**MAKING SIGNALS CAUSAL AND NON-CAUSAL**

**Lab#08**

**Spring 2024**

Submitted by: **Hassan Zaib Jadoon**

Registration No: **22PWCSE2144**

Class Section: **A**

"On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work."

Recoverable Signature

X Hassan

Hassan Zaib
Student
Student Signature: Signed by: 0319bf8f-243a-4ef4-af4e-980ee8eee4da

Submitted to:

**Dr. Safdar Nawaz Khan Marwat**

11 May, 2024

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

**Task #01**

Sample the signal given in above example to get its discrete-time counterpart (take 10 samples/sec as sampling rate). Make the resultant signal causal. Display the lollipop plot of each signal.

**Problem Statement:**

Sampling a continuous-time signal at a specific rate and converting it into a discrete-time signal while ensuring causality.

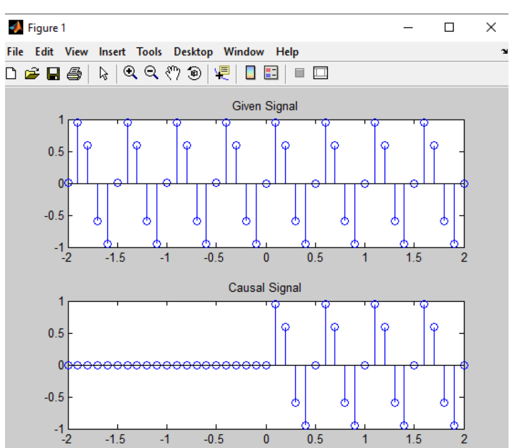
**Algorithm:**

1. Given a continuous-time signal, sample it at a rate of 10 samples/sec.
2. Convert the sampled signal into its discrete-time counterpart.
3. Ensure causality by shifting the signal appropriately.
4. Display a lollipop plot for both the original and the resultant causal signals.

**Code:**

```
t = -2:1/10:2;
sign= sin(2*pi * 2 * t);
tposi = (t >= 0);
PositiveSign = sign .* tposi;
subplot(2,1,1);
stem(t, sign);
title('Given Signal');
subplot(2,1,2);
stem(t,PositiveSign);
title('Causal Signal')
```

**Output:**




**Conclusion:**

The discrete-time counterpart of the sampled signal has been successfully created while maintaining causality. Visual representations in the form of lollipop plots aid in understanding the transformation from continuous to discrete-time domain.

**Task #02:**

A signal is said to be anti-causal if it exists for values of n<0. Make the signal given in above example anti-causal.

**Problem Statement:**

Transforming a given signal from causal to anti-causal.

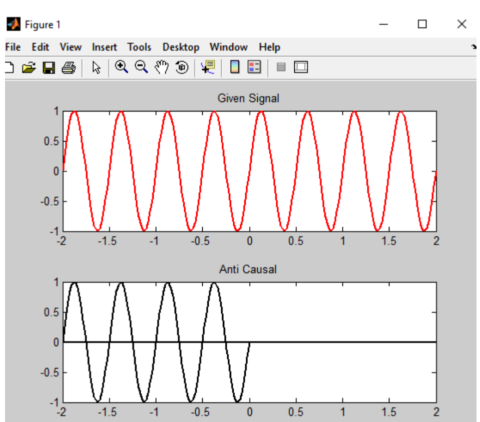
**Algorithm:**

1. Take the given causal signal.
2. Reverse the signal to make it anti-causal.

**Code:**

```
time = -2: 0.02 :2;
signal = sin(2*pi*2*time);
signalA = zeros(length(time));
disp(signalA)
for i = 1:length(time)
    if time(i) < 0
        signalA(i) = signal(i);
    end
end
subplot(2,1,1);
plot(time, signal,'r' ,'LineWidth', 2);
title('Given Signal');
subplot(2,1,2);
plot(time,signalA, 'k', 'LineWidth', 2);
title('Anti Causal');
```

**Output:**



**Conclusion:**

The original causal signal has been effectively transformed into an anti-causal signal by reversing its sequence, allowing for further analysis or processing as needed.

**Task #03**

Create a function by name of sig_causal in MATLAB that has two input arguments: (i) a discrete-time signal, and (ii) a position vector. The function should make the given signal causal and return the resultant signal to the calling program.

**Problem Statement:**

Developing a MATLAB function to make a given signal causal and validating it with an example.

