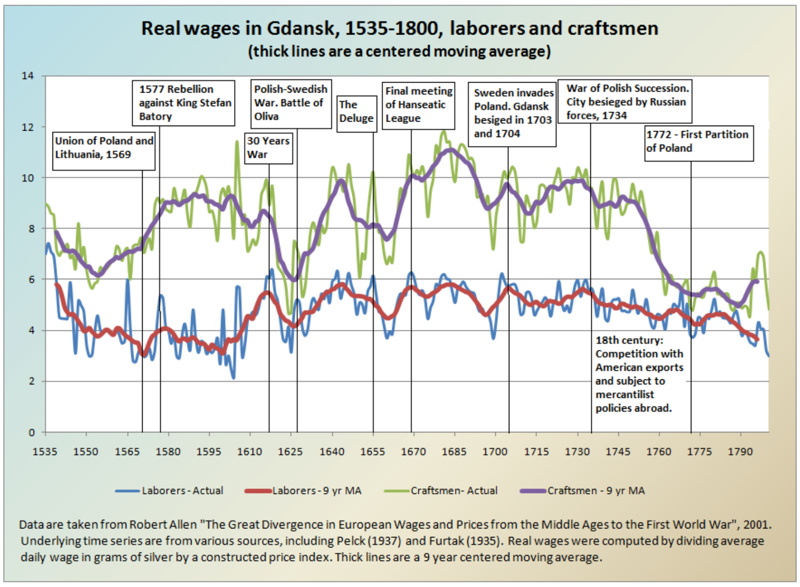
Signals and Systems
Lab 1: Convolution and correlation



Data Science and Engineering, 2019-2020

Contents

1 Introduction

The goal of this lab is to work experimentally with convolution, an important operation in signal and image processing. An operation closely related to convolution is correlation, which is useful to automatically locate particular patterns in a signal, as well as to measure delays or periodicities.

During the lab session complete the work of section 2. You may find useful starting your code as follows:

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import convolve
import pandas as pd


def myconvolve(x, h):
    N = x.size
    M = h.size
    P = XXX #Modify this line
    x_padded = np.concatenate((np.zeros(P), x, np.zeros(P)))
    L = M + N - 1
    y = np.zeros(L)
    h_rev = h[::-1]
    for n in range(L):
        y[n] = np.sum(x_padded[n:n+M]*h_rev)
    return y


def plot3(i, nx, x, nh, h, ny, y):
    plt.figure(i)
    ax3 = plt.subplot(3,1,3)
    plt.stem(ny, y)
    plt.subplot(3,1,1, sharex=ax3)
    plt.stem(nx, x)
    plt.subplot(3,1,2, sharex=ax3)
    plt.stem(nh, h)


#Your part starts here
```

If you are using Spyder, an IDE currently shipped with the Anaconda installer (http://anaconda.com/downloads), separating sections with #%% will allow you to execute each section independently. To know how to use the function `convolve`, type `convolve?` in the IPython console.

When using Matplotlib to plot graphs, instead of the default inline drawing, you may want to have the figures in a separate window so that interactions can be enabled. To display figures in a separate window, in Spyder click Tools, Preferences, Ipython Console, Graphics and under Graphics Backend select "automatic" instead of "inline".

For interactive plots with Jupyter Notebooks execute the command `%matplotlib notebook`.

2 Lab work part 1: Convolution and correlation of 1D signals

1. Choose a partner to work with. In class you will have to open an Atenea quiz (one per group) to introduce the answers to the questions that follow.

Discrete Time (DT) convolution

Let $y[n]$ represent the DT signal that results when $x[n]$ is convolved with $h[n]$:

$$\begin{aligned} y[n] = x[n] * h[n] &= \sum_{k=-\infty}^{\infty} x[k]h[n-k] = \sum_{k=-\infty}^{\infty} h[k]x[n-k] \\ &= \ldots + h[2]x[n-2] + h[1]x[n-1] + h[0]x[n] + \underbrace{h[-1]x[n+1] + h[-2]x[n+2] + \ldots}_{\text{Non causal part: future samples}} \end{aligned} \tag{1}$$

We will start by considering two linear and time invariant (LTI) systems with impulse responses: $h_1[n] = u[n]$ and $h_2[n] = \delta[n] - \delta[n-1]$. We will also consider two different signals $x_1[n] = p_6[n]$ and $x_2[n] = u[n]$.

Determine the closed-form expressions for each of the following:

(a) $x_1[n] * h_1[n] = p_6[n] * u[n]$

(b) $x_1[n] * h_2[n] = p_6[n] * (\delta[n] - \delta[n-1])$

(c) $x_2[n] * h_1[n] = u[n] * u[n]$

(d) $x_2[n] * h_2[n] = u[n] * (\delta[n] - \delta[n-1])$

Let us see now how to visualize previous signals in Python along with the result of their convolution.

Firstly, we will visualize $x_1[n]$, $x_2[n]$, $h_1[n]$, and $h_2[n]$ in the interval $n = -5, \ldots, 20$. To that end, we will generate the interval first with `n=np.arange(-5,21)`, and then we will generate $h_1[n]$, $h_2[n]$, $x_1[n]$, and $x_2[n]$ in that interval.

