THE CITY COLLEGE OF NEW YORK
Department of Electrical Engineering
EE425 Computer Engineering Laboratory - Spring 2020

Exp. 6: Discrete-time Series Averaging Filters -Part II

<u>Objective</u>: Implementation of moving average filters with application to discrete-time series.

Consider the moving average filter given by:

$$y[n] = \frac{x[n] + x[n-1] + x[n-2] + x[n-3]}{4}$$

In Figure 1 we show the output time series **yi[n]**, for **i=1,...,4**, produced by this filter when it is convolved with each of the five different periodic times series inputs **xi[n]**, for **i=1,...,4**. In this figure, the inputs **xi[n]** are shown in blue, while the outputs **yi[n]** are shown in red, and it should be evident to you, by inspection, why we call such a filter a moving average filter: note how in Figure 1 all the outputs appear "smoother" than the inputs, in some sense. Indeed, averaging can be interpreted as a smoothing process.

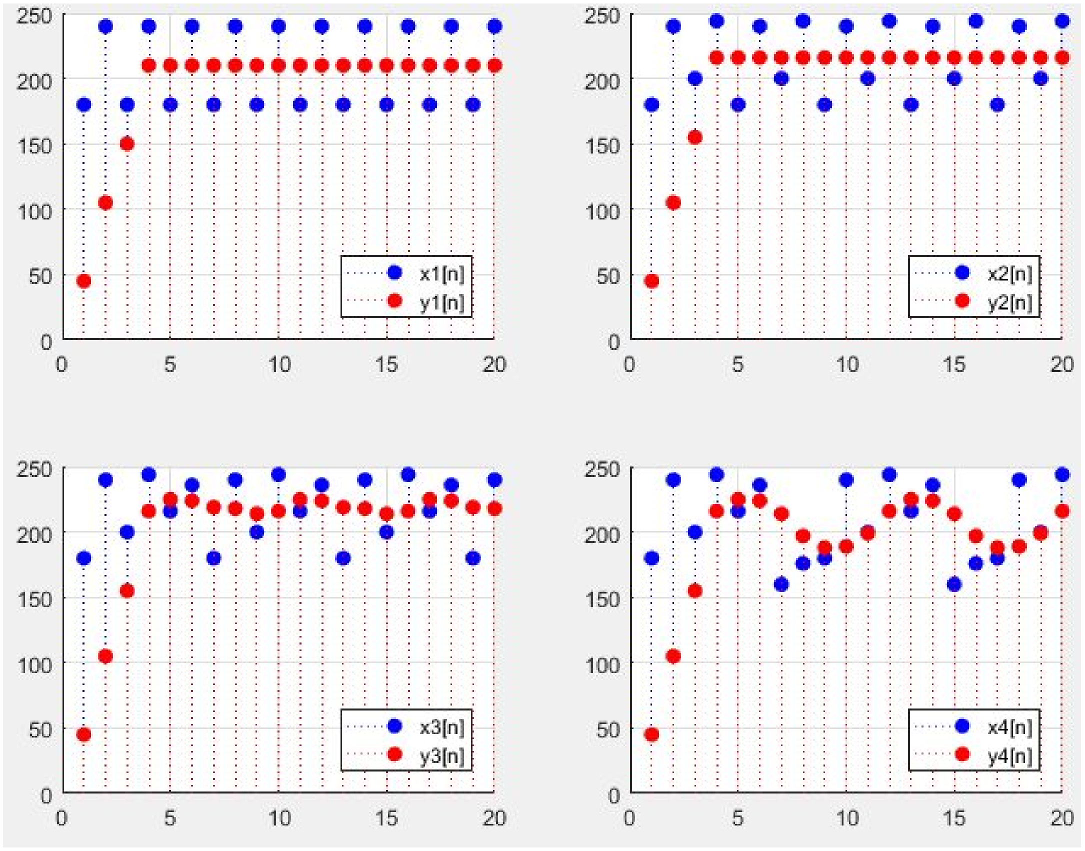
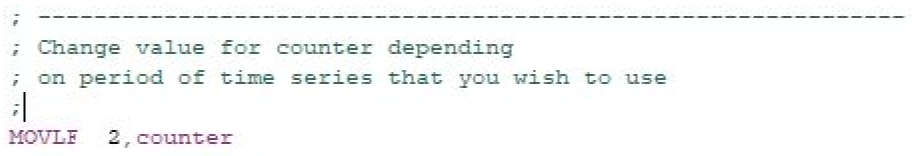


Figure 1. Inputs **xi[n]** and outputs **yi[n]**, for **i=1,...,4.**

IMPORTANT NOTE 1: Please note that the outputs **yi[n]**, for **i=1,...,4,** in Figure 1 are showing both the transient phase and the steady-state phase.