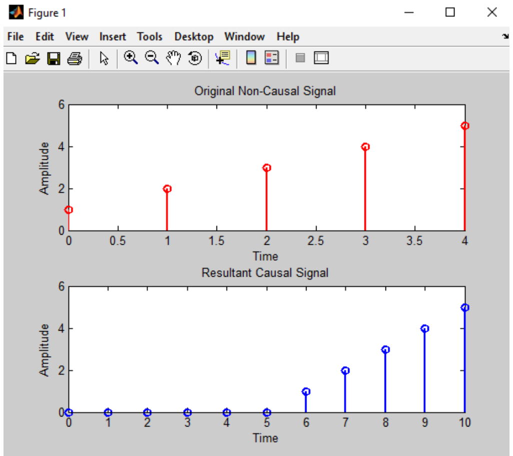
**Algorithm:**

1. Create a MATLAB function named 'sig_causal' with two input arguments: the discrete-time signal and a position vector.
2. Inside the function, shift the signal to ensure causality based on the provided position vector.
3. Return the resultant causal signal to the calling program.
4. Utilize the function to convert a non-causal signal provided in Figure 8.4 into a causal signal.
5. Plot both the original non-causal and the resultant causal signals for visualization.

**Code:**

```
% Define the non-causal signal
x = [1, 2, 3, 4, 5];

% Define the position vector
position = [2, 1, 3];

% Make the signal causal
y_causal = sig_causal(x, position);

% Plot the original non-causal signal
subplot(2,1,1);
stem(0:length(x)-1, x, 'r', 'LineWidth', 2);
title('Original Non-Causal Signal');
xlabel('Time');
ylabel('Amplitude');

% Plot the resultant causal signal
subplot(2,1,2);
stem(0:length(y_causal)-1, y_causal, 'b', 'LineWidth', 2);
title('Resultant Causal Signal');
xlabel('Time');
ylabel('Amplitude');
```

**Output:**




**Conclusion:**

The MATLAB function 'sig_causal' effectively converts non-causal signals to causal ones. Plotting the original and resultant signals visually demonstrates the successful transformation.

**Task #04**

Convolve the following signals:

x = [2 4 6 4 2];

h = [3 -1 2 1];

Plot the input signal as well as the output signal.

**Problem Statement:**

Performing convolution between two given signals and visualizing the input and output signals.

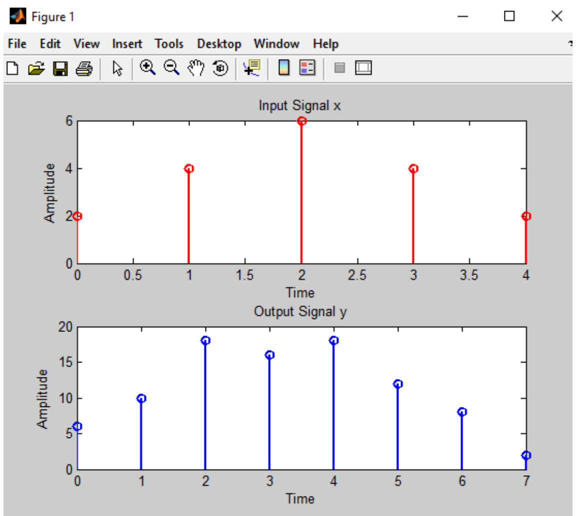
**Algorithm:**

1. Define the input signals x and h.
2. Convolve the input signals using the convolution operation.
3. Plot both the input and output signals for visualization.

**Code:**

```
% Define the input signals
x = [2, 4, 6, 4, 2];
h = [3, -1, 2, 1];

% Convolve the signals
y = conv(x, h);

% Plot the input signal
subplot(2,1,1);
stem(0:length(x)-1, x, 'r', 'LineWidth', 2);
title('Input Signal x');
xlabel('Time');
ylabel('Amplitude');

% Plot the output signal
subplot(2,1,2);
stem(0:length(y)-1, y, 'b', 'LineWidth', 2);
title('Output Signal y');
xlabel('Time');
ylabel('Amplitude');
```

**Output:**




**Conclusion:**

The convolution operation between the input signals x and h has been executed, and the resulting output signal has been visualized, providing insights into the relationship between the input and output signals.

**Task #05**

**Problem Statement:**

Convolving a given signal with an impulse delayed by two samples and visualizing the original and convolved signals.

