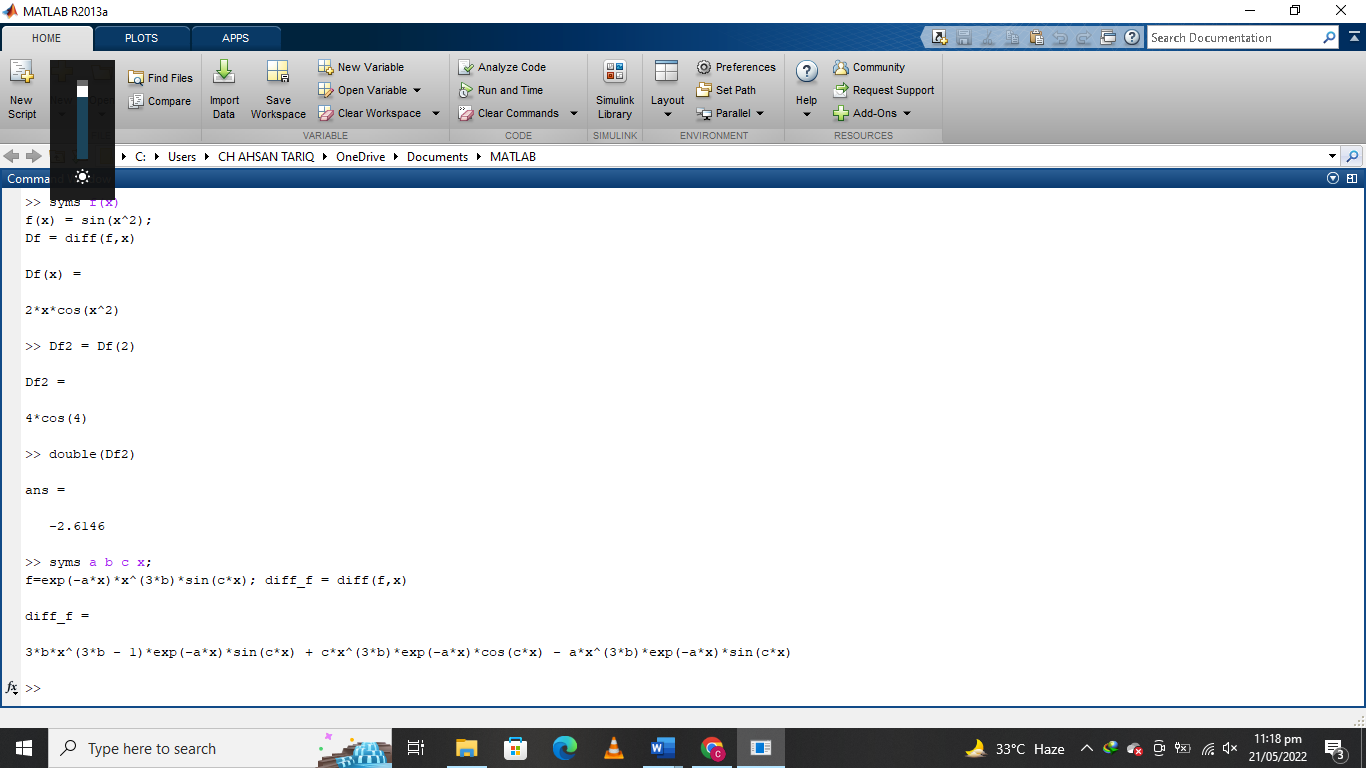
where f = e-axx3bsin(cx) and a, b and c are unspecified constants,

clear;

syms a b c x;

f=exp(-a\*x)\*x^(3\*b)\*sin(c\*x); diff\_f = diff(f,x)

**Output=**



# Integration:

Let f is a symbolic expression, thenint(f) returns the indefinite integral or anti-derivative

of f.

**Syntax:** int(expr,var)

int(expr,var,a,b)

int([expr](https://www.mathworks.com/help/symbolic/int.html" \l "inputarg_expr)[,var](https://www.mathworks.com/help/symbolic/int.html#inputarg_var)) computes the indefinite integral of expr with respect to the symbolic scalar variable var. Specifying the variable var is optional. If you do not specify it, int uses the default variable determined by [symvar.](https://www.mathworks.com/help/symbolic/symvar.html) If expr is a constant, then the default variable is x.

int([expr](https://www.mathworks.com/help/symbolic/int.html" \l "inputarg_expr)[,var](https://www.mathworks.com/help/symbolic/int.html#inputarg_var)[,a,](https://www.mathworks.com/help/symbolic/int.html#inputarg_a)[b](https://www.mathworks.com/help/symbolic/int.html#inputarg_b)) computes the definite integral of expr with respect to var from a to b. If you do not specify it, int uses the default variable determined by [symvar.](https://www.mathworks.com/help/symbolic/symvar.html) If expr is a constant, then the default variable is x.

Two types : A) Definite

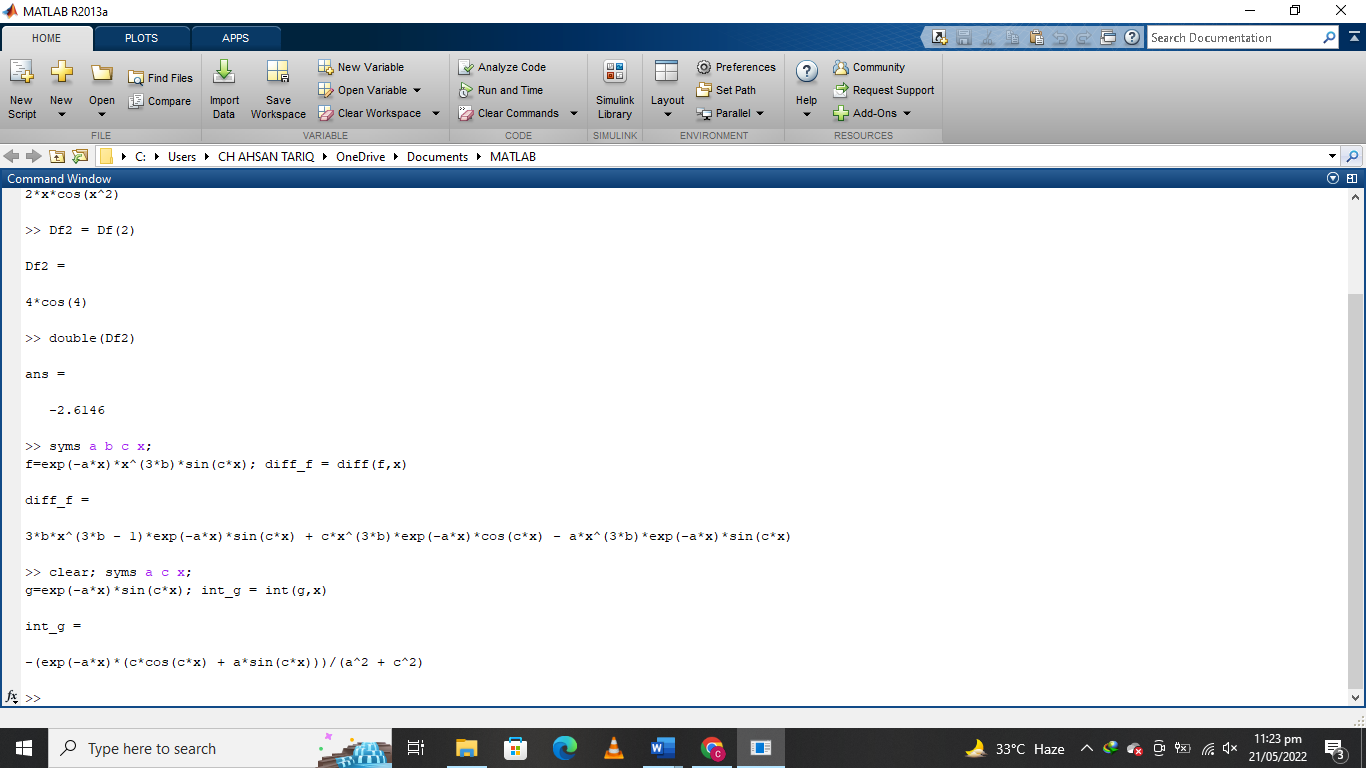
1. Indefinite integrals

## Indefinite integrals:

**Example**: Find int\_g = ʃ gdx where g = e^-axsin(cx),

clear; syms a c x;

g=exp(-a\*x)\*sin(c\*x); int\_g = int(g,x)



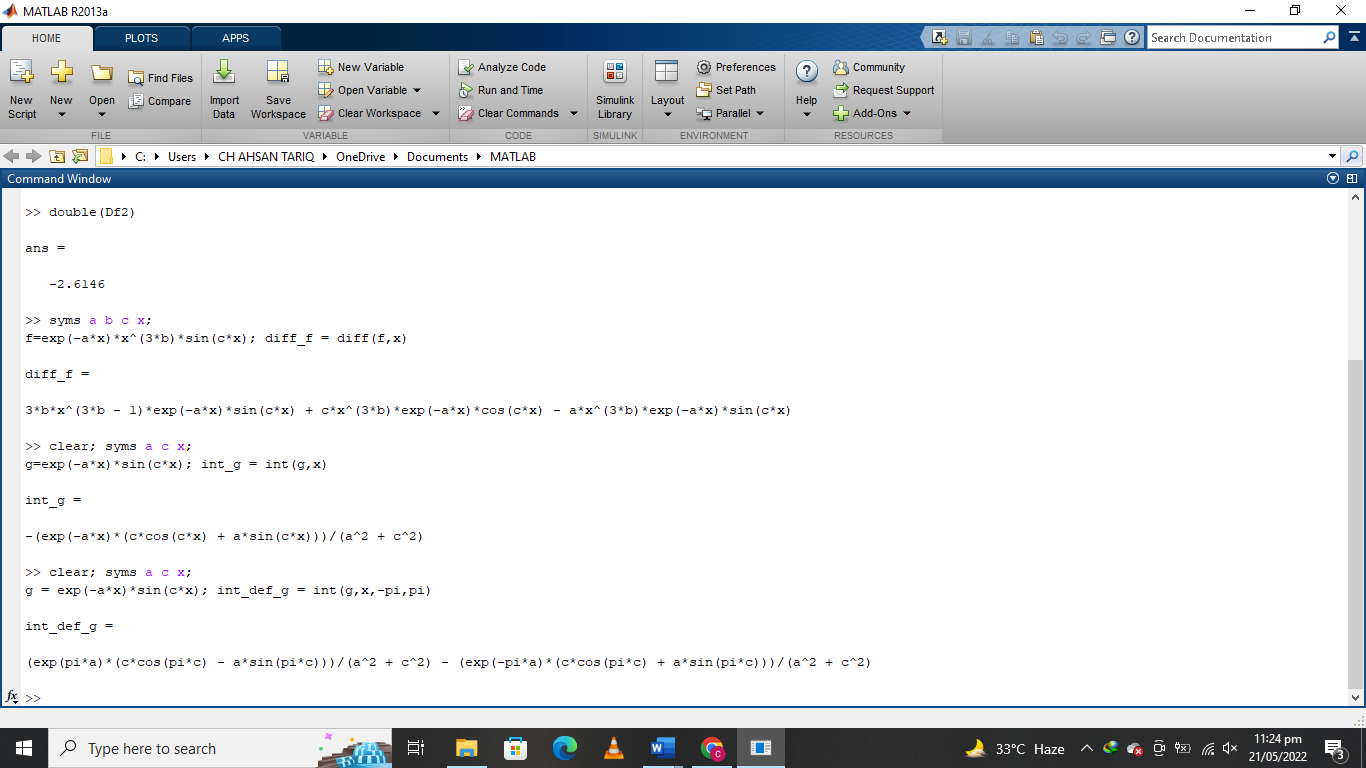
* 1. **Definite integrals**: **Example**: Find int\_def\_g= ∫+𝜋 𝑔𝑑𝑥

