The assembly source files can for this experiment can be found here.

Contents

Objective

The objective of this experiment is to further analyze moving average filters using the PIC18F452 by considering more complex filters.

Introduction (Task 1)

The first task requires simulating the given template file to better understand what will be required to implement later. The `counter` value corresponds to the period to be selected which in turn, selects what data table (i.e., values of $x_i[n]$) to use as shown in figures 1a and 1b. If, for example, `counter=2`, the time series $x_1[n]$ is to be selected. Likewise, if `counter=6`, then the time series $x_3[n]$ is to be selected. Note that these tables must be commented or uncommented accordingly.

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

Figure 1a: `Counter` variable as a period selector

```
;  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 1b: The four possible data tables

Task 2

Having experimented with the template code, we must now consider the following time series:

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

It is noticeably more complex than previous time series we've looked at but the logic to implement it remains the same. We still require a *memory buffer* to compute necessary values (i.e., $x[n-1]$). Similarly, we require an adder and divider to compute the correct value of $y[n]$.

Implementing the Memory Buffer

The memory buffer will be responsible for storing the values $x[n]$ to $x[n-3]$. Utilizing a counter to select the appropriate period, the appropriate values for $x[n]$ to $x[n-3]$ can be assigned. Figure 2a shows the assembly code for the memory buffer by utilizing the `movff` instruction.

```
; --------------------------------
; (1) WRITE CODE FOR MEMORY BUFFER HERE
;        you may write the full code
;        here or call a subroutine

; Determine value for x[n] to x[n-3]
movff xn2, xn3 ;x[n-3] = x[n-2]
movff xn1, xn2 ;x[n-2] = x[n-1]
movff valueL, xn1 ;x[n-1] = x[n]
movwf valueL ; x[n] = TABLAT (current value
```

Figure 2a: Memory buffer for $y_i[n]$

Implementing the Adder/Divider

Figure 2b below shows how the summations and division will occur. Recall that a register's contents cannot exceed 8-Bits (i.e., value cannot be > 255). Hence, we may need to use more than 1 register to store the summations. For this reason, a summation result will span across two registers to store the lower and upper 8 bits. Namely, we will utilize arbitrary *L* and *U* registers as shown below. Consequently, addition cannot occur for more than two registers at a time. Since there are four elements that need to be added, we cannot rely on a single summation variable. The figure blow shows that we require three summation variables (16-Bit length): `SUM1`, `SUM2`, and `Res`. `SUM1` will add $x[n] + x[n-1]$ and `SUM2` will add $x[n-2] + x[n-3]$. `Res` will store the final value by storing the result `SUM1` and `SUM2`. Note that these registers are comprised of two 8-bit upper and lower registers. For example, `SUM1 = SUM1U + SUML` where '+' is the concatenation of `SUM1U` and `SUM1L`.

$$y[n] = \frac{\overbrace{\overbrace{x[n] + x[n-1]}^{SUM1 = SUM1U + SUM1L} + \overbrace{x[n-2] + x[n-3]}^{SUM2 = SUM2U + SUM2L}}^{Res = SUM1 + SUM2}}{4}$$

Rotate right *Res* twice (rrcf)

Figure 2b: Breakdown of summations and division

The assembly code for the adder is shown in figure 2c below. Note the use of two variables `TEMP1` and `TEMP2` which serve as temporary registers to load any carry into `SUM1U` and `SUM2U`.

```
; -----------------------------------
;                ADDER
; -----------------------------------
; First Summation (SUM1)
movf xn1, W ; changes wreg to x[n-1]
addwf xn,W ; x[n]+x[n-1]
movwf SUM1L ; Store above result into SUM1L
movf TEMP1,W
addwfc TEMP2,W ;x[n]+x[n-1]
movwf SUM1U ;result into SUM1U

; Second Summation (SUM2)
movf xn2,W ; x[n-2] = WREG
addwf xn3,W ; x[n-2]+x[n-3]
movwf SUM2L ;result into SUM2L
movf TEMP1,W
addwfc TEMP2,w ;x[n-2]+x[n-3] = SUM2U
movwf SUM2U ;result into SUM2U

; Final Summation (Res = SUM1 + SUM 2)
movf SUM1L, W
addwf SUM2L, W ; SUM1L + SUM2L
movwf ResL ; Store result into ResL
movf SUM1U, W
addwfc SUM2U, W ; SUM1U + SUM2U
movwf ResU ; Store result into ResU
```

Figure 2c: Adder for $y_i[n]$

Lastly, we must divide the entire summation by 4. To achieve this, we simply perform two right rotations both the `ResU` and `ResL` registers via the `rrcf` instruction as shown in figure 2d.

```
; ---------------------------------
;               DIVIDER
; ---------------------------------
;First Rotation for ResU and ResL
rrcf ResU, W ; ResU/2
movwf ResU
rrcf ResL, W ; ResL/2
movwf ResL
; Second Rotation for ResU and ResL (effective div by 4)
rrcf ResU, W ; ResU/2
movwf ResU ; Final ResU Value
rrcf ResL, W ; ResL/2
movwf ResL ; Final ResL value
```

Figure 2d: Divider for $y_i[n]$

To summarize this entire process, table 1 describes the purpose of each variable used.

Variable	Description
`xn`	Stores the value of $x[n]$
`xn1`	Stores the value of $x[n-1]$
`xn2`	Stores the value of $x[n-2]$
`xn3`	Stores the value of $x[n-3]$
`SUM1U`	Stores the upper 8 bits of `SUM1`
`SUM1L`	Stores the lower 8 bits of `SUM1`
`SUM2U`	Stores the upper 8 bits of `SUM2`
`SUM2L`	Stores the lower 8 bits of `SUM2`
`RESU`	Stores the upper 8 bits of `Res`
`RESL`	Stores the lower 8 bits of `Res`

Generating $y_1[n]$

Table 2 shows the expected values of $y_1[n]$. Note that only the *Steady State* values are shown on the table. We expect there to be transient values as well. To compute $y_1[n]$, it requires that `counter` is set to 2 so that appropriate period is selected. In addition, we must uncomment the corresponding data as described in the table.

n	…	k	k+1	k+2	k+3	k+4	k+5	k+6	k+7	k+8	k+9	…
$x_1[n]$	…	180	240	180	240	180	240	180	240	180	240	…
$y_1[n]$	…	210	210	210	210	210	210	210	210	210	210	…

Table 2: Expected output of $y_1[n]$

```
; time series X1
SimpleTable ; ---> period 2
db 180,240
```

Figure 3: Data table of time series $x_1[n]$

The following results are obtained. Figures 4a to 4c show the transient values of $y_1[n]$. Figures 4d and 4e show the values of $y_1[k]$ and $y_1[k+9]$.

Update	Address	Symbol Name	Value	Decimal	Binary
	000	counter	0x02	2	00000010
	FE8	WREG	0x2D	45	00101101
	FF5	TABLAT	0xB4	180	10110100
	002	xn	0xB4	180	10110100
	003	xn1	0x00	0	00000000
	004	xn2	0x00	0	00000000
	005	xn3	0x00	0	00000000
	006	SUM1U	0x00	0	00000000
	007	SUM1L	0xB4	180	10110100
	008	SUM2U	0x00	0	00000000
	009	SUM2L	0x00	0	00000000
	00D	ResU	0x00	0	00000000
	00C	ResL	0x2D	45	00101101

Fiugre 4a: First transient value of $y_1[n]$

Update	Address	Symbol Name	Value	Decimal	Binary
	000	counter	0x01	1	00000001
	FE8	WREG	0x69	105	01101001
	FF5	TABLAT	0xF0	240	11110000
	002	xn	0xF0	240	11110000
	003	xn1	0xB4	180	10110100
	004	xn2	0x00	0	00000000
	005	xn3	0x00	0	00000000
	006	SUM1U	0x01	1	00000001
	007	SUM1L	0xA4	164	10100100
	008	SUM2U	0x00	0	00000000
	009	SUM2L	0x00	0	00000000
	00D	ResU	0x00	0	00000000
	00C	ResL	0x69	105	01101001

Fiugre 4b: Second transient value of $y_1[n]$

Update	Address	Symbol Name	Value	Decimal	Binary
	000	counter	0x02	2	00000010
	FE8	WREG	0x96	150	10010110
	FF5	TABLAT	0xB4	180	10110100
	002	xn	0xB4	180	10110100
	003	xn1	0xF0	240	11110000
	004	xn2	0xB4	180	10110100
	005	xn3	0x00	0	00000000
	006	SUM1U	0x01	1	00000001
	007	SUM1L	0xA4	164	10100100
	008	SUM2U	0x00	0	00000000
	009	SUM2L	0xB4	180	10110100
	00D	ResU	0x00	0	00000000
	00C	ResL	0x96	150	10010110

Figure 4c: Third transisent value of $y_1[n]$

Update	Address	Symbol Name	Value	Decimal	Binary
	000	counter	0x01	1	00000001
	FE8	WREG	0xD2	210	11010010
	FF5	TABLAT	0xF0	240	11110000
	002	xn	0xF0	240	11110000
	003	xn1	0xB4	180	10110100
	004	xn2	0xF0	240	11110000
	005	xn3	0xB4	180	10110100
	006	SUM1U	0x01	1	00000001
	007	SUM1L	0xA4	164	10100100
	008	SUM2U	0x01	1	00000001
	009	SUM2L	0xA4	164	10100100
	00D	ResU	0x00	0	00000000
	00C	ResL	0xD2	210	11010010

Figure 4d: Value of $y_1[k]$

Update	Address	Symbol Name	Value	Decimal	Binary
	000	counter	0x02	2	00000010
	FE8	WREG	0xD2	210	11010010
	FF5	TABLAT	0xB4	180	10110100
	002	xn	0xB4	180	10110100
	003	xn1	0xF0	240	11110000
	004	xn2	0xB4	180	10110100
	005	xn3	0xF0	240	11110000
	006	SUM1U	0x01	1	00000001
	007	SUM1L	0xA4	164	10100100
	008	SUM2U	0x01	1	00000001
	009	SUM2L	0xA4	164	10100100
	00D	ResU	0x00	0	00000000
	00C	ResL	0xD2	210	11010010

Figure 4e: Value of $y_1[k+9]$

The plot below shows the behavior of the moving average filter with the transisent phase higlighted in yellow and the steady-state phase in green.

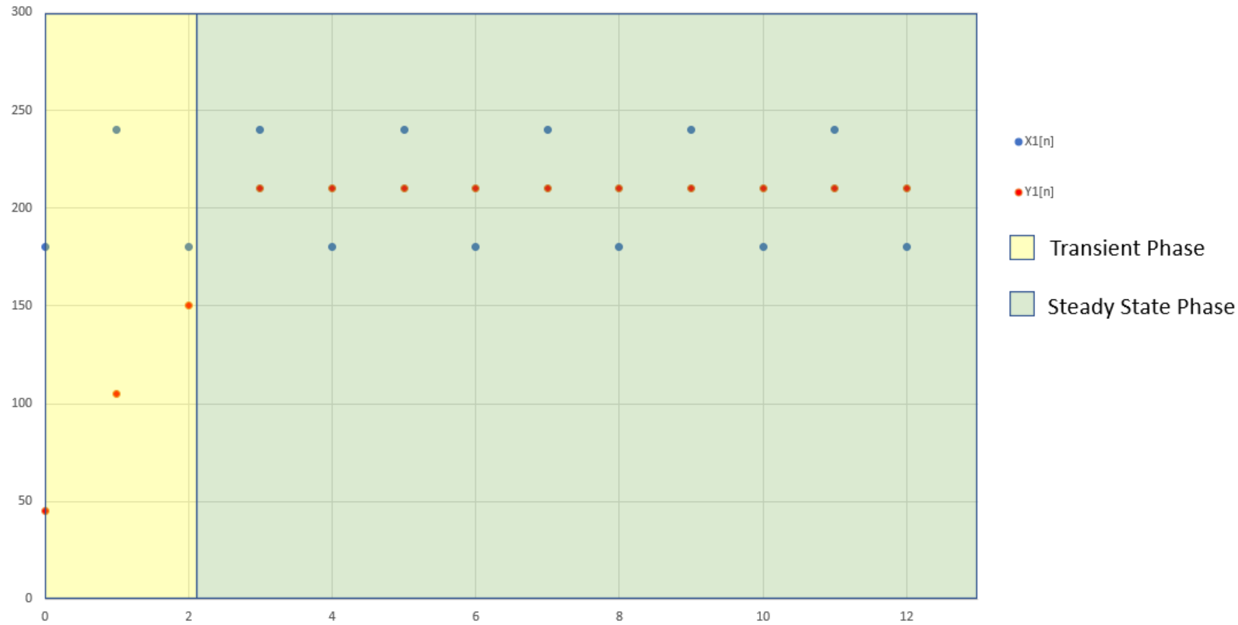


Figure 5: Average moving filter $y_1[n]$

Generating $y_2[n]$

To compute $y_2[n]$, we only need to change the period (i.e., the `counter` value). It was previous set to 2 but is now changed to 4. Also, we uncomment the appropriate data table.

