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

November 16, 2024

1 3RD ASSIGNMENT

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

1.1 Question 1

$$f(t) = e^t \quad ; \quad g(t) = \sin(t)$$

convolution $\rightarrow \quad y(t) = \int_{-\infty}^{\infty} f(t) \, g(t-\tau) \, d\tau = f * g$

$$= \int_{-\infty}^{\infty} e^{-\tau} \sin(t-\tau) \, d\tau$$

$$= \int_{-\infty}^{\infty} \sin(t) \, e^{-(t-\tau)} \, d\tau \qquad \because f(t) * g(t) = g(t) * f(t)$$

from Integration by parts :- $\int u \, dv = uv - \int v \, du$

$$\int \sin t \, e^{-(t-\tau)} \, d\tau = \sin \tau \cdot e^t - \int e^\tau \cdot \cos \tau \, d\tau$$

let $u = \sin \tau \qquad dv = e^{-(t-\tau)} \, d\tau = e^{\tau-t} \, d\tau$
$du = \cos \tau \, d\tau \qquad v = e^{\tau-t}$

$$= \sin \tau \, e^{\tau-t} \pm \left[\cos \tau \, e^{\tau-t} \pm \int e^{\tau-t} (\sin \tau) \, d\tau \right]$$

let $u = \cos \tau \qquad dv = e^{\tau-t} d\tau$
$du = -\sin \tau \, d\tau \qquad v = e^{\tau-t}$

$$= \sin \tau \, e^{\tau-t} \pm \cos \tau \, e^{\tau-t} \pm \int e^{\tau-t} \sin \tau \, d\tau$$

$$\therefore \quad 2 \int \sin \tau \, e^{\tau-t} \, d\tau = \sin \tau \, e^{\tau-t} - \cos \tau \, e^{\tau-t}$$

$$\int \sin \tau \, e^{\tau-t} \, d\tau = \frac{e^{\tau-t}}{2} \left[\sin \tau - \cos \tau \right]$$

assuming convolution Interval $\rightarrow \quad 0 \leq \tau \leq t :-$

$$y(t) = \int_0^t \sin(\tau) \, e^{-(t-\tau)} \, d\tau = \frac{e^{\tau-t}}{2} \left[\sin \tau - \cos \tau \right] \Big|_0^t$$

$$= \left(\frac{e^0}{2} \left[\sin t - \cos t \right] \right) - \left(\frac{e^{0-t}}{2} \left[\sin 0 - \cos 0 \right] \right)$$

$$\boxed{y(t) = \frac{1}{2} (\sin t - \cos t) + \frac{e^{-t}}{2}} = \boxed{\frac{e^{-t}}{2} + \sin\left(\frac{\pi}{4}\right) \cos\left(\frac{\pi}{4} + t\right)} \quad \text{sum to product}$$

