CENG 384 - Signals and Systems for Computer Engineers
Spring 2023
Homework 2

Geçit, Emre
e2521581@ceng.metu.edu.tr

Yancı, Baran
e2449015@ceng.metu.edu.tr

April 22, 2023

1. (a)

$$y(t) = x(t) - 5\dot{y}(t)$$

(b)

$$y(t) = (e^{-t} + e^{-3t})u(t) - 5\dot{y}(t)$$
$$y(t) + 5\dot{y}(t) = (e^{-t} + e^{-3t})u(t)$$
$$y(t) = y_p(t) + y_h(t)$$
$$y_p(t) = Ke^{-t}u(t) + Le^{-3t}u(t)$$
$$Ke^{-t}u(t) + Le^{-3t}u(t) + 5(-Ke^{-t}u(t) - 3Le^{-3t}u(t)) = (e^{-t} + e^{-3t})u(t)$$
$$Ke^{-t}u(t) + Le^{-3t}u(t) - 5Ke^{-t}u(t) - 15Le^{-3t}u(t) = (e^{-t} + e^{-3t})u(t)$$
$$e^{-t}u(t)(K - 5K) + e^{-3t}u(t)(L - 15L) = (e^{-t} + e^{-3t})u(t)$$
$$K - 5K = 1$$
$$K = -1/4$$
$$L - 15L = 1$$
$$L = -1/14$$
$$y_p(t) = \frac{-1}{4}e^{-t}u(t) + \frac{-1}{14}e^{-3t}u(t)$$
$$y_h(t) = c_1 e^{\alpha t}$$
$$c_1 e^{\alpha t} + 5\alpha c_1 e^{\alpha t} = 0$$
$$c_1 + 5\alpha c_1 = 0$$
$$\alpha = \frac{-1}{5}$$
$$y_h(t) = c_1 e^{\frac{-1}{5}t}$$
$$y(t) = y_p(t) + y_h(t)$$
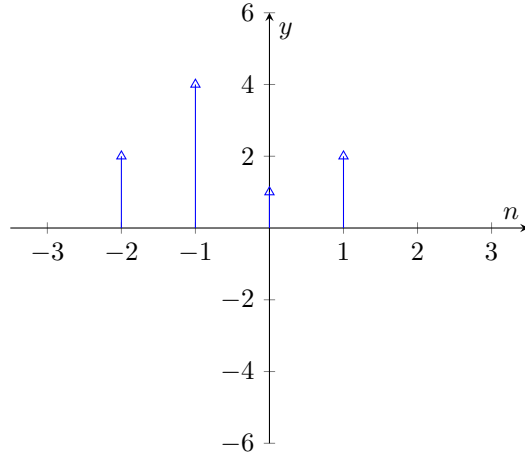$$= \frac{-1}{4}e^{-t}u(t) + \frac{-1}{14}e^{-3t}u(t) + c_1 e^{\frac{-1}{5}t}$$
$$y(0) = 0$$
$$0 = \frac{-1}{4} + \frac{-1}{14} + c_1$$
$$c_1 = \frac{9}{28}$$
$$y(t) = \frac{-1}{4}e^{-t}u(t) + \frac{-1}{14}e^{-3t}u(t) + \frac{9}{28}e^{\frac{-1}{5}t}$$

2. (a)

$$y[n] = x[n] * h[n]$$

$$= \sum_{k=0}^{n} x[k]h[n-k]$$

$$= \sum_{k=0}^{n} (2\delta[k] + \delta[k+1]) (\delta[n-(1+k)] + 2\delta[n+1-k])$$

$$= 2\sum_{k=0}^{n} \delta[k]\delta[n-(1+k)] + 4\sum_{k=0}^{n} \delta[k]\delta[n+1-k] + \sum_{k=0}^{n} \delta[k+1]\delta[n-(1+k)] + 2\sum_{k=0}^{n} \delta[k+1]\delta[n+1-k]$$

$$= 2\delta[n-1] + 4\delta[n+1] + \delta[n] + 2\delta[n+2]$$

(b)

$$y(t) = \frac{dx(t)}{dt} * h(t)$$

$$= \frac{d}{dt}(u(t-1) + u(t+1)) * e^{-t}\sin(t)u(t)$$

$$= (\delta(t-1) + \delta(t+1)) * e^{-t}\sin(t)u(t)$$

$$= \int_{-\infty}^{\infty} (\delta(\tau-1) + \delta(\tau+1)) e^{-t-\tau}\sin(t-\tau)d\tau$$

$$= \int_{-\infty}^{\infty} \delta(\tau-1)e^{-t-\tau}\sin(t-\tau)d\tau + \int_{-\infty}^{\infty} \delta(\tau+1)e^{-t-\tau}\sin(t-\tau)d\tau$$