```
; time series X2
SimpleTable ; ---> period 4
db 180,240,200,244
```

Figure 6: Data table of time series $x_2[n]$

Table 3 shows the $y_2[n]$ expected steady state values.

n	…	k	k+1	k+2	k+3	k+4	k+5	k+7	k+8	k+9	k+10	…
$x_2[n]$	…	180	240	200	244	180	240	200	244	180	240	…
$y_2[n]$	…	216	216	216	216	216	216	216	216	216	216	…

Table 3: Steady State values of $y_2[n]$

Figures 7a to 7c show the transisent values of $y_2[n]$. Figures 7d and 7e show the values of $y_2[k]$ and $y_2[k+4]$.

Update	Address	Symbol Name	Value	Decimal	Binary
	000	counter	0x04	4	00000100
	FE8	WREG	0x2D	45	00101101
	FF5	TABLAT	0xB4	180	10110100
	003	xn	0xB4	180	10110100
	004	xn1	0x00	0	00000000
	005	xn2	0x00	0	00000000
	006	xn3	0x00	0	00000000
	007	SUM1U	0x00	0	00000000
	008	SUM1L	0xB4	180	10110100
	009	SUM2U	0x00	0	00000000
	00A	SUM2L	0x00	0	00000000
	00C	ResU	0x00	0	00000000
	00B	ResL	0x2D	45	00101101

Figure 7a: First transient vlaue

Update	Address	Symbol Name	Value	Decimal	Binary
	000	counter	0x03	3	00000011
	FE8	WREG	0x69	105	01101001
	FF5	TABLAT	0xF0	240	11110000
	003	xn	0xF0	240	11110000
	004	xn1	0xB4	180	10110100
	005	xn2	0x00	0	00000000
	006	xn3	0x00	0	00000000
	007	SUM1U	0x01	1	00000001
	008	SUM1L	0xA4	164	10100100
	009	SUM2U	0x00	0	00000000
	00A	SUM2L	0x00	0	00000000
	00C	ResU	0x00	0	00000000
	00B	ResL	0x69	105	01101001

Figure 7b: Second transient vlaue

Update	Address	Symbol Name	Value	Decimal	Binary
	000	counter	0x02	2	00000010
	FE8	WREG	0x9B	155	10011011
	FF5	TABLAT	0xC8	200	11001000
	003	xn	0xC8	200	11001000
	004	xn1	0xF0	240	11110000
	005	xn2	0xB4	180	10110100
	006	xn3	0x00	0	00000000
	007	SUM1U	0x01	1	00000001
	008	SUM1L	0xB8	184	10111000
	009	SUM2U	0x00	0	00000000
	00A	SUM2L	0xB4	180	10110100
	00C	ResU	0x00	0	00000000
	00B	ResL	0x9B	155	10011011

Figure 7c: Third transisent vlaue

Update	Address	Symbol Name	Value	Decimal	Binary
	000	counter	0x01	1	00000001
	FE8	WREG	0xD8	216	11011000
	FF5	TABLAT	0xF4	244	11110100
	003	xn	0xF4	244	11110100
	004	xn1	0xC8	200	11001000
	005	xn2	0xF0	240	11110000
	006	xn3	0xB4	180	10110100
	007	SUM1U	0x01	1	00000001
	008	SUM1L	0xBC	188	10111100
	009	SUM2U	0x01	1	00000001
	00A	SUM2L	0xA4	164	10100100
	00C	ResU	0x00	0	00000000
	00B	ResL	0xD8	216	11011000

Figure 7d: Value of $y_2[k]$

Update	Address	Symbol Name	Value	Decimal	Binary
	000	counter	0x02	2	00000010
	FE8	WREG	0xD8	216	11011000
	FF5	TABLAT	0xC8	200	11001000
	003	xn	0xC8	200	11001000
	004	xn1	0xF0	240	11110000
	005	xn2	0xB4	180	10110100
	006	xn3	0xF4	244	11110100
	007	SUM1U	0x01	1	00000001
	008	SUM1L	0xB8	184	10111000
	009	SUM2U	0x01	1	00000001
	00A	SUM2L	0xA8	168	10101000
	00C	ResU	0x00	0	00000000
	00B	ResL	0xD8	216	11011000

Figure 7e: Value of $y_2[k+4]$

The plot below shows the behavior of this moving average filter with the transisent phase higlighted in yellow and the steady-state phase in green.

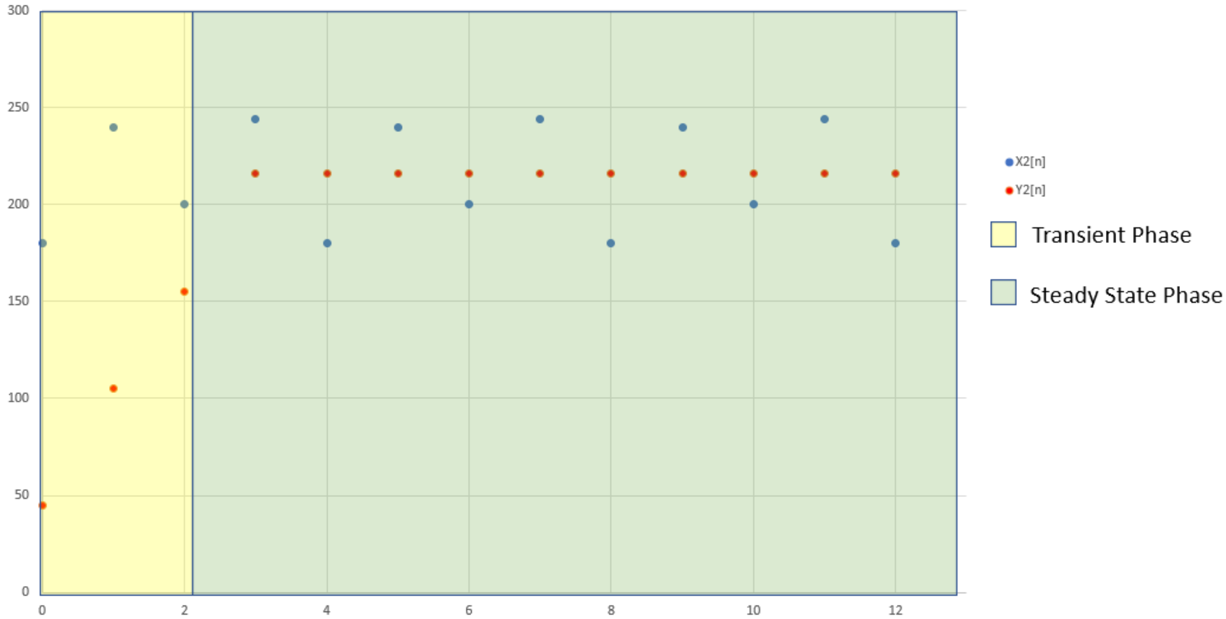


Figure 8: Average moving filter $y_2[n]$

Implementing $y_3[n]$

To compute $y_3[n]$, we only need to change the period (i.e., the `counter` value). It was previous set to 4 but is now changed to 6. Also, we uncomment the appropriate data table.

