Digital Signal Processing Laboratory
Department of Electrical & Computer Engineering

Digital signal processing

Lab 3

Digital filters

Heraklion 2025

Dr. Konstantinos Karampidis

Convolution

With the sum of the convolution, we can find the response of a discrete-time system for an input x(n), if we know its impulse response h(n).

The output y(n) of the system will be equal to the convolution of the input x(n) and the impulse response h(n) of the system and is defined as follows:

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

where the symbol * corresponds to the convolution operator.

Convolution sum

The Sum of Convolution can also be written as follows:

$$y(n) = x(n) * h(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k)$$

When the system and the input sequence are causal (i.e.

$$y(n) = \sum_{k=0}^{n} x(k)h(n-k) \qquad \text{Για } n=0,\ldots,N+M-2$$

h(n)=x(n)=0, for any n<0 and have sample lengths of M and N respectively) then the limits of the sum change and the convolution equation takes the form:

Therefore the response y has magnitude N+M-1

Convolve Python function

To calculate the sum of the convolution you can use the Python function.

```
import numpy as np
np.convolve (A, B)
np.convolve (A, B, SHAPE)
```

Convolve two vectors A and B.

np.convolve (A, B, mode='full') Return the full convolution. The result is a vector with length equal to 'length (A) + length (B) - 1'.

When A and B are the coefficient vectors of two polynomials, the convolution represents the coefficient vector of the product polynomial.

Example

Suppose we have the following input signal x(n) and its impulse response h(n). Calculate the convolution using the Python function.

$x(n) = 3n^3 + n^2 + 2n + 1$
$h(n) = n^2 + 2n + 3$

First, we find the vectors a and b where we need to insert as arguments to the convolve function.

```
a = [3, 1, 2, 1];
b = [1, 2, 3];
y = np.convolve(a, b)
```

Output

```
y = 3   7   13   8   8   3
```

Example

Perform the convolution calculation for the same signals x(n) and h(n) by hand.

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

$(3n^3 + n^2 + 2n + 1)(n^2 + 2n + 3) = 3n^5 + 6n^4 + 9n^3 + n^4 + 2n^3 + 3n^2 + 2n^3 + 4n^2 + 6n + n^2 + 2n + 3 = \mathbf{3}n^5 + \mathbf{7}n^4 + \mathbf{13}n^3 + \mathbf{8}n^2 + \mathbf{8}n + \mathbf{3}$

Circular Convolution

Circular convolution is commonly used in the context of periodic discrete-time signals and the discrete Fourier transform (DFT). It involves summing in a circular or periodic shift of one of the signals.

For two discrete-time sequences $x[n]$ and $h[n]$, both of length N, the circular convolution $y[n]$ is defined as follows:

$$y[n] = (x \circledast h)[n] = \sum_{k=0}^{N-1} x[k]h[(n-k) \mod N]$$

Circular Convolution

Circular convolution steps

Ensuring the same length: Make sure that both sequences $x[n]$ and $h[n]$ have the same length N. If not, fill in the shorter sequence with zeros.

Circular shift: Circular shift (to the right or left) of a sequence and multiply it by the other sequence (element wise).

Sum of products: For each shift, sum the products to get the output sequence.

Circular Convolution

Example

Consider two sequences:
$x[n]$={1,2,3}
$h[n]$={4,5,6}

To calculate the circular convolution we follow the steps below:

Zero Padding: not necessary here, as both sequences have length 3:
$x[n]$={1,2,3}
$h[n]$={4,5,6}

Circular Convolution

Circular shift:

1) Initial sequence h[0]= [4,5,6]
2) One position right shift: h[1]=[6,4,5]
3) Two positions right shift: h[2]=[5,6,4]

Then calculate the sum of the products for each shift.