−𝜋

clear; syms a c x;

g = exp(-a\*x)\*sin(c\*x); int\_def\_g = int(g,x,-pi,pi)

**Output=**



**Example**: Find the roots λ1 and λ2 of the polynomial λ2 + 4λ + *k* for three values of *k* :

1. *k* = 3
2. *k* = 4
3. *k* = 40

## a)

>> r = roots ([1 4 3]);

>>disp (['Case (k=3): roots = [',num2str(r.'),']']);

>> y\_0 = dsolve('D2y + 4\*Dy + 3\*y = 0','y(0) = 3','Dy(0) = -7','t');

>>disp(['(a) k = 3; y\_0 = ',char(y\_0)])

**b)**

>> r = roots ([1 4 4]);

>>disp (['Case (k=4): roots = [',num2str(r.'),']']);

>> y\_0 = dsolve('D2y + 4\*Dy + 4\*y = 0','y(0) = 3','Dy(0) = -7','t');

>>disp(['(b) k = 4; y\_0 = ',char(y\_0)])

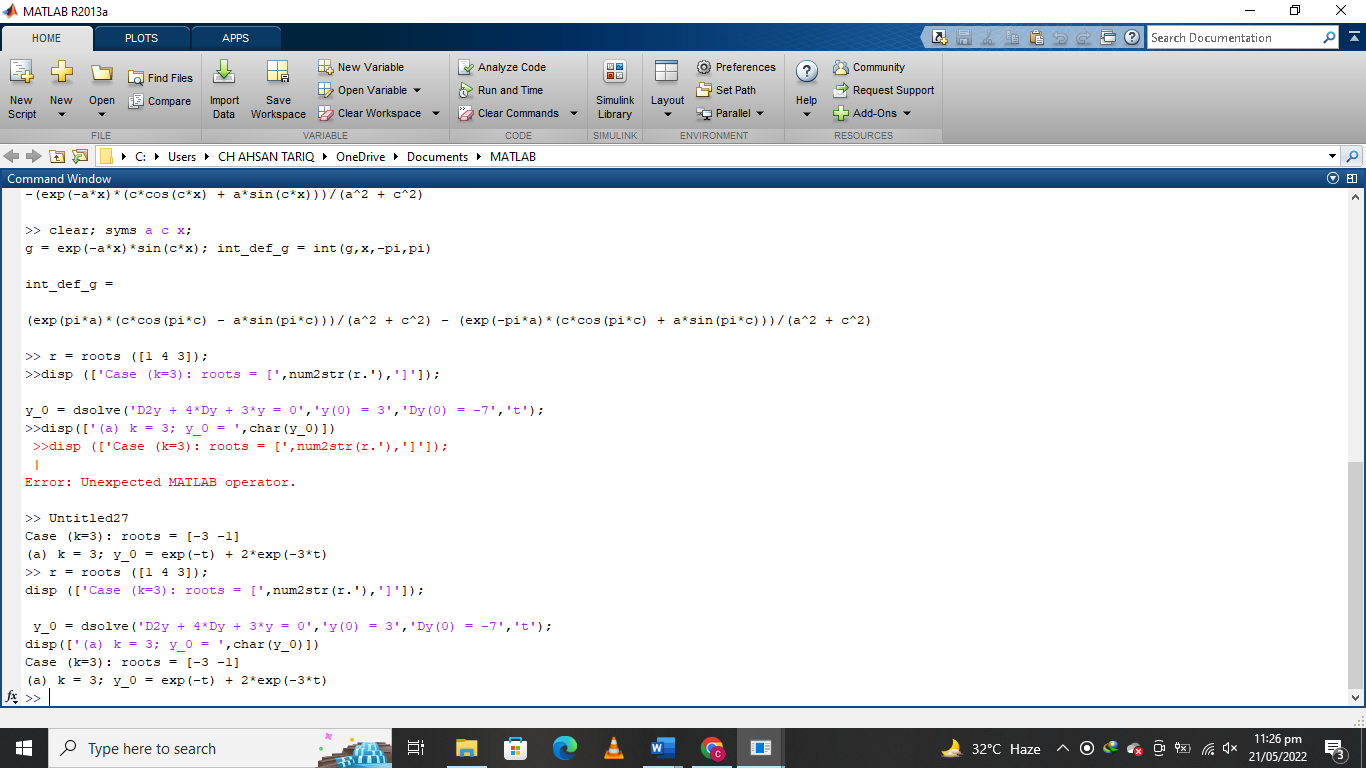
## c)

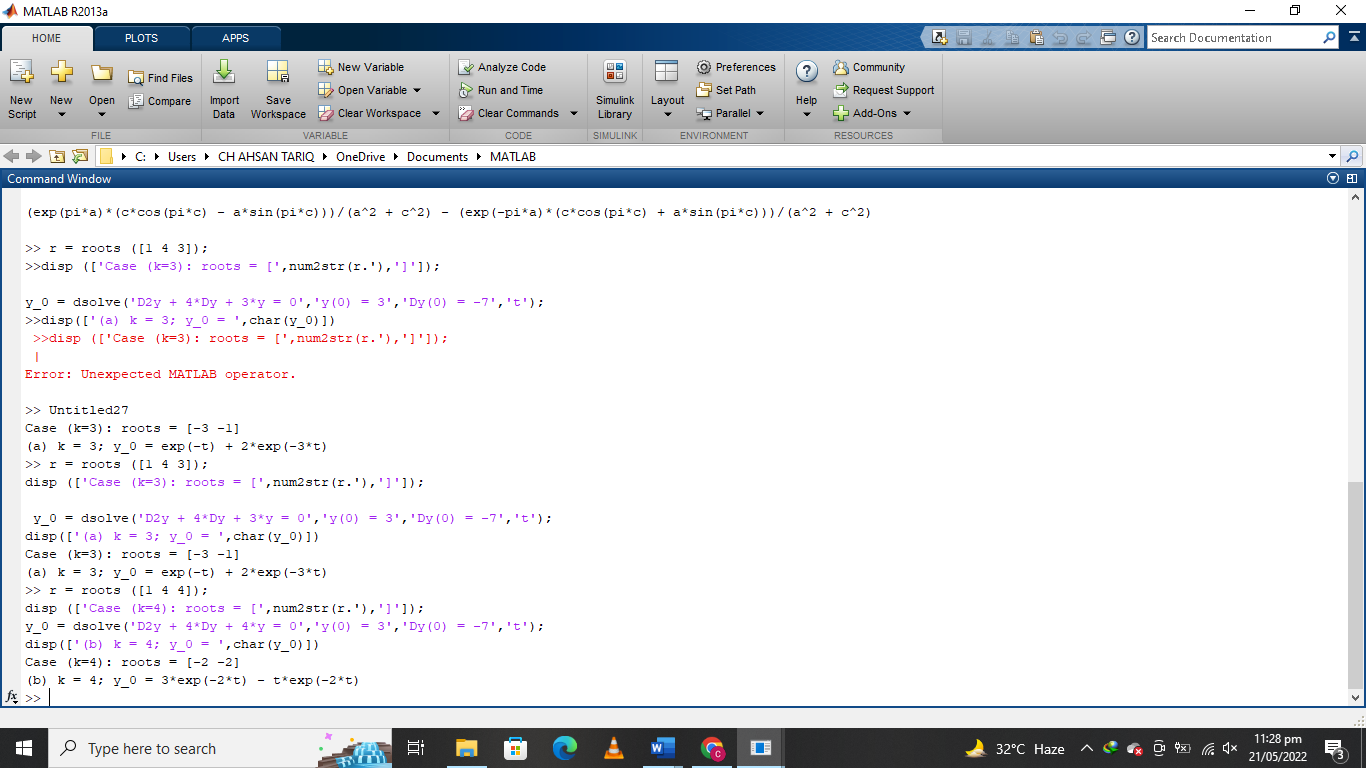
>> r = roots ([1 4 40]);

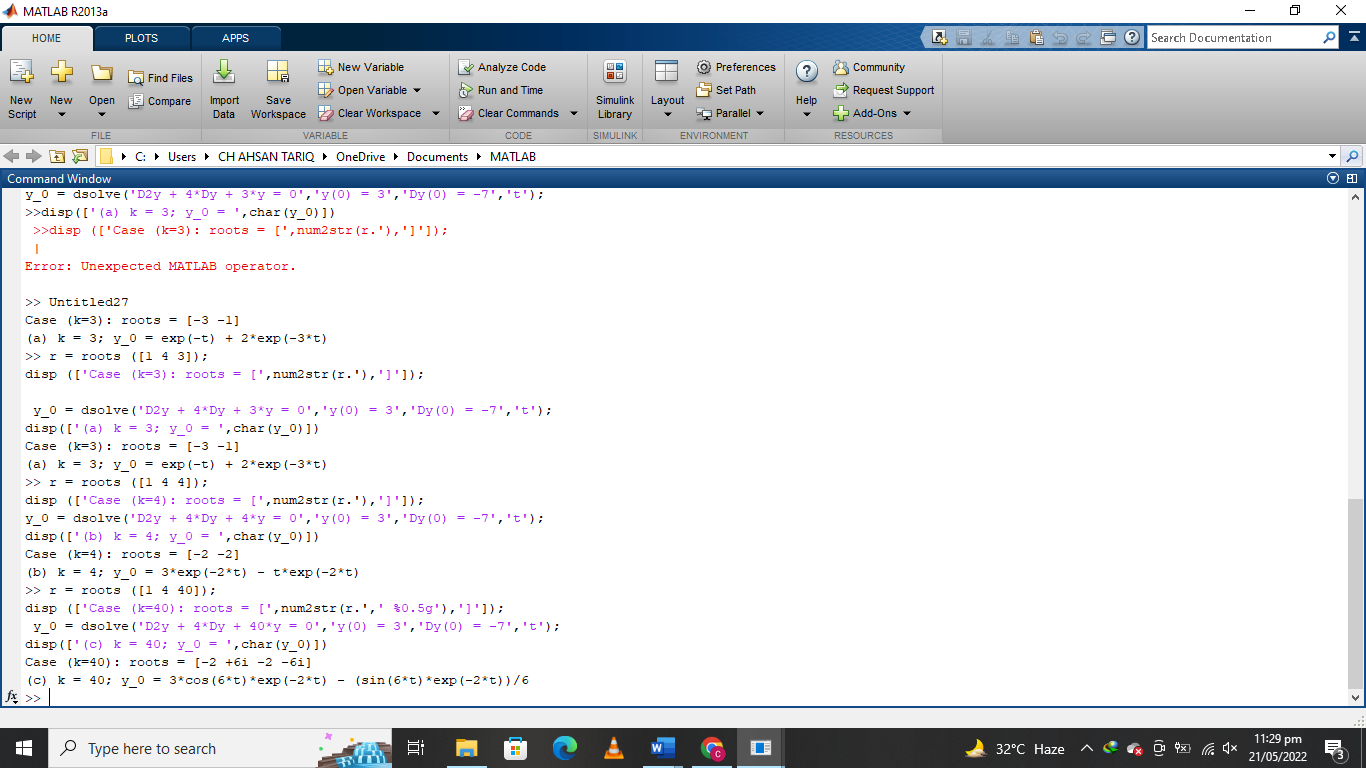
>>disp (['Case (k=40): roots = [',num2str(r.',' %0.5g'),']']);