```
n = np.arange(-5, 21)
h1 = 1.0*(n >= 0)
h2 = np.zeros_like(n); h2[(n==0)]=1; h2[(n==1)]=-1
x1 = np.zeros_like(n) + (n >= 0)*(n < 6)
x2 = h1
```

To visualize the result of the convolution between $x_i[n]$ (with $i = 1, 2$) and $h_i[n]$ (with $i = 1, 2$) in the specified interval you may convolve the signals generated by using the function `convolve` from the scipy.signal package, i.e. `y = convolve(h, x)`. Once you have computed the convolution, use the provided function `plot3(i, nx, x, nh, h, ny, y)` to visualize the signals and the result of the convolution in the specified interval.

```
#Compute convolutions
ya = convolve(x1, h1)
yb = convolve(x1, h2)
yc = convolve(x2, h1)
yd = convolve(x2, h2)

plt.close('all')
ny = XXX #Modify this line
plot3(1, n, x1, n, h1, ny, ya)
```

5. Is it correct to set `ny=n` to plot `ya`?
6. Set `ny = np.arange(2*min(n), 2*max(n)+1)`. Can you visualize y_a now?
7. Until what value of n do you consider that y_a is correct, i.e. equal to the convolution $p_6[n] * u[n]$?
8. Until what value of n do you consider y_b is correct, i.e. equal to the convolution $p_6[n] * (\delta[n] - \delta[n-1])$?
9. Until what value of n do you consider y_c is correct, i.e. equal to the convolution $u[n] * u[n]$?
10. Until what value of n do you consider y_d is correct, i.e. equal to the convolution $u[n] * (\delta[n] - \delta[n-1])$?

The previous example was an academic example and serve its purpose of visualizing signals and the result of their convolution. In practice, for sequences of finite length, you may generate vectors only for the interval the contains all the non-zero samples of the signal (notice that, in this interval, you may have zero samples between non-zero samples). Such vectors will start at index 0 and end at index $N - 1$, being N the length of the interval that contains all the non-zero samples. After performing the convolution, if needed, you may represent the result in an appropriate time axis.

Following the consideration explained in the previous paragraph, let us compute the convolution $z[n] = p_6[n] * p_6[n-3]$. The first signal, $p_6[n]$ starts at $n = 0$ and ends at $n = 5$, the second signal, $p_6[n-3]$ starts at $n = 3$ and ends at $n = 8$. To compute this convolution, we do:

```
p = np.ones(6)
z = convolve(p, p)
plt.close('all')
n = np.arange(0, 6)
nz = XXX #Modify this line
plt.stem(nz, z)
```

11. With previous code we want to correctly represent $z[n]$ from `z = convolve(p, p)` with

`plt.stem(nz, z)`. To that end, how do we generate the time axis `nz`?

Although, we have used the function `convolve` to compute convolutions, the convolution operation between a signal and a L.I. filter with impulse response $h[n]$ is nothing else than a weighted sum of the samples of the input signal, with the weights given by the coefficients of $h[n]$ (as seen in eq. 1). Therefore, the provided function `myconvolve(x, h)` should do the same task, if the variable `P` is correctly expressed in terms of M and/or N.

12. Set `P` so that the function `myconvolve(x, h)` provides the same result as `convolve(x, h)`.

Removing short-term variations from a signal

In this section, we will experiment with a very simple LTI filter, the average filter, applied to a time series contained in the file Ibex35.xlsx. We will use the Pandas package to read the excel file, and generate a DataFrame, `df`, with columns for date and opening price. The frame rate is 1 sample per day.

```
df = pd.read_excel('Ibex35.xlsx', sheet_name='Hoja1')
x = df['Ibex 35'].values
f = df['fecha'].values
plt.close('all')
plt.plot(x, label='input')
h = XXX #Modify this line
y = convolve(x, h)
plt.plot(y, label='output')
plt.legend(loc='best')
```

To smooth out short-term fluctuations and highlight longer-term trends or cycles, we are interested in performing a moving average which is simply the unweighted mean of the present value and previous $M-1$ data, i.e. $y[n] = \frac{1}{M}\sum_{k=0}^{M-1} x[n-k]$.

13. Write the impulse response of the linear invariant system that performs the moving average to the data. Is this a causal or non-causal system?

14. Compute the output through convolution for $M = 5$, $M = 119$, $M = 257$. Which value of M removes more the high frequencies (fast fluctuations) of the signal?

15. Are the variations in the ouput aligned in time with the variations in the data? (compare the position of the peaks of the input with the position of the peaks of the output)

16. The centered moving average computes the mean from an equal number of data on either side of a central value. Write the impulse response of the linear invariant system that performs the centered moving average. Is this a causal or non-causal system?

17. Numpy arrays start at index 0. As we cannot generate an array with negative indexes, explain how could we obtain the result of the centered moving average from `y = convolve(x, h)`

18. The function `convolve` allows for computing the convolution with different modes, namely `full`, `same`, and `valid`. You may type `convolve?` in the IPython console to learn the details. Which one do you think would be more useful in this case?

Correlation

Correlation is a useful operation to detect similarities between signals (either aligned in time or not) or patterns in a signal. Also autocorrelation (i.e. the correlation between a signal and itself) can be very useful to estimate periodicities.

In this section, we will consider correlation to automatically locate a particular pattern, $p[n]$, within a signal $x[n]$: the maximum of the correlation between $x[n]$ and $p[n]$ will indicate the location of the pattern. The correlation between $x[n]$ and $p[n]$ can be computed by flipping $p[n]$, i.e., $f[n] = p[-n]$ and convolving $f[n]$ (called matched filter) with $x[n]$.

The file signals.npy contains 10 signals. Each signal has length 1024 and contains a random pattern of -1 and 1 as well as one or several instances of the pattern $p[n] = \{\underline{1}, -1, -1, 1, 1, 1, -1, 1, -1, -1, 1, 1, 1\}$.

19. Determine the matched filter $f[n] = p[-n]$ to find the occurrences of the pattern $p[n]$ in the signal $x[n]$
20. Select one of the signals of the file signals.npy and detect the position of the first element of the pattern within the sequence. To find the position of the maximum in a sequence `y`, you may use `np.argmax(y)`