```
; time series X3
SimpleTable ; ---> period 6
db 180,240,200,244,216,236
```

Figure 9: Data table of time series $x_3[n]$

Table 4 shows the $y_3[n]$ show only the expected Steady State values.

n	…	k	k+1	k+2	k+3	k+4	k+5	k+7	k+8	k+9	k+10	k+11	k+12	…
$x_3[n]$	…	180	240	200	244	216	236	180	240	200	244	216	236	…
$y_3[n]$	…	216	225	224	219	218	214	216	225	224	219	218	214	…

Table 4: Steady State values of $y_3[n]$

Figures 10a to 10c show the transisent values of $y_3[n]$. Figures 10d through 10g show the first three values of $y_3[n]$ along with the last value.

Update	Address	Symbol Name	Value	Decimal	Binary
	000	counter	0x06	6	00000110
	FE8	WREG	0x2D	45	00101101
	FF5	TABLAT	0xB4	180	10110100
	003	xn	0xB4	180	10110100
	004	xn1	0x00	0	00000000
	005	xn2	0x00	0	00000000
	006	xn3	0x00	0	00000000
	007	SUM1U	0x00	0	00000000
	008	SUM1L	0xB4	180	10110100
	009	SUM2U	0x00	0	00000000
	00A	SUM2L	0x00	0	00000000
	00C	ResU	0x00	0	00000000
	00B	ResL	0x2D	45	00101101

Figure 10a: First transisent vlaue

Update	Address	Symbol Name	Value	Decimal	Binary
	000	counter	0x05	5	00000101
	FE8	WREG	0x69	105	01101001
	FF5	TABLAT	0xF0	240	11110000
	003	xn	0xF0	240	11110000
	004	xn1	0xB4	180	10110100
	005	xn2	0x00	0	00000000
	006	xn3	0x00	0	00000000
	007	SUM1U	0x01	1	00000001
	008	SUM1L	0xA4	164	10100100
	009	SUM2U	0x00	0	00000000
	00A	SUM2L	0x00	0	00000000
	00C	ResU	0x00	0	00000000
	00B	ResL	0x69	105	01101001

Figure 10b: Second transisent vlaue

Update	Address	Symbol Name	Value	Decimal	Binary
	000	counter	0x04	4	00000100
	FE8	WREG	0x9B	155	10011011
	FF5	TABLAT	0xC8	200	11001000
	003	xn	0xC8	200	11001000
	004	xn1	0xF0	240	11110000
	005	xn2	0xB4	180	10110100
	006	xn3	0x00	0	00000000
	007	SUM1U	0x01	1	00000001
	008	SUM1L	0xB8	184	10111000
	009	SUM2U	0x00	0	00000000
	00A	SUM2L	0xB4	180	10110100
	00C	ResU	0x00	0	00000000
	00B	ResL	0x9B	155	10011011

Figure 10c: Third transisent vlaue

Update	Address	Symbol Name	Value	Decimal	Binary
	000	counter	0x03	3	00000011
	FE8	WREG	0xD8	216	11011000
	FF5	TABLAT	0xF4	244	11110100
	003	xn	0xF4	244	11110100
	004	xn1	0xC8	200	11001000
	005	xn2	0xF0	240	11110000
	006	xn3	0xB4	180	10110100
	007	SUM1U	0x01	1	00000001
	008	SUM1L	0xBC	188	10111100
	009	SUM2U	0x01	1	00000001
	00A	SUM2L	0xA4	164	10100100
	00C	ResU	0x00	0	00000000
	00B	ResL	0xD8	216	11011000

Figure 10d: Value of $y_3[k]$

Update	Address	Symbol Name	Value	Decimal	Binary
	000	counter	0x02	2	00000010
	FE8	WREG	0xE1	225	11100001
	FF5	TABLAT	0xD8	216	11011000
	003	xn	0xD8	216	11011000
	004	xn1	0xF4	244	11110100
	005	xn2	0xC8	200	11001000
	006	xn3	0xF0	240	11110000
	007	SUM1U	0x01	1	00000001
	008	SUM1L	0xCC	204	11001100
	009	SUM2U	0x01	1	00000001
	00A	SUM2L	0xB8	184	10111000
	00C	ResU	0x00	0	00000000
	00B	ResL	0xE1	225	11100001

Figure 10e: Value of $y_3[k+1]$

Update	Address	Symbol Name	Value	Decimal	Binary
	000	counter	0x01	1	00000001
	FE8	WREG	0xE0	224	11100000
	FF5	TABLAT	0xEC	236	11101100
	003	xn	0xEC	236	11101100
	004	xn1	0xD8	216	11011000
	005	xn2	0xF4	244	11110100
	006	xn3	0xC8	200	11001000
	007	SUM1U	0x01	1	00000001
	008	SUM1L	0xC4	196	11000100
	009	SUM2U	0x01	1	00000001
	00A	SUM2L	0xBC	188	10111100
	00C	ResU	0x00	0	00000000
	00B	ResL	0xE0	224	11100000

Figure 10f: Value of $y_3[k+2]$

Update	Address	Symbol Name	Value	Decimal	Binary
	000	counter	0x04	4	00000100
	FE8	WREG	0xD6	214	11010110
	FF5	TABLAT	0xC8	200	11001000
	003	xn	0xC8	200	11001000
	004	xn1	0xF0	240	11110000
	005	xn2	0xB4	180	10110100
	006	xn3	0xEC	236	11101100
	007	SUM1U	0x01	1	00000001
	008	SUM1L	0xB8	184	10111000
	009	SUM2U	0x01	1	00000001
	00A	SUM2L	0xA0	160	10100000
	00C	ResU	0x00	0	00000000
	00B	ResL	0xD6	214	11010110

Figure 10g: Value of $y_3[k+5]$

The plot below shows the behavior of this moving average filter with the transisent phase higlighted in yellow and the steady-state phase in green.

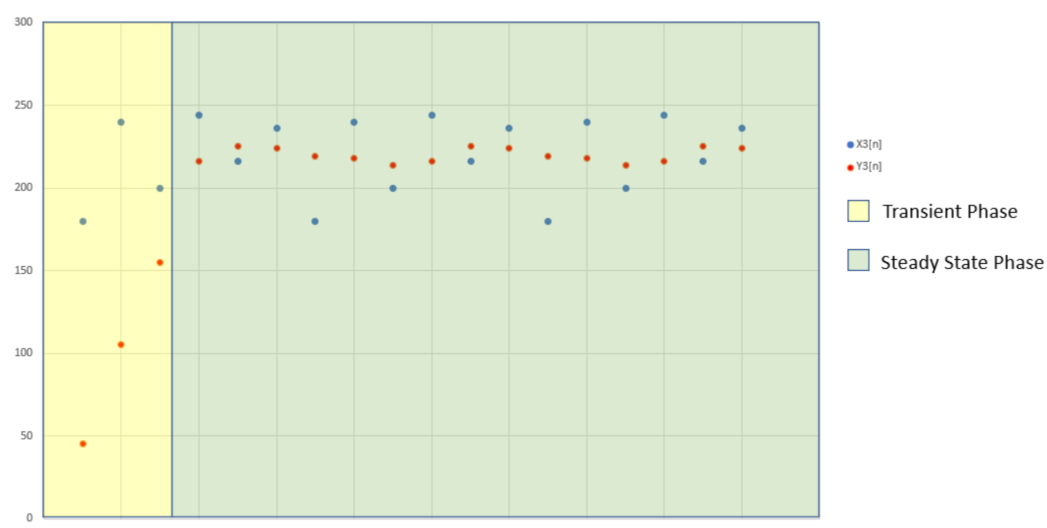


Figure 11: Average moving filter $y_3[n]$

Implementing $y_4[n]$

To compute $y_4[n]$, we only need to change the period (i.e., the `counter` value). It was previous set to 6 but is now changed to 8. Also, we comment out the appropriate data table.

```
; time series X4
SimpleTable ; ---> period 8
db 180,240,200,244,216,236,160,176
```

Figure 12: Data table of time series $x_4[n]$

Table 5 shows only the Steady State values of $y_4[n]$.

$\boldsymbol{n}$	**…**	**k**	**k+1**	**k+2**	**k+3**	**k+4**	**k+5**	**k+6**	**k+7**	**k+8**	**k+9**	**k+10**	**k+11**	**…**
$\boldsymbol{x_4[n]}$	…	180	240	200	244	216	236	160	176	180	240	244	216	…
$\boldsymbol{y_4[n]}$	…	216	225	224	214	197	188	189	199	216	225	224	214	…

Table 5: Steady State values of $y_4[n]$

Figures 13a to 13c show the transisent values of $y_4[n]$. Figures 13d through 13g show the first and last three values of $y_4[n]$.

Update	Address	Symbol Name	Value	Decimal	Binary
	000	counter	0x08	8	00001000
	FE8	WREG	0x2D	45	00101101
	FF5	TABLAT	0xB4	180	10110100
	003	xn	0xB4	180	10110100
	004	xn1	0x00	0	00000000
	005	xn2	0x00	0	00000000
	006	xn3	0x00	0	00000000
	007	SUM1U	0x00	0	00000000
	008	SUM1L	0xB4	180	10110100
	009	SUM2U	0x00	0	00000000
	00A	SUM2L	0x00	0	00000000
	00C	ResU	0x00	0	00000000
	00B	ResL	0x2D	45	00101101

Figure 13a: First transisent vlaue

Update	Address	Symbol Name	Value	Decimal	Binary
	000	counter	0x07	7	00000111
	FE8	WREG	0x69	105	01101001
	FF5	TABLAT	0xF0	240	11110000
	003	xn	0xF0	240	11110000
	004	xn1	0xB4	180	10110100
	005	xn2	0x00	0	00000000
	006	xn3	0x00	0	00000000
	007	SUM1U	0x01	1	00000001
	008	SUM1L	0xA4	164	10100100
	009	SUM2U	0x00	0	00000000
	00A	SUM2L	0x00	0	00000000
	00C	ResU	0x00	0	00000000
	00B	ResL	0x69	105	01101001

Figure 13b: Second transisent vlaue

Update	Address	Symbol Name	Value	Decimal	Binary
	000	counter	0x06	6	00000110
	FE8	WREG	0x9B	155	10011011
	FF5	TABLAT	0xC8	200	11001000
	003	xn	0xC8	200	11001000
	004	xn1	0xF0	240	11110000
	005	xn2	0xB4	180	10110100
	006	xn3	0x00	0	00000000
	007	SUM1U	0x01	1	00000001
	008	SUM1L	0xB8	184	10111000
	009	SUM2U	0x00	0	00000000
	00A	SUM2L	0xB4	180	10110100
	00C	ResU	0x00	0	00000000
	00B	ResL	0x9B	155	10011011

Figure 13c: Third transisent vlaue

Update	Address	Symbol Name	Value	Decimal	Binary
	000	counter	0x05	5	00000101
	FE8	WREG	0xD8	216	11011000
	FF5	TABLAT	0xF4	244	11110100
	003	xn	0xF4	244	11110100
	004	xn1	0xC8	200	11001000
	005	xn2	0xF0	240	11110000
	006	xn3	0xB4	180	10110100
	007	SUM1U	0x01	1	00000001
	008	SUM1L	0xBC	188	10111100
	009	SUM2U	0x01	1	00000001
	00A	SUM2L	0xA4	164	10100100
	00C	ResU	0x00	0	00000000
	00B	ResL	0xD8	216	11011000

Figure 13d: Value of $y_4[k]$

Update	Address	Symbol Name	Value	Decimal	Binary
	000	counter	0x08	8	00001000
	FE8	WREG	0xBC	188	10111100
	FF5	TABLAT	0xB4	180	10110100
	003	xn	0xB4	180	10110100
	004	xn1	0xB0	176	10110000
	005	xn2	0xA0	160	10100000
	006	xn3	0xEC	236	11101100
	007	SUM1U	0x01	1	00000001
	008	SUM1L	0x64	100	01100100
	009	SUM2U	0x01	1	00000001
	00A	SUM2L	0x8C	140	10001100
	00C	ResU	0x00	0	00000000
	00B	ResL	0xBC	188	10111100

Figure 13e: Value of $y_4[k + 5]$

Update	Address	Symbol Name	Value	Decimal	Binary
	000	counter	0x07	7	00000111
	FE8	WREG	0xBD	189	10111101
	FF5	TABLAT	0xF0	240	11110000
	003	xn	0xF0	240	11110000
	004	xn1	0xB4	180	10110100
	005	xn2	0xB0	176	10110000
	006	xn3	0xA0	160	10100000
	007	SUM1U	0x01	1	00000001
	008	SUM1L	0xA4	164	10100100
	009	SUM2U	0x01	1	00000001
	00A	SUM2L	0x50	80	01010000
	00C	ResU	0x00	0	00000000
	00B	ResL	0xBD	189	10111101

Figure 13f: Value of $y_4[k+6]$

Update	Address	Symbol Name	Value	Decimal	Binary
	000	counter	0x06	6	00000110
	FE8	WREG	0xC7	199	11000111
	FF5	TABLAT	0xC8	200	11001000
	003	xn	0xC8	200	11001000
	004	xn1	0xF0	240	11110000
	005	xn2	0xB4	180	10110100
	006	xn3	0xB0	176	10110000
	007	SUM1U	0x01	1	00000001
	008	SUM1L	0xB8	184	10111000
	009	SUM2U	0x01	1	00000001
	00A	SUM2L	0x64	100	01100100
	00C	ResU	0x00	0	00000000
	00B	ResL	0xC7	199	11000111

Figure 13g: Value of $y_4[k+7]$

The plot below shows the behavior of this moving average filter with the transisent phase higlighted in yellow and the steady-state phase in green.

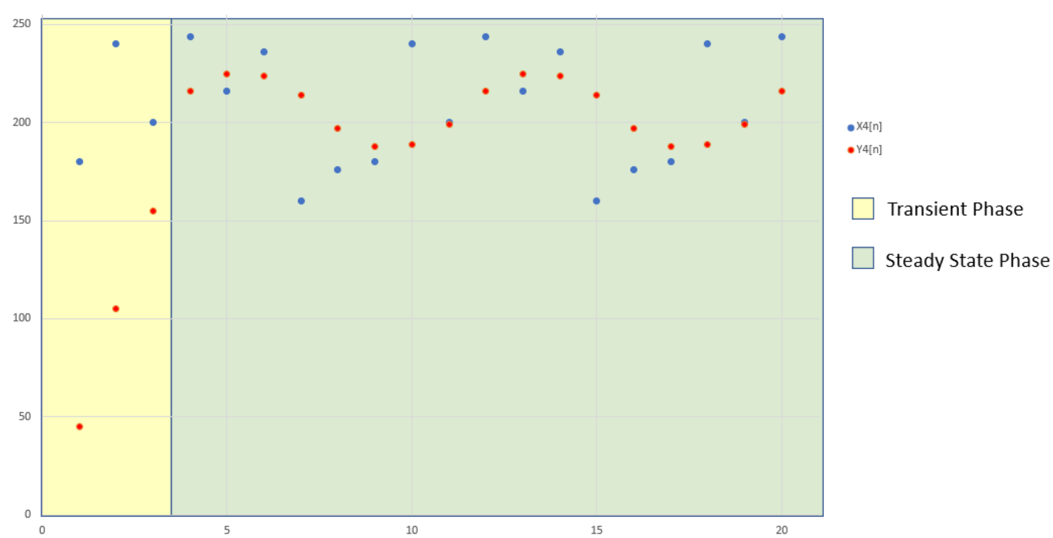


Figure 14: Average moving filter $y_4[n]$

Task 3

We extend the program to now implement the following time series:

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

We now need to calculate the values of $x[n-2]$, $x[n-4]$, and $x[n-6]$. Figure 15 below shows a modified memory buffer to compute these values. Again, we wish to compute $A_1[n]$ to $A_4[n]$ by setting the appropriate `counter` value.