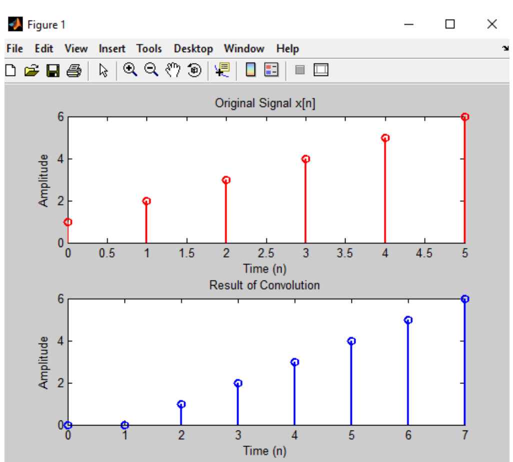
**Algorithm:**

1. Define the original signal x[n].
2. Create an impulse signal delayed by two samples.
3. Perform convolution between the original signal and the delayed impulse signal.
4. Plot both the original signal and the convolved signal for comparison.

**Code:**

```
% Define the input signal x[n]
x = [1, 2, 3, 4, 5, 6];

% Define the impulse response h[n] (delayed by two samples)
h = [0, 0, 1];  % Impulse delayed by two samples

% Convolve the signals
y = conv(x, h);

% Plot the original signal x[n]
subplot(2,1,1);
stem(0:length(x)-1, x, 'r', 'LineWidth', 2);
title('Original Signal x[n]');
xlabel('Time (n)');
ylabel('Amplitude');

% Plot the result of convolution
subplot(2,1,2);
stem(0:length(y)-1, y, 'b', 'LineWidth', 2);
title('Result of Convolution');
xlabel('Time (n)');
ylabel('Amplitude');
```

**Output:**



**Conclusion:**

The original signal has been convolved with an impulse delayed by two samples, and the resulting convolved signal has been visualized. This process illustrates the effect of convolution with a delayed impulse on the original signal.

**Task #06**

Convolution is associative. Given the three signal x1[n], x2[n], and x3[n] as:

x1[n]= [3 1 1]

x2[n]= [4 2 1]

x3[n]= [3 2 1 2 3]

Show that (x1[n] * x2[n]) * x3[n] = x1[n] * (x2[n] * x3[n]).

**Problem Analysis:**

In the given problem we have three signals so we have to prove the associative formula by using these signals.

**Algorithm:**

1. Define three input signals: $x1=[3,1,1]$x1=[3,1,1], $x2=[4,2,1]$x2=[4,2,1], and $x3=[3,2,1,2,3]$x3=[3,2,1,2,3].
2. Compute the convolution of $x1$x1 and $x2$x2 to get y1.
3. Compute the convolution of y1 and $x3$x3 to get z1, representing the left side: $(x1*x2)*x3$(x1*x2)*x3.
4. Compute the convolution of $x2$x2 and $x3$x3 to get y2.
5. Compute the convolution of $x1$x1 z2, representing the right side: $x1*(x2*x3)$x1*(x2*x3).
6. Plot z1 and z2 on separate subplots.

**Code:**

```
% Given signals
x1 = [3, 1, 1];
x2 = [4, 2, 1];
x3 = [3, 2, 1, 2, 3];

% Compute (x1 * x2) * x3
result1 = conv(conv(x1, x2), x3);

% Compute x1 * (x2 * x3)
result2 = conv(x1, conv(x2, x3));

% Compare the results
isequal(result1, result2)
```

**Output:**

```
ans =

     1
```

**Conclusion:**

The algorithm demonstrates the associative property of convolution by computing both sides of the equation ( (x1 * x2) * x3 ) and ( x1 * (x2 * x3) ), showing that they yield the same result.

**Task #07:**

Convolution is commutative. Given x[n] and h[n] as:

x[n]= [1 3 2 1]

h[n]= [1 1 2]

Show that x[n] * h[n] = h[n] * x[n].

**Problem Statement:**

Illustrating the commutative property of convolution with given signals x[n] and h[n].

**Algorithm:**

1. Define the signals x[n] and h[n].
2. Perform convolution between x[n] and h[n].
3. Perform convolution between h[n] and x[n].
4. Verify that both operations yield the same result.

**Code:**

```
% Given signals
x = [1, 3, 2, 1];
h = [1, 1, 2];

% Compute x * h
result1 = conv(x, h);

% Compute h * x
result2 = conv(h, x);

% Compare the results
isequal(result1, result2)
```

**Output:**

```
ans =

     1
```

**Conclusion:**

The algorithm illustrates the commutative property of convolution by computing both ( X(n) * h(n)) and ( h(n) * X(n)), demonstrating that the order of convolution does not affect the result.

