EEE 3005 Signals and Systems

HOMEWORK #3

CONVOLUTION AND ITS APPLICATIONS

Student Name

Ahmad Zameer Nazarı

220702706

Name of Lecturers

Dr. Öğr. Üyesi Kutlu Karayahşi

Arş. Gör. Ali Can Erüst

Date:

011/11/2024 – 18/11/2024

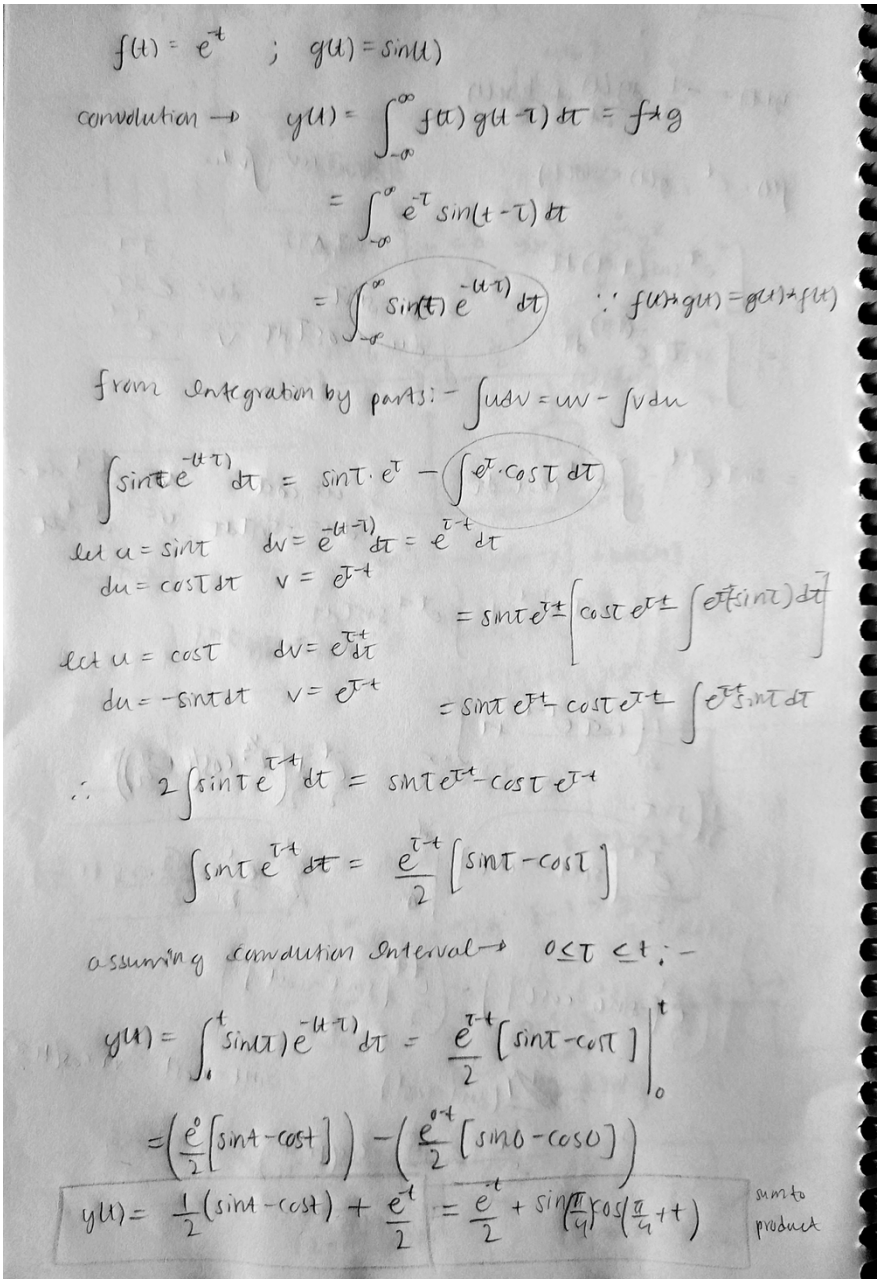
[EEE-3005]. HW3. 220702706 - Ahmad Zameer Nazarı

November 16, 2024

1 3RD ASSIGNMENT

[1]:
```
# importing some essential libraries
import numpy as np
import matplotlib.pyplot as plt
```