$$= e^{-t-1}\sin(t-1)u(t-1) + e^{-t+1}\sin(t+1)u(t+1)$$

3. (a)

$$y(t) = x(t) * h(t)$$

$$= \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau$$

$$= \int_{-\infty}^{\infty} x(t-\tau)h(\tau)d\tau$$

$$= \int_{-\infty}^{\infty} e^{-(t-\tau)}e^{-2\tau}d\tau$$

$$= \int_{0}^{t} e^{-(t-\tau)}e^{-2\tau}d\tau$$

$$= e^{-t}\int_{0}^{t} e^{\tau}d\tau$$

$$= e^{-t}\left(1 - e^{-t}\right)u(t)$$

(b)

$$y(t) = x(t) * h(t)$$

$$= \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau$$

$$= \int_{-\infty}^{\infty} x(t-\tau)h(\tau)d\tau$$

$$= \int_{-\infty}^{\infty} \left(u(t-\tau) - u(t-(\tau+1)) \right) e^{3\tau} d\tau$$

$$= \int_{-\infty}^{\infty} u(t-\tau)e^{3\tau}d\tau - \int_{-\infty}^{\infty} u(t-(\tau+1))e^{3\tau}d\tau$$

$$= \int_{-\infty}^{t} e^{3\tau}d\tau - \int_{-\infty}^{t-1} e^{3\tau}d\tau$$

$$= \frac{e^{3t}}{3} - \frac{e^{3t-3}}{3}$$

4. (a)

$$y[n] - y[n-1] - y[n-2] = 0$$
$$y[n] = y[n-1] + y[n-2]$$
$$y[2] = y[1] - y[0] = 2$$
$$y[3] = y[2] - y[1] = 3$$
$$y[4] = y[3] - y[2] = 5$$
$$y[5] = y[4] - y[3] = 8$$
$$\dots$$

It is the Fibonacci sequence.

$$y[n] = \frac{(1+\sqrt{5})^n - (1-\sqrt{5})^n}{2^n\sqrt{5}}$$

(b)

$$y^{(3)}(t) - 6y''(t) + 13y'(t) - 10y(t) = 0$$
$$K^3 - 6K^2 + 13K - 10 = 0$$
$$K(K-5)(K-2) + (K+5)(K-2) = 0$$
$$(K-2)(K^2 - 4K + 5) = 0$$

$$K = 2, 2-j, 2+j$$

$$y_h(t) = c_1 e^{2t} + c_2 e^{(2+j)t} + c_3 e^{(2-j)t}$$
$$y_h(t) = c_1 e^{2t} + c_2 \left(e^{2t}\cos(t) + je^{2t}\sin(t) \right) + c_3 \left(e^{2t}\cos(t) - je^{2t}\sin(t) \right)$$
$$y_h(t) = c_1 e^{2t} + c_2 e^{2t}\cos(t) + c_3 e^{2t}\cos(t) + c_2 je^{2t}\sin(t) - c_3 je^{2t}\sin(t)$$
$$y_h(t) = c_1 e^{2t} + (c_2 + c_3)e^{2t}\cos(t) + j(c_2 - c_3)e^{2t}\sin(t)$$
$$y_h(t) = C_1 e^{2t} + C_2 e^{2t}\cos(t) + C_3 e^{2t}\sin(t)$$

$$y''(0) = 3$$
$$= 4C_1 + 3C_2 + 4C_3$$
$$y'(0) = 1.5$$
$$= 2C_1 + 2C_2 + C_3$$
$$y(0) = 1$$
$$= C_1 + C_2$$

$$C_1 = 2$$
$$C_2 = -1$$
$$C_3 = -0.5$$

$$y_h(t) = 2e^{2t} - e^{2t}\cos(t) - \frac{1}{2}e^{2t}\sin(t)$$

5. (a)

$$y''(t) + 5y'(t) + 6y(t) = \cos(5t)$$
$$= \frac{e^{j5t} + e^{-j5t}}{2}$$
$$= \frac{e^{j5t}}{2} + \frac{e^{-j5t}}{2}$$

$$y_p(t) = c_1 e^{j5t} + c_2 e^{-j5t}$$

$$-25c_1 e^{j5t} - 25c_2 e^{-j5t} + 25jc_1 e^{j5t} - 25jc_2 e^{-j5t} + 6c_1 e^{j5t} + 6c_2 e^{-j5t} = \frac{e^{j5t}}{2} + \frac{e^{-j5t}}{2}$$
$$e^{j5t}\left(-19c_1 + 25jc_1\right) + e^{-j5t}\left(-19c_2 - 25jc_2\right) = \frac{e^{j5t}}{2} + \frac{e^{-j5t}}{2}$$

$$c_1 = \frac{5j}{-38 + 50j}$$
$$c_2 = \frac{5j}{38 + 50j}$$
$$y_p(t) = \frac{5j}{-38 + 50j}e^{j5t} + \frac{5j}{38 + 50j}e^{-j5t}$$

