Lab 5: FIR filters

Grading

This Lab consists of three exercises. Once you have submitted your code in Matlab Grader AND once the deadline has past, your code will be checked for correctness. Note here, that upon submission, your code is already subjected to some basic checks that are aimed to verify whether your code will compile; these basics checks don't say anything about the correctness of your submission. You can visit Matlab Grader again after the deadline (give the servers some time to do all the assessments; this might even take a few days) to see how well you did. In case Matlab Grader indicates you failed an exercise, this does not automatically imply that you failed the entire exercise. Each exercise is subjected to n tests, where the number of tests can vary between exercises. In case Matlab Grader indicates you failed the exercise, this means that not all tests were passed (e.g. in an exercise with 7 tests, you could have passed 6 and Matlab Grader will indicate you failed the exercise). Your grade is calculated based on the number of tests you passed and not on the number of exercises you passed.

1 Introduction

In this Lab you will perform digital signal processing by means of finite impulse response (FIR) filtering. In the last lab you have seen that any signal can be sampled with a certain sampling frequency and that this can yield some (un)wanted results (aliasing). The signals you use as input for these FIR filters have already been converted by a continuous-time to discrete-time (C-to-D) converter (x(t) $\rightarrow$ x[n]). The goal of this lab is to learn how to apply the convolution-sum and experience the effect FIR filters can have on digital signals. In Lab 6, you will understand why FIR filters affect the signals in specific ways.

All exercises must be submitted via Matlab Grader.

2 Filtering a signal

Assume a signal processing system containing the following elements:

The system in figure 1 has a filter that is described in figure 2

Every sampled signal consists of a collection of samples. These samples can be individually described as delta-pulses. The systems response to a delta-

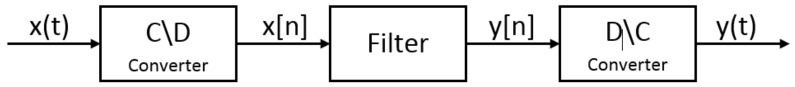

Figure 1: Signal Processing System

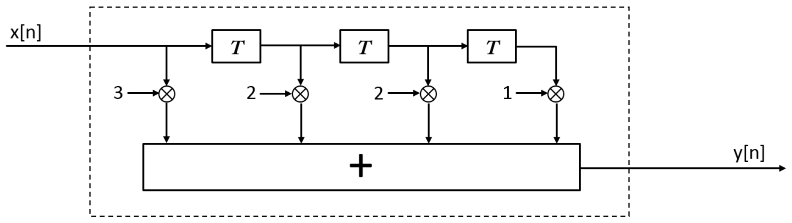

Figure 2: FIR Filter

pulse (impulse response) can be determined by having a look at the filter and is denoted in the table below as $h[n]$. As can be seen, the impulse response $h[n]$ equals zero after a couple of samples. Such a filter has a Finite length Impulse Response and is abbiviated by FIR filter. Consider the discrete-time signal given by the following finite sequence of samples $x[n] = \delta[n] + 2\delta[n-1] - \delta[n-3]$. This discrete-time signal is shown in figure 3.

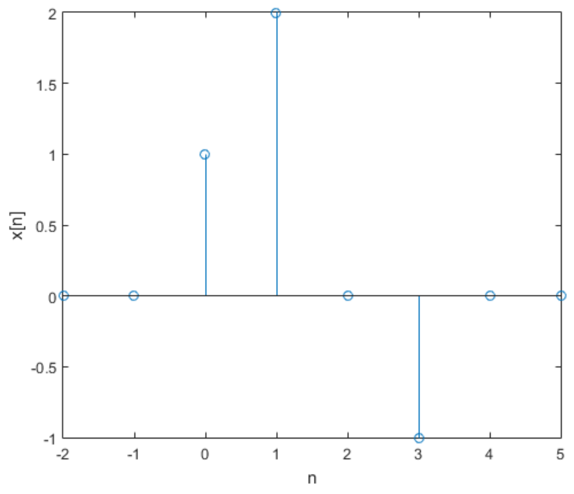

Figure 3: The signal x[n]

The $x[n]$ to $y[n]$ conversion is illustrated below. The starting point of this conversion is the impulse response $h[n]$ and the input signal $x[n]$. Knowing $x[n]$ and $h[n]$, $y[n]$ can be calculated using the convolution-sum, as discussed during the lectures:

$$y[n] = \sum_{k=0}^{N-1} h[k]x[n-k] = h[n] \star x[n]. \tag{1}$$

Here, N is the filter length (the sample duration of the impulse response, in this case $N = 4$).