1.1 Question 1



1.2 Question 2

1.2.1 MATLAB results:

I computed convolution in two ways:

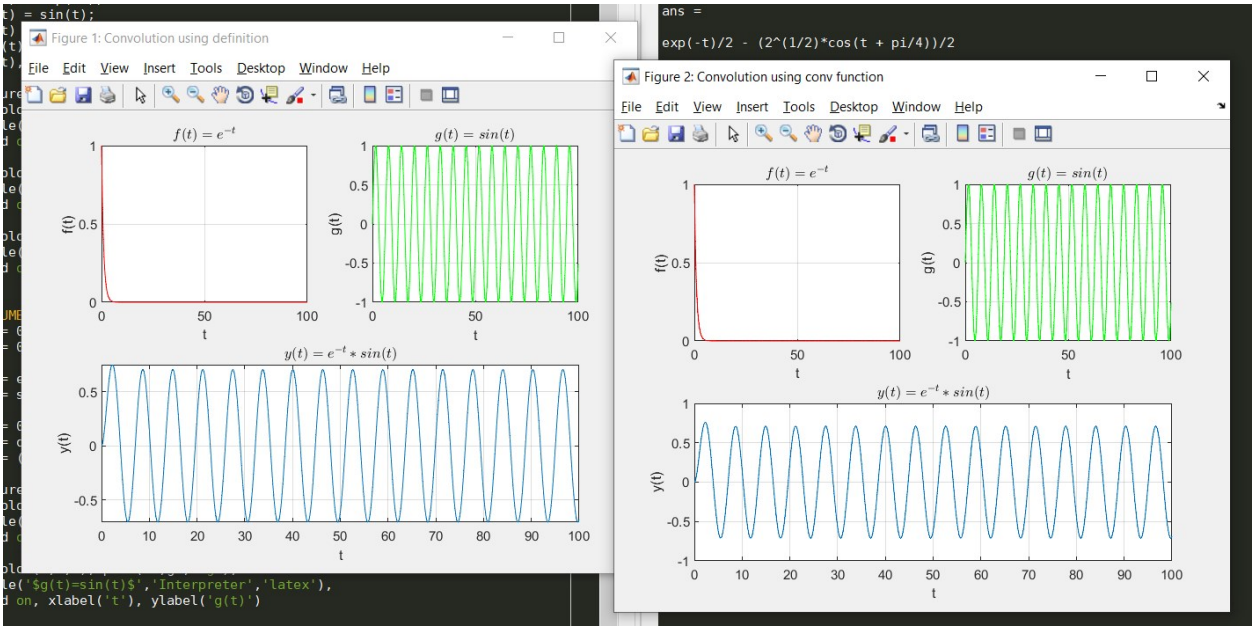
question_1_2.m | question_3.m | question_4.m

```
clc; clear; close all;

% SYMBOLIC COMPUTATION USING CONVOLUTION DEFINITION
syms t tau real
fs(t) = exp(-t);
gs(t) = sin(t);
ys(t) = int(fs(t-tau)*gs(tau),tau);        % symbolic expression for y=f*g
yst(t) = int(fs(t-tau)*gs(tau),tau, 0, t); % result with bounds 0<tau<t
ys(t), yst(t)                              % display result

figure('Name','Convolution using definition');
subplot(2,2,1), fplot(fs(t), [0,100], 'r'),
title('$f(t)=e^{-t}$','Interpreter','latex'),
grid on, xlabel('t'), ylabel('f(t)')

subplot(2,2,2), fplot(gs(t), [0,100], 'g'),
title('$g(t)=sin(t)$','Interpreter','latex'),
grid on, xlabel('t'), ylabel('g(t)')

subplot(2,2,[3;4]), fplot(yst(t), [0,100]),
title('$y(t)=e^{-t}*sin(t)$','Interpreter','latex'),
grid on, xlabel('t'), ylabel('y(t)')


% NUMERICAL COMPUTATION USING BUILT-IN CONV() FUNCTION
dt = 0.01;                     % delta time step
t1 = 0:dt:100;

ft = exp(-t1);
gt = sin(t1);

st = 0;                        % corrective time shift in conv time axis = 0
yt = dt*conv(ft,gt);           % multipyling by dt to normalize yt amplitude
t2 = (1:length(yt)).*dt - st;  % multiplying by dt to normalize time axis

figure('Name','Convolution using conv function');
subplot(2,2,1), plot(t1,ft,'r'),
title('$f(t)=e^{-t}$','Interpreter','latex'),
grid on, xlabel('t'), ylabel('f(t)')

subplot(2,2,2), plot(t1,gt, 'g'),
title('$g(t)=sin(t)$','Interpreter','latex'),
grid on, xlabel('t'), ylabel('g(t)')

subplot(2,2,[3;4]), plot(t2, yt),
title('$y(t)=e^{-t}*sin(t)$','Interpreter','latex'),
grid on, xlabel('t'), ylabel('y(t)'), xlim([0 100])
```