>> y\_0 = dsolve('D2y + 4\*Dy + 40\*y = 0','y(0) = 3','Dy(0) = -7','t');

>>disp(['(c) k = 40; y\_0 = ',char(y\_0)])

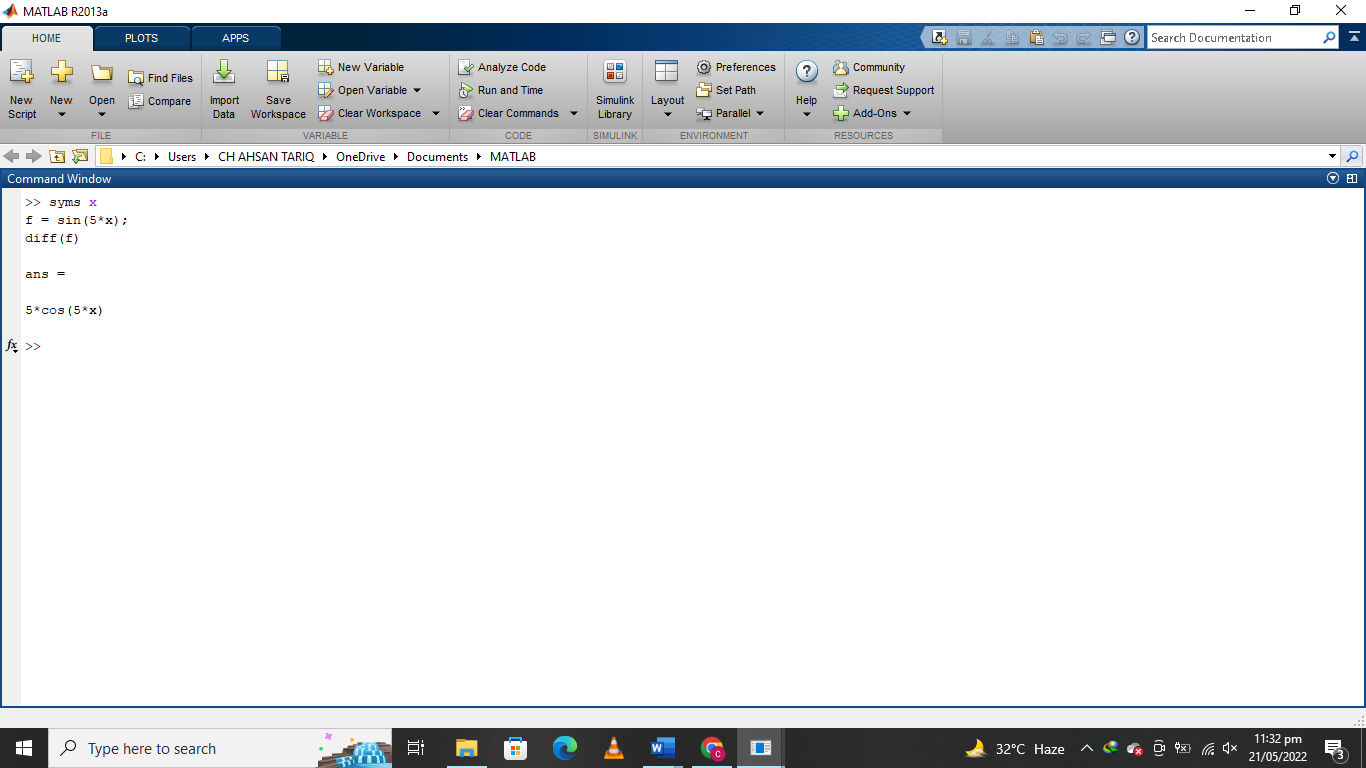




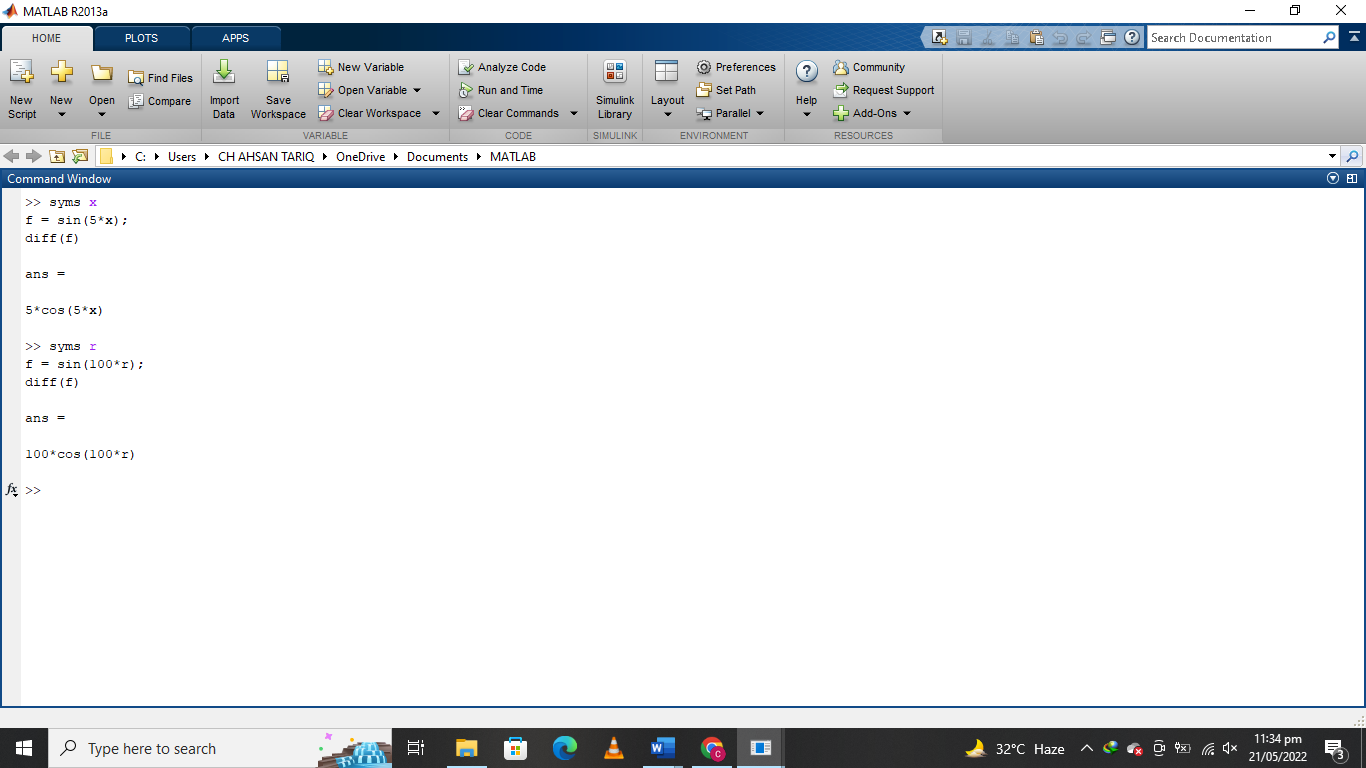


Q1.Write Code for finding derivatives of

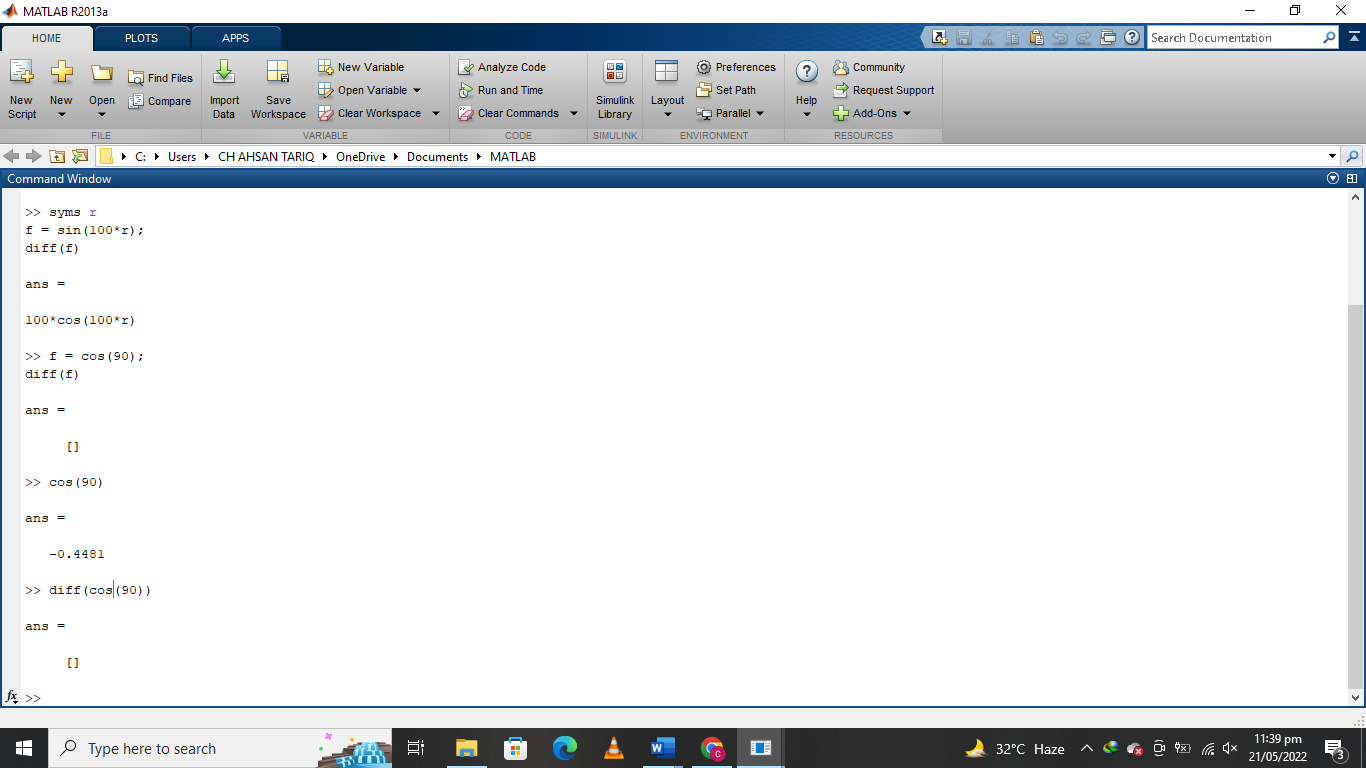
* + sin 5x,



* + sin 100𝑟,

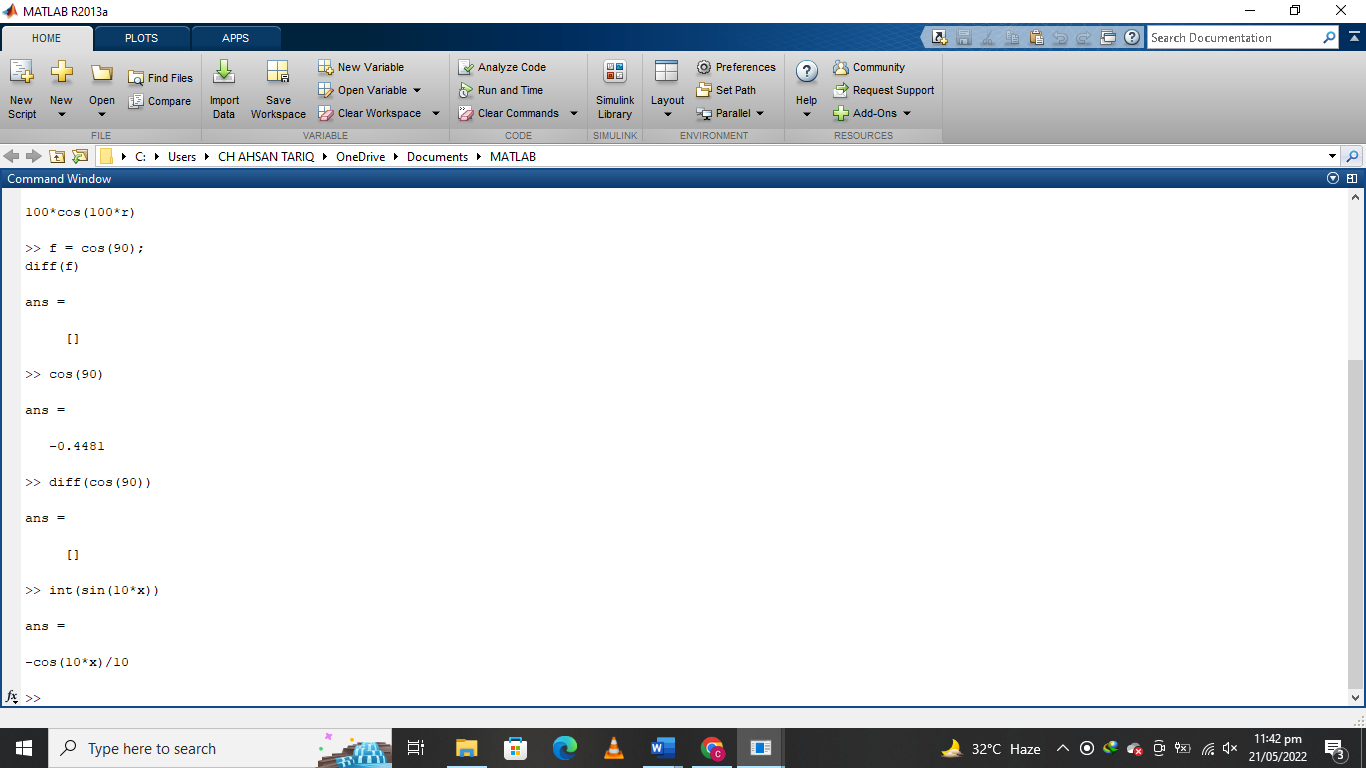


* + cos 90

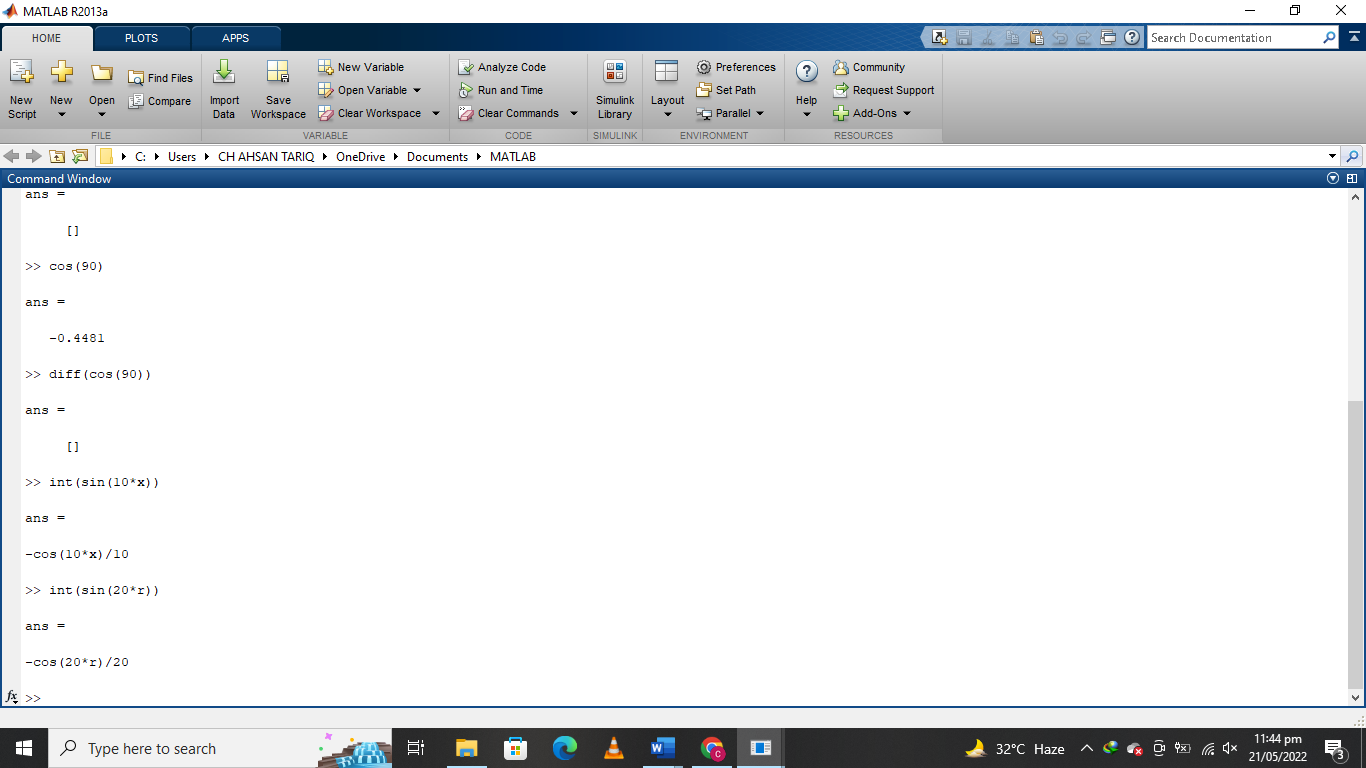


Q2. Write Code for finding integrals of

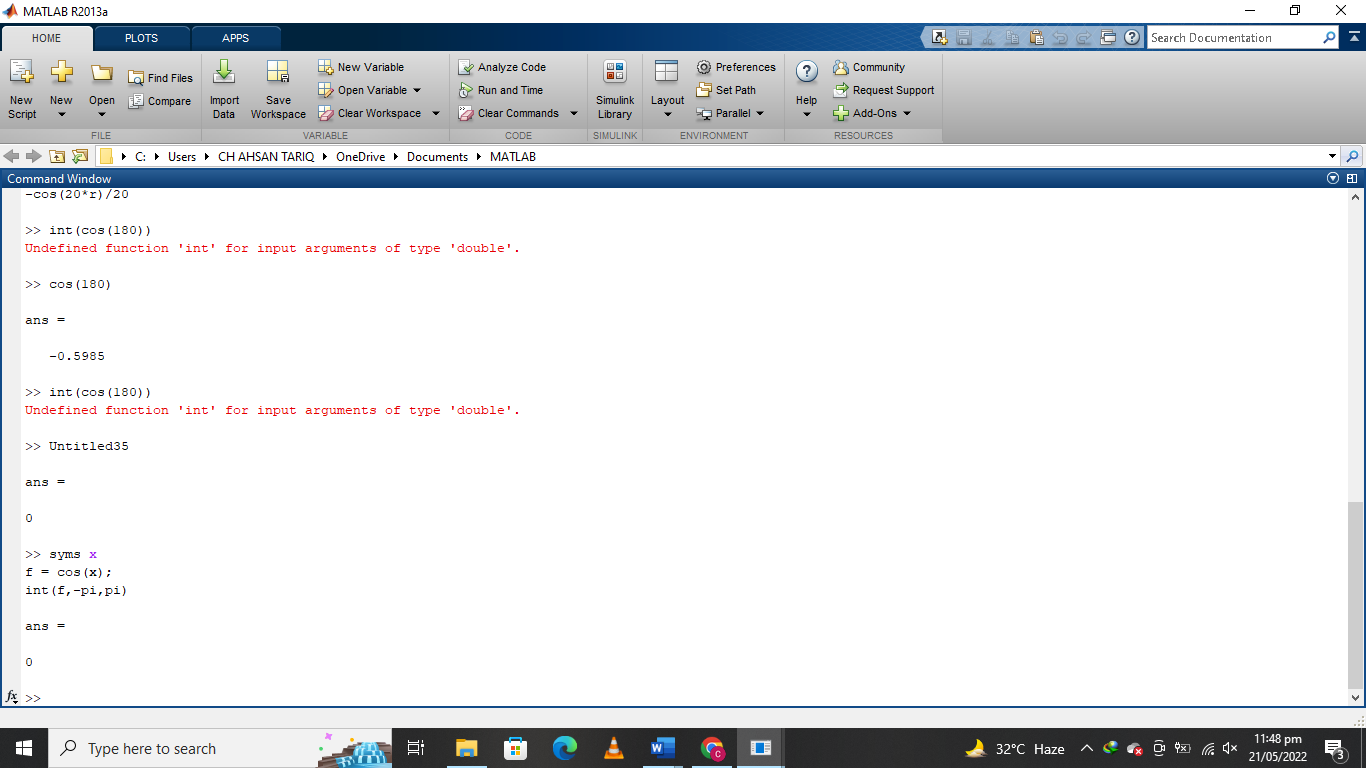
* + sin 10x,



* + sin 20𝑟,



* + cos 180



The Islamia University of Bahawalpur **U**niversity **C**ollege of **E**ngineering **&T**echnology **D**epartment of **C**omputer **S**ystem **E**ngineering

**LAB MANUAL SIGNALS AND SYSTEMS EE-311 4thSemester**

**LAB EXPERIMENT # 05**

**Computing convolution & correlation of signals in MATLAB**

**Student Name: Ahsan Tariq Roll No: 20CSE-26**

**Lab Instructor Signatures: Date:**

**OBJECTIVE:**

 To study convolution & correlation using MATLAB

1. **Convolution:**

A linear, time-invariant system h {.} is completely characterized by its impulse response. h (t)=T{δ(t)}. Convolution determines the system's output from knowledge of the input and the system's impulse response. This will result in the convolution integral and its properties

( ) ∫ ( ) ( ) ( )( )

A discrete time LTI system is completely characterized by its impulse response

[ ] ∑ [ ] [ ]

Where h[n] is impulse response of LTI systems where is applied to the system above equation is the convolution sum in DT Signal i.e.

