EE 313 Linear Signals & Systems (Fall 2018)

***Solution Set for Homework #4 on Finite Impulse Response (FIR) Filter***

By: Houshang Salimian & Prof. Brian L. Evans

**Problem 1:**

**Solution:**

**Part a:**

**Using a binomial expansion.**

1. 
2. 









**Part b:**

1. 



1. 

**Part c:**

**i.**

We’ll derive the formula by using the result from part ii below.

because

**ii.**

Solution: We’ll derive a closed-form answer. Let’s start with a slightly different indexing for n:

We’ll reorder the addition of the terms to go from highest exponent to lowest:

The terms in parenthesis are from the result of dividing *aN*-1 by *a*-1. We’ll compute the polynomial division using long division:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | *aN-1* | *aN-2* | … | *a* | 1 |
| *a*-1 | *aN* | 0 | 0 | … | 0 | -1 |
| - | *aN* | *-aN-1* |  |  |  |  |
|  | 0 | *aN-1* |  |  |  |  |
| - |  | *aN-1* | *-aN-2* |  |  |  |
|  |  | 0 | *aN-2* | … |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  | *a*2 |  |  |
| - |  |  |  | *a*2 | *-a* |  |
|  |  |  |  | 0 | *a* | -1 |
| - |  |  |  |  | *a* | -1 |
|  |  |  |  |  |  | 0 |

We can connect this summation with the form in the question:

I based the above derivation on the content at

<https://www.purplemath.com/modules/series7.htm>

**Second solution:**

**Let assume:**



By multiplying “a” to both sides of this equation:













**Problem 2:**

**Solution:**

**Part a:**

  
The values for x[n] and y[n] are given in the following table.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| n | < 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| x[n] | 0 | 1 | 2 | 3 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| x[n-1] | 0 | 0 | 1 | 2 | 3 | 2 | 1 | 1 | 1 | 1 | 1 | 1 |
| x[n-2] | 0 | 0 | 0 | 1 | 2 | 3 | 2 | 1 | 1 | 1 | 1 | 1 |
| y[n] | 0 | 2 | 1 | 2 | -1 | 2 | 3 | 1 | 1 | 1 | 1 | 1 |

**Part b:**





**Part c:**





|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| n | < 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| δ[n] | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| δ[n-1] | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| δ[n-2] | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| h[n] | 0 | 2 | -3 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Part d:**





**Problem 3:**

**Prologue:** This problem introduces the convolution sum, and asks you to calculate it for a simple finite impulse response filter (*L*-point averaging filter) given an infinitely long input signal. The unit step signal models a physical action such as turning on a switch and leaving it on indefinitely. In discrete time, the unit step function *u*[*n*] is zero in amplitude for *n* < 0, and one in amplitude for *n* ≥ 0.

**Part a:**

The MATLAB function stepfun(n, n0) implements *u*[*n-n0*] and is plotted on the right:

Unit-step signal turns on at n=0 so



MATLAB code:

n = -4:6;

u = stepfun(n,0);

stem(n,u)

xlabel('n')

ylabel('u[n]')

ylim([-0.5 1.5])



**Part b:**

**MATLAB Code:**

n = -4:6;

u = stepfun(n,0);

x = (0.5.^n).\*u;

stem(n,x)

xlabel('n')

ylabel('x[n]')

ylim([-0.5 1.5])



**Part c:**

In order to calculate y[n], the value of x[n] should be calculated for different parts of discrete-time range.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| n | X[n] | X[n-1] | X[n-2] | X[n-3] | Y[n] |
| -5 | 0 | 0 | 0 | 0 | 0 |
| -4 | 0 | 0 | 0 | 0 | 0 |
| -3 | 0 | 0 | 0 | 0 | 0 |
| -2 | 0 | 0 | 0 | 0 | 0 |
| -1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1/4 |
| 1 | 1/2 | 1 | 0 | 0 | 3/8 |
| 2 | 1/4 | 1/2 | 1 | 0 | 7/16 |
| 3 | 1/8 | 1/4 | 1/2 | 1 | 15/32 |
| 4 | 1/16 | 1/8 | 1/4 | 1/2 | 15/64 |
| 5 | 1/32 | 1/16 | 1/8 | 1/4 | 15/128 |
| 6 | 1/64 | 1/32 | 1/16 | 1/8 | 15/256 |
| 7 | 1/128 | 1/64 | 1/32 | 1/16 | 15/512 |
| 8 | 1/256 | 1/128 | 1/64 | 1/32 | 15/1024 |
| 9 | 1/512 | 1/256 | 1/128 | 1/64 | 15/2048 |
| 10 | 1/1024 | 1/512 | 1/256 | 1/128 | 15/4096 |

**Part d:**





**Problem 4-**

Prologue: For a discrete-time finite impulse response (FIR) filter with M+1 coefficients, the values of the coefficients are equal to the impulse response h[n].  Given input x[n], the output y[n] is given by



This formula determines y[n] by computing the discrete-time convolution of x[n] and h[n].

**Deconvolution**attempts to determine h[n] when knowing the input x[n] and the output y[n].

**Application**.  An application is in determining the acoustic response of a concert hall.  One places an audio speaker on stage and a microphone at one of the seats at head height.  A laptop controls the discrete-time signal being played over the audio speaker x[n] and records the output of the microphone in discrete-time as y[n].  The values computed for h[n] give a model for the acoustic response of the room.  That is, given an audio signal x[n], we can compute what a person in the concert hall would hear by convolving h[n] and x[n].  This emulation of a concert hall is available on certain audio playback systems.

**Approach**.  There are many methods for deconvolution, i.e. determining h[n] when knowing the input x[n] and the output y[n].  The method below uses the convolution formula for an FIR filter to compute the impulse response h[n]:



Assuming that h[n] and x[n] are causal signals, i.e. their amplitude values are zero when n < 0, the formula for the first output sample y[0] gives us one equation in one unknown h[0] because we know the values of x[0] and y[0]:



We then solve for h[0], which works as long as x[0] is not zero.  The next output sample gives us one equation in one unknown h[1]:



We then solve for h[1], which works as long as x[0] is not zero.

**Solution:**

**Part a:**



Using the formula on prologue, the value of h[n] can be calculated.



So the value of h[0] can be calculated as: h[0] = 0/1 = 0





|  |  |  |  |
| --- | --- | --- | --- |
| n | x[n] | y[n] | h[n] |
| 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |
| 2 | 1 | 1 | 0 |

We've stopped calculating values for h[n] to see if we've finished.  We can now compute the convolution of h[n] and x[n] = u[n] to see if we get y[n] = u[n-1]



Now, if we place x[n] = δ[n], the output of system is y[n] = h[n]



**Part b:**





|  |  |  |  |
| --- | --- | --- | --- |
| n | x[n] | y[n] | h[n] |
| 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | -1 |
| 2 | 1 | 0 | 0 |

We've stopped calculating values for h[n] to see if we've finished.  We can now compute the convolution of h[n] and x[n] = u[n] to see if we get y[n] = δ[n]



Now, if we place x[n] = δ[n], the output of system is y[n] = h[n]



**Part c:**





|  |  |  |  |
| --- | --- | --- | --- |
| n | x[n] | y[n] | h[n] |
| 0 | 1 | 0 | 0 |
| 1 | 1/2 | 1 | 1 |
| 2 | 1/4 | 0 | -0.5 |
| 3 | 1/8 | 0 | 0 |

We've stopped calculating values for h[n] to see if we've finished.  We can now compute the convolution of h[n] and  to see if we get y[n] = δ[n-1]



Now, if we place x[n] = δ[n], the output of system is y[n] = h[n]