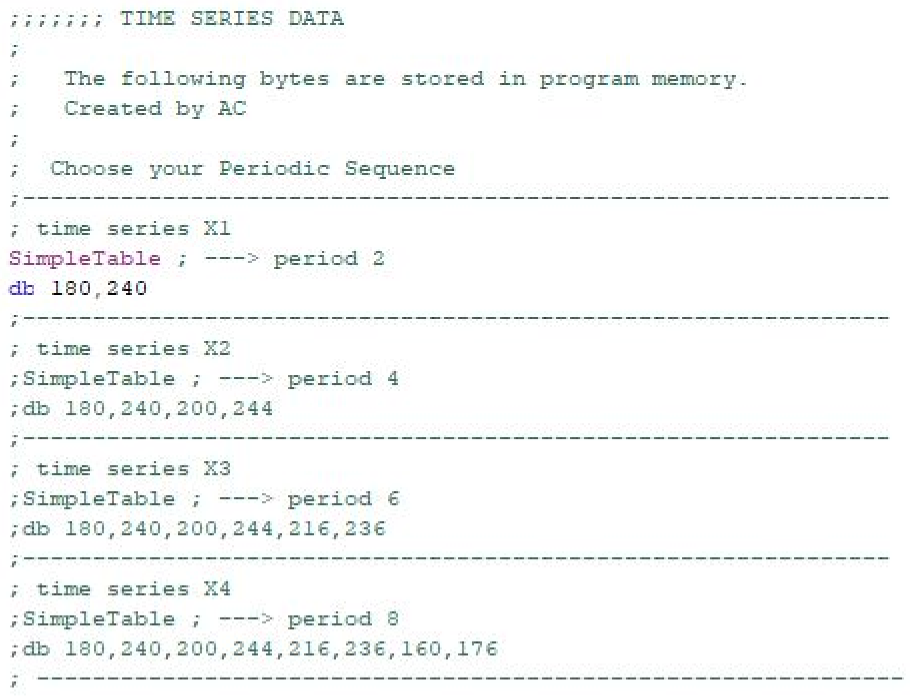
Task 1

1. The first thing for you to do is to compile and simulate the given *.asm* template titled "**template_for_moving_average_by_AC_Part_II.**" While you do that, please note the following:
    a. The variable of interest in this template is, again, the register called *value*.
    b. Simulate the template as is (i.e., without making any changes) and note that the contents of *value* are exactly the values for time series **x1[n]** shown in Figure 1.
2. Go back to the template and locate the sections of code shown in Figures 2 and 3.



```
; ---------------------------------------------------------------
; Change value for counter depending
; on period of time series that you wish to use
;
MOVLF  2,counter
```

Figure 2. *counter* variable, where the literal **2** is the period of time series **x1[n]**.



```
;;;;;;; TIME SERIES DATA
;
;   The following bytes are stored in program memory.
;   Created by AC
;
;  Choose your Periodic Sequence
;--------------------------------------------------------------
; time series X1
SimpleTable ; ---> period 2
db 180,240
;--------------------------------------------------------------
; time series X2
;SimpleTable ; ---> period 4
;db 180,240,200,244
;--------------------------------------------------------------
; time series X3
;SimpleTable ; ---> period 6
;db 180,240,200,244,216,236
;--------------------------------------------------------------
; time series X4
;SimpleTable ; ---> period 8
;db 180,240,200,244,216,236,160,176
; --------------------------------------------------------------
```

Figure 3. Periodic time series of different periods.

3. Consider Figures 2 and 3, and perform the following changes in the template:
    a. Corresponding to Figure 2, replace the literal **2** with the new literal **4**. This new literal **4** is simply the period of the second time series **x2[n],** of period **4,** shown in the code in Figure 3.
    b. Corresponding to Figure 3, <u>comment out</u> the following lines of code, which correspond to time series **x1[n]**:
        i. SimpleTable ; ---> period 2,
        ii. db 80,240
    c. Corresponding to Figure 3, <u>uncomment</u> the following lines of code, which correspond to time series **x2[n]**:
        i. ;SimpleTable ; ---> period 4
        ii. ;db 80,240,160,60
4. Simulate the code again, with the changes made in step 3, and note how the contents of register *value* change according to the time series **x2[n],** also shown in Figure 1.
5. Repeat step 3 above for each of the remaining time series in Figure 3 and note that the contents of *value* will vary according to the time series that you choose. All the available time series in the template are shown in the time domain in Figure 1 as **xi[n]**, for **i=1,...,4,** and shown in code in Figure 3. Each time series has a different period, please verify this through simulations before moving on to the next task.