The convolution signal matches with my result when i calculated manually above.

```
Command Window                                  Workspace

ans =

-exp(tau - t)*(cos(tau)/2 - sin(tau)/2)


ans =

exp(-t)/2 - (2^(1/2)*cos(t + pi/4))/2

>>
```

the graphs are:

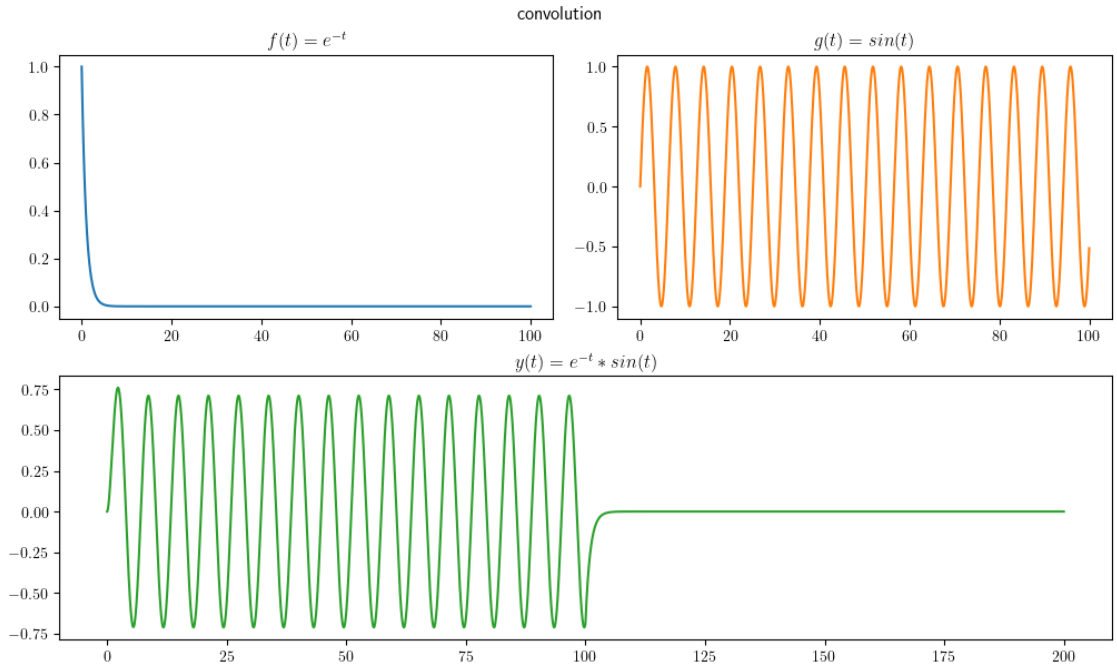


as can be seen both methods gave the same results. I used an interval of [0, 100] mostly because the conv() function starts the signal at t=0.