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  ✕   +
 1 -    clc; clear; close all;
 2
 3      % SYMBOLIC COMPUTATION USING CONVOLUTION DEFINITION
 4 -    syms t tau real
 5 -    fs(t) = exp(-t);
 6 -    gs(t) = sin(t);
 7 -    ys(t) = int(fs(t-tau)*gs(tau),tau);          % symbolic expression for y=f*g
 8 -    yst(t) = int(fs(t-tau)*gs(tau),tau, 0, t); % result with bounds 0<tau<t
 9 -    ys(t), yst(t)                                % display result
10
11 -    figure('Name','Convolution using definition');
12 -    subplot(2,2,1), fplot(fs(t), [0,100], 'r'),
13 -    title('$f(t)=e^{-t}$','Interpreter','latex'),
14 -    grid on, xlabel('t'), ylabel('f(t)')
15
16 -    subplot(2,2,2), fplot(gs(t), [0,100], 'g'),
17 -    title('$g(t)=sin(t)$','Interpreter','latex'),
18 -    grid on, xlabel('t'), ylabel('g(t)')
19
20 -    subplot(2,2,[3;4]), fplot(yst(t), [0,100]),
21 -    title('$y(t)=e^{-t}*sin(t)$','Interpreter','latex'),
22 -    grid on, xlabel('t'), ylabel('y(t)')
23
24
25      % NUMERICAL COMPUTATION USING BUILT-IN CONV() FUNCTION
26 -    dt = 0.01;                          % delta time step
27 -    t1 = 0:dt:100;
28
29 -    ft = exp(-t1);
30 -    gt = sin(t1);
31