(b)

$$y''(t) + 5y'(t) + 6y(t) = 0$$
$$K^2 + 5K + 6 = 0$$
$$K = -3, -2$$

$$y_h(t) = c_1 e^{-3t} + c_2 e^{-2t}$$

(c)

$$y(t) = y_p(t) + y_h(t)$$
$$= \frac{5j}{50j - 38}e^{j5t} + \frac{5j}{50j + 38}e^{-j5t} + c_1 e^{-3t} + c_2 e^{-2t}$$

$$y(0) = y'(0) = 0$$

$$\frac{5j}{50j - 38} + \frac{5j}{50j + 38} + c_1 + c_2 = 0$$

$$\frac{-25}{50j - 38} + \frac{25}{50j + 38} - 3c_1 - 2c_2 = 0$$

$$c_1 = \frac{25}{34}$$

$$c_2 = \frac{-25}{29}$$

$$y(t) = \frac{5j}{50j - 38} e^{j5t} + \frac{5j}{50j + 38} e^{-j5t} + \frac{25}{34} e^{-3t} - \frac{-25}{29} e^{-2t}$$

6. (a)

$$w[n] - \frac{1}{2}w[n - 1] = x[n]$$

$$w[n] = 0, \forall n < 0$$

$$w[0] - \frac{1}{2}w[-1] = x[0]$$

$$w[0] = x[0]$$

$$w[1] - \frac{1}{2}w[0] = x[1]$$

$$w[1] = x[1] + \frac{1}{2}x[0]$$

$$w[2] - \frac{1}{2}w[1] = x[2]$$

$$w[2] = x[2] + \frac{1}{2}x[1] + \frac{1}{4}x[0]$$

$$\ldots$$

$$w[n] = \sum_{k=0}^{n} 2^{-k} x[n - k]$$

$$h_0[n] = \sum_{k=0}^{n} 2^{-k} \delta[n - k]$$

$$= 2^{-n} \qquad\qquad (\delta[n - k] = 0 \text{ for } k! = n)$$

(b) If we feed the first system with the unit impulse, we get $h_0[n]$. If we feed the second system with $h_0[n]$, we get

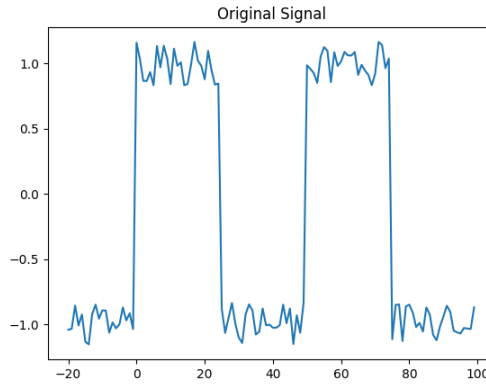$$h[n] = \sum_{k=0}^{n} 2^{-k} 2^{-n+k}$$

$$= \sum_{k=0}^{n} 2^{-n}$$

$$= (n + 1) * 2^{-n}$$

(c)

$$y[n] = x[n] * h[n]$$

$$= \sum_{k=-\infty}^{\infty} x[n-k]h[k]$$

$$= \sum_{k=0}^{\infty} x[n-k]h[k] \qquad\qquad (h[k] = 0 \text{ for } k < 0)$$

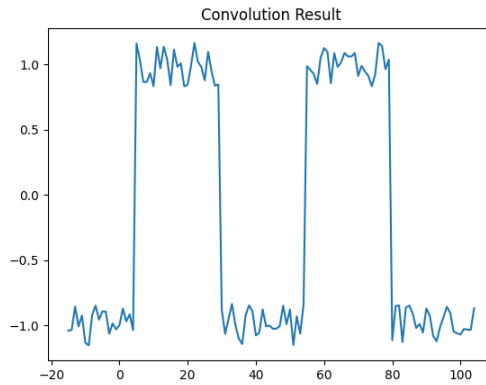$$= \sum_{k=0}^{\infty} x[k] * (n+1) * 2^{k-n}$$

7. The code used to generate the plots is at the end of the answer.

Figure 1: Original Signal

(a) The effect of convoluting with $\delta[n-5]$ is to shift the signal by 5 units to the right.

Figure 2: Convolution with $h[n] = \delta[n-5]$

(b) By convolving a signal with an n-point moving average filter, it is possible to effectively smooth the abrupt transitions or edges in the signal, as well as attenuate any unwanted noise present in the signal.