[ ] [ ] [ ]

**Code:**

clc; clear all; close all;

figure (1); % Create figure window and make visible on screen

x = inline('1.5\*sin(pi\*t).\*(t>=0 & t<1)');

h = inline('1.5\*(t>=0&t<1.5) - (t>=2&t<2.5)');

dtau = 0.005; tau = -1:dtau:4;

ti = 0; tvec = -.25:.1:3.75;

y = NaN\*zeros(1, length (tvec)); % Pre-allocate memory for t = tvec,

xh = x(t-tau).\*h(tau); lxh = length(xh);

y(ti) = sum(xh.\*dtau); % Trapezoidal approximation of interal

subplot (2,1,1);

plot(tau, h(tau), 'k-', tau, x(t-tau), 'k--', t, 0, 'ok');

axis([tau(1) tau(end) -2.0 2.5]);

patch([tau(1:end-1);tau(1:end-1); tau(2:end); tau(2:end)],... [zeros(1,lxh-1);xh(1:end-1);xh(2:end);zeros(1,lxh-1)],...

[.8 .8 .8],'edgecolor', 'none');

xlabel('\tau'); legend('h(\tau)', 'x(t-\tau)','t ', 'h(\tau) x (t-

\tau)', 3);

c = get(gca, 'children'); set (gca, 'children', [c(2); c(3); c(4);

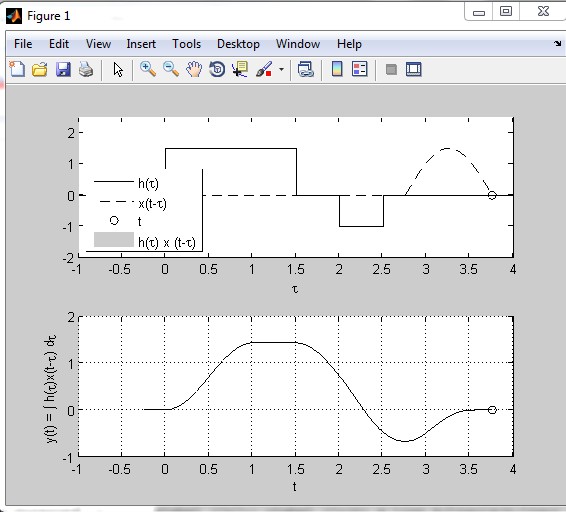
c(1)]);

subplot (2, 1, 2), plot (tvec, y, 'k', tvec (ti), y(ti), 'ok'); xlabel ('t'); ylabel ('y(t) = \int h(\tau)x(t-\tau) d\tau'); axis ([tau(1) tau(end) -1.0 2.0]); grid;

drawnow;

end

**Output:**



2. **Correlation:**

Correlation is a measure of the similarity between two signals as a function to time shift between them. Correlation is a maximum when two signals are similar in shape, and are in phase.

Correlation can be characterized in two types

**a)** Autocorrelation

**b)** Cross-correlation

**Autocorrelation** is a mathematical tool used frequently in signals processing for analyzing function or series of values, such as time domain signals. Informally, it is measure of how well a signal matches a time-shifted version of itself, as a function of the amount of time shift.

**Cross-correlation** (or sometimes “cross-covariance”) is a measure of similarity of two signals,

commonly used to find features in an unknown signal by comparing it to a known one.

**Example:**

Two discrete time signals are convolved, generate output sequence for given code:

**Code:**

x=[1 2 2 1 2];

nx=[-2:2]; % define sequence x[n] and its range h=[2 2 -1 1 2 2 1];

nh=[-3:3]; % define sequence h[n] and its range nmin=min(nx)+min(nh); % specify the lower bound of convolved sequences

nmax=max(nx)+max(nh); % specify the upper bound of convolved sequences subplot(311)

stem(nx,x,'filled','m'); title('x(n)') subplot(312) stem(nh,h,'filled','r') axis([-4.5 4.5 -2 3]) title('h(n)') y=conv(x,h);

n=[nmin:nmax]; % compute convolution and spec its range subplot(313)

stem(n,y,'filled');

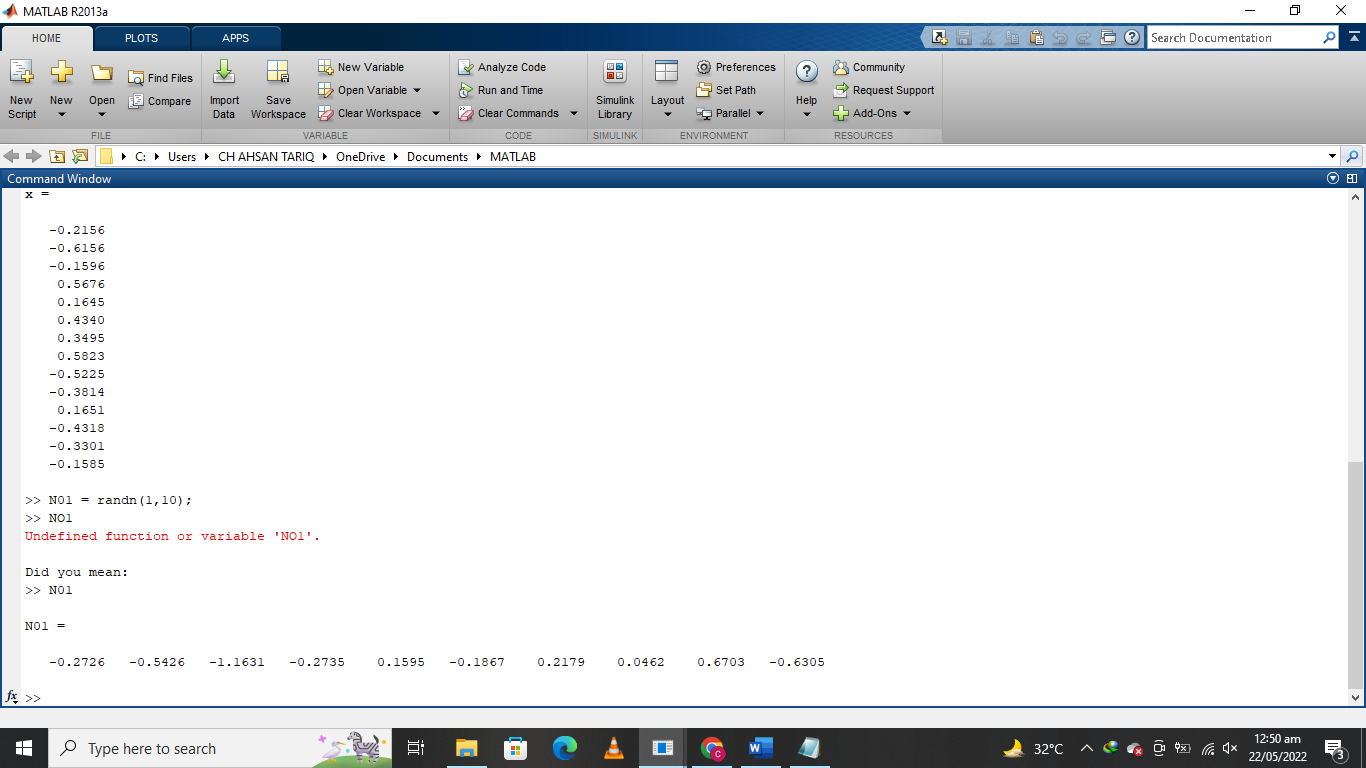
grid on; % plot the resulting sequence y[n] title('convolution of two sequence') % add title to the plot ylabel('y[n]=x[n]\*h[n]') % label the y-axis xlabel('index,[n]') % label the horizontal axis