```
p = np.array([ 1, -1, -1,  1,  1,  1, -1,  1, -1, -1,  1,  1,  1])
s = np.load('signals.npy')
id = 0 #select one of the signals
x = s[id]
#Find the pattern
```

3 Lab work part 2: Convolution and correlation of 2D signals

Upload a file with your answers for this part (include figures and code).

Detecting edges in an image

In this example, we will work with another simple filter which performs a simple estimation of the derivative: $y[n] = x[n+1] - x[n-1]$. This operation is useful to detect changes, and, since the transformation is linear and invariant, can be computed as a convolution between the signal and a filter with a certain impulse response $h[n]$.

1. Which is the impulse response, $h[n]$, of the filter to obtain $y[n]$ as the convolution between $x[n]$ and $h[n]$?

We are going to apply previous filter to a 2D signal of size $N \times N$. Firstly, we will generate first the signal. Being this one a 2D signal, we will visualize it as an image with `plt.imshow`.

```
N=128
#We create the artificial 2D signals X1 and X2
X1 = np.zeros((N,N))
X2 = np.zeros((N,N))
```

```
X1[:, int(0.5*N):] = 1
X2[:int(0.5*N), 0:int(0.5*N)] = 1
X2[int(0.5*N):, int(0.5*N):] = 1
plt.imshow(X1, cmap='gray') #visualize X1 as an image
plt.imshow(X2, cmap='gray')
```

Write a code to compute the convolution of each row of matrix `X1` with $h[n]$. The result will be a new matrix `Y1` where each row is the result of the convolution of $h[n]$ and the corresponding row in `X1`. Store the result of each convolution in the row of a new matrix `Y1`. Visualize the result[1] with `plt.imshow(Y1)`. Do the same with `X2`.

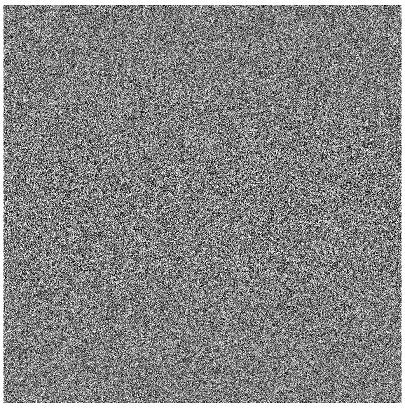
2. What do you observe when you visualize `Y1`?
3. What do you observe when you visualize `Y2`?
4. Explain how you would estimate all the edges of the image. Test your procedure with the previous artificial images and also with a real gray-level image.

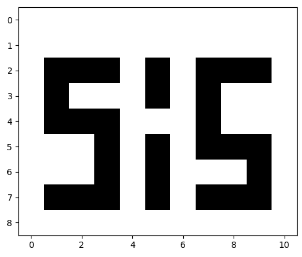
Pattern matching in images

We have seen previously that correlation can be used to automatically locate particular patterns within a 1D signal $x[n]$. The same approach can be extended to find patterns in images by generalizing the convolution operator to two dimensions:

$$y[m,n] = x[m,n] * f[m,n] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} x[k,l] f[m-k, n-l] \tag{2}$$

The files called find_sis1.png, find_sis2.png, and find_sis3.png of size 1024×1024 pixels contain a random pattern of white pixels (coded as 1) and black pixels (0) shown in figure 1a as well as a single instance of the image of figure 1b.


(a) Finding SIS



(b) SIS logo

Figure 1

5. Write a code to find automatically the SIS pattern in the images find_sis1.png, find_sis2.png and find_sis3.png. Explain your procedure and your results.

[1]When working with images it is usual to restrict the size of the output image to be equal to the size of the input image. You may delete yourself the unwanted pixels, or select the option `mode='same'` in the function `convolve`.