32 -    st = 0;                             % corrective time shift in conv time axis = 0
33 -    yt = dt*conv(ft,gt);                % multipyling by dt to normalize yt amplitude
34 -    t2 = (1:length(yt)).*dt - st;  % multiplying by dt to normalize time axis
35
36 -    figure('Name','Convolution using conv function');
37 -    subplot(2,2,1), plot(t1,ft,'r'),
38 -    title('$f(t)=e^{-t}$','Interpreter','latex'),
39 -    grid on, xlabel('t'), ylabel('f(t)')
40
41 -    subplot(2,2,2), plot(t1,gt, 'g'),
42 -    title('$g(t)=sin(t)$','Interpreter','latex'),
43 -    grid on, xlabel('t'), ylabel('g(t)')
44
45 -    subplot(2,2,[3;4]), plot(t2, yt),
46 -    title('$y(t)=e^{-t}*sin(t)$','Interpreter','latex'),
47 -    grid on, xlabel('t'), ylabel('y(t)'), xlim([0 100])
```

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

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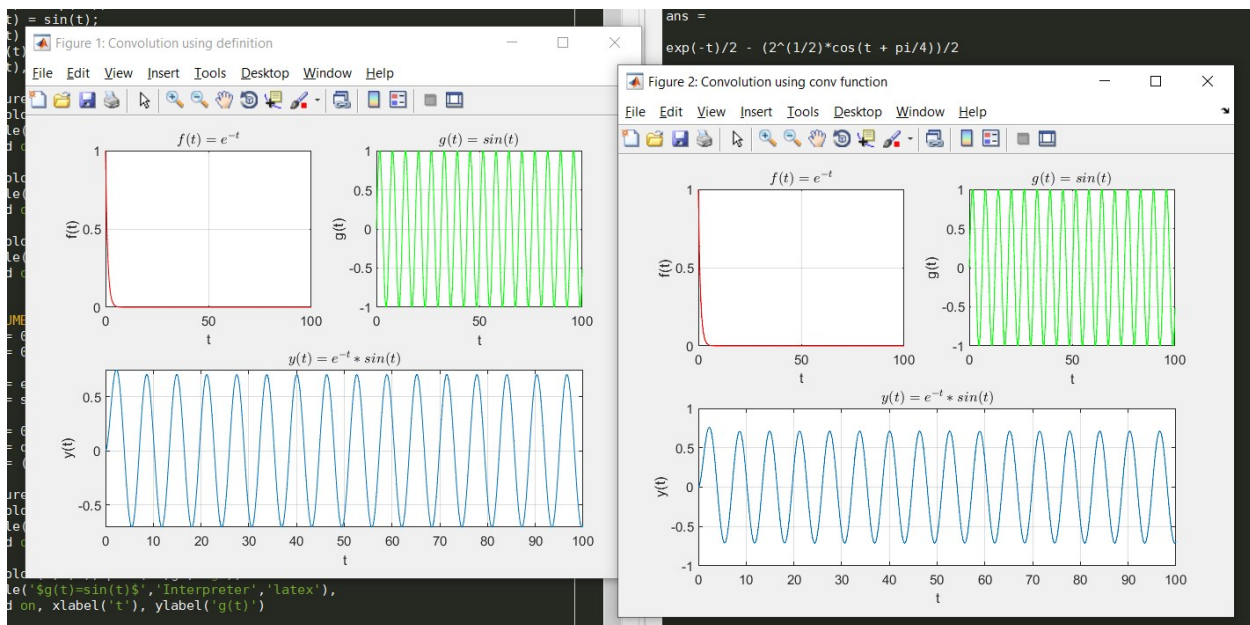
as can be seen both methods gave the same results. I used an interval of [0, 100] mostly because the conv() function starts the signal at t=0.

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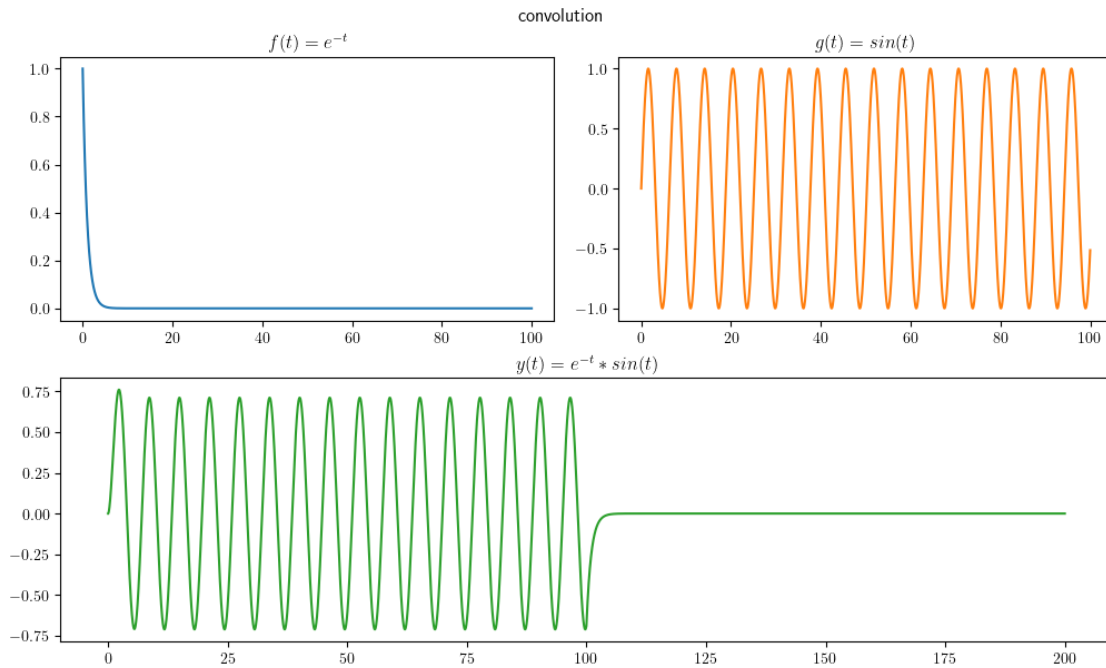
```
plt.rcParams.update({"text.usetex": True})

