hugomcp@di.ubi.pt, 2024/25

Ficha Prática 3

Signals and Convolution

1. Discrete-Time Convolution Implementation

Convolution in discrete time is defined as:

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

This operation represents how a system with impulse response h[n]processes an input signal x[n]. The length of the output signal is given by:

$$L=M+N-1,$$

where M and N are the lengths of x[n] and h[n], respectively.

a) Implement a Python function that manually compute the convolution for each value of n.

b) Use NumPy's np.convolve(x, h, mode="X"), with X ∈ {'valid', 'full', 'same'} and compare the results.

c) Plot the original signals and their convolution result using Matplotlib.

2. Edge Detection with Convolution

Edge detection relies on high-pass filters that emphasize intensity changes in images. For instance, the Sobel filter detects vertical or horizontal edges by computing intensity gradients. The operation applied to an image is 2D convolution:

$$I_{\text{out}}(x,y) = \sum_i \sum_j I_{\text{in}}(x-i, y-j)h(i,j)$$

a) Implement a Python function that perform convolution between an image and a 2D filter
b) Use OpenCV (cv2.filter2D) or SciPy (convolve2d) for efficient convolution.

c) Convert the image to grayscale before applying the filter and visualize the filtered image with Matplotlib.

3. System Impulse Response Identification

The impulse response h[n]characterizes a linear time-invariant (LTI) system. If y[n]=x[n]∗h[n], then knowing y[n] for an impulse input x[n]=δ[n] directly gives h[n].

Convolution allows us to predict future outputs given any input.

a) Extract h[n] directly from the system's response to an impulse.

b) Compute the output for a new x[n] by convolving it with h[n].

c) Use NumPy’s convolve() to validate results.

4. Audio Filtering with FIR Filter

A low-pass filter removes high-frequency noise by averaging neighboring samples. A moving average filter is a simple FIR filter:

$$y[n] = \frac{1}{N}\sum_{k=0}^{N-1} x[n-k]$$

where N is the filter length.

a) Use SciPy’s convolve() for filtering.

b) Use scipy.io.wavfile.read/write to process .wav files.

5. Real-Time Signal Processing Simulation

Streaming processing is crucial for real-time DSP applications like audio and communication systems. A buffer-based approach processes data in small chunks rather than all at once. An exponential moving average (EMA) filter is useful for real-time smoothing:

$$y[n]=\alpha\, x[n]+(1-\alpha)\, y[n-1]$$

a) Use a loop-based approach to process signal chunks.

b) Adjust alpha to control smoothing strength.