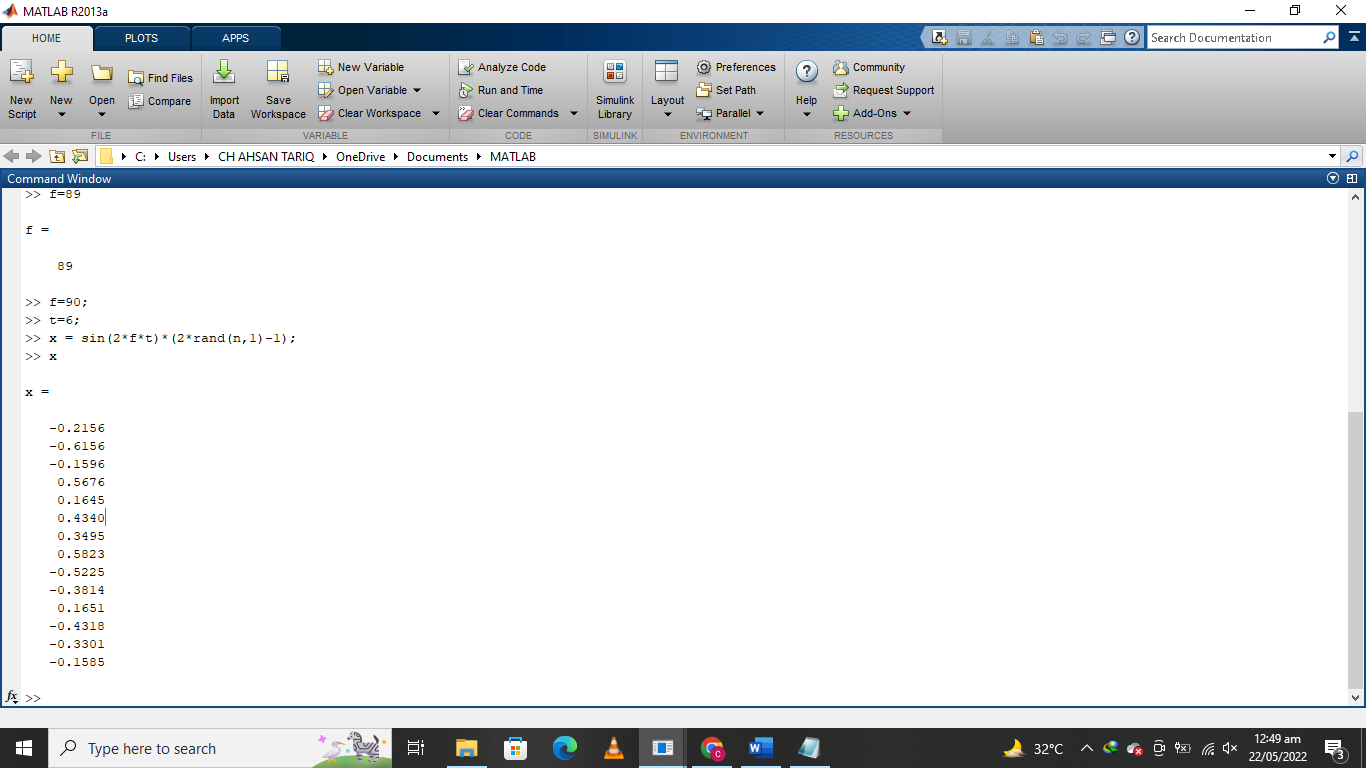


**Homework Tasks:**

1. Plot the cross-correlation of the following signal attach output graph comment on final result:

i. **y( n )= x( n ) + w( n )**

ii. **x( n )= sin( f t ) with f1= 1 Hz** 



The Islamia University of Bahawalpur

**U**niversity **C**ollege of **E**ngineering **&T**echnology **D**epartment of **C**omputer **S**ystem **E**ngineering

|  |  |  |
| --- | --- | --- |
| **LAB MANUAL** | **SIGNALS AND SYSTEMS EE-311** | **4thSemester** |

**LAB EXPERIMENT # 06**

**Trigonometric Fourier Series in MATLAB**

|  |  |
| --- | --- |
| **Student Name: Ahsan Tariq** | **Roll No: 20-CSE-26** |
| **Lab Instructor Signatures:** | **Date:** |

### OBJECTIVE:

Fourier Synthesis and Fourier Series Implementation

## Fourier Series:

The representation of a periodic signal (with period 𝑇𝑜) as a linear combination of harmonically related oscillating functions (sines and cosines) or complex

exponentials is called “***Fourier series”***.

## Fourier analysis:

Fourier analysis is the study of the way general functions may be represented or approximated by sums of simpler trigonometric functions. In the sciences and engineering, the process of decomposing a function into simpler pieces is often called Fourier analysis, while the operation of rebuilding the function from these pieces is known as Fourier synthesis.

**Example**: Fourier Analysis of a square wave

**Code:**

function[] = fourier() clc;

clearall;

symsa0 ak k T0 T1 T2 t w0 y T0=4\*T1;

w0=(2\*pi)/T0;

% calculate the DC comonent a0 a0=1/T0\*int(1,t,-T1,T1);

a0=simplify(a0);

%calculate the kth harmonic ak ak=1/T0\*int(exp(-1i\*k\*w0\*t),t,-T1,T1); ak=simplify(ak);

index=[-9:11]; fori=1:1:21 ifi-10==0

y(1,10)=subs(a0,k,i-10); else

y(1,i)=subs(ak,k,i-10); end

end

y = double(y);

%axis([-20 20 -2 2]);

display(size(index)); display(size(y)); axis([-20 20 -5 5]);

stem(index,y); end

# Fourier Synthesis:

Fourier synthesis is a method of electronically constructing a [signal](http://searchnetworking.techtarget.com/definition/signal) with a specific, desired periodic [waveform](http://searchcio-midmarket.techtarget.com/definition/waveform) . It works by combining a [sine wave](http://searchcio-midmarket.techtarget.com/definition/waveform) signal and sine-wave or cosine-wave harmonics (signals at multiples of the lowest, or fundamental, frequency) in certain proportions.

Let’s implement the synthesis equation i.e. add the Fourier components as follows:

x(t) = a0

x(t) = a-1e-jwot + a0 + a1ejw t

0

x(t) = a-2e-j2w t + a

e-jw t + a

+ a ejw t + a ej2w t

0 -1 0

### .

**.**

### .

0 1 0 2 0

Until we get a close approximated shape of x (t).

### CODE:

function[] = fourier\_synthesis( ) clc;

clearall; x=1/2; T=1;

w0=(2\*pi)/T; t=-1:0.001:1;

fork=1:20 ak=sin(k\*pi/2)/(k\*pi);

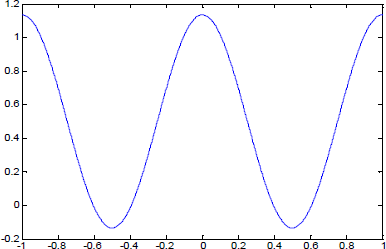
x=x+ak\*(exp(1i\*k\*w0\*t)+exp(-1i\*k\*w0\*t)); end

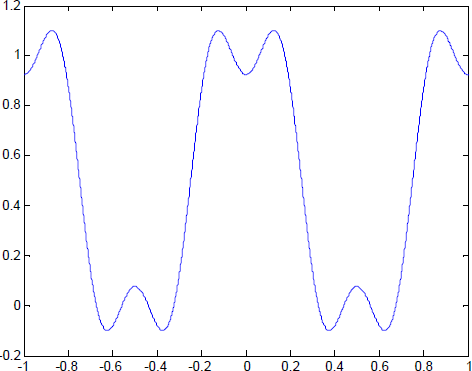
plot(t,x); end

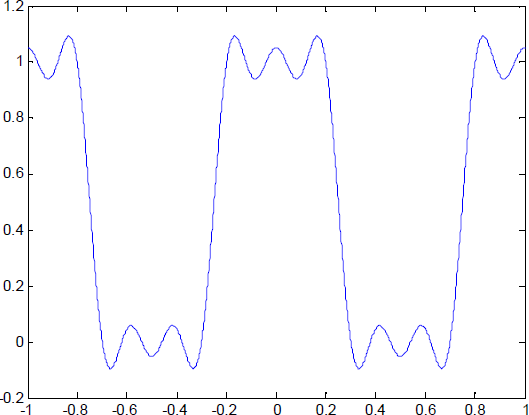
DC + first harmonic

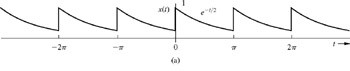
**Output:**

DC + first + third harmonics







**Example**: compute and plot the Fourier coefficients for the periodic signal in Fig. In this example, *T*0 = πand ω0 = 2.

### Code

n = 1:10;

a\_n(1) = 0.504; a\_n(n+1) = 0.504\*2./(1+16\*n.^2);

b\_n(1) = 0; b\_n(n+1) = 0.504\*8\*n./(1+16\*n.^2);

c\_n(1) = a\_n(1); c\_n(n+1) = sqrt (a\_n(n+1).^2+b\_n(n+1).^2); theta\_n(1) = 0; theta\_n(n+1) = atan2(-b\_n(n+1),a\_n(n+1)); n = [0,n];

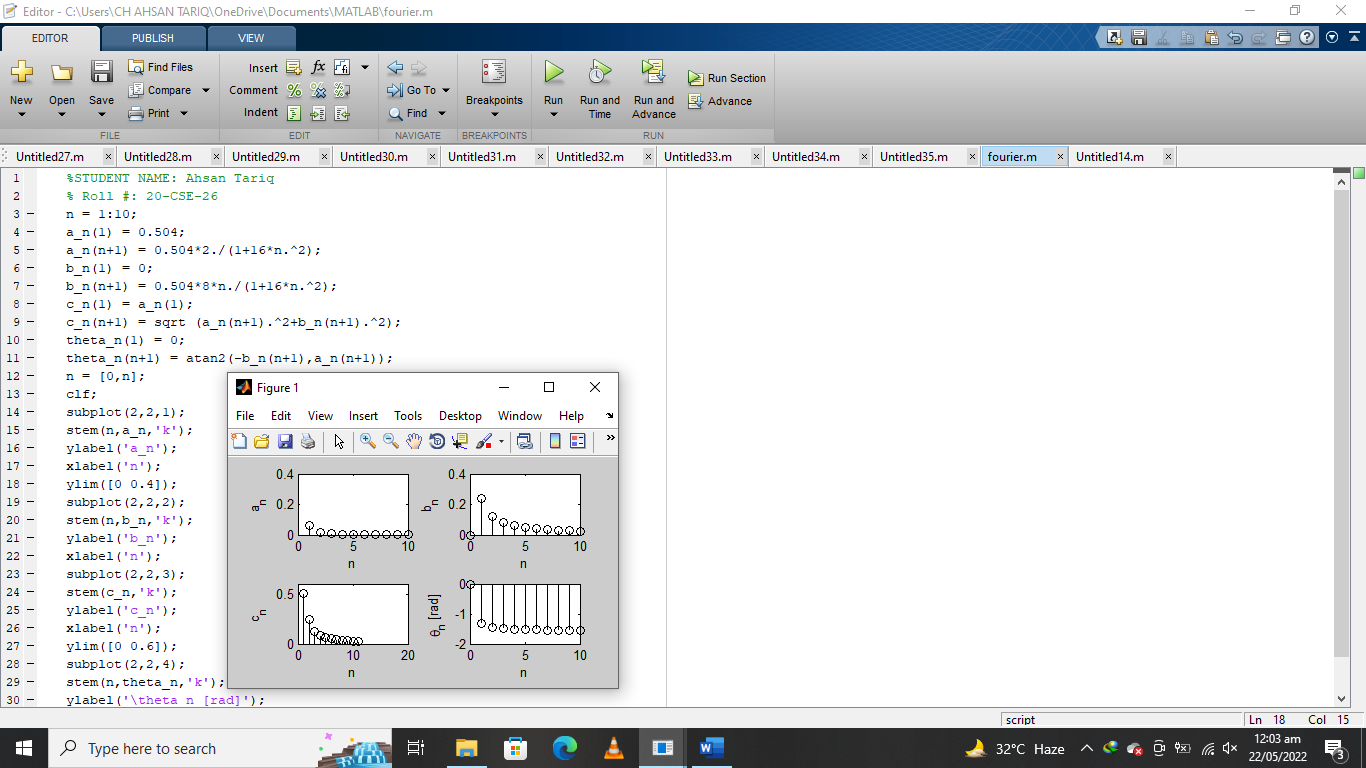
clf; subplot(2,2,1); stem(n,a\_n,'k'); ylabel('a\_n'); xlabel('n');ylim([0 0.4]);

subplot(2,2,2); stem(n,b\_n,'k'); ylabel('b\_n'); xlabel('n');

subplot(2,2,3); stem(c\_n,'k'); ylabel('c\_n'); xlabel('n');ylim([0 0.6]);

subplot(2,2,4); stem(n,theta\_n,'k'); ylabel('\theta\_n [rad]'); xlabel('n');