Calculating the convolution-sum for the example above, yields:

$$\begin{aligned}
y[0] &= x[0] \cdot h[0] \\
y[1] &= x[0] \cdot h[1] + x[1] \cdot h[0] \\
y[2] &= x[0] \cdot h[2] + x[1] \cdot h[1] + x[2] \cdot h[0] \\
y[3] &= x[0] \cdot h[3] + x[1] \cdot h[2] + x[2] \cdot h[1] + x[3] \cdot h[0] \\
y[4] &= x[1] \cdot h[3] + x[2] \cdot h[2] + x[3] \cdot h[1] + x[4] \cdot h[0] \\
y[5] &= \ldots
\end{aligned}$$

Filling in the appropriate values for the exemplified filter $h[n]$ and signal $x[n]$ now gives:

n	-2	-1	0	1	2	3	4	5	6	7	8
$\delta[n]$	0	0	1	0	0	0	0	0	0	0	0
$2\delta[n-1]$	0	0	0	2	0	0	0	0	0	0	0
$-\delta[n-3]$	0	0	0	0	0	-1	0	0	0	0	0
$x[n]$	0	0	1	2	0	-1	0	0	0	0	0
$h[n]$	0	0	3	2	2	1	0	0	0	0	0
$h[0] \cdot x[n]$	0	0	3	6	0	-2	0	0	0	0	0
$h[1] \cdot x[n-1]$	0	0	0	2	4	0	-2	0	0	0	0
$h[2] \cdot x[n-2]$	0	0	0	0	2	4	0	-2	0	0	0
$h[3] \cdot x[n-3]$	0	0	0	0	0	1	2	0	-1	0	0
$y[n]$	0	0	3	8	6	2	0	-2	-1	0	0

Exercise 1 [5 tests]

For this exercise you will make use of unit step functions u[n], defined below, to create a block signal as input to a FIR filter.

$$u[n] = \begin{cases} 0 & \text{for} \quad n < 0 \\ 1 & \text{for} \quad n \geq 0 \end{cases}$$

MATLAB does not offer a unit step function. It does offer a Heaviside step function, but this function does not fulfill the specifications. Have a look at the internet to see options for how to plot a unit step function. The use of a

conditional statement in the function might be required. *Remember: true equals 1 and false equals 0!*
By making use of two unit step functions, one of which is time-shifted and negative with respect to the other, a block signal can be constructed. In figure 4 two unit steps are plotted. When added, the resulting signal would be a block-shaped function.

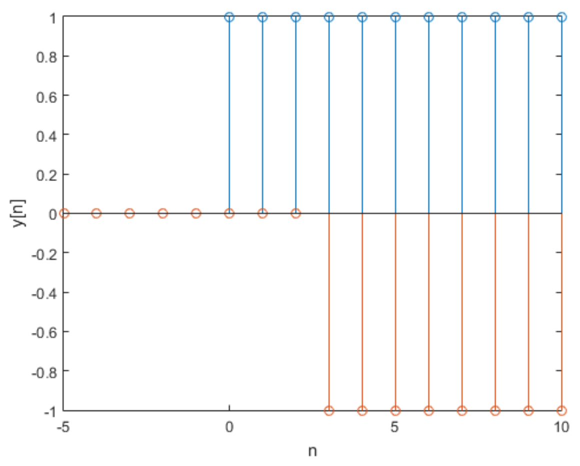

Figure 4: Two individual unit step functions

This block signal can be adapted into a 'decaying' block signal:

$$x[n] = A\left(u[n] - u[n-L]\right) \cdot a^n \tag{2}$$

Create a function that generates a decaying block signal using the following input variables:

1. **Amplitude** - Block Amplitude A
2. **Decay** - Decay parameter a
3. **Length** - Block sample length L
4. **Offset** - Block starting point offset

The function should make a stem plot of this decaying block on the time span $n \in [-10, 20]$. The first sample of the block should have the normal (non-decayed) block amplitude, so keep in mind to implement this offset in the power of the decay parameter. Also give the axes names. The x-axis should have the label n, the y-axis $y[n]$.

An example output has been given in figure 5

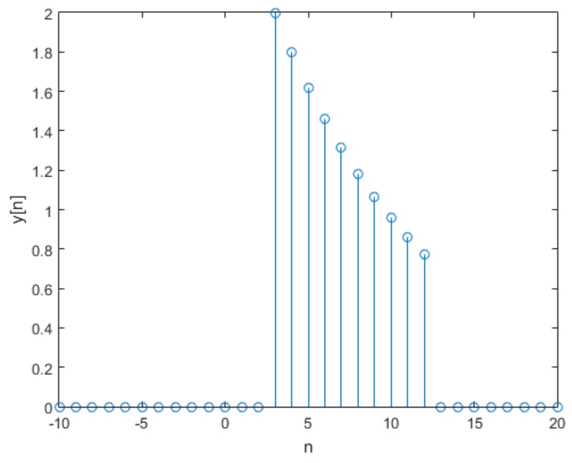

Figure 5: Example output for: Amplitude = 2, Decay = 0.9, Length = 10, Offset = 3

3 FIR filters

The convolution sum can be used to describe the output of any system with a finite impulse response. The FIR filter can be generalized as shown in Figure 6.