```
; --------------------------------
; (1) WRITE CODE FOR MEMORY BUFFER HERE
;       you may write the full code
;       here or call a subroutine

; Determine values for x[n] to x[n-6]
movff xn5, xn6 ;x[n-6] = x[n-5]
movff xn4, xn5 ;x[n-5] = x[n-4]
movff xn3, xn4 ;x[n-4] = x[n-3]
movff xn2, xn3 ;x[n-3] = x[n-2]
movff xn1, xn2 ;x[n-2] = x[n-1]
movff xn, xn1 ;x[n-1] = x[n]
movwf xn ; x[n] = TABLAT (current value)
```

Figure 15: Memory buffer for $A_i[n]$

Implementing $A_1[n]$

Table 7 shows the obtained transient and Steady State values of $A_1[n]$

$\boldsymbol{n}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**…**
$\boldsymbol{x_1[n]}$	180	240	180	240	180	240	180	240	180	240	180	240	…
$\boldsymbol{A_1[n]}$	45	60	90	120	135	180	180	240	180	240	180	240	…

Table 7: Values of $A_1[n]$

Figure 16 shows all 6 transient values and the period values of $A_1[n]$

00F	ResU	0x00	0	00000000
00E	ResL	0x2D	45	00101101
00F	ResU	0x00	0	00000000
00E	ResL	0x3C	60	00111100
00F	ResU	0x00	0	00000000
00E	ResL	0x5A	90	01011010
00F	ResU	0x00	0	00000000
00E	ResL	0x78	120	01111000
00F	ResU	0x00	0	00000000
00E	ResL	0x87	135	10000111
00F	ResU	0x00	0	00000000
00E	ResL	0xB4	180	10110100
00F	ResU	0x00	0	00000000
00E	ResL	0xB4	180	10110100
00F	ResU	0x00	0	00000000
00E	ResL	0xF0	240	11110000

Figure 16: Transient and period values of $A_1[n]$

Figure 17 shows a plot of the behavior of this moving average filter with the transisent phase higlighted in yellow and the steady-state phase in green.

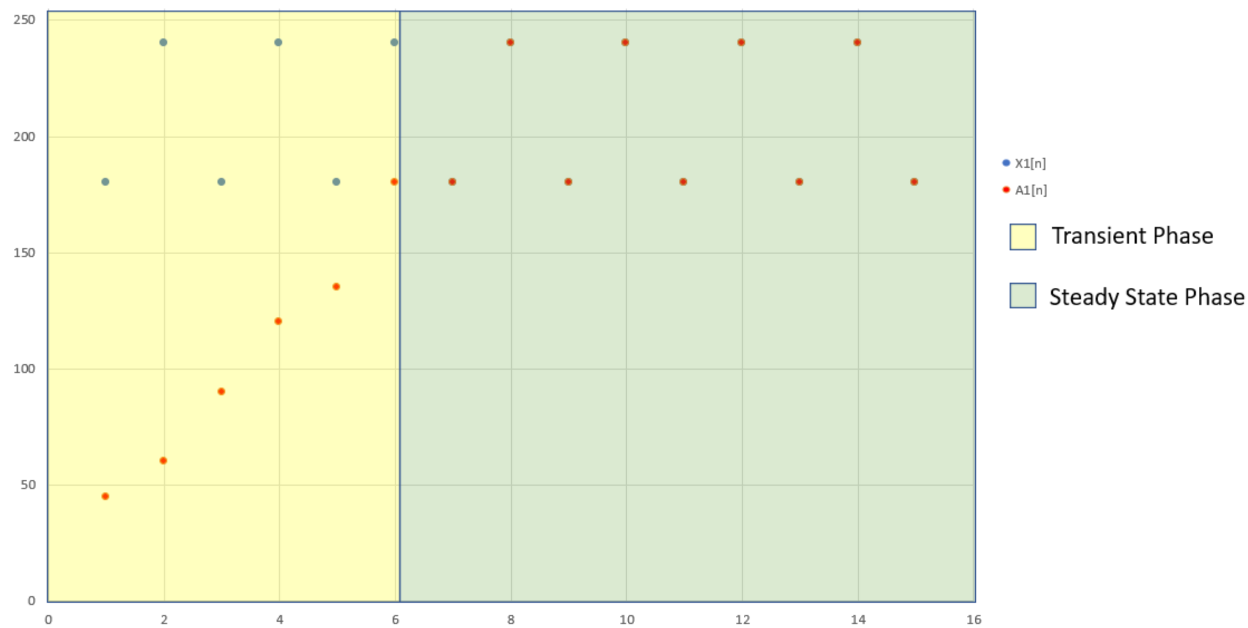


Figure 17: Average moving filter $A_1[n]$

Implementing $A_2[n]$

Table 8 shows the obtained transient and Steady State values of $A_1[n]$

$\boldsymbol{n}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**…**
$\boldsymbol{x_2[n]}$	180	240	200	244	180	240	200	244	180	240	200	244	…
$\boldsymbol{A_2[n]}$	45	60	95	121	140	181	190	242	190	242	190	242	…

Table 8: Values of $A_2[n]$

Figure 18 shows all 6 transient values and the period values of $A_2[n]$

00F	ResU	0x00	0	00000000
00E	ResL	0x2D	45	00101101
00F	ResU	0x00	0	00000000
00E	ResL	0x3C	60	00111100
00F	ResU	0x00	0	00000000
00E	ResL	0x5F	95	01011111
00F	ResU	0x00	0	00000000
00E	ResL	0x79	121	01111001
00F	ResU	0x00	0	00000000
00E	ResL	0x8C	140	10001100
00F	ResU	0x00	0	00000000
00E	ResL	0xB5	181	10110101
00F	ResU	0x00	0	00000000
00E	ResL	0xBE	190	10111110
00F	ResU	0x00	0	00000000
00E	ResL	0xF2	242	11110010
00F	ResU	0x00	0	00000000
00E	ResL	0xBE	190	10111110
00F	ResU	0x00	0	00000000
00E	ResL	0xF2	242	11110010

Figure 18: Transient and period values of $A_2[n]$

Figure 19 shows a plot of the behavior of this moving average filter with the transisent phase higlighted in yellow and the steady-state phase in green.

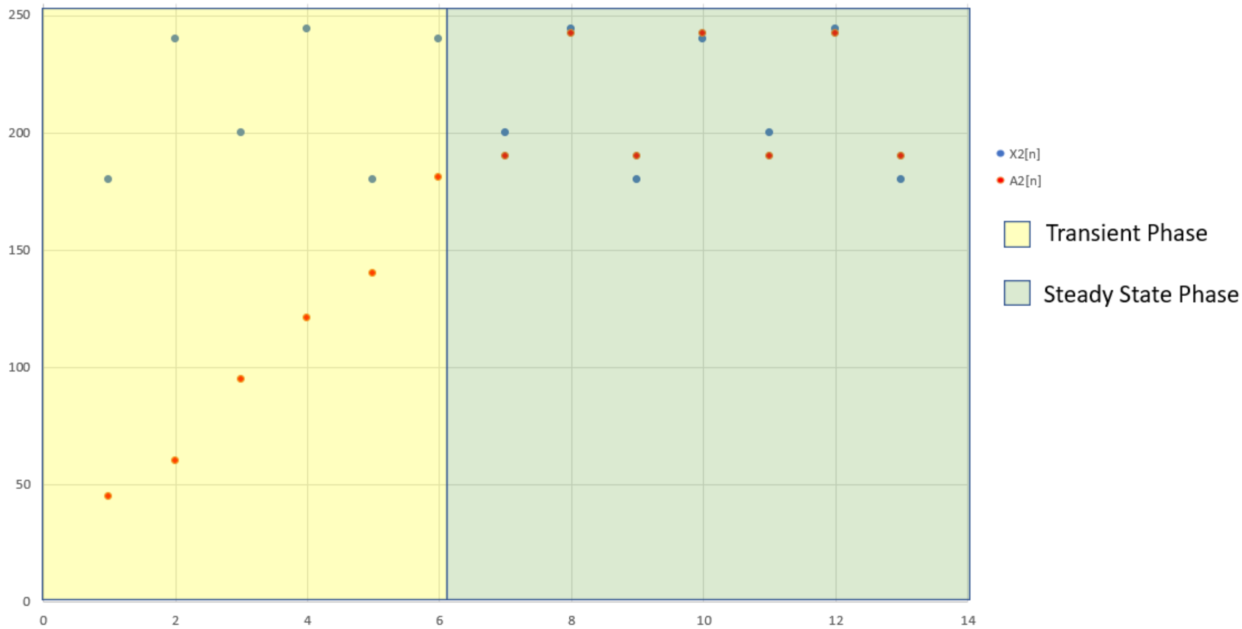


Figure 19: Average moving filter $A_2[n]$

Implementing $A_3[n]$

Table 9 shows the obtained transient and Steady State values of $A_3[n]$

