Holon Institute of Technology, Department of Applied Mathematics

Final Project in Digital Signal Processing – 2023

Dr. Yochay Jerby

Submission Policy

The project should be submitted by 12.2.2023. The project can be submitted in pairs. For each exercise, fully justified answers are required. When a Python function is requested, you must both write the code in your theoretical answer and also provide a Python file ready to run.

I. Echo generation and cancelation

Given a signal $x(n)$ and $\alpha \in (0,1)$, one can simulate an echo by creating the following signal

$$y(n) = x(n) + \alpha \cdot x(n-d),$$

where $d \in \mathbb{N}$ is the delay factor.

1. Explain in your own words why this is an echo filter.
2. Write the impulse response of the filter, $h(n)$, that transforms $x(n)$ to $y(n)$. Write its Z-transorm and compute its Fourier transform. What kind of filter is it?
3. Write the inverse filter that transforms $y(n)$ to $x(n)$ in the z-plane. Compute the poles. Is it stable? Does it have a Fourier transform? If yes, compute it, what kind of filter is it?
4. Let us define the auto-correlation

$$\phi_{xx}(k) = \sum_{n=-\infty}^{\infty} x(n)x(n-k)$$

compute the relationship between ϕ_{xx} and ϕ_{yy}.

5. Explain why ϕ_{yy} has a strong local maximum at $0, d, -d$.
6. Design an algorithm for finding a, d from $y(n)$.
7. Implement your algorithm in Python and test it on your own recorded voice for $a = 0.5, d = 3$.

II. Adaptive filters

In applications filters might need to adapt their defining parameters based on real-time needs, such filters are called adaptive filters. Adaptive filtering is necessary in many cases, like adaptive antenna systems, digital communication receivers, adaptive noise canceling techniques, systems modeling...

In the following exercise we will implement a simple adaptive filter via least mean square method.

Least mean square for coefficient adjustment

Suppose that we have a FIR-filter with adaptable coefficients

$$\{h(k) | 0 \leq k \leq N-1\}$$

Denote by $x(n)$ and $y(n)$ the input and output signals of the filter, respectively. Suppose we have a desired output signal $d(n)$ which we can compare to the output $y(n)$. Define

$$e(n) = y(n) - d(n)$$

and set

$$\Lambda = \sum_{n=0}^{M} e(n)^2$$

where M is the common length of $y(n)$ and $d(n)$.

1. Explain in your own words why Λ can be considered as the distance or error between the required output $d(n)$ and the resulting output $y(n)$.
2. Prove the following equality

$$\Lambda = \sum_{n=0}^{M} d(n)^2 - 2\sum_{k=0}^{N-1} h(k) r_{dx}(k) + \sum_{k=0}^{N-1}\sum_{l=0}^{N-1} h(k)h(l) r_{xx}(k-l)$$

   where:

$$r_{dx}(k) = \sum_{n=0}^{M} d(n)x(n-k)$$

   and

$$r_{xx}(k) = \sum_{n=0}^{M} x(n)x(n+k).$$

   We refer to $r_{dx}(k)$ as the cross correlation between $d(n)$ and $x(n)$ and to $r_{xx}(k)$ as the autocorrelation of $x(n)$. We implicitly assume in the definition that $x(n) = 0$ for $0 \le n \le M$.
3. Prove that the filter is optimal if the following equation holds

$$\sum_{k=0}^{N-1} h(k) r_{xx}(k-m) = 2 \cdot r_{dx}(m)$$

   for $0 \le m \le N-1$.
4. Theoretically, in order to find the optimal filter, we can solve the previous equations. However, we would rather proceed by an alternative method without explicitly computing the correlation sequences $r_{xx}(k)$ and $r_{dx}(k)$.

Let us start with a non-optimal filter whose impulse response is $h_0(k)$ with $0 \leq k \leq N-1$. After each new input sample $x(n)$ enters the adaptive FIR filter, compute the corresponding output, say $y(n)$, compute the error $e(n) = d(n) - y(n)$ and update the filter coefficient according to the equation

$$h_s(k) = h_{s-1}(k) + \Delta \cdot e(n) \cdot x(n-k)$$

where $0 \leq k \leq N-1$ and $s = 1,2, \ldots$ where Δ is called the step size parameter. To ensure stability, Δ must be chosen

$$0 < \Delta < \frac{1}{10NP_x}$$

where N is the length of the adaptive FIR filter and P_x is the power in the input signal, which can be approximated by

$$P_x \approx \frac{1}{M+1}\sum_{n=0}^{M} x(n)^2 = \frac{r_{xx}(0)}{M+1}.$$

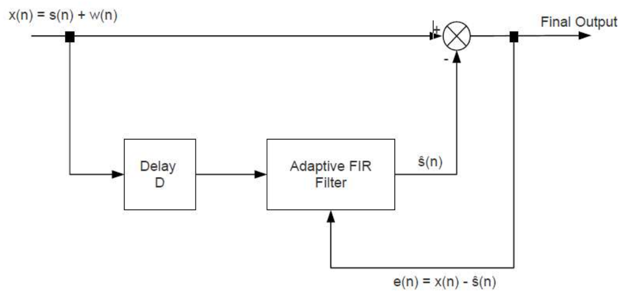
Implement the adaptive LMS-algorithm described above in Python by writing a function of the following general form and parameters:

```
function [h,y] = lms(x,d,delta,N)
% LMS Algorithm for Coefficient Adjustment
% [h,y] = lms(x,d,delta,N)
% h = estimated FIR filter
% y = output array
% x = input array
% d = desired array, length must be same as x.
% delta = step size
% N = length of the FIR filter
```

Noise cancelation

5. We shall now use the LMS algorithm for canceling an additive noise from a signal. Denote by $s(n)$ a signal and $w(n)$ a noise. The input

signal is $x(n) = s(n) + w(n)$. We will want to build a filter that restores only $s(n)$ from $x(n)$ according to the following configuration



The delay is chosen sufficiently large so that the signal noise $w(n)$ and $w(n-D)$ are uncorrelated. The output of the adaptive FIR filter is the estimate

$$\hat{s}(n) = \sum_{k=0}^{N-1} h(k)x(n-k-D)$$

The error signal that is used in optimizing the FIR filter coefficients is

$$e(n) = x(n) - \hat{s}(n).$$

Due to the delay D, the LMS algorithm for adjusting the coefficients recursively becomes

$$h_s(k) = h_{s-1}(k) + \Delta \cdot e(n) \cdot x(n-k-D)$$

for $0 \leq k \leq N-1, s = 1,2 \ldots$. Write in Python a function *lmsdelay,* which implements this modified LMS algorithm. Test your function on the following input:

$s(n) = \sin\left(\frac{\pi}{5n}\right)$

$w(n)$ a Gaussian noise with zero mean and standard deviation equals to 0.2 (use the function np.random.normal),

$N = 11, M = 1000$

Does the computed FIR filter have a linear phase? Numerically draw the DFT of its impulse response with resolution of 200 samples. Is it a low-pass filter, high-pass filter,…? Compute the zeros of its Z-transform.