Biomedical Data Science Lab
Fall 2019

Introduction to digital signal processing in Python

Benjamín Béjar Haro
Assistant Research Professor
Department of Biomedical Engineering
320A Clark Hall, Johns Hopkins University
E-mail: bbejar@jhu.edu

1 Pre-Lab Description

In this pre-lab we will get familiar with the working environment (Google Colab) and we will acquire the basic Python programming skills, as well as the necessary digital signal processing background (see Section 2 of this document) to solve Lab I. This pre-lab assignment needs to be solved in the companion IPython Notebook.

Getting started. Go to the class website BMDS Lab and download the IPython notebook for this pre-lab assignment. The notebook is a file with an extension `.ipynb`. Go to Google Colaboratory and upload *(File → Upload notebook)* the downloaded IPython notebook. You will need to create a Google (gmail) account to access Colab. Follow the instructions in the notebook and on this manual to complete the assignment. Each exercise below is worth 10 points.

Exercise 1. Create a sequence (list) of numbers corresponding to sampled values of the function $\sin(2\pi t)$ over one period, and with a sampling frequency $f_s = 8$ Hz. In other words, create a sequence:

$$x_n = \sin(2\pi nT), \quad n = 0, 1, \ldots, \lfloor f_s \rfloor, \quad T = \frac{1}{f_s},$$

where $\lfloor x \rfloor$ denotes the largest integer smaller than x. Print the sequence of values. For evaluating math functions you can use the 'math' module of Python. You will need to import the corresponding module before invoking the $\sin(\cdot)$ function as shown in the example below:

```
import math
x = math.sin(math.pi/2)
print(x)
```

Exercise 2. Create two sequences x_n and y_n by sampling $\sin(2\pi t)$ and $\cos(4\pi t)$ with sampling rate $f_s = 64$ Hz, and store them in two separate lists. Write a function `add_sequence(x,y)` that returns the addition of the two lists. Plot the result of the addition in a graph adding the corresponding labels for the axes. Repeat the addition task before using vector array operations with the `numpy` module.

Exercise 3. Using the definition of the DTFT show the property that convolution in time is equivalent to multiplication in the frequency domain. In other words, show that for $y_n = x_n * h_n$, its DTFT $Y(e^{j\omega}) = X(e^{j\omega})H(e^{j\omega})$.

Exercise 4. Generate two box sequences as defined below:

$$x_n = \begin{cases} 1 & 0 \leq n \leq 10 \\ 0 & \text{else} \end{cases}, \quad h_n = \begin{cases} 1 & 0 \leq n \leq 20 \\ 0 & \text{else} \end{cases}, \quad n = 0, \ldots, 63.$$

Compute $y_n = x_n * h_n$ using `numpy.convolve` and plot (`plt.stem`) the resulting sequence y_n. Compute the DFT Y_k and plot its **magnitude** over the frequency range $[-\pi, \pi]$. (Hint. Use `numpy.fft.fftshift`). Label the horizontal and vertical axes.

Exercise 5. For this exercise, generate the sequence defined by the following finite difference equation:

$$x_n = \alpha x_{n-1} + \beta x_{n-2} + \epsilon_n, \quad x_0 = x_1 = 0, \quad n = 0, 1, \ldots, 127,$$

where the driving noise sequence ϵ_n consists of independent and identically distributed random numbers from a standard, normal distribution `numpy.random.randn`. For the coefficients, generate three random pairs of (α, β) where $\alpha, \beta \sim \mathcal{U}(-1, 1)$ (uniformly distributed between -1 and 1). Plot the three time-series on the same figure. Include a title, axis labels, and a legend that lists the generated (α, β) values of each time-series.

Exercise 6. In this exercise you will perform a basic denoising operation by filtering a noisy signal with a low-pass filter. You are provided with a function `get_noisy_signal()` that returns a signal corrupted with noise. Let a Gaussian filter g_n be defined as:

$$g_n = \frac{1}{K} e^{-(n/\sigma)^2}, \quad -10 \leq n \leq 10,$$

where K is a constant such that the weights of the filter add up to 1, and where σ is a parameter. Filter the signal for three different values of $\sigma = 1, 5, 10$. Observe the trade-off between denoising and sharpness of the signal. Compare the original signal with its denoised versions by plotting them in the same figure. Add labels and legends as appropriate. Why do we loose sharp transitions when filtering with a wider kernel?.

2 Background Notes on Digital Signal Processing

This notes are intended to provide a brief overview of the necessary signal processing tools for successfully completing this lab experience. For a more detailed and thorough treatment of the subject we refer the interested reader to the excellent book [1].