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

Figure 2: Convolution using conv function
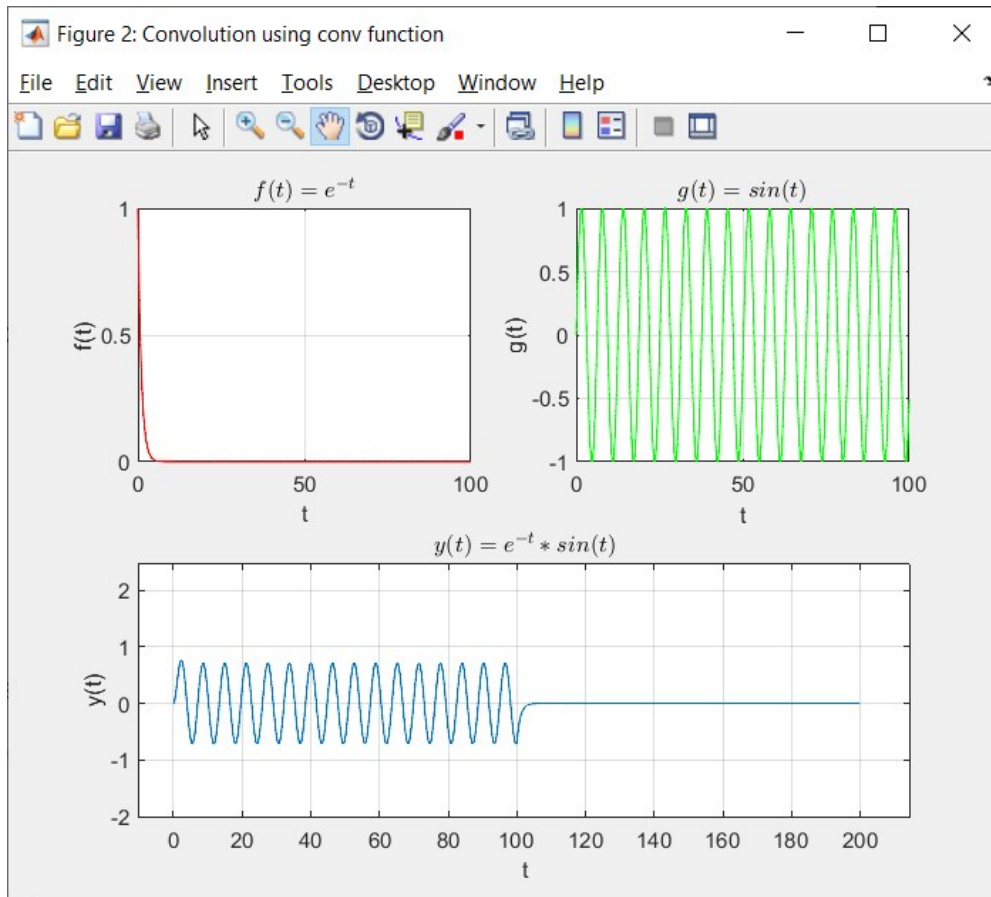
1.3 Question 3

