**PRACTICE 2 – LINEAR TIME-INVARIANT (LTI) DISCRETE -TIME SYSTEMS**

**OBJECTIVES:**

1. Define basic systems.
2. Check LTI systems.
3. Obtain the impulse response of LTI systems.
4. The convolution sum.
5. Causal LTI systems defined as Linear Difference Equations with constant coefficients.

|  |  |
| --- | --- |
| **Name 1:** | Alejandro Serrano López |
| **Name 2:** | Mihnea-Iulian Popescu |

1. **Basic systems**

Define the following basic systems:

* Ideal delay system:

function[y,m] = delay(x,n,n0) % x is the input signal and n0 is the delay.

|  |
| --- |
| **delay.m** |
| function[y,m] = delay(x,n,n0) % x is the input signal and n0 is the delay  arguments  x  n  n0  end    if n0 == 0  y = x;  m = n;  else  z(abs(n0)) = 0;  if n0 > 0  m = (n(1)):(n(end)+n0);  y = [z x];  else  m = (n(1)+n0):(n(end));  y = [x z];  end  end  end |

* Accumulator system:

function [y,m] = accum(x,n) % x is the input signal.

|  |
| --- |
| **accum.m** |
| function [y,m] = accum(x,n) % x is the input signal.  arguments  x  n  end  m = n;  y = cumsum(x(:));  end |

* Backward difference system:

function [y,m] = bdiff(x,n,ac) % x is the input signal and ac is the auxiliary condition x[-1].

|  |
| --- |
| **bdiff.m** |
| function [y,m] = bdiff(x,n,ac) % x is the input signal and ac is the auxiliary condition x[-1].  arguments  x  n  ac  end  m = n;  y(length(x)) = 0;  y(1) = x(1)-ac;  for a=2:length(x)  y(a) = x(a)-x(a-1);  end  end |

* Moving average system:

function [y,m] = sma(x,n,M,ac) % x is the input sequence and M+1 is the number of samples averaged and ac are the M auxiliary conditions.

|  |
| --- |
| **sma.m** |
| function [y,m] = sma(x,n,M,ac) % x is the input sequence and M+1 is the number of samples averaged and ac are the M auxiliary conditions.  arguments  x  n  M  ac  end  m = n;  y = zeros(1, length(x));  aux = [ac x];  if M > length(ac)  M = length(ac)  end  for a=1:length(x)  y(a) = sum(aux(a-M+length(ac):a+length(ac)))/(M+1);  end  end |

OPTIONAL

------------------------------------------------------------------------------------------------------------------------------------

* Compressor system:

function [y,m] = comp(x,n,F) % x is the input sequence and F is the compression factor.

|  |
| --- |
| **comp.m** |
| function [y,m] = comp(x,n,F) % x is the input sequence and F is the compression factor.  arguments  x  n  F  end  m = (floor(n(1)/F)):(floor(n(end)/F));  startPos = 1;  if n(1) < 0 && n(end) >= 0  startPos = find(n==0);  end  a1= 1;  aux = zeros(1, length(m));  %y(round(startPos/F))=x(startPos);  while (startPos - (F \* a1)) >= 1  aux(a1) = x(-F\*a1 + startPos);  %y(round(startPos/F) + a) = x(F\*a + startPos);  a1 = a1 +1;  end  ladoDerecho = fliplr(aux(1:a1-1));  a1 = 1;  aux2 = zeros(1, length(m));  while (startPos + F \* a1) <= length(x)  aux2(a1) = x(F\*a1 + startPos);  %y(round(startPos/F) + a) = x(F \* a + startPos);  a1 = a1 + 1;  end  ladoIzquierdo = aux2(1:a1-1);  y = [ladoDerecho x(startPos) ladoIzquierdo];    stem(m, y)  end |

------------------------------------------------------------------------------------------------------------------------------------

1. **LTI discrete-time systems.**

Check graphically if the above systems are time-invariant. To do so:

1. Create a rectangular pulse x[n] as the input of the systems. Use: [x,n] = rectan (7,12,0,20)

NOTE: rectan is not available, we used Rectangle from the previous practice instead:

>> [x,n] = Rectangle (7, 12, 0, 20);

Gráfico, Histograma

Descripción generada automáticamente

1. Calculate the output y[n] of every system.
2. Calculate the new input x[n-3].

>> [xdelay, ndelay] = delay(x,n,3)