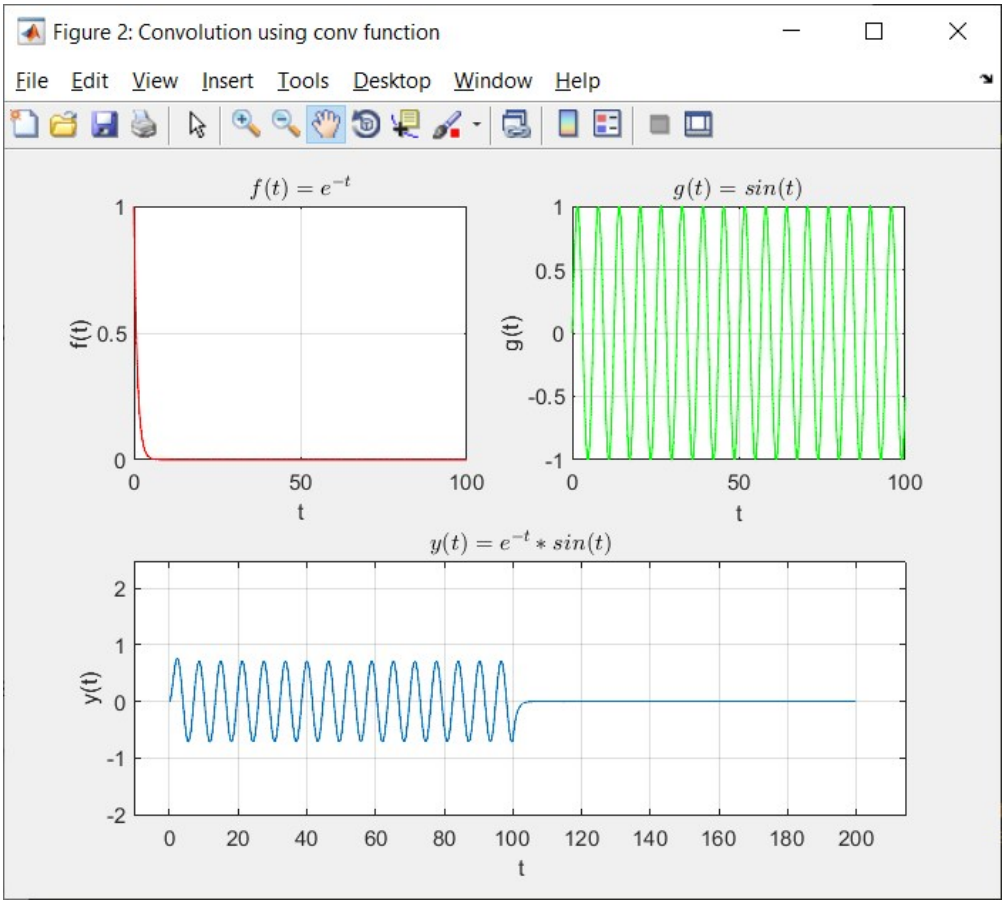
1.2.2 Python results:

[4]:
```
dt = 0.01
t = np.arange(0,100,dt)

ft = np.exp(-t)
gt = np.sin(t)

yt = dt*np.convolve(ft,gt,mode='full')
t2 = dt*np.arange(0,len(yt))

fig, axd = plt.subplot_mosaic("""AB;CC""", constrained_layout=True,␣
 ↪figsize=(10,6))
fig.suptitle('convolution')
```

```
plt.rcParams.update({"text.usetex": True})

axd['A'].plot(t, ft, 'C0')
axd['A'].set_title('$f(t)=e^{-t}$')
axd['B'].plot(t, gt, 'C1')
axd['B'].set_title('$g(t)=sin(t)$')
axd['C'].plot(t2, yt, 'C2')
axd['C'].set_title('$y(t)=e^{-t}*sin(t)$')

plt.show()
```



As can be seen the convolution signal output has a legnth of ~200, even though the initial signals were limited to a range [0, 100]. This is because the size of the convolution is always the length of the sum of the two initial signal (both 100 in this case) minus one. So this result here shows the full convolution. The same could also be seen in the matlab results, if we slide the time axis forward



1.3 Question 3

$f = 10$ and $f = 15$ in the question, i assume means frequencies of respective signals.