$f = 10$ 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  ×   +
 1 -   clc; clear; close all;
 2
 3 -   dt = 1;                              % delta time step
 4 -   f1 = 10;                             % sine frequency
 5 -   f2 = 15;                             % cosine frequency
 6
 7 -   figure('Name','Decreasing convolution step size');
 8
 9 -   subplot(2,2,1)
10 -   p1 = plot(NaN,NaN,'r');
11 -   title('$g(t)=sin(2\pi10t)$','Interpreter','latex'),
12 -   grid on, xlabel('t'), ylabel('g(t)')
13
14 -   subplot(2,2,2)
15 -   p2 = plot(NaN,NaN, 'g');
16 -   title('$h(t)=cos(2\pi15t)$','Interpreter','latex'),
17 -   grid on, xlabel('t'), ylabel('h(t)')
18
19 -   subplot(2,2,[3;4])
20 -   p3 = plot(NaN,NaN);
21 -   title('$y(t)=sin(2\pi10t)*cos(2\pi15t)$','Interpreter','latex'),
22 -   grid on, xlabel('t'), ylabel('y(t)'), xlim([0 2])
23
24
25     % ANIMATE PLOT WHILE DECREASING STEP SIZE
26 -   for i = 1:10
27
28 -       t1 = 0:dt:2;                      % defined time interval
29 -       gt = sin(2*pi*f1*t1);            % sine function
30 -       ht = cos(2*pi*f2*t1);            % cosine function
31
32 -       yt = dt*conv(gt,ht);            % their convolution
33 -       t2 = (1:length(yt)).*dt;        % convolution time axis
34
35 -       set(p1, 'xdata', t1, 'ydata', gt);  % update sine plot with new dt
36 -       set(p2, 'xdata', t1, 'ydata', ht);  % update cosine plot with new dt
37 -       set(p3, 'xdata', t2, 'ydata', yt);  % update conv plot with new dt
38
39 -       dt = dt/2;                       % reduce dt in half each iter for i=10