$\boldsymbol{n}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**…**
$\boldsymbol{x_3[n]}$	180	240	200	244	216	236	180	240	200	244	216	236	…
$\boldsymbol{A_3[n]}$	45	60	95	121	149	180	194	240	199	241	203	239	…

Table 9: Values of $A_3[n]$

Figure 20 shows all 6 transient values and the period values of $A_3[n]$

00F	ResU	0x00	0	00000000
00E	ResL	0x2D	45	00101101
00F	ResU	0x00	0	00000000
00E	ResL	0x3C	60	00111100
00F	ResU	0x00	0	00000000
00E	ResL	0x5F	95	01011111
00F	ResU	0x00	0	00000000
00E	ResL	0x79	121	01111001
00F	ResU	0x00	0	00000000
00E	ResL	0x95	149	10010101
00F	ResU	0x00	0	00000000
00E	ResL	0xB4	180	10110100
00F	ResU	0x00	0	00000000
00E	ResL	0xC2	194	11000010
00F	ResU	0x00	0	00000000
00E	ResL	0xF0	240	11110000
00F	ResU	0x00	0	00000000
00E	ResL	0xC7	199	11000111
00F	ResU	0x00	0	00000000
00E	ResL	0xF1	241	11110001
00F	ResU	0x00	0	00000000
00E	ResL	0xCB	203	11001011
00F	ResU	0x00	0	00000000
00E	ResL	0xEF	239	11101111

Figure 20: Transient and period values of $A_3[n]$

Figure 21 shows a plot of the behavior of this moving average filter with the transisent phase higlighted in yellow and the steady-state phase in green.

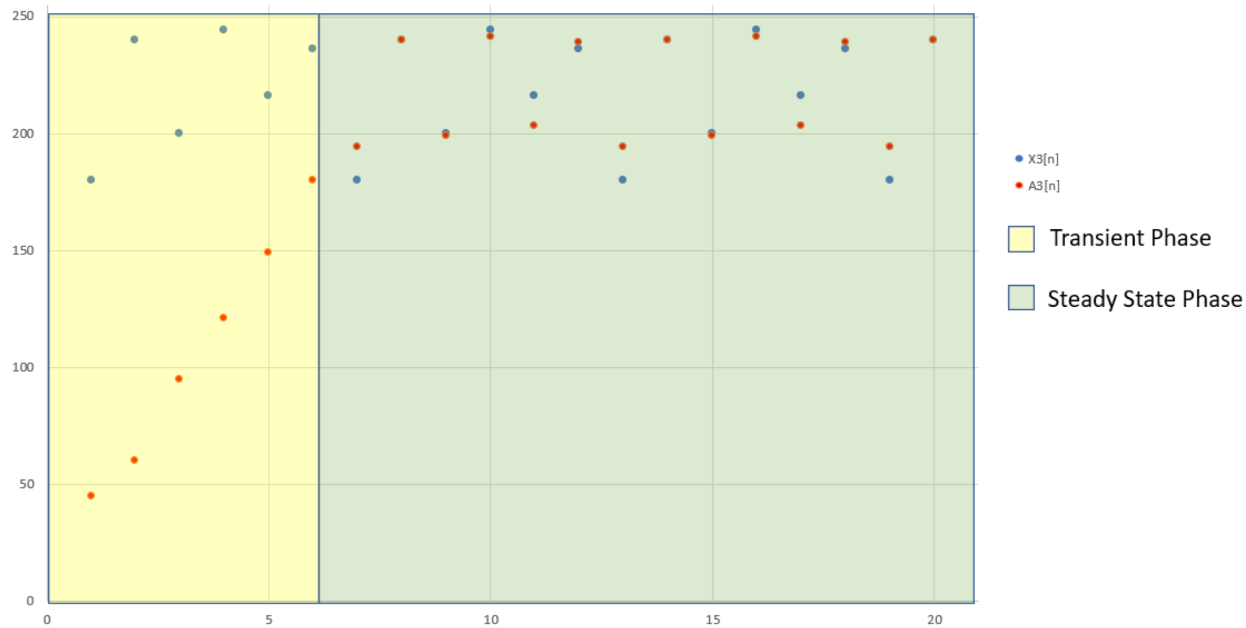


Figure 21: Average moving filter $A_3[n]$

Implementing $A_4[n]$

Table 10 shows the obtained transient and Steady State values of $A_4[n]$

$\boldsymbol{n}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**…**
$\boldsymbol{x_4[n]}$	180	240	200	244	216	236	160	176	180	240	200	244	…
$\boldsymbol{A_4[n]}$	45	60	95	121	149	180	189	224	189	224	189	224	…

Table 10: Values of $A_4[n]$

Figure 22 shows all 6 transient values and the period values of $A_4[n]$

00F	ResU	0x00	0	00000000
00E	ResL	0x2D	45	00101101
00F	ResU	0x00	0	00000000
00E	ResL	0x3C	60	00111100
00F	ResU	0x00	0	00000000
00E	ResL	0x5F	95	01011111
00F	ResU	0x00	0	00000000
00E	ResL	0x79	121	01111001
00F	ResU	0x00	0	00000000
00E	ResL	0x95	149	10010101
00F	ResU	0x00	0	00000000
00E	ResL	0xB4	180	10110100
00F	ResU	0x00	0	00000000
00E	ResL	0xBD	189	10111101
00F	ResU	0x00	0	00000000
00E	ResL	0xE0	224	11100000

Figure 22: Transient and period values of $A_4[n]$

Figure 23 shows a plot of the behavior of this moving average filter with the transisent phase higlighted in yellow and the steady-state phase in green.

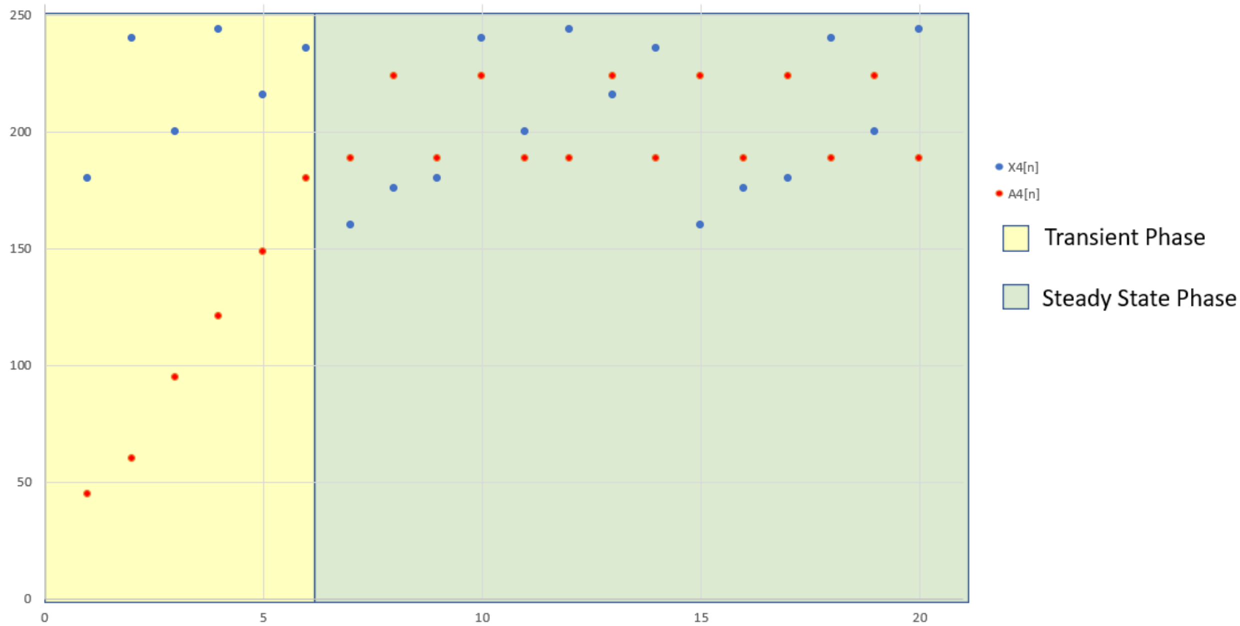


Figure 23: Average moving filter $A_4[n]$

Task 4

We extend the program to now implement the following time series:

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

We now need to calculate the values of $x[n-3]$, $x[n-6]$, and $x[n-9]$. Figure 24 below shows a modified memory buffer to compute these values. Again, we wish to compute $B_1[n]$ to $B_4[n]$ by setting the appropriate `counter` value.

```
; --------------------------------
; (1) WRITE CODE FOR MEMORY BUFFER HERE
;        you may write the full code
;        here or call a subroutine