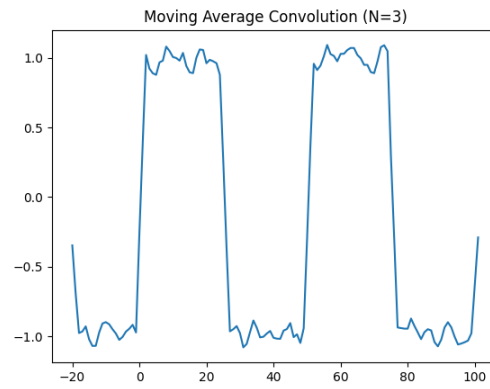Figure 3: Convolution with the moving average filter, N = 3

Moving Average Convolution (N=3)

Figure 4: Convolution with the moving average filter, N = 5

Moving Average Convolution (N=5)
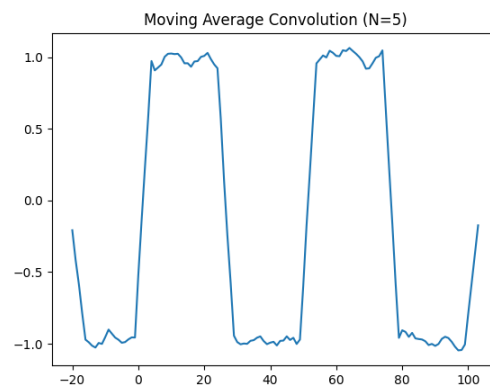
Figure 5: Convolution with the moving average filter, N = 10

Moving Average Convolution (N=10)
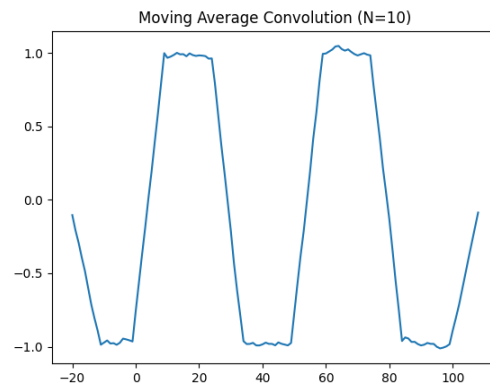
Figure 6: Convolution with the moving average filter, N = 20

Moving Average Convolution (N=20)
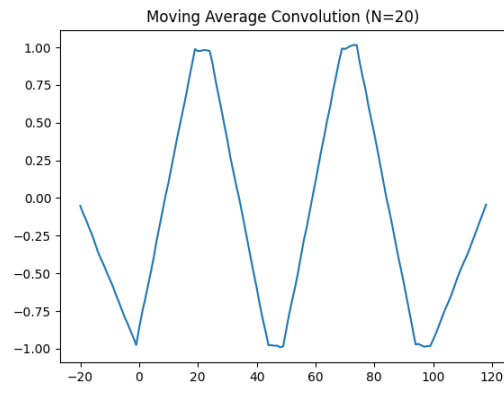
Figure 7: Python Code

```python
from matplotlib import pyplot


class Signal:
    def __init__(self, data, start):
        self.data = data
        self.start = start
        self.end = start + len(data) - 1

    def __getitem__(self, index):
        if index < self.start or index > self.end:
            return 0
        return self.data[index - self.start]

    def __mul__(self, other):
        return ConvolutionResult(self, other)

    def plot(self, title, save_path):
        pyplot.title(title)
        pyplot.plot(range(self.start, self.end + 1), self.data)
        pyplot.savefig(save_path)
        pyplot.clf()


class ConvolutionResult(Signal):
    def __init__(self, signal1, signal2):
        self.signal1 = signal1
        self.signal2 = signal2
        self.start = signal1.start + signal2.start
        self.end = signal1.end + signal2.end
        self.data = [self[i] for i in range(self.start, self.end + 1)]

    def __getitem__(self, index):
        if index < self.start or index > self.end:
            return 0
        result = 0
        for i in range(self.start, self.end + 1):
            result += self.signal1[i] * self.signal2[index - i]
        return result


with open("hw2_signal.csv", "r", encoding="ascii") as file:
    raw_data = [float(item) for item in file.read().split(",")]

x = Signal(raw_data[1:], int(raw_data[0]))
x.plot("Original Signal", "images/original.png")

h = Signal([1], 5)
convolution = h * x
convolution.plot("Convolution Result", "images/convolution.png")

for N in (3, 5, 10, 20):
    m = Signal([1 / N] * N, 0)
    moving_average = m * x
    moving_average.plot(f"Moving Average Convolution (N={N})", f"images/moving_average_{N}.png")
```