40
41 -       pause(1); drawnow;               % animate every 1 sec
42
43 -   end
44
```

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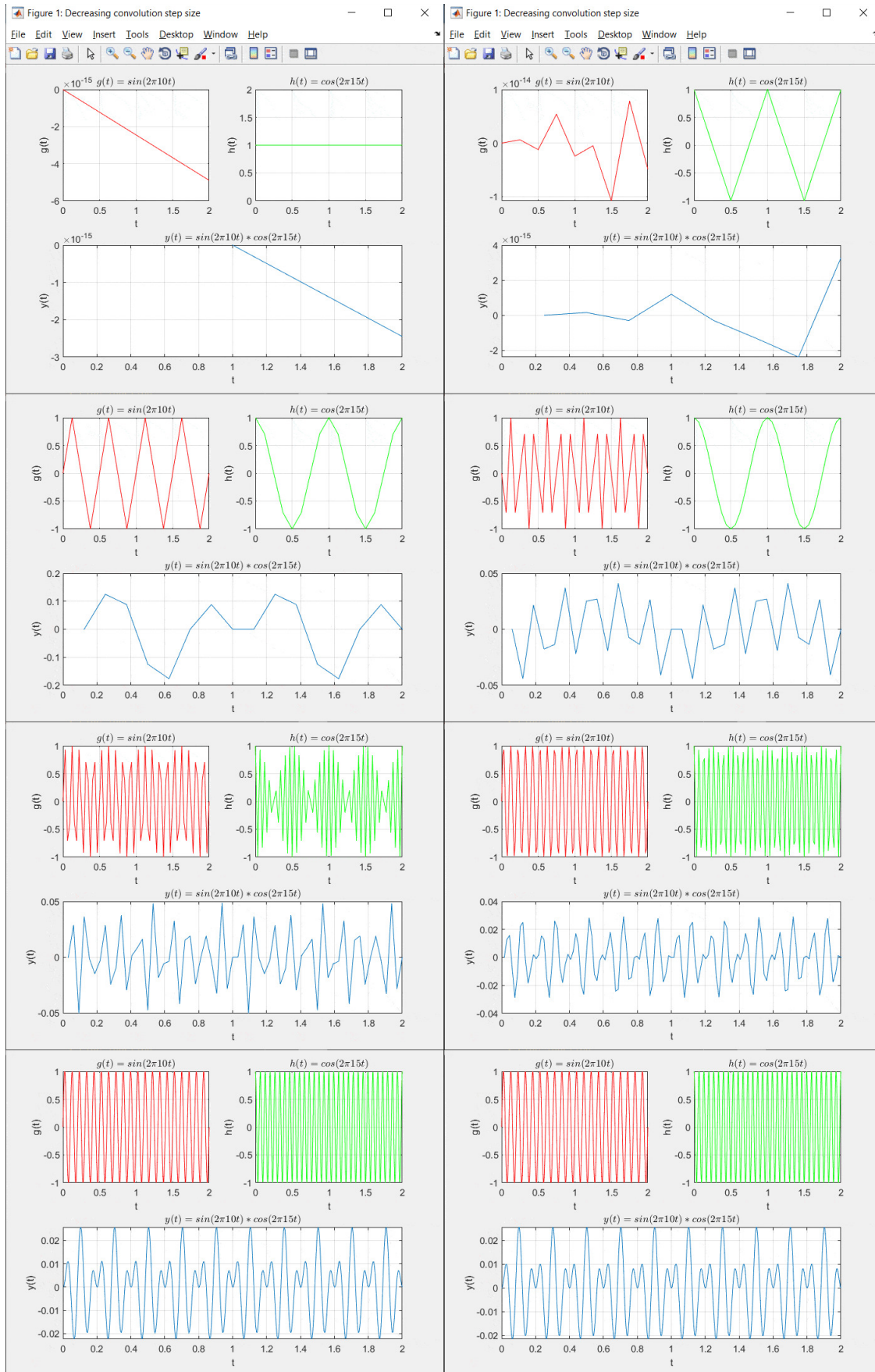
With the final result with the smallest step size of 2^{-10}:
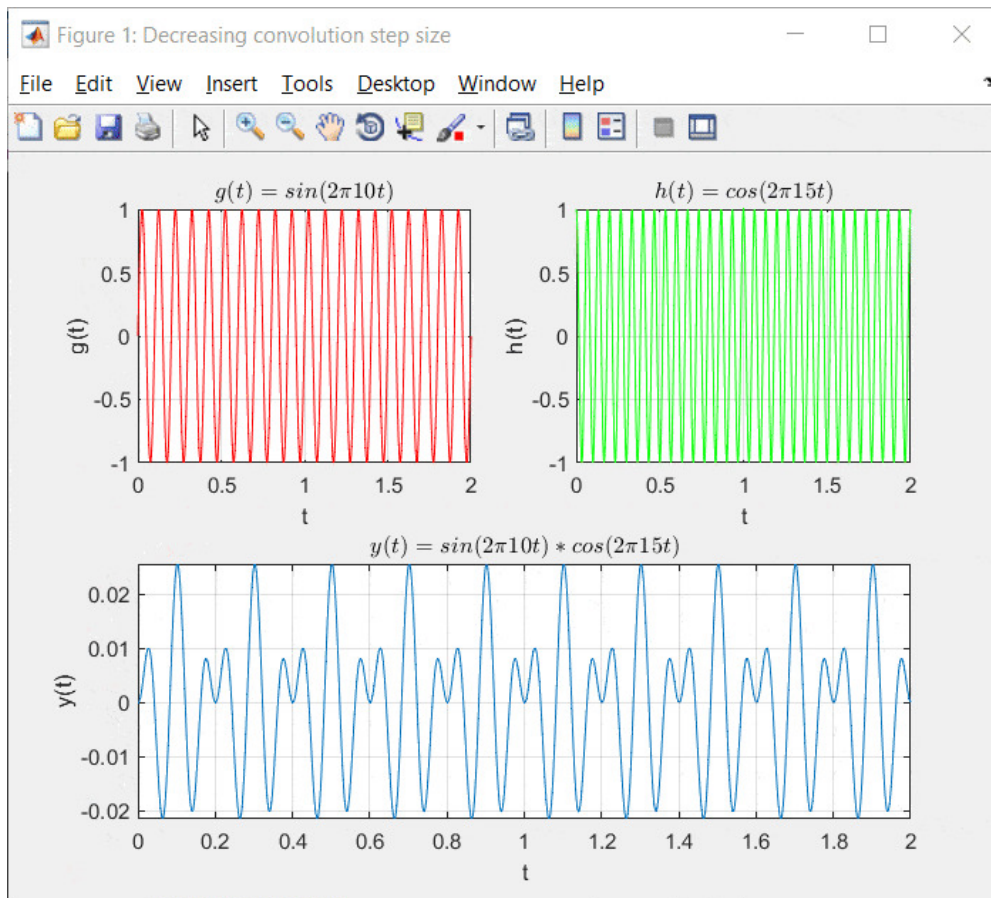
1.3.2 Python results

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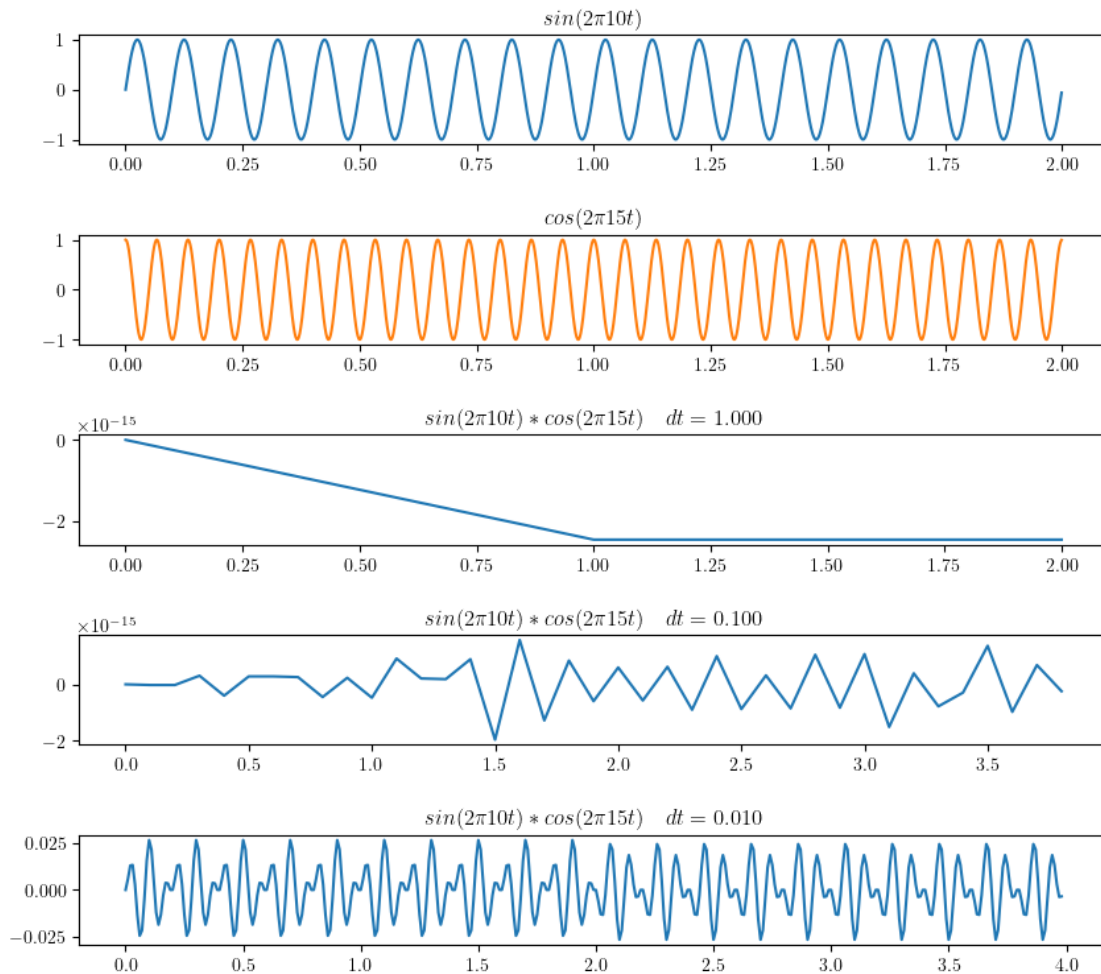
```
dtv = 1
for i in range(1,4):
    tv = np.arange(0,2,dtv)
    gt2 = np.sin(2*np.pi*f1*tv)
    ht2 = np.cos(2*np.pi*f2*tv)
    yt = dtv*np.convolve(gt2,ht2,mode='full')
    t2 = dtv*np.arange(0,len(yt))
    plt.subplot(5,1,i+2)
    plt.plot(t2,yt)
    plt.title('$sin(2 \pi 10t)*cos(2 \pi 15t) \quad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