1.3.1 Matlab results

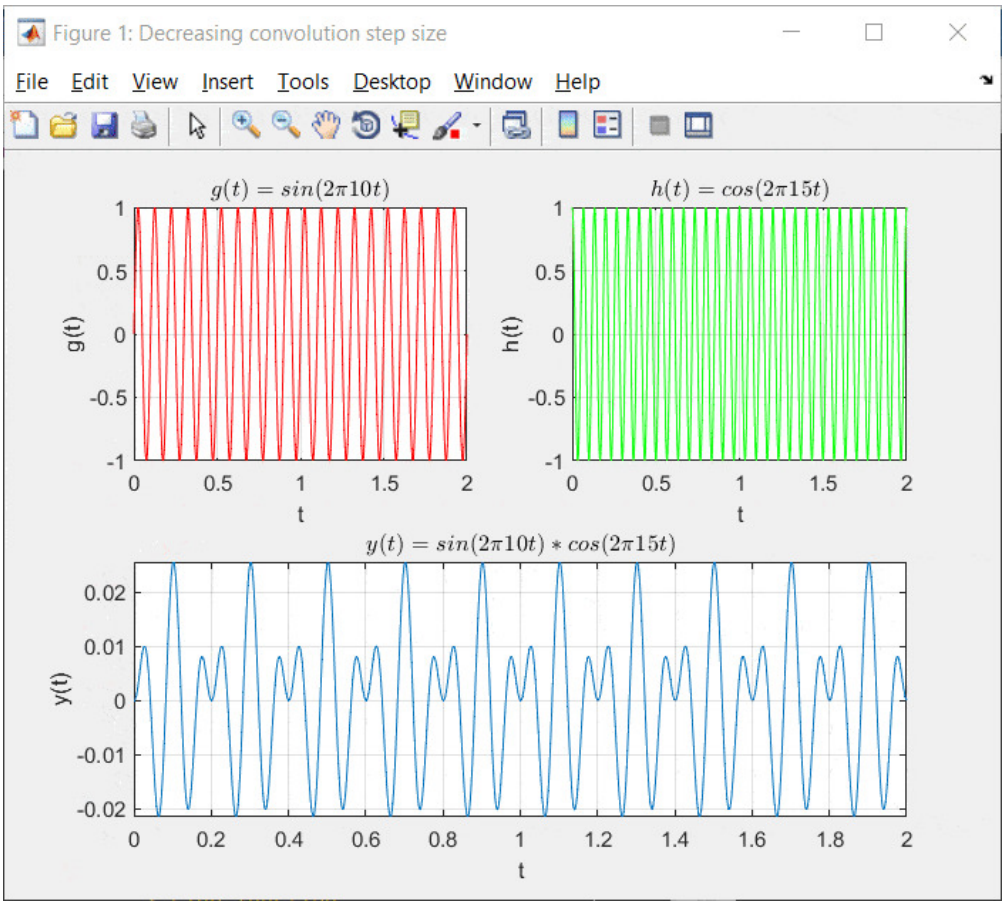
question_1_2.m | question_3.m | question_4.m

```
clc; clear; close all;

dt = 1;                                 % delta time step
f1 = 10;                                % sine frequency
f2 = 15;                                % cosine frequency

figure('Name','Decreasing convolution step size');

subplot(2,2,1)
p1 = plot(NaN,NaN,'r');
title('$g(t)=sin(2\pi10t)$','Interpreter','latex'),
grid on, xlabel('t'), ylabel('g(t)')

subplot(2,2,2)
p2 = plot(NaN,NaN, 'g');
title('$h(t)=cos(2\pi15t)$','Interpreter','latex'),
grid on, xlabel('t'), ylabel('h(t)')

subplot(2,2,[3;4])
p3 = plot(NaN,NaN);
title('$y(t)=sin(2\pi10t)*cos(2\pi15t)$','Interpreter','latex'),
grid on, xlabel('t'), ylabel('y(t)'), xlim([0 2])


% ANIMATE PLOT WHILE DECREASING STEP SIZE
for i = 1:10

    t1 = 0:dt:2;                        % defined time interval
    gt = sin(2*pi*f1*t1);               % sine function
    ht = cos(2*pi*f2*t1);               % cosine function

    yt = dt*conv(gt,ht);                % their convolution
    t2 = (1:length(yt)).*dt;            % convolution time axis

    set(p1, 'xdata', t1, 'ydata', gt);  % update sine plot with new dt
    set(p2, 'xdata', t1, 'ydata', ht);  % update cosine plot with new dt
    set(p3, 'xdata', t2, 'ydata', yt);  % update conv plot with new dt

    dt = dt/2;                          % reduce dt in half each iter for i=10

    pause(1); drawnow;                  % animate every 1 sec

end

```

I've wrote the code in a loop so that with each iteration, the step size is decreased and the plot is updated as an animation. The gif file showing the animation is included in my github repository. For the purpose of this pdf, the sequence of images showing decreased step size is as below, moving from left to right then downwards:


Figure 1: Decreasing convolution step size
File Edit View Insert Tools Desktop Window Help
$g(t) = sin(2\pi 10t)$
$h(t) = cos(2\pi 15t)$
$y(t) = sin(2\pi 10t) * cos(2\pi 15t)$
g(t)
h(t)
y(t)
t