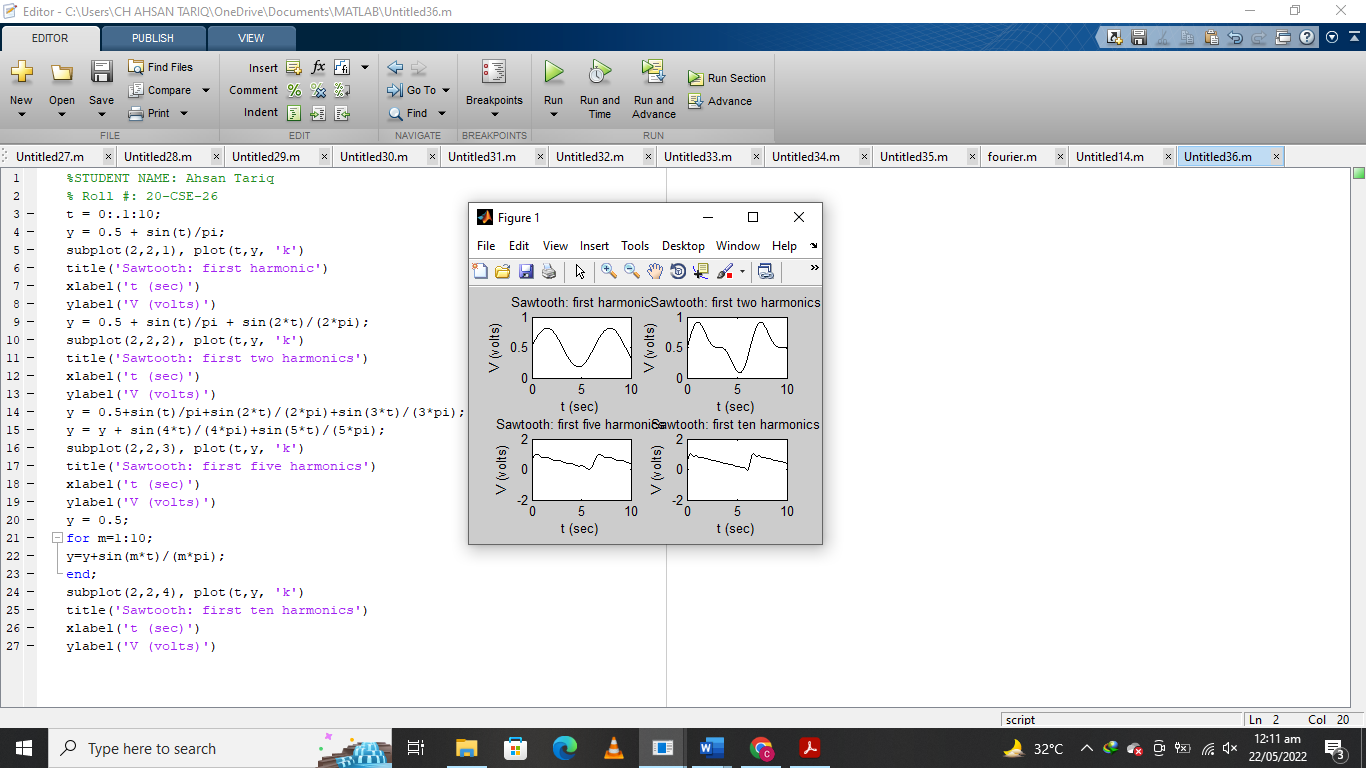
**Output** :



# Task:

* Implement a complete harmonic set for Triangular wave.

Ans:



The Islamia University of Bahawalpur

**U**niversity **C**ollege of **E**ngineering **&T**echnology **D**epartment of **C**omputer **S**ystem **E**ngineering

|  |  |  |
| --- | --- | --- |
| **LAB MANUAL** | **SIGNALS AND SYSTEMS EE-311** | **4thSemester** |

**LAB EXPERIMENT # 07**

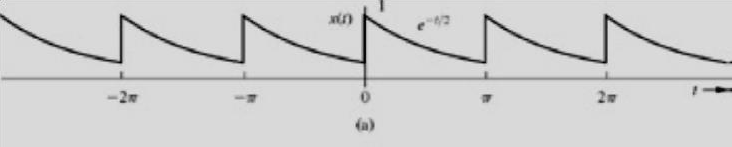
**Exponential Fourier Series in MATLAB**

|  |  |
| --- | --- |
| **Student Name: Ahsan Tariq** | **Roll No: 20-CSE-26** |
| **Lab Instructor Signatures:** | **Date:** |

# OBJECTIVE:

* Fourier Series

**Example:** Compute and Plot the trigonometric & exponential Fourier spectra for the periodic signal



Code:

T\_0 = pi; N\_0 = 256; T = T\_0/N\_0; t = (0:T:T\*(N\_0-1))'; M = 10;

x = exp(-t/2);

D\_n = fft (x)/N\_0; n = [-N\_0/2:N\_0/2-1]';

clf; subplot (2, 2, 1); stem(n, abs(fftshift (D\_n)),'k');

axis ([-M M -.1 .6]); xlabel('n'); ylabel('|D\_n|');

subplot (2, 2, 2); stem(n, angle(fftshift(D\_n)),'k');

axis([-M M -pi pi]); xlabel ('n'); ylabel('\angle D n [rad]');

n = [0:M]; C\_n(1) = abs(D\_n(1)); C\_n(2:M+1) = 2\*abs (D\_n(2:M+1));

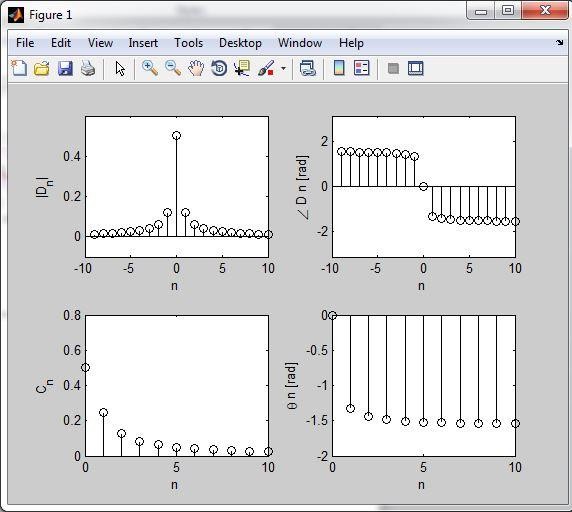
theta\_n(1) = angle(D\_n(1)); theta\_n(2:M+1) = angle(D\_n(2:M+1)); subplot (2, 2, 3); stem(n,C\_n,'k');

xlabel ('n'); ylabel('C\_n');

subplot (2, 2, 4); stem(n,theta\_n,'k');

xlabel ('n'); ylabel('\theta n [rad]');

**Output**:



**Representation of Gibbs Phenomenon:**

**Code:**

t=linspace (-pi/4,2\*pi+pi/4, 1000); % Time vector exceeds one period.

x = inline (['mod(t,2\*pi)/A.\*(mod(t,2\*pi)<A)+','((mod(t,2\*pi)>=A)&(mod(t,2\*pi)< pi))'],'t','A');

sumterms = zeros (101, length (t)); % Pre-allocate memory sumterms (1,:) = (2\*pi-A)/(4\*pi); % Compute DC term

for n = 1:100, % Compute N remaining terms D\_n=1/(2\*pi\*n)\*((exp(-j\*n\*A)-1)/(n\*A) + j\*exp(-j\*n\*pi)); sumterms (1+n,:)=real (D\_n\*exp(j\*n\*t) + conj(D\_n)\*exp(-j\*n\*t)); end

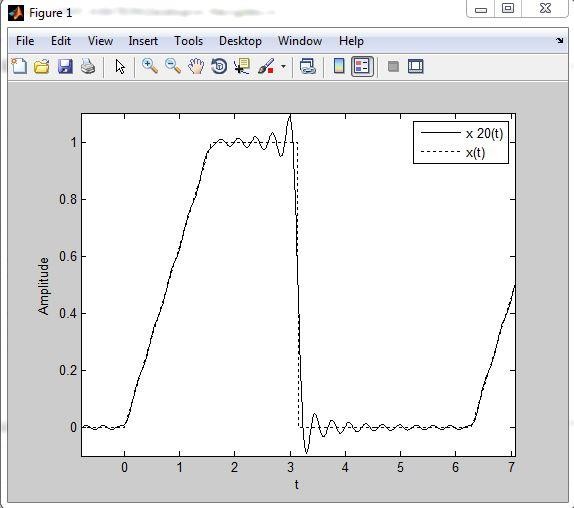
x\_N = cumsum (sumterms); A = pi/2;

Z=x(t,A);

plot(t,x\_N(21,:),'k',t,Z,'k:'); axis([-pi/4 2\*pi+pi/4 -0.1 1.1]);

xlabel ('t'); ylabel ('Amplitude'); legend ('x 20(t)','x(t)',0);

**Output:**

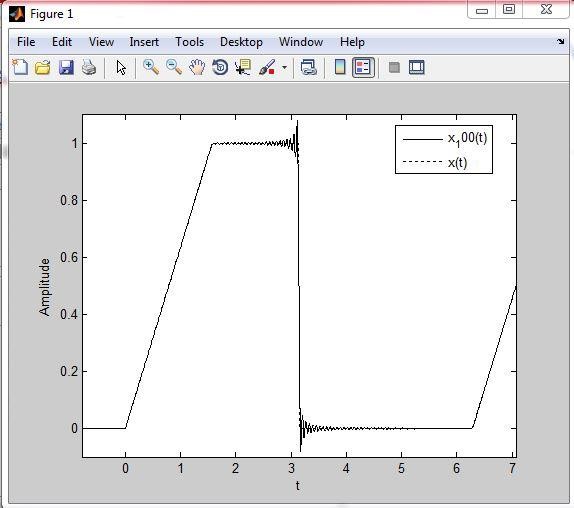


Increasing *N* to 100,improves the approximation but does not reduce the overshoot.