- $y[0] = x[0]h_0[0] + x[1]h_0[1] + x[2]h_0[2] = 1 \cdot 4 + 2 \cdot 5 + 3 \cdot 6 = 31$
- $y[1] = x[0]h_1[0] + x[1]h_1[1] + x[2]h_1[2] = 1 \cdot 6 + 2 \cdot 4 + 3 \cdot 5 = 29$
- $y[2] = x[0]h_2[0] + x[1]h_2[1] + x[2]h_2[2] = 1 \cdot 5 + 2 \cdot 6 + 3 \cdot 4 = 29$

Therefore the result of the circular convolution will be y[n]=[31,29,29]

Circular Convolution

```
# Circular shift visualization
import numpy as np
# Define the signal
h = np.array([4, 5, 6])

# Circular right shift by 1
h_right_1 = np.roll(h, 1)
print("Right Shift by 1:", h_right_1)

# Circular right shift by 2
h_right_2 = np.roll(h, 2)
print("Right Shift by 2:", h_right_2)

# Circular left shift by 1
h_left_1 = np.roll(h, -1)
print("Left Shift by 1:", h_left_1)

# Circular left shift by 2
h_left_2 = np.roll(h, -2)
print("Left Shift by 2:", h_left_2)
```

Circular Convolution

#1st way

```
import numpy as np

def circular_convolution(x, h):
    N = len(x)
    y = np.zeros(N)
    for n in range(N):
        for k in range(N):
            y[n] += x[k] * h[(n - k) % N]
    return y

# Define the signals
x = np.array([1, 2, 3])
h = np.array([4, 5, 6])
# Compute the circular convolution
y = circular_convolution(x, h)

print("Circular Convolution Result:", y)
```

Circular Convolution

#2nd way

```
import numpy as np
# Define the signals
x = np.array([1, 2, 3])
h = np.array([4, 5, 6])
# Compute the circular convolution using numpy
y = np.fft.ifft(np.fft.fft(x) * np.fft.fft(h))
print("Circular Convolution Result:", np.real(y))
```

Circular Convolution

y = np.fft.ifft(np.fft.fft(x) * np.fft.fft(h)) : Calculates the circular convolution of x and h using the Fast Fourier Transform (FFT). The function np.fft.fft computes the discrete Fourier transform (DFT) for the input sequences x and h.

np.fft.fft(x): It computes the DFT of the sequence x.

np.fft.fft(h): It computes the DFT of the sequence h.

np.fft.fft(x) * np.fft.fft(h): It multiplies the results of the two DFTs element-wise. This step corresponds to multiplication in the frequency domain, which is equivalent to convolution in the time domain.

np.fft.ifft(...): Calculates the inverse DFT of the product of the DFTs to obtain the result of the circular convolution in the time domain. Returns a complex matrix where the real part represents the result.

Frequency Response

The frequency response characterises the way in which the amplitude and phase of an output signal varies with the frequency of the input signal. It shows how the various frequency components are amplified or attenuated by the system and how their phases are shifted.

Key components of the frequency response

- **Magnitude response**: $|H(f)|$
  Shows the amplitude of the output signal as a function of frequency.
- **Phase Response** : $\angle H(f)$

Frequency response

For a discrete system with an impulse response h(n), its frequency response will be the Fourier transform of the impulse response.

$$H(e^{j\omega}) = \sum_{n=-\infty}^{\infty} h[n]e^{-j\omega n}$$

$H(e^{j\omega})$ is, in the general case, a complex number and depends on the frequency ω of the complex exponential function.
Therefore, it fully characterizes y(n) from the time domain to that of frequencies.

Frequency Response - Example

Consider a low-pass filter:

- **Magnitude Response:** allows low frequency components to pass while attenuating high frequency components.
- **Phase Response:** Shows how the filter affects the phase of various frequency components.

Bode diagram

A Bode diagram is a graphical representation of the frequency response of a system. It consists of two diagrams:
Magnitude Plot: Shows the magnitude response |H(f)| versus frequency (often on a logarithmic scale).

Phase plot: shows the phase response $\angle H(f)$ versus frequency.

The freqz function of Python

The Python frequency response calculation function is freqz.

```
import scipy
 H, W = scipy.signal.freqz(B, A,N)
```

Return the complex frequency response H of the rational IIR filter whose numerator and denominator coefficients are B and A, respectively.
If N is omitted, a value of 512 is assumed. For fastest computation, N should factor into a small number of small primes.

Exercise 1

For the system with an equation of diffrences:

$y(n) = 1.1y(n-1)-0.5y(n-2)-0.3y(n-4)+0.5x(n)-0.2x(n-1)$

Calculate:

a) The impulse response for the range [0,10].

b) The system response for the following input using the convolve function.

x = [5 1 1 1 0 0 1 1 1 0]

c) The response for the above input using the lfilter function.