1. Calculate the new output y’[n] of every system.
2. Plot for every system: x[n] vs x[n-3], y[n] vs y’[n]. (Use the subplot instruction)

* Accumulator system:

|  |  |
| --- | --- |
| **Subplot x[n] vs x[n-3]** | **Subplot y[n] vs y’[n]** |
| Gráfico, Histograma  Descripción generada automáticamente | >> [y,m] = accum (x,n);  >> subplot(2,1,1)  >> stem(m,y)  >> subplot(2,1,2)  >> [y2,m2] = accum (xdelay,ndelay);  >> stem(m2,y2)  Gráfico, Histograma  Descripción generada automáticamente |

* Backward difference system:

|  |  |
| --- | --- |
| **Subplot x[n] vs x[n-3]** | **Subplot y[n] vs y’[n]** |
| Gráfico, Histograma  Descripción generada automáticamente | >> [y,m] = bdiff (x,n,0);  >> subplot(2,1,1)  >> stem(m,y)  >> subplot(2,1,2)  >> [y2,m2] = bdiff (xdelay,ndelay,0);  >> stem(m2,y2)  Gráfico, Gráfico de líneas  Descripción generada automáticamente |

* Moving average system:

|  |  |
| --- | --- |
| **Subplot x[n] vs x[n-3]** | **Subplot y[n] vs y’[n]** |
| Gráfico, Histograma  Descripción generada automáticamente | >> [y,m] = sma(x,n,3, [0, 0, 0]);  >> subplot(2,1,1)  >> stem(m,y)  >> subplot(2,1,2)  >> [y2,m2] = sma(xdelay,ndelay,3, [0, 0, 0]);  >> stem(m2,y2)  Gráfico  Descripción generada automáticamente |

OPTIONAL

------------------------------------------------------------------------------------------------------------------------------------

* Compressor system:

The compressor is not time-invariant

|  |  |
| --- | --- |
| **Subplot x[n] vs x[n-3]** | **Subplot y[n] vs y’[n]** |
| Gráfico, Histograma  Descripción generada automáticamente | >> [y,m] = comp(x,n,2);  >> subplot(2,1,1)  >> stem(m,y)  >> subplot(2,1,2)  >> [y2,m2] = comp(xdelay,ndelay,2);  >> stem(m2,y2) |

------------------------------------------------------------------------------------------------------------------------------------

1. **Impulse response.**

Obtain graphically the impulse response of the previous systems (if possible). To do so:

1. Create the unit impulse [n] as the input of the systems. Use: [d,n]=delta(0,0,20)

>> [ddelta,ndelta]=Delta(0,0,20)

1. Plot the output y[n]=h[n] of every LTI system.

|  |  |
| --- | --- |
| **Ideal delay system h1[n]** | **Accumulator system h2[n]** |
| >> [ h1, m1 ] = delay(ddelta, ndelta, 3);  >> stem(m1,h1);  Gráfico, Histograma  Descripción generada automáticamente | >> [ h2, m2 ] = accum(ddelta, ndelta);  >> stem(m2,h2);  Gráfico, Histograma  Descripción generada automáticamente |

|  |  |
| --- | --- |
| **Backward difference system h3[n]** | **Moving average system h4[n]** |
| >> [ h3, m3 ] = bdiff(ddelta, ndelta, 0);  >> stem(m3,h3);  Gráfico  Descripción generada automáticamente | >> [ h4, m4 ] = sma(ddelta, ndelta, 3, [0, 0, 0]);  >> stem(m4,h4);  Gráfico, Histograma  Descripción generada automáticamente |

Considering the results of the h[n]. What can you say about causality and stability of the systems?

|  |
| --- |
| Causality moves the impulse response h(n) to the right as many times it goes back (f.ex. if it’s delayed by 3 or needs information from 3 steps ago it moves the response 3 units to the right), if the impulse response is on the negatives we know it depends on future events/states.  We also know that if for a bounded input, ∑|h(n) | < ∞ then the system is inherently stable, else having an specific signal inputting forever will cause a system instability.  From this, we can conclude that:  The 4 systems measured are causal. The delay requires the values n0 steps back from the present, the Backward difference system only needs the immediately past input, the moving average uses the previous M values apart from the present one, while the accumulator needs all the inputs.  Regarding stability, delay and moving average system are stable, backward difference specially; while the accumulator is unstable if receiving a constant input (f.ex. a simple step signal). |