; Determine values for x[n] to x[n-9]
movff xn8, xn9 ;x[n-9] = x[n-8]
movff xn7, xn8 ;x[n-8] = x[n-7]
movff xn6, xn7 ;x[n-7] = x[n-6]
movff xn5, xn6 ;x[n-6] = x[n-5]
movff xn4, xn5 ;x[n-5] = x[n-4]
movff xn3, xn4 ;x[n-4] = x[n-3]
movff xn2, xn3 ;x[n-3] = x[n-2]
movff xn1, xn2 ;x[n-2] = x[n-1]
movff xn, xn1 ;x[n-1] = x[n]
movwf xn ; x[n] = TABLAT (current value)
```

Figure 24: Memory buffer for $B_i[n]$

Implementing $B_1[n]$

Table 11 shows the obtained transient and Steady State values of $B_1[n]$

$\boldsymbol{n}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**…**
$\boldsymbol{x_1[n]}$	180	240	180	240	180	240	180	240	180	240	180	240	…
$\boldsymbol{B_1[n]}$	45	60	45	105	105	105	150	165	150	210	210	210	…

Table 11: Values of $B_1[n]$

Figure 25 shows all 9 transient values and the period values of $B_1[n]$

012	ResU	0x00	0	00000000
011	ResL	0x2D	45	00101101
012	ResU	0x00	0	00000000
011	ResL	0x3C	60	00111100
012	ResU	0x00	0	00000000
011	ResL	0x2D	45	00101101
012	ResU	0x00	0	00000000
011	ResL	0x69	105	01101001
012	ResU	0x00	0	00000000
011	ResL	0x69	105	01101001
012	ResU	0x00	0	00000000
011	ResL	0x69	105	01101001
012	ResU	0x00	0	00000000
011	ResL	0x96	150	10010110
012	ResU	0x00	0	00000000
011	ResL	0xA5	165	10100101
012	ResU	0x00	0	00000000
011	ResL	0x96	150	10010110
012	ResU	0x00	0	00000000
011	ResL	0xD2	210	11010010

Figure 25: Transient and period values of $B_1[n]$

Figure 26 shows a plot of the behavior of this moving average filter with the transisent phase higlighted in yellow and the steady-state phase in green.

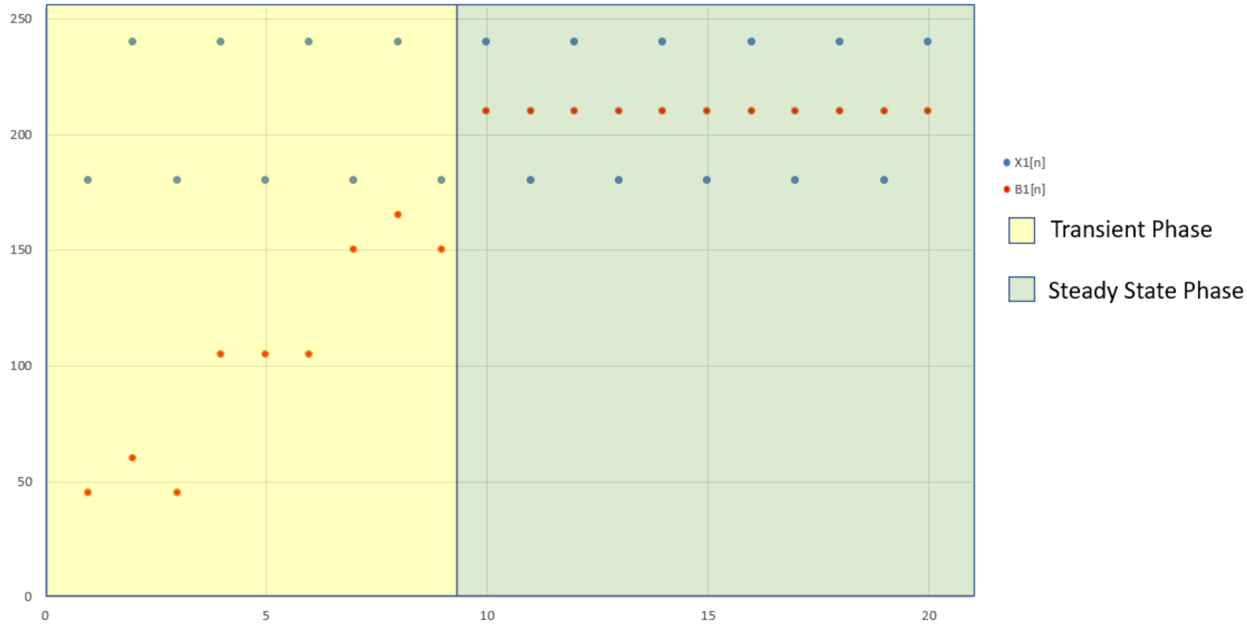


Figure 26: Average moving filter $B_1[n]$

Implementing $B_2[n]$

Table 12 shows the obtained transient and Steady State values of $B_2[n]$

$\boldsymbol{n}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**…**
$\boldsymbol{x_2[n]}$	180	240	200	244	180	240	200	244	180	240	200	244	…
$\boldsymbol{B_2[n]}$	45	60	50	106	105	110	156	166	155	216	216	216	…

Table 12: Values of $B_2[n]$

Figure 27 shows all 9 transient values and the period values of $B_2[n]$

ResU	0x00	0	00000000
ResL	0x2D	45	00101101
ResU	0x00	0	00000000
ResL	0x3C	60	00111100
ResU	0x00	0	00000000
ResL	0x32	50	00110010
ResU	0x00	0	00000000
ResL	0x6A	106	01101010
ResU	0x00	0	00000000
ResL	0x69	105	01101001
ResU	0x00	0	00000000
ResL	0x6E	110	01101110
ResU	0x00	0	00000000
ResL	0x9C	156	10011100
ResU	0x00	0	00000000
ResL	0xA6	166	10100110
ResU	0x00	0	00000000
ResL	0x9B	155	10011011
ResU	0x00	0	00000000
ResL	0xD8	216	11011000

Figure 27: Transient and period values of $B_2[n]$

Figure 28 shows a plot of the behavior of this moving average filter with the transisent phase higlighted in yellow and the steady-state phase in green.

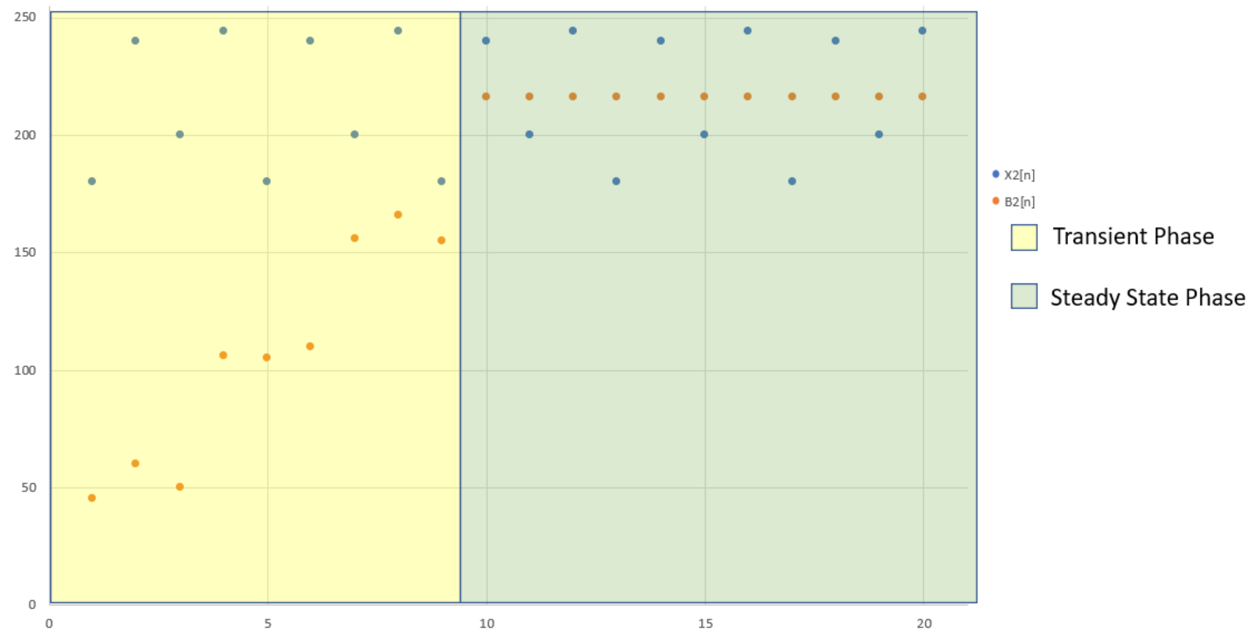


Figure 28: Average moving filter $B_2[n]$

Implementing $B_3[n]$

Table 13 shows the obtained transient and Steady State values of $B_3[n]$