Solution

From the differences equation we see that in order to calculate the impulse response with our code we need to calculate the initial conditions for n=[0,4].

$h(n) = 1.1h(n-1)-0.5h(n-2)-0.3h(n-4)+0.5\delta(n)-0.2\delta(n-1)$

$h(0) = 0 - 0 - 0 + 0.5 - 0 = 0.5$

$h(1) = 1.1(0.5) - 0 - 0 + 0 - 0.2 = 0.55 - 0.2 = 0.35$

$h(2) = 1.1(0.35) - 0.5(0.5) - 0 + 0 - 0 = 0.385 - 0.25 = 0.135$

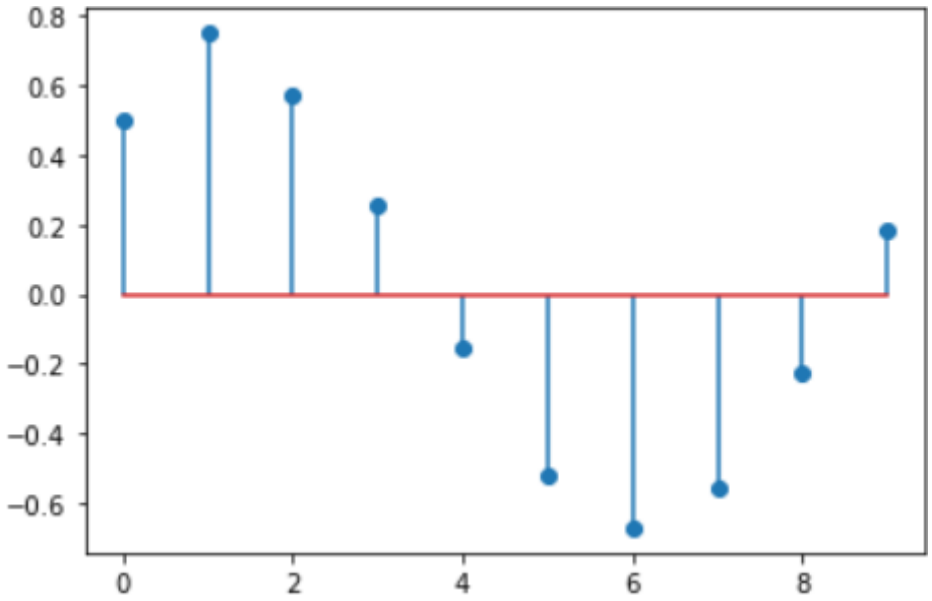
$h(3) = 1.1(0.135) - 0.5(0.35) - 0 + 0 - 0 = 0.1485 - 0.175 = -0.0265$

$h(4) = 1.1(-0.0265) - 0.5(0.135) - 0.3(0.5) + 0 - 0 = -0.0215 - 0.0675 - 0.15$

$= -0.24665$

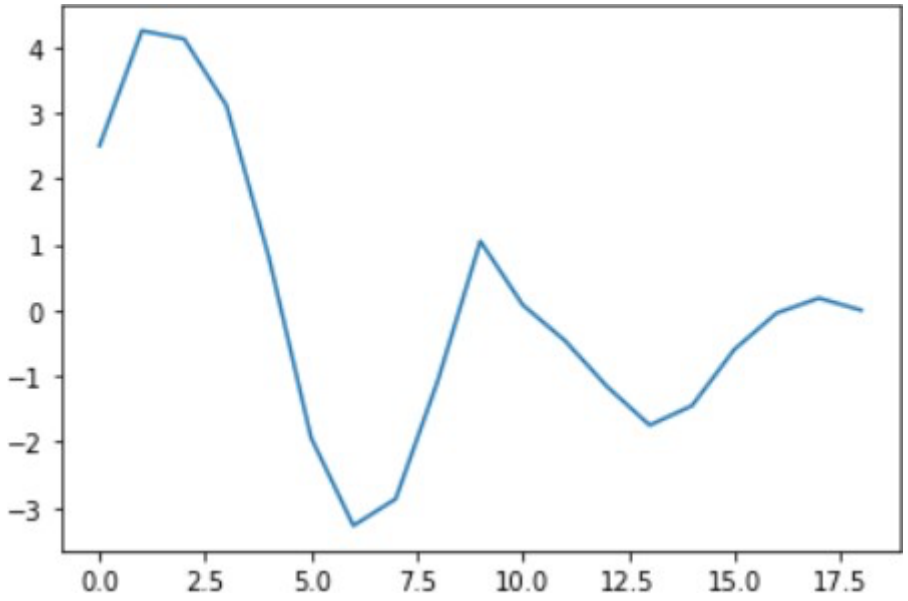
Using the lfilter of Python

```
delta = lambda n: n==0
a = [1, -1.1, 0.5, 0, 0.3];
b = [0.5, -0.2, 0, 0, 0];
x=np.zeros(10)
for n in range(0,10,1):
    temp=int(delta(n))
    x[n]=temp
n=range(0,10,1)
h= scipy.signal.lfilter(b,a,x)
plt.stem(n,h)
```