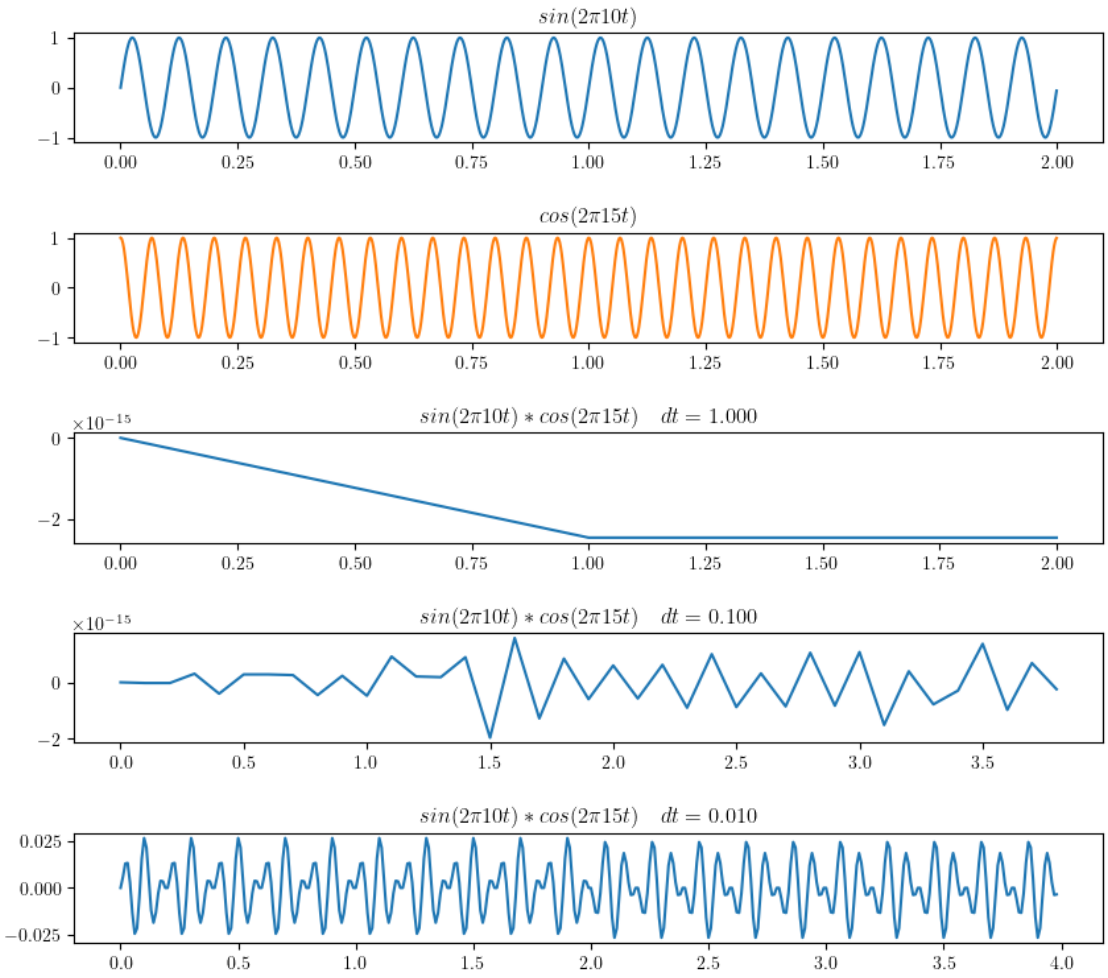
With the final result with the smallest step size of 2^{-10}:



1.3.2 Python results

[3]:
```
dt = 0.001
t = np.arange(0,2,dt)

f1 = 10
f2 = 15

gt = np.sin(2*np.pi*f1*t)
ht = np.cos(2*np.pi*f2*t)

fig, ax = plt.subplots(5, figsize=(9, 8))
fig.tight_layout(pad=3)
plt.rcParams.update({"text.usetex": True})

ax[0].plot(t, gt, 'C0')
ax[0].set_title('$sin(2 \pi 10t)$')
ax[1].plot(t, ht, 'C1')
ax[1].set_title('$cos(2 \pi 15t)$')
```

```
dtv = 1
for i in range(1,4):
    tv = np.arange(0,2,dtv)
    gt2 = np.sin(2*np.pi*f1*tv)
    ht2 = np.cos(2*np.pi*f2*tv)
    yt = dtv*np.convolve(gt2,ht2,mode='full')
    t2 = dtv*np.arange(0,len(yt))
    plt.subplot(5,1,i+2)
    plt.plot(t2,yt)
    plt.title('$sin(2 \pi 10t)*cos(2 \pi 15t) \quad dt = %1.3f$' % dtv)
    dtv = dtv/10

plt.show()
```