>> plot(t,x\_N(101,:),'k',t,x(t,A),'k:'); axis([-pi/4,2\*pi+pi/4,-0.1,1.1]);

>>xlabel('t'); ylabel('Amplitude'); legend('x\_100(t)','x(t)',0);

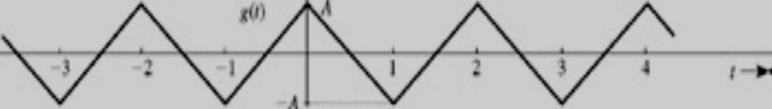
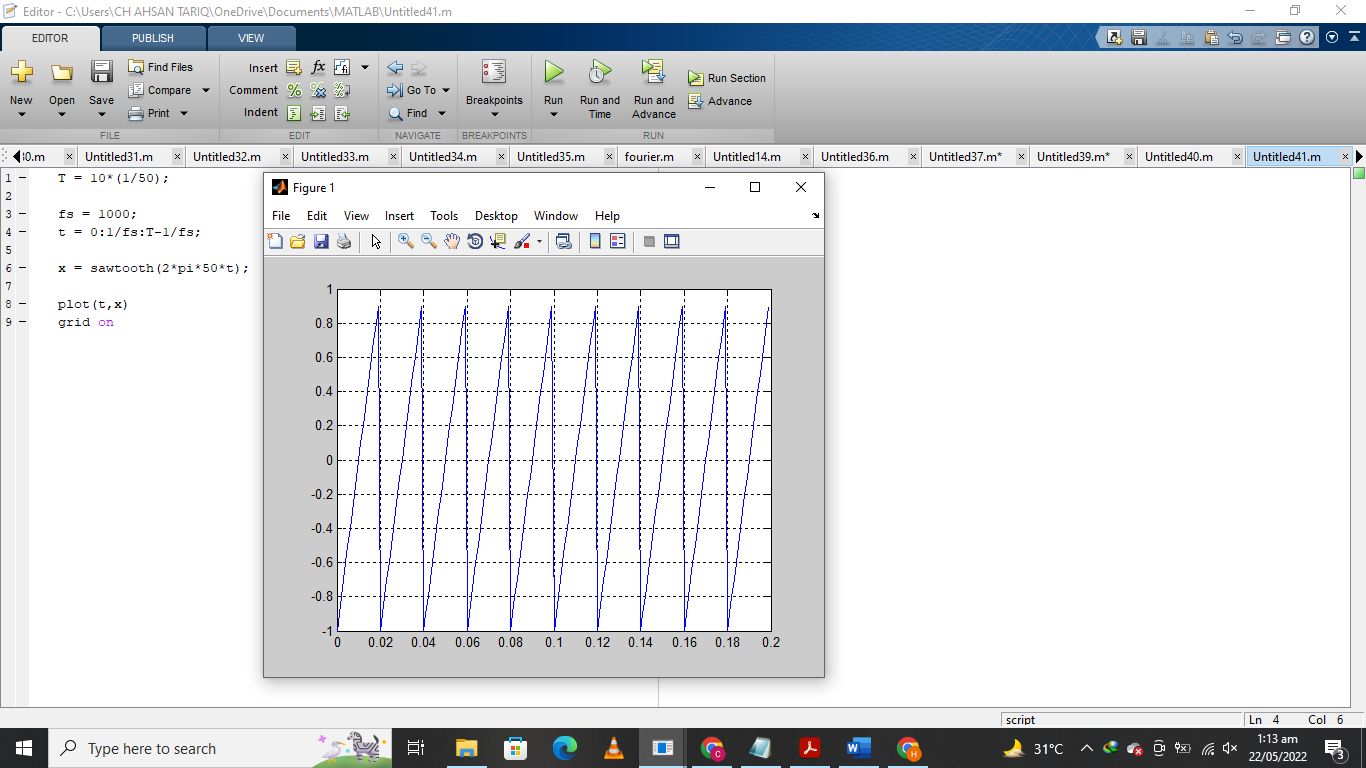
**Output:**



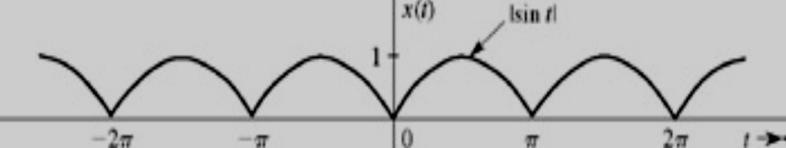
**Tasks:**

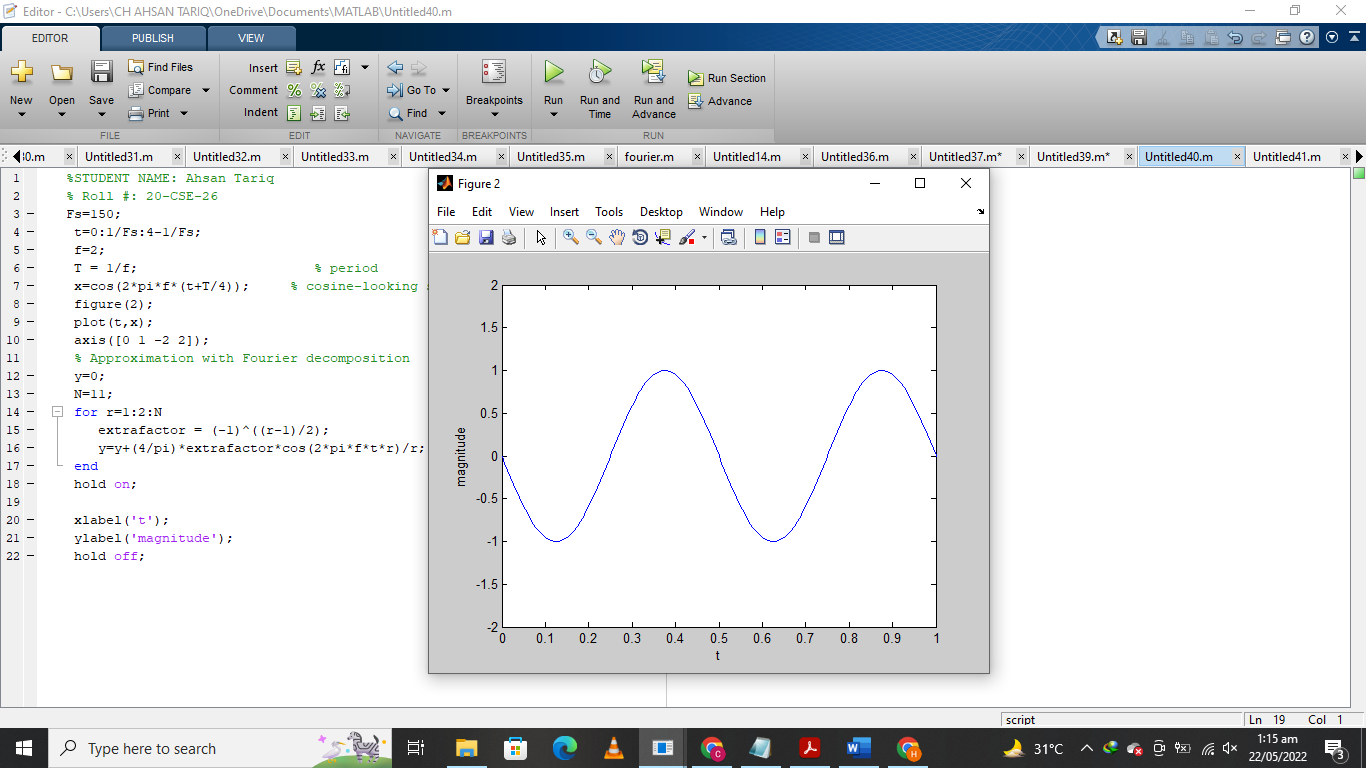
**1.**

1. Plot magnitude and phase graph of the following given sequence.
2. Plot waveform by its magnitude and phase spectrum (Synthesis equation)



**2.**

1. Plot magnitude and phase graph of the following given sequence.
2. Plot waveform by its magnitude and phase spectrum (Synthesis equation)



The Islamia University of Bahawalpur

**U**niversity **C**ollege of **E**ngineering **&T**echnology **D**epartment of **C**omputer **S**ystem **E**ngineering

|  |  |  |
| --- | --- | --- |
| **LAB MANUAL** | **SIGNALS AND SYSTEMS EE-311** | **4thSemester** |

**LAB EXPERIMENT # 08**

**Implementation of Fourier Transform in MATLAB**

|  |  |
| --- | --- |
| **Student Name: Ahsan Tariq** | **Roll No: 20-CSE-26** |
| **Lab Instructor Signatures:** | **Date:** |

## OBJECTIVE:

* Familiarization with Fourier transform using Matlab
* Basic Plotting in Matlab i.e. Sin Wave, Cosine Wave

# Introduction:

Fourier transform is an operator that transforms a signal from time domain to frequency domain. The Fourier transform of a continuous time signal is defined as:

|  |  |
| --- | --- |
| **Fourier Transform** | **Inverse Fourier Transform** |
| ∞  X(j𝜔) = ∫ 𝑥(𝑡)𝑒−j𝜔𝑡𝑑𝑡  −∞ | ( ) 1 ∞ j𝜔𝑡  𝑥 𝑡 = 2𝜋 ∫ X(j𝜔)𝑒 𝑑𝜔  −∞ |

# Fourier Transform in MATLAB:

MATLAB has a built in function ***fourier()*** and ***ifourier()*** for calculating the Fourier transform and inverse Fourier transform respectively.

**Example:** 𝑥(𝑡) = {1 |𝑡| ≤ 𝑎/2

0 𝑜𝑡ℎ𝑒𝑟𝑤i𝑠𝑒

Fourier Transform of the signal can be found by performing the integration, using the following MATLAB commands:

**Code**:

>> syms a t f

>> sig=t/abs(t);

>> ut=(1+sig)/2;

>> pulse=subs(ut,t+a/2)-subs(ut,t-a/2) pulse =

1/2\*(t+1/2\*a)/abs(t+1/2\*a)-1/2\*(t-1/2\*a)/abs(-t+1/2\*a)

>> pretty(pulse)

# Continuous Time Fourier Transform:

t + 1/2 a

t - 1/2 a

1/2 - 1/2

| t + 1/2 a | | -t + 1/2 a |

>> Xjw=int( exp(-j\*2\*pi\*f\*t)\*subs(pulse,a,1),t,-inf,inf) Xjw =

-1/2\*i\*(exp(i\*pi\*f)-exp(-i\*pi\*f))/pi/f

>> pretty(simple(Xjw))

sin(pi f)

pi f

In Matlab a built in function for **‘fft’** exists which is the implementation of Fast Fourier transform, however, there is no function available for CTFT.

The following function will calculate CTFT of any given signal.

function [w,X]=CTFT(t,x) w= -pi:0.01:pi;

ee=exp(-j\*w'\*t); w=w./pi\*1000; X=x\*ee';

## Task:

1. Find and plot the Fourier Transform of the following signals:
   * 𝑔(𝑡) = { 𝑒−𝑡 ƒ𝑜𝑟 𝑡 > 0

0 𝑜𝑡ℎ𝑒𝑟𝑤i𝑠𝑒

𝑔(𝑡) = 𝑒−|𝑡|

# 

# Basic plotting in MATLAB:

MATLAB has an excellent set of graphic tools. Plotting a given data set or the results of computation is possible with very few commands.The MATLAB command to plot a graph is plot(x,y), e.g.

x = 0:pi/100:2\*pi; y = sin(x);

plot(x,y)

MATLAB enables you to add axis Labels and titles, e.g.

xlabel('x=0:2\pi'); ylabel('Sine of x'); tile('Sine function')