In these notes, we start by introducing discrete-time signals and their representation using the delta sequence δ_n. We then introduce linear operations on sequences and their Fourier domain characterization.

2.1 Sequences

A sequence $x \in \mathbb{C}^{\mathbb{Z}}$ is an ordered list of (complex) numbers x_n that take values on the integers $\mathbb{Z} = \{\ldots, -1, 0, 1, 2, \ldots\}$. The *delta* sequence is defined as:

$$\delta_n = \begin{cases} 1 & n = 0 \\ 0 & n \neq 0 \end{cases}, \quad n \in \mathbb{Z}. \tag{1}$$

Basic operations. The basic operations on sequences are scaling, shifting by some integer, and addition. For sequences $x = \{x_n\}_{n\in\mathbb{Z}}$ and $y = \{y_n\}_{n\in\mathbb{Z}}$ these operations are defined, respectively as:

$$\alpha x = \{\alpha x_n\}_{n\in\mathbb{Z}}, \quad \alpha \in \mathbb{C} \tag{2}$$

$$S_k x = \{x_{n-k}\}_{n\in\mathbb{Z}}, \quad k \in \mathbb{Z} \tag{3}$$

$$x + y = \{x_n + y_n\}_{n\in\mathbb{Z}}. \tag{4}$$

For notational convenience we will often refer to a sequence as x_n where we make explicit the dependence on the "time" index n. Likewise, we will often use x_{n-k} as a shorthand for $S_k\, x$.

Representation in terms of the delta sequence. Note that due to the special form of the delta sequence we can use it to express any sequence as a linear combination of scaled and shifted delta sequences since:

$$x = \ldots + x_{-2}S_{-2}\,\delta_n + x_{-1}S_{-1}\,\delta_n + x_0\delta_n + x_1S_1\,\delta_n + \ldots = \sum_{k\in\mathbb{Z}} x_k\,\delta_{n-k} \tag{5}$$

2.2 Filtering

Now suppose we want to perform some operations on our sequence and, for that purpose, we pass our input sequence x through a system H and obtain an output sequence y as a result. We denote this operation as $y = Hx$. Now, let us put some restrictions on the type of systems that we consider. In particular, we will be interested in *linear and shift-invariant* systems. Linearity implies that if the

input is a linear combination of two sequences then the output is the same linear combination of the outputs to the individual sequences:

$$\textit{Linearity:} \qquad H(\alpha x + \beta z) = \alpha H x + \beta H z. \tag{6}$$

The system being shift-invariant means that the shifting and system operations commute or, in other words, if a sequence x produces an output y then if we input a shifted version of x then this will translate into the same shifting of the original output:

$$\textit{Shift-invariant:} \qquad Hx = y \iff H\big(S_k\, x\big) = S_k\big(Hx\big) = \{y_{n-k}\}_{n\in\mathbb{Z}}. \tag{7}$$

Impulse response. Now recall the representation of a sequence x as a sum of shifted and scaled delta sequences (5). Let us now use that representation and pass it through the linear and shift-invariant system H:

$$\begin{aligned} y = Hx &= H\big(\sum_{k\in\mathbb{Z}} x_k S_k \delta_n\big) \\ &\overset{(a)}{=} \sum_{k\in\mathbb{Z}} x_k H\big(S_k\delta_n\big) \\ &\overset{(b)}{=} \sum_{k\in\mathbb{Z}} x_k S_k \underbrace{H\big(\delta_n\big)}_{h}, \end{aligned} \tag{8}$$

where (a) follows from linearity and (b) from the shift-invariant property of the system.

Convolution. Now, if we denote the sequence $h = H\delta_n = \{h_n\}_{n\in\mathbb{Z}}$ as the *impulse response* of the system H, then it follows from (8) that the output of the system $y = h * x = \{y_n\}_{n\in\mathbb{Z}}$ is a sequence whose entries are given by the convolution equation:

$$y_n = (h * x)_n = \sum_{k\in\mathbb{Z}} x_k\, h_{n-k} = \sum_{k\in\mathbb{Z}} h_k\, x_{n-k}\,. \tag{9}$$

2.3 Fourier Transform

In this section we will introduce the Fourier transform for sequences and we will illustrate its use in filtering.