$\boldsymbol{n}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**…**
$\boldsymbol{x_3[n]}$	180	240	200	244	216	236	180	240	200	244	216	236	180	…
$\boldsymbol{B_3[n]}$	45	60	50	106	114	109	151	174	159	212	228	218	212	…

Table 13: Values of $B_3[n]$

Figure 29 shows all 9 transient values and the period values of $B_3[n]$

ResU	0x00	0	00000000
ResL	0x2D	45	00101101
ResU	0x00	0	00000000
ResL	0x3C	60	00111100
ResU	0x00	0	00000000
ResL	0x32	50	00110010
ResU	0x00	0	00000000
ResL	0x6A	106	01101010
ResU	0x00	0	00000000
ResL	0x72	114	01110010
ResU	0x00	0	00000000
ResL	0x6D	109	01101101
ResU	0x00	0	00000000
ResL	0x97	151	10010111
ResU	0x00	0	00000000
ResL	0xAE	174	10101110
ResU	0x00	0	00000000
ResL	0x9F	159	10011111
ResU	0x00	0	00000000
ResL	0xD4	212	11010100
ResU	0x00	0	00000000
ResL	0xE4	228	11100100
ResU	0x00	0	00000000
ResL	0xDA	218	11011010

Figure 29: Transient and period values of $B_3[n]$

Figure 30 shows a plot of the behavior of this moving average filter with the transisent phase higlighted in yellow and the steady-state phase in green.

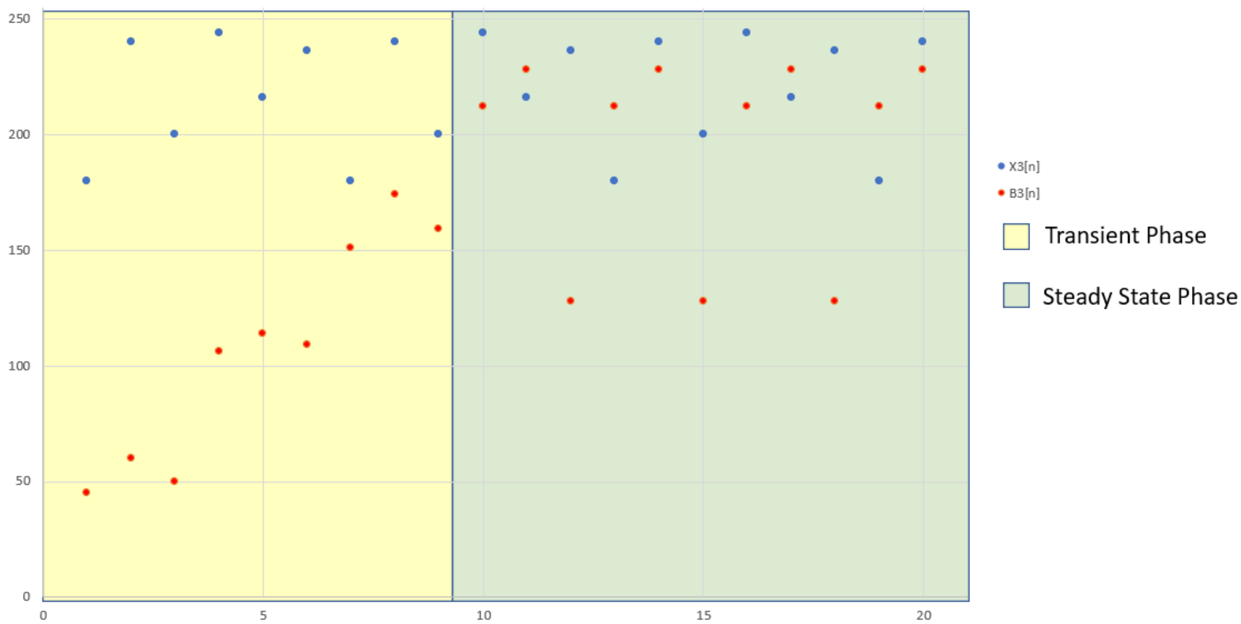


Figure 30: Average moving filter $B_3[n]$

Implementing $B_4[n]$

Table 14 shows the obtained transient and Steady State values of $B_4[n]$

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	…
$x_4[n]$	180	240	200	244	216	236	160	176	180	240	200	244	216	236	160	…
$B_4[n]$	45	60	50	106	114	109	146	158	154	206	208	215	215	207	205	…

Table 14: Values of $B_4[n]$

Figure 31 shows all 9 transient values and the period values of $B_4[n]$

ResU	0x00	0	00000000
ResL	0x2D	45	00101101
ResU	0x00	0	00000000
ResL	0x3C	60	00111100
ResU	0x00	0	00000000
ResL	0x32	50	00110010
ResU	0x00	0	00000000
ResL	0x6A	106	01101010
ResU	0x00	0	00000000
ResL	0x72	114	01110010
ResU	0x00	0	00000000
ResL	0x6D	109	01101101
ResU	0x00	0	00000000
ResL	0x92	146	10010010
ResU	0x00	0	00000000
ResL	0x9E	158	10011110
ResU	0x00	0	00000000
ResL	0x9A	154	10011010
ResU	0x00	0	00000000
ResL	0xCE	206	11001110
ResU	0x00	0	00000000
ResL	0xD0	208	11010000
ResU	0x00	0	00000000
ResL	0xD7	215	11010111
ResU	0x00	0	00000000
ResL	0xD7	215	11010111

ResU	0x00	0	00000000
ResL	0xCF	207	11001111

ResU	0x00	0	00000000
ResL	0xCD	205	11001101

Figure 31: Transient and period values of $B_4[n]$

Figure 32 shows a plot of the behavior of this moving average filter with the transisent phase higlighted in yellow and the steady-state phase in green.

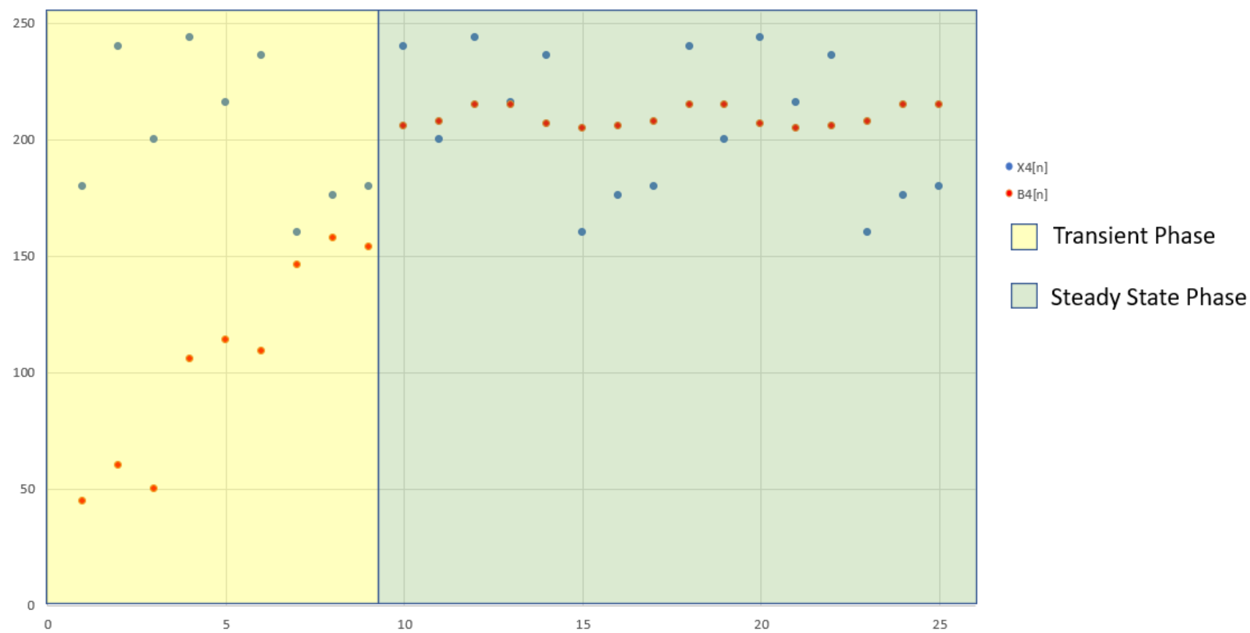


Figure 32: Average moving filter $B_4[n]$

Task 5

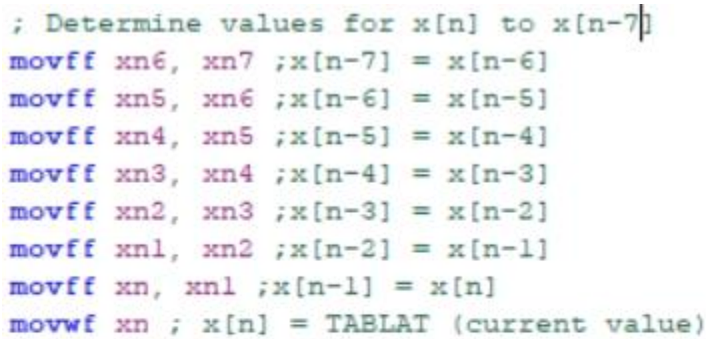
In this final task, we wish to implement the following time series:

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

Where i, j, and k, are integer values less than 15. For this time series, let $i = 2, j = 5, k = 7$ such that $C[n]$ now becomes:

$$C[n] = \frac{x[n] + x[n-2] + x[n-5] + x[n-7]}{4}$$

The modified memory buffer is shown in figure 33 below.