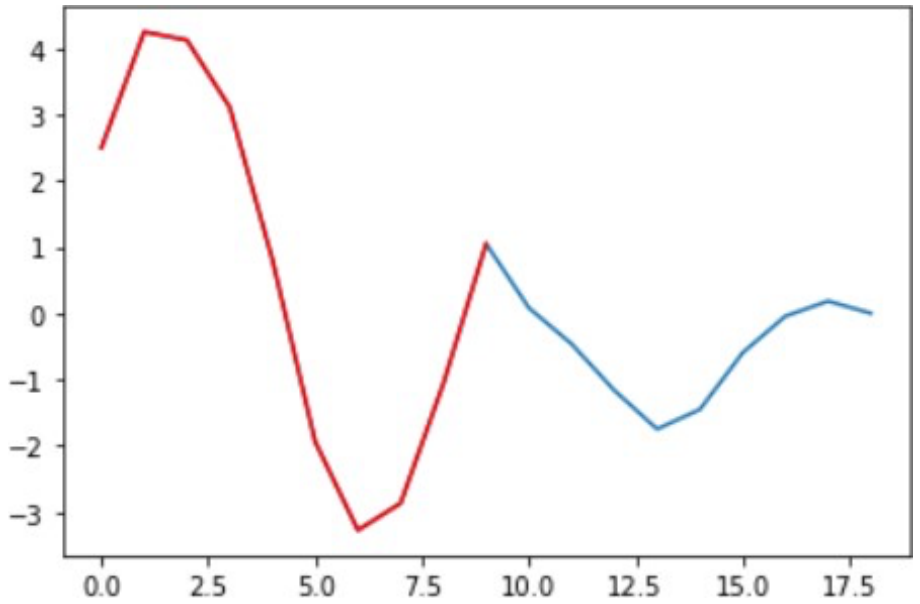
To calculate the response (using the function convolve) add at the end of the previous code:

```
x=[5, 1, 1, 1, 0, 0, 1, 1, 1, 0];

y=np.convolve(x, h);

plt.plot(y)

plt.show()
```



To calculate the response (using the lfilter function) add at the end of the previous code:

```
y2=scipy.signal.lfilter(b, a, x);
plt.plot(y);
plt.plot(y2,'r');
```

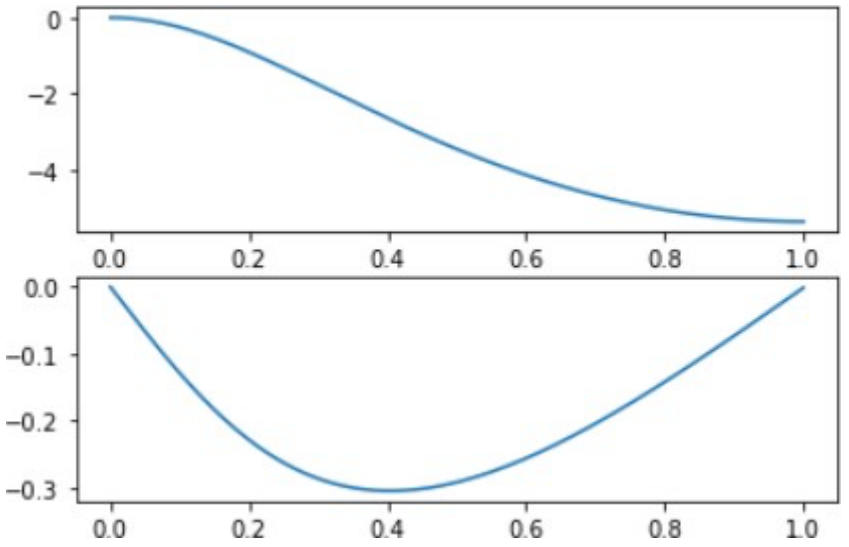


Exercise 2

Calculate and show the graph of the frequency response (magnitude and phase) of the following filter.
y(n) = 0.3y(n-1) + 0.7x(n) για n>=0