plt.show()
```
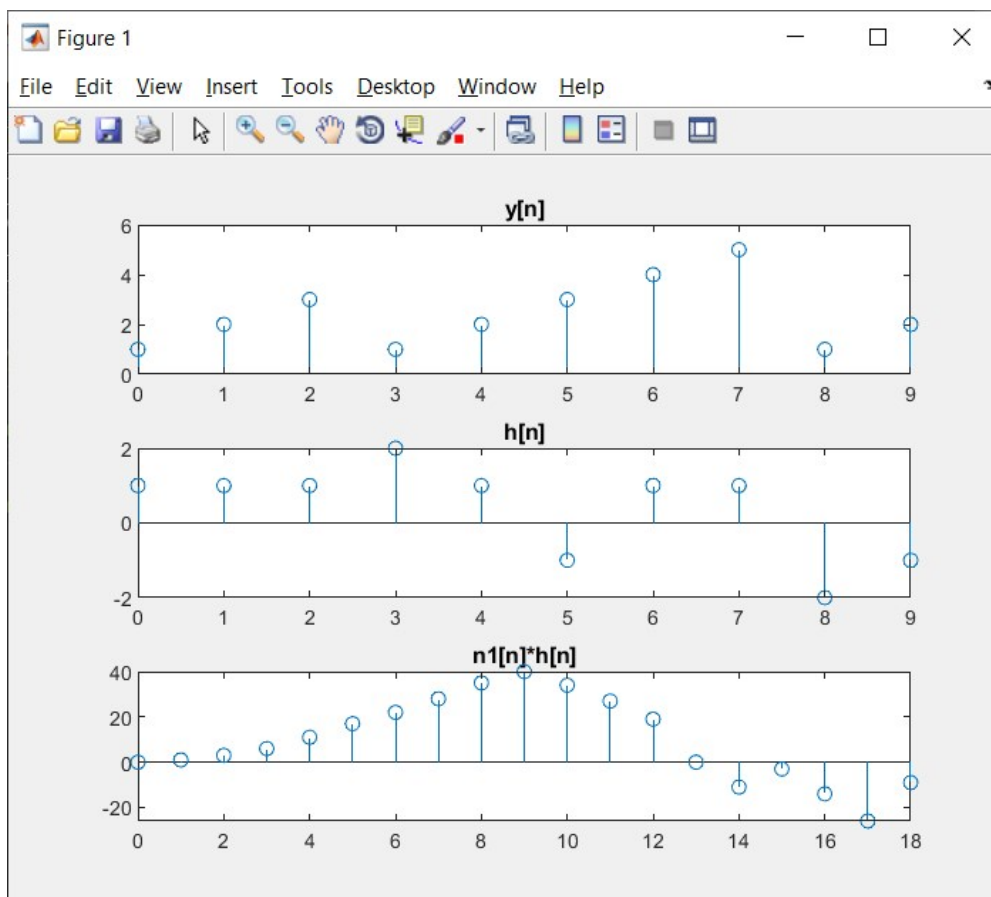

1.4 Question 4

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

```
question_1_2.m  ✕   question_3.m  ✕   question_4.m  ✕   +
 1 -    clear;
 2 -    close all;
 3 -    n1 = 0:1:9;                          % discrete time n1: 0 and first 9 integers
 4 -    y1 = [ 1 2 3 1 2 3 4 5 1 2];         % 1st functin y1
 5 -    h1 = [ 1 1 1 2 1 -1 1 1 -2 -1];      % 2nd function h1
 6 -    X = conv(n1, h1);                    % convolution of n1 (ramp) and h1
 7 -    n2 = 0:length(X)-1;                  % setting up new time axis n2 for X
 8 -    figure(1)
 9 -    subplot(3,1,1)
10 -    stem(n1, y1)                         % plotting 1st function y1 to n1
11 -    title('y[n]');
12 -    subplot(3,1,2)
13 -    stem(n1, h1)                         % plotting 2nd function h1 to n1
14 -    title('h[n]');
15 -    subplot(3,1,3)
16 -    stem(n2, X)                          % plotting convolution to its axis n2
17 -    title('n1[n]*h[n]');
18
```

The result of the plot is a discrete time convolution:

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

11