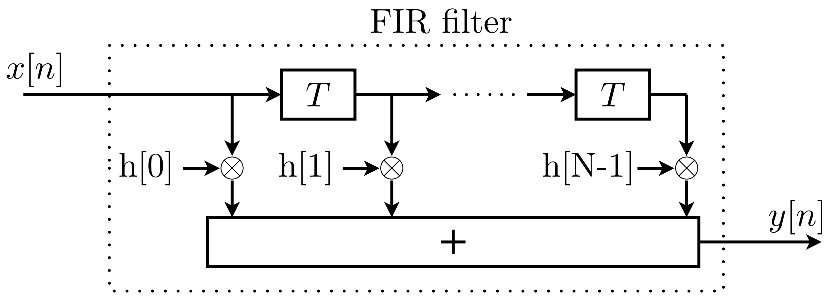

Figure 6: FIR filter

Exercise 2 [5 tests]

For this exercise you will have to import data. In the oase folder there is a file called '`lab5data.mat`'. Download this file and select the `Import Data` button under the `Home` tab in MATLAB. Select the file and import the data. If done correctly, a variable called $x1$ should appear in your workspace. This is the signal $x[n]$ you will be working with. The filter used to manipulate the input signal will be a N-point averaging filter. The impulse response of a 3-point averaging filter can be described as:

$$h[n] = \frac{1}{3}\delta[n] + \frac{1}{3}\delta[n-1] + \frac{1}{3}\delta[n-2] \tag{3}$$

An N-point averaging filter can be described with the following impulse response:

$$h[n] = \frac{1}{N}\sum_{k=0}^{N-1}\delta[n-k] \tag{4}$$

For this exercise you will run the imported data through a N-point averaging filter. Do not overwrite the value(s) of x1. You will make 3 plots underneath each other in one figure. The first plot should contain the original data on the time span $n \in [0, 99]$. The second plot should contain the plot of the original data *and* a plot of the original data convoluted with a 2-point averaging filter. The third plot should contain a plot of the original data *and* a plot of the original data convoluted with a 5-point averaging filter.
Since the impulse response is known, you can use the MATLAB function `conv` to calculate the output signal. If done correctly, you should see a change in the vector length of the output with respect to the original data. Define your axes in such a way that the output can be plotted on the appropriate time span, meaning that the entire output of the filter is exactly (i.e. nothing more and nothing less) visible in your plot. **The original signal has to be plotted first, followed by the output signal.** Use the normal plot function.

4 Cascading multiple FIR filters

In the real world, sampled signals can get distorted by some external interference. Assume the following system, where interference causes a certain distortion that is similar to a FIR filter with the system response as described in equation 5.

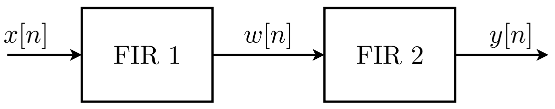

Figure 7: Cascade of two FIR filters

$$w[n] = x[n] + q \cdot x[n-1] \tag{5}$$

$$y[n] = \sum_{l=0}^{M} r^l w[n-l] \tag{6}$$

To negate this unwanted distortion, a restoration filter is added. This restoration filter is used to undo the effects of the first FIR filter. The second FIR filter performs restoration, but it only does this approximately. The impulse response of the second FIR filter is displayed in equation 6.

Exercise 3 [5 tests]

Create a script that makes two plots underneath each other in one figure. Both plots containing the input signal: `xn = 256*(rem(0:100, 50)<10)`. In the first plot, the signal $w[n]$ should also be added. Now, in the second plot, the signal $y[n]$ should be added instead of the signal $w[n]$. Since the length of the vectors of the signals will not be equal after convolution, you need to define some new time spans. Make sure the x-axis only shows the time span $n \in [0, 75]$ using `xlim`. Use the normal plot function! Also create a variable `ErrorMax` that saves the value of the biggest (absolute) error between $x[n]$ and $y[n]$ on the domain $n \in [0, 75]$. **The original signal has to be plotted first, followed by the output signal.** The values of the variables are given below:

$\mathbf{q} = -0.9$

$\mathbf{r} = 0.9$

$\mathbf{M} = 22$

your answer should look like the figure below (excluding axis-labels and legends).

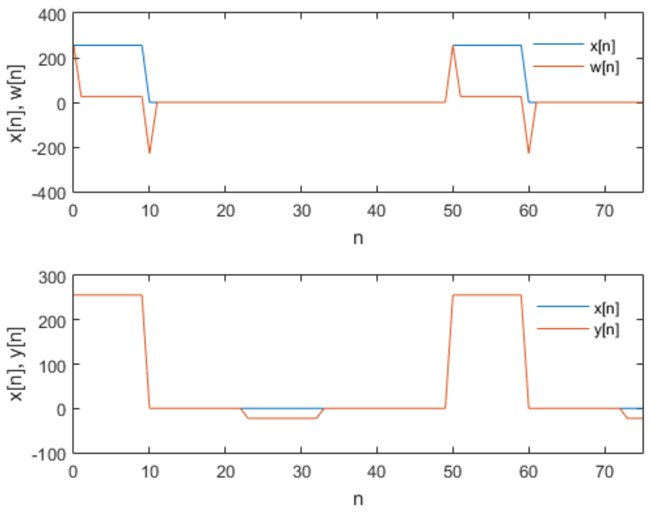

Figure 8: Answer of exercise 3