Solution

```
#y(n)-0.3y(n-1) = 0.7x(n)
a=[1, -0.3]
b=[0.7, 0]
w,h= scipy.signal.freqz(b,a)
angles = np.unwrap(np.angle(h))


plt.subplot(2, 1, 1)
plt.plot(w/max(w), 20 * np.log10(abs(h)))
plt.subplot(2, 1, 2)
plt.plot(w/max(w), angles)
```

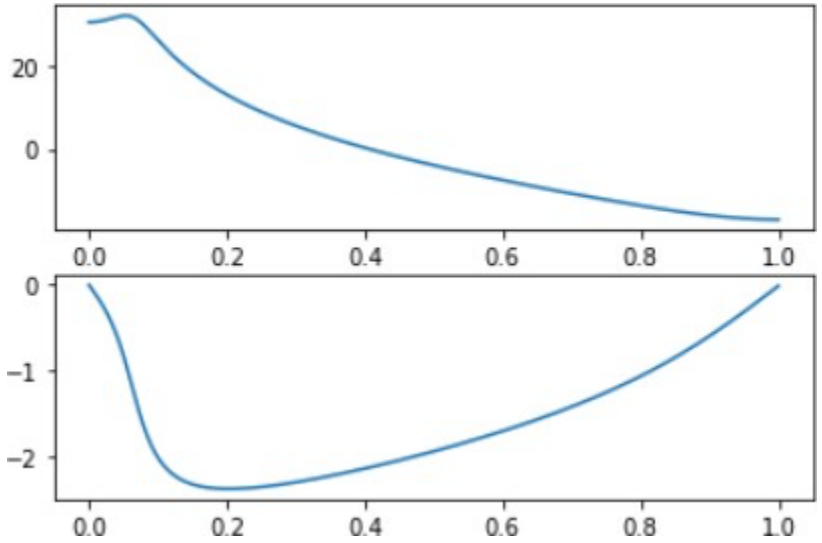


Exercise 3

Calculate and plot the frequency response (magnitude and phase) with 512 samples in the upper half of the unit cycle (ω=0-π) of the filter with the following transfer function.

$$H(z) = \frac{1 + 0.5z^{-1}}{1 - 1.8\cos(\frac{\pi}{16})z^{-1} + 0.81z^{-2}}$$

```
a=[1,-1.8*np.cos(np.pi/16), 0.81]
b=[1, 0.5, 0]
w, h = scipy.signal.freqz(b,a)
angles = np.unwrap(np.angle(h))

plt.subplot(2, 1, 1)
plt.plot(w/max(w), 20 * np.log10(abs(h)))
plt.subplot(2, 1, 2)
plt.plot(w/max(w), angles)
```



Exercise 4

Suppose we have an FIR filter with the following impulse response: h[n]={0.25,0.5,0.25}
Calculate and plot the frequency response.

Solution

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import freqz

# Define the impulse response of the FIR filter
h = np.array([0.25, 0.5, 0.25])

# Compute the frequency response. The w array contains the frequencies (in radians per sample), and the H array contains the corresponding frequency response values.
w, H = freqz(h)
```

```
# Plot the magnitude response
plt.figure()
plt.subplot(2, 1, 1)
plt.plot(w, np.abs(H), 'b')
plt.title('Frequency Response')
plt.xlabel('Frequency
(rad/sample)')
plt.ylabel('Magnitude')
plt.grid()

# Plot the phase response
plt.subplot(2, 1, 2)
plt.plot(w, np.unwrap(np.angle(H)), 'b')
plt.xlabel('Frequency (rad/sample)')
plt.ylabel('Phase (radians)')
plt.grid()
plt.tight_layout()
plt.show()
```