<u>Task 2</u>

1. Having completed Task 1, please write the .asm code to implement the moving average filter **y[n]** specified above.
    a. Convolve your implemented filter with <u>each</u> of the time series **xi[n]**, for **i=1,...,4,** in the template, one at a time, of course.
        i. Note that this Task is an extension of your work for Lab 5.
        ii. You must implement a linear memory buffer exactly like the one that you wrote for Lab 5.
        iii. Just like for the filter in Lab 5, you should also <u>add all the values first</u>, and then "divide" the entire sum after (in this case it will be "division" by 4). Note that now you will be adding four (4) variables and not just two (2) like in Lab 5. Consequently, your adder/divider from Lab 5 will have to be extended to account for this. Recall that addition can only be performed with two registers at a time. This means that you may want to create additional variables to help you keep track of the full sum, <u>which will necessarily span two registers</u>. There are many ways to perform this addition. I suggest that you draw out the process on a piece of paper, before writing any code, to help you get your ideas straight and so that you can get a clear idea of what you are trying to do here. Note that this adder/divider is not a difficult thing to do, but you really have to be clear on what is going on when you are trying to add four variables (two at a time), considering that each of these variables is 8 bits long.

b. Verify that your filter was properly implemented by comparing your results with each output **yi[n]**, for **i=1,...,4,** in Figure 1. In order to help you keep track of all the values, you should create a table for each pair **xi[n]** and **yi[n]** as shown in Table 1 for **x1[n]**.
c. For this task, you should fill up the tables for each pair on your own, by hand, before implementing the filter. This is so that you know what outputs to expect (Hint: see Figure 1). Include all tables in your report.

n	**...**	**k**	**k+1**	**k+2**	**k+3**	**k+4**	**k+5**	**k+6**	**k+7**	**k+8**	**k+9**	**k+10**	**k+11**	...
x1[n]	**...**	180	240	180	240	180	240	180	240	180	240	180	240	...
y1[n]	**...**	210	210	210	210	210	210	210	210	210	210	210	210	...

Table 1. Periodic discrete-time series input **x1[n]** and corresponding output **y1[n].**

IMPORTANT NOTE 2: Please note that the output **y1[n]**, in Table 1, is only showing the steady-state phase, and not the transient phase.

IMPORTANT NOTE 3: The remaining tasks in this assignment are simple extensions of your work in Task 2. Thus, it is important that you simulate your code for Task 2 extensively to make sure that it is correct. If you do that, then the following tasks will only require trivial changes to the code that you wrote for Task 2. Furthermore, you should take advantage of the simulator and use it to generate the values for the tables (similar to Table 1) for all the cases below. In short, for the following tasks, you should not be calculating the values for the desired tables by hand anymore. Instead, use the simulator and just fill up the tables with the values it generates!

Task 3

2. Repeat Task 2 for the following moving average filter **A[n]**:

$$A[n] = \frac{x[n] + x[n-2] + x[n-4] + x[n-6]}{4}$$

Task 4

3. Repeat Task 2 for the following moving average filter **B[n]**:

$$B[n] = \frac{x[n] + x[n-3] + x[n-6] + x[n-9]}{4}$$

Task 5

4. Choose three arbitrary and distinct positive integers **j, k,** and **l**, each no greater than 15, and then repeat Task 2 for the following moving average filter **C[n]**:

$$C[n] = \frac{x[n] + x[n-j] + x[n-k] + x[n-l]}{4}$$

Task 6 OPTIONAL AND FOR A LOT OF EXTRA CREDIT

5. Repeat Task 2 for the filter D[n] shown below:
    a. This filter may look perverse and intractable but it is not really that bad.
    b. This filter D[n] is NOT known as a moving average. But it is, by inspection, a simple FIR filter.
    c. It is relevant in wavelet applications.
    d. Hints:
        i. First, try to crack it on your own.
        ii. Note that in this case you will have to use direct multiplications, which we have avoided like the plague up to now.
        iii. Then, if interested, email me for further hints after you have done some work toward a solution for this.

$$D[n] = \frac{(1+\sqrt{3})\, x[n] + (3+\sqrt{3})\; x[n-1] + (3-\sqrt{3})\, x[n-2] + (1-\sqrt{3})\, x[n-3]}{4\sqrt{2}}$$