Eigenfunctions and eigenvalues. Before that, let us continue our discussion about linear and shift-invariant systems by introducing the concept of *eigenfunction* of a system. In our case, and since our system deals with sequences can refer to

them as *eigensequences*. In general, an eigenfunction u of a system H is an input such that the output of the system is again the same input up to scaling:

$$Hu = \lambda u, \tag{10}$$

where λ is called an *eigenvalue* associated to the eigenfunction u. Now consider a complex exponential sequence of the form $x_n = e^{j\omega n}$ and pass it through a linear shift-invariant system with impulse response h. Then we have:

$$y_n = \sum_{k\in\mathbb{Z}} h_k x_{n-k} = \sum_{k\in\mathbb{Z}} h_k e^{-j\omega(n-k)} = e^{j\omega n} \sum_{k\in\mathbb{Z}} h_k e^{-j\omega k} = H(e^{j\omega}) x_n. \tag{11}$$

From (11) we see that complex exponentials (actually all exponential sequences) are eigenfunctions of linear shift-invariant systems. In particular, $x_n = e^{j\omega n}$ has an associated eigenvalue given by $\lambda = H(e^{j\omega}) = \sum_{k\in\mathbb{Z}} h_k e^{-j\omega k}$ which is called the *transfer function* of the system h at frequency ω. Note that for the transfer function to be meaningful we need that the above sum converges. A sufficient condition for $H(e^{j\omega})$ to exist is that the sequence h is absolute summable (i.e. $\sum_n |h_n| < \infty$). Note also that for finite sequences that condition is trivially satisfied.

Discrete-Time Fourier Transform. The discrete-time Fourier transform (DTFT) of a sequence is defined as:

$$\mathcal{F}x = X(e^{j\omega}) = \sum_{n\in\mathbb{Z}} x_n \, e^{-j\omega n}, \quad \omega \in \mathbb{R}, \tag{12}$$

where the notation $X(e^{j\omega})$ is used to emphasize the 2π–periodicity of the DTFT. The *inverse* DTFT of a 2π–periodic function $X(e^{j\omega})$ is given by

$$x_n = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega}) \, e^{j\omega n} \, d\omega, \quad n \in \mathbb{Z}. \tag{13}$$

Filtering in Fourier domain. The Fourier transform gives us an alternative representation of the signal that is sometimes more convenient to work with. For instance, a filtering operation (e.g. convolution) can also be characterized in the frequency domain. Let $y = h * x$ be a sequence that is the output of a linear shift-invariant filter h. Then, in the frequency domain we have that:

$$\mathcal{F}y = Y(e^{j\omega}) = H(e^{j\omega}) X(e^{j\omega}), \tag{14}$$

which means that convolution in time is equivalent to multiplication in frequency.

2.4 DSP with two-dimensional sequences (images)

All of the above definitions naturally extend to the case of multi-dimensional sequences such as images. We will state the definitions for two-dimensional sequences since in practice we will be mostly dealing with images. Our sequences now will be indexed by two elements (i.e. $x = \{x_{mn}\}_{m,n\in\mathbb{Z}}$).

Convolutions in $2D$. Analogous to the definition for $1D$ sequences, and assuming linear and shift-invariant systems, we can define filtering of two-dimensional sequences as:

$$y_{mn} = \big(x * h\big)_{mn} = \sum_{m\in\mathbb{Z}} \sum_{n\in\mathbb{Z}} h_{k\ell}\, x_{m-k,n-\ell} = \sum_{m\in\mathbb{Z}} \sum_{n\in\mathbb{Z}} x_{k\ell}\, h_{m-k,n-\ell}, \tag{15}$$

where x is an input sequence and h is a two-dimensional filter (sequence).

Fourier Transform. Analogous to (12) and (17) we define the Fourier Transform of two-dimensional sequences as:

$$\mathcal{F}x = X(e^{j\boldsymbol{\omega}}) = \sum_{m,n\in\mathbb{Z}} x_{mn}\, e^{-j(\omega_1 m+\omega_2 n)}, \quad \boldsymbol{\omega} = \begin{bmatrix}\omega_1\\ \omega_2\end{bmatrix} \in \mathbb{R}^2, \tag{16}$$

where $\boldsymbol{\omega}$ is now a two-dimensional vector of frequencies.

The inverse Fourier Transform is then given by

$$x_{mn} = \frac{1}{(2\pi)^2} \int_{-\pi}^{\pi}\int_{-\pi}^{\pi} X(e^{j\boldsymbol{\omega}})\, e^{j(\omega_1 m+\omega_2 n)}\, d\omega_1\, d\omega_2, \quad m,n \in \mathbb{Z}. \tag{17}$$

References

[1] M. Vetterli, J. Kovačević, and V. K. Goyal. *Foundations of Signal Processing*. Cambridge University Press, 2014.