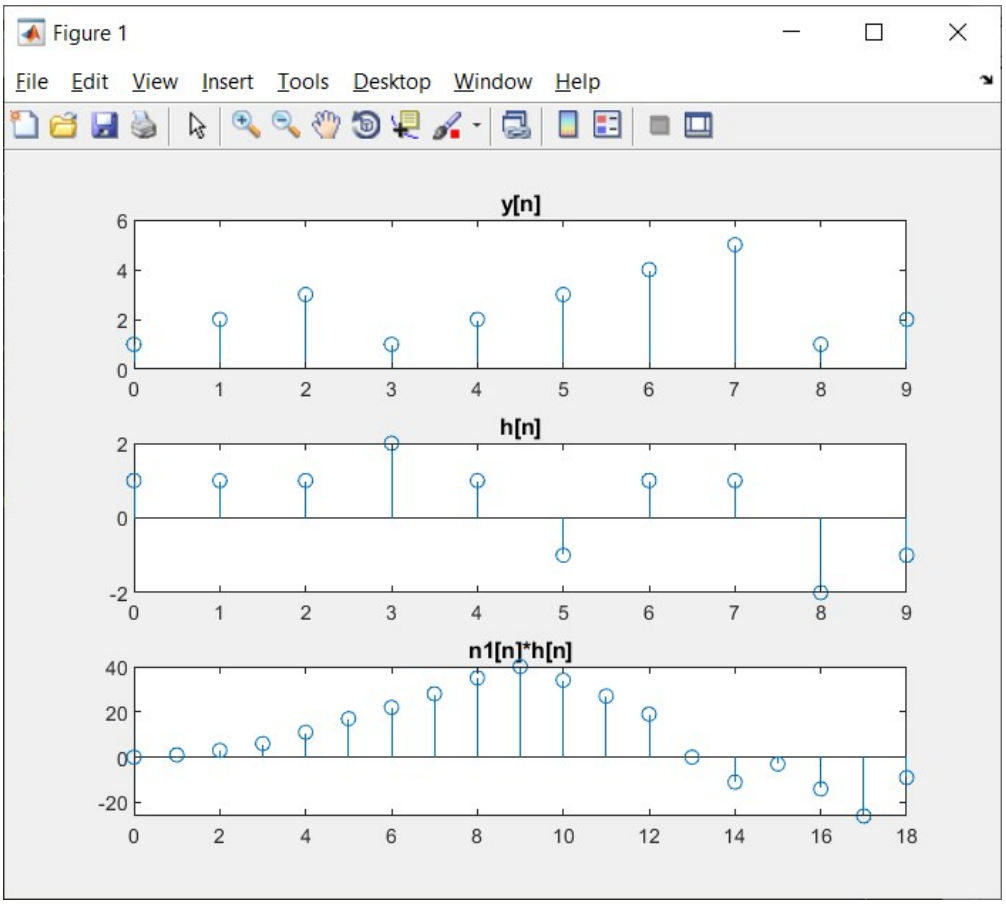
1.4 Question 4

The corrected MATLAB code, with comments that explain lines is as follows:

question_1_2.m | question_3.m | question_4.m

```
clear;
close all;
n1 = 0:1:9;                          % discrete time n1: 0 and first 9 integers
y1 = [ 1 2 3 1 2 3 4 5 1 2];         % 1st functin y1
h1 = [ 1 1 1 2 1 -1 1 1 -2 -1];      % 2nd function h1
X = conv(n1, h1);                    % convolution of n1 (ramp) and h1
n2 = 0:length(X)-1;                  % setting up new time axis n2 for X
figure(1)
subplot(3,1,1)
stem(n1, y1)                         % plotting 1st function y1 to n1
title('y[n]');
subplot(3,1,2)
stem(n1, h1)                         % plotting 2nd function h1 to n1
title('h[n]');
subplot(3,1,3)
stem(n2, X)                          % plotting convolution to its axis n2
title('n1[n]*h[n]');

```

The result of the plot is a discrete time convolution:



---

all files available at: https://github.com/az-yugen/EEE-3005.-Signals-Systems-LAB