1. **The convolution sum.**

Use the convolution sum (MATLAB function “conv”) to calculate the output of the systems to a rectangular pulse input. Use: [x,n] = rectan(7,12,0,20)

NOTE: rectan is not available, we used Rectangle from the previous practice instead:

|  |  |
| --- | --- |
| **Ideal delay system: y[n]=x[n]h1[n]** | **Accumulator system: y[n]=x[n]h2[n]** |
| >> yconv1 = conv(x, h1);  >> stem(0:43, yconv1);  Gráfico, Histograma  Descripción generada automáticamente | >> yconv2 = conv(x, h2)  >> stem(0:length(yconv2)-1, yconv2)  NOTE: Due to how the convolution gives a longer-than-n response and how the delta function given only considers delta for an specific finite n, any values past that n will not be valid if the h(n) is not finite (such as the impulse response to the accumulator).  Since we did it for n = 20, we can see it behaves exactly like the accumulator system up to that point.  Gráfico  Descripción generada automáticamente |

|  |  |
| --- | --- |
| **Backward difference system: y[n]=x[n]h3[n]** | **Moving average system: y[n]=x[n]h4[n]** |
| >> yconv3 = conv(x, h3);  >> stem(0:length(yconv3)-1, yconv3)  Gráfico  Descripción generada automáticamente | >> yconv4 = conv(x, h4);  >> stem(0:length(yconv4)-1, yconv4)  Gráfico  Descripción generada automáticamente |

Compare these outputs with the results obtained in section 2. What can you say about them?

|  |
| --- |
| On a Linear, Time-Invariant system, Y = conv(x, h) is the same as calculating [y, m] = systemToFilter(x, n) |

1. **Causal LTI discrete-time systems defined as Linear Difference Equations with constant coefficients (LDEcc).**

Check graphically that the accumulator system can be implemented as a LDEcc with initial-rest conditions.

**Initial-rest conditions**: if the input x[n] is zero for n less than some time , then the output y[n] is constrained to be zero for n less than .

To do so:

1. Write a new MATLAB function for this implementation.

* Accumulator system (Forward difference system):

function [y,m] = acc\_LDE(x,n,ac) % x is the input sequence starting at n0=n(1) and ac is the auxiliary condition y[n0-1].

|  |
| --- |
| **acc\_LDE.m** |
| function [y,m] = acc\_LDE(x,n,ac) % x is the input sequence starting at n0=n(1) and ac is the auxiliary condition y[n0-1].  y(1) = ac + x(1);  for i=2:length(n)  y(i) = y(i-1) + x(i);  end  m = n;  stem(m,y)  end |

1. Create a rectangular pulse x[n] as the input of the systems. Use: [x,n] = rectan (7,12,0,20)
2. Calculate the output y1[n] of the system. Use: [y1,m1] = acc\_LDE(x,n,ac)
3. Compare graphically the result with section 2 (use subplot). Use: [y2,m2] = accum(x,n)

|  |  |
| --- | --- |
| **Instructions** | **Subplot y1[n] vs y2[n]** |
| >> [x,n] = Rectangle(7,12,0,20);  >> [y1,m1] = acc\_LDE(x,n,0);  >> subplot(2,1,1);  >> [y1,m1] = acc\_LDE(x,n,0);  >> subplot(2,1,2);  >> [y2,m2] = accum(x,n); | Gráfico  Descripción generada automáticamente |

(OPTIONAL: What happens if the auxiliary condition y[-1] is not zero (non-initial-rest condition)?)

|  |
| --- |
| >> [x,n] = Rectangle(7,12,0,20);  >> [y1,m1] = acc\_LDE(x,n,0);  >> subplot(2,1,1);  >> [y1,m1] = acc\_LDE(x,n,0);  >> subplot(2,1,2);  >> [y1v2,m1v2] = acc\_LDE(x,n,-2)  Gráfico  Descripción generada automáticamente  It will shift the y value by the initial condition in all the regions the function exists, since the initial conditions are not zero, the output is not constrained to zero, either. On this case the initial position counts before a certain n0 |