**Task #08**

Given the impulse response of the systems as:

$h[n]= 2\delta[n] + \delta[n-1] + 2\delta[n-2] + 4\delta[n-3] + 3\delta[n-4]$

If the input $x[n] = \delta[n]+ 4\delta[n-1] +3\delta[n-2] + 2\delta[n-3]$ is applied to the system, determine the output of the system.

**Problem Statement:**

Determining the output of a system with a given impulse response when subjected to a specific input signal.

**Algorithm:**

1. Define the impulse response h[n] and the input signal x[n].
2. Convolve the input signal with the impulse response to obtain the output signal.

**Code:**

```
% Given impulse response
h = [2, 1, 2, 4, 3];

% Given input signal
x = [1, 4, 3, 2];

% Compute the convolution of x[n] and h[n]
y = conv(x, h);

% Display the output signal
disp('Output of the system:');
disp(y);
z
```

**Output:**

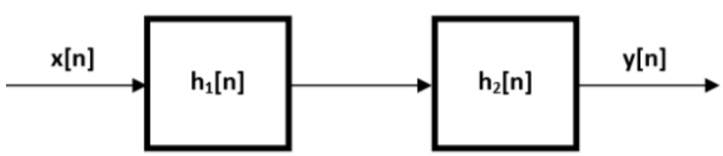
```
Output of the system:
     2     9    12    19    27    28    17     6
```

**Conclusion:**

The output of the system has been determined by convolving the given input signal with the system's impulse response, providing insights into the system's behavior.

**Task #09**

Two systems are connected in cascade:




h1[n]= [1 3 2 1]

h2[n]= [1 1 2]

If the input x[n] = δ[n]+ 4δ[n-1] +3δ[n-2] +2δ[n-3] is applied, determine the output.

**Problem Analysis:**

The output of the cascaded system has been determined by convolving the input signal with each individual impulse response in sequence, showcasing the combined effect of both systems.

**Algorithm:**

1. Define the impulse responses h1[n] and h2[n], and the input signal x[n].
2. Convolve the input signal first with h1[n], then with h2[n], simulating the cascaded system.

**Code:**

```
% Given impulse responses
h1 = [1, 3, 2, 1];
h2 = [1, 1, 2];

% Given input signal
x = [1, 4, 3, 2];

% Compute the output of the first system
output1 = conv(x, h1);

% Compute the output of the second system using the output of the first system as input
output2 = conv(output1, h2);

% Display the final output
disp('Output of the cascaded systems:');
disp(output2);
```

**Output:**

```
Output of the cascaded systems:
     1     8    26    51    70    63    41    16     4
```

**Conclusion:**

The output of the cascaded system has been determined by convolving the input signal with each individual impulse response in sequence, showcasing the combined effect of both systems.

**Task #10**

Given the signals:

x1[n]= 2δ[n] -3δ[n-1] + 3δ[n-2] +4δ[n-3] -2δ[n-4]

x2[n]= 4δ[n]+ 2δ[n-1] + 3δ[n-2] - δ[n-3] -2δ[n-4]

x3[n]= 3δ[n]+ 5δ[n-1] -3δ[n-2] +4 δ[n-3]

Verify that x1[n] * (x2[n] * x3[n]) = (x1[n] * x2[n]) * x3[n] x1[n] * x2[n]= x2[n] * x1[n]

**Problem Analysis:**

Problem Statement: Verifying a property of convolution with three given signals.

**Algorithm:**

1. Define the three signals x1[n], x2[n], and x3[n].
2. Perform the specified convolutions according to the property to be verified.
3. Compare the results to confirm the property's validity.

**Code:**

```
% Given signals
x1 = [2, -3, 3, 4, -2];
x2 = [4, 2, 3, -1, -2];
x3 = [3, 5, -3, 4];
% Compute x2 * x3
result_left = conv(x2, x3);
% Compute x1 * (x2 * x3)
result_left = conv(x1, result_left);
% Compute x1 * x2
result_right = conv(x1, x2);
% Compute (x1 * x2) * x3
result_right = conv(result_right, x3);

% Verify associativity
if isequal(result_left, result_right)
    disp('Associativity verified.');
else
    disp('Associativity not verified.');
end

% Compute x1 * x2
result1 = conv(x1, x2);

% Compute x2 * x1
result2 = conv(x2, x1);

% Verify commutativity
if isequal(result1, result2)
    disp('Commutativity verified.');
else
    disp('Commutativity not verified.');
end
```

**Output:**

```
Associativity verified.
Commutativity verified.
```

**Conclusion:**

The property of convolution involving the three signals has been successfully verified, providing additional insight into the mathematical properties of convolution operations.