```
; Determine values for x[n] to x[n-7]
movff xn6, xn7 ;x[n-7] = x[n-6]
movff xn5, xn6 ;x[n-6] = x[n-5]
movff xn4, xn5 ;x[n-5] = x[n-4]
movff xn3, xn4 ;x[n-4] = x[n-3]
movff xn2, xn3 ;x[n-3] = x[n-2]
movff xn1, xn2 ;x[n-2] = x[n-1]
movff xn, xn1 ;x[n-1] = x[n]
movwf xn ; x[n] = TABLAT (current value)
```

Figure 33: Memory Buffer for $C_i[n]$

Implementing $C_1[n]$

Table 15 shows the obtained transient and Steady State values of $C_1[n]$

n	1	2	3	4	5	6	7	8	9	10	11	12	…
$x_1[n]$	180	240	180	240	180	240	180	240	180	240	180	240	…
$C_1[n]$	45	60	90	120	90	165	150	210	210	210	210	210	…

Table 15: Values of $C_1[n]$

Figure 34 shows all 7 transient values and the period values of $C_1[n]$

ResU	0x00	0	00000000
ResL	0x2D	45	00101101
ResU	0x00	0	00000000
ResL	0x3C	60	00111100
ResU	0x00	0	00000000
ResL	0x5A	90	01011010
ResU	0x00	0	00000000
ResL	0x78	120	01111000
ResU	0x00	0	00000000
ResL	0x5A	90	01011010
ResU	0x00	0	00000000
ResL	0xA5	165	10100101
ResU	0x00	0	00000000
ResL	0x96	150	10010110
ResU	0x00	0	00000000
ResL	0xD2	210	11010010

Figure 34: Transient and period values of $C_1[n]$

Figure 35 shows a plot of the behavior of this moving average filter with the transisent phase higlighted in yellow and the steady-state phase in green.

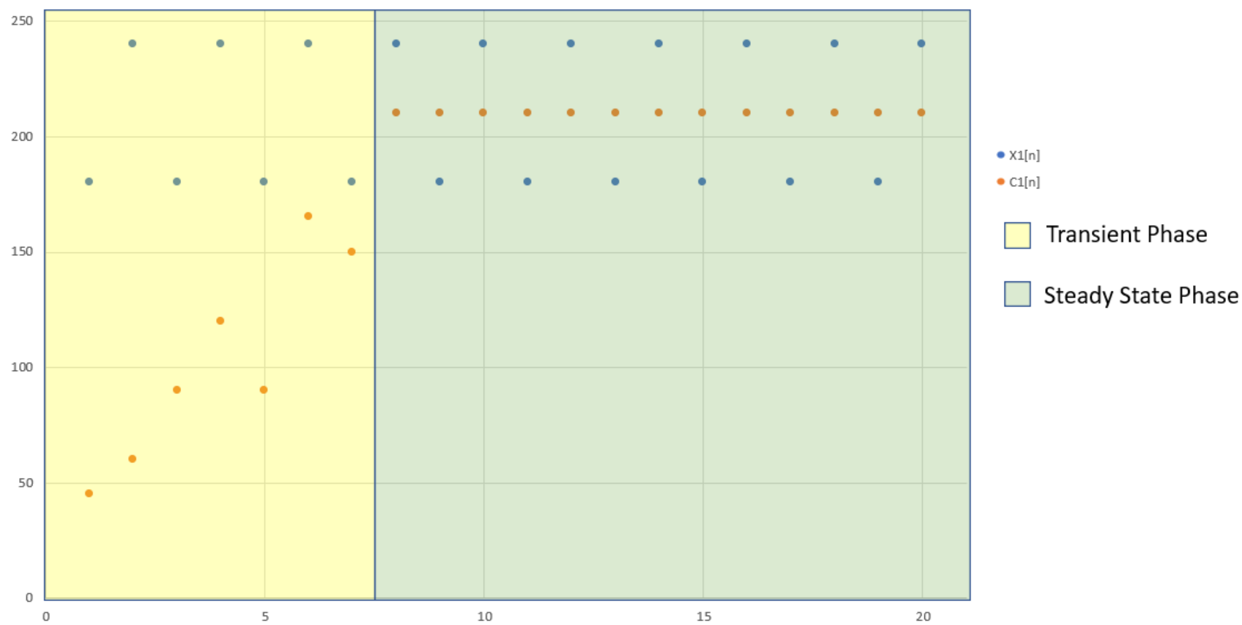


Figure 35: Average moving filter $C_1[n]$

Implementing $C_2[n]$

Table 16 shows the obtained transient and Steady State values of $C_2[n]$

n	1	2	3	4	5	6	7	8	9	10	11	12	…
$x_2[n]$	180	240	200	244	180	240	200	244	180	240	200	244	…
$C_2[n]$	45	60	95	121	95	166	155	216	216	216	216	216	…

Table 16: Values of $C_2[n]$

Figure 36 shows all 7 transient values and the period values of $C_2[n]$

ResU	0x00	0	00000000
ResL	0x2D	45	00101101
ResU	0x00	0	00000000
ResL	0x3C	60	00111100
ResU	0x00	0	00000000
ResL	0x5F	95	01011111
ResU	0x00	0	00000000
ResL	0x79	121	01111001
ResU	0x00	0	00000000
ResL	0x5F	95	01011111
ResU	0x00	0	00000000
ResL	0xA6	166	10100110
ResU	0x00	0	00000000
ResL	0x9B	155	10011011
ResU	0x00	0	00000000
ResL	0xD8	216	11011000

Figure 36: Transient and period values of $C_2[n]$

Figure 37 shows a plot of the behavior of this moving average filter with the transisent phase higlighted in yellow and the steady-state phase in green.

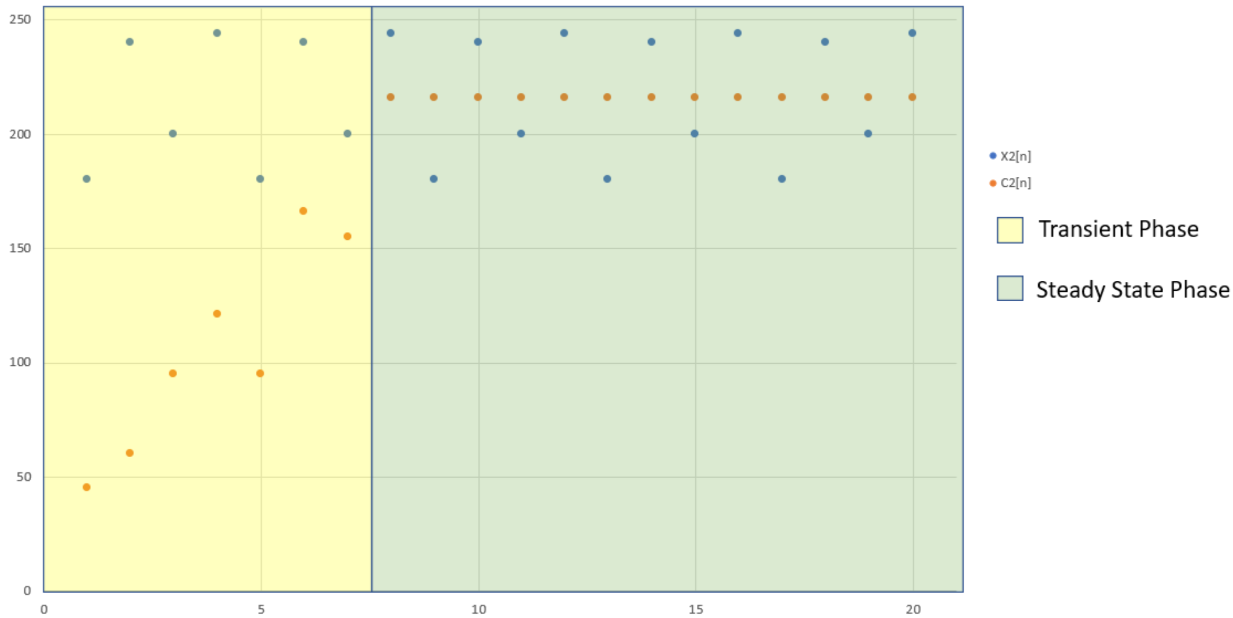


Figure 37: Average moving filter $C_2[n]$

Implementing $C_3[n]$

Table 17 shows the obtained transient and Steady State values of $C_3[n]$

$\boldsymbol{n}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**…**
$\boldsymbol{x_3[n]}$	180	240	200	244	216	236	180	240	200	244	216	236	180	…
$\boldsymbol{B_3[n]}$	45	60	95	121	201	165	159	214	216	225	224	219	218	…

Table 17: Values of $C_3[n]$

Figure 38 shows all 7 transient values and the period values of $C_3[n]$

ResU	0x00	0	00000000
ResL	0x2D	45	00101101
ResU	0x00	0	00000000
ResL	0x3C	60	00111100
ResU	0x00	0	00000000
ResL	0x5F	95	01011111
ResU	0x00	0	00000000
ResL	0x79	121	01111001
ResU	0x00	0	00000000
ResL	0x68	104	01101000
ResU	0x00	0	00000000
ResL	0xA5	165	10100101
ResU	0x00	0	00000000
ResL	0x9F	159	10011111
ResU	0x00	0	00000000
ResL	0xD6	214	11010110
ResU	0x00	0	00000000
ResL	0xD8	216	11011000
ResU	0x00	0	00000000
ResL	0xE1	225	11100001
ResU	0x00	0	00000000
ResL	0xE0	224	11100000
ResU	0x00	0	00000000
ResL	0xDB	219	11011011
ResU	0x00	0	00000000
ResL	0xDA	218	11011010

Figure 38: Transient and period values of $C_3[n]$

Figure 39 shows a plot of the behavior of this moving average filter with the transisent phase higlighted in yellow and the steady-state phase in green.

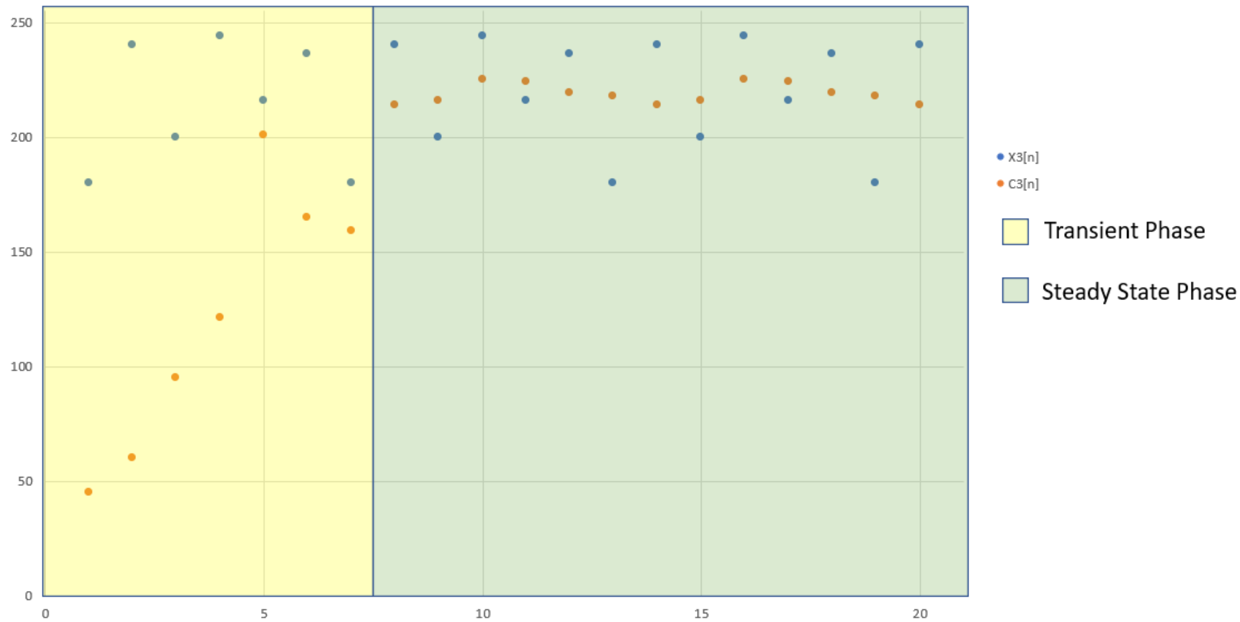


Figure 39: Average moving filter $C_2[n]$

Implementing $C_4[n]$

Table 18 shows the obtained transient and Steady State values of $C_4[n]$

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	…
$x_4[n]$	180	240	200	244	216	236	160	176	180	240	200	244	216	236	160	…
$C_4[n]$	45	60	95	121	104	165	154	198	206	208	215	215	207	205	198	…

Table 18: Values of $C_4[n]$

Figure 40 shows all 7 transient values and the period values of $C_4[n]$

ResU	0x00	0	00000000
ResL	0x2D	45	00101101
ResU	0x00	0	00000000
ResL	0x3C	60	00111100
ResU	0x00	0	00000000
ResL	0x5F	95	01011111
ResU	0x00	0	00000000
ResL	0x79	121	01111001
ResU	0x00	0	00000000
ResL	0x68	104	01101000
ResU	0x00	0	00000000
ResL	0xA5	165	10100101
ResU	0x00	0	00000000
ResL	0x9A	154	10011010
ResU	0x00	0	00000000
ResL	0xC6	198	11000110
ResU	0x00	0	00000000
ResL	0xCE	206	11001110
ResU	0x00	0	00000000
ResL	0xD0	208	11010000
ResU	0x00	0	00000000
ResL	0xD7	215	11010111
ResU	0x00	0	00000000
ResL	0xD7	215	11010111
ResU	0x00	0	00000000
ResL	0xCF	207	11001111
ResU	0x00	0	00000000
ResL	0xCD	205	11001101

ResU	0x00	0	00000000
ResL	0xC6	198	11000110

Figure 40: Transient and period values of $C_4[n]$

Figure 41 shows a plot of the behavior of this moving average filter with the transisent phase higlighted in yellow and the steady-state phase in green.

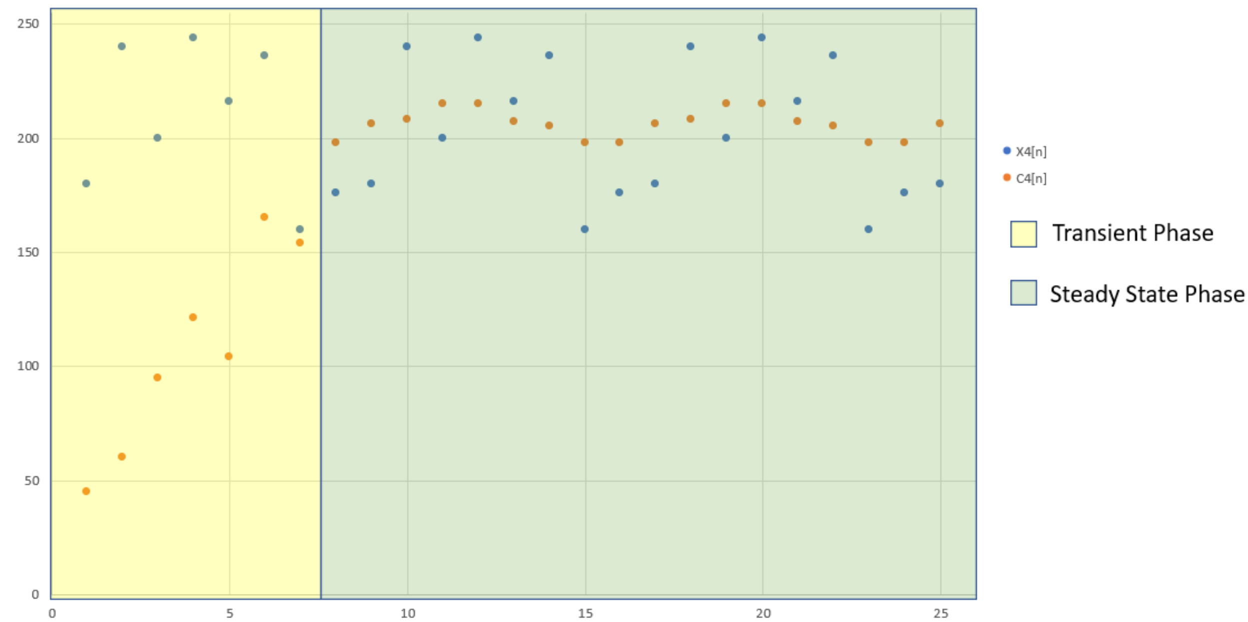


Figure 41: Average moving filter $C_4[n]$