MATLAB Assignment

Submitted by:

Saurabh Kumar

SC22B146

Course: AV223 - Signals and Systems

Answers on the line 162 (Linearity), 223 (Causality) and 306 (Stability).

Linearity, Causality, Stability

Impulse Response

The impulse response of a discrete-time system is the system's output when the input is an impulse function.

Let $h[n]$ be the impulse response of a discrete-time system. If the system is linear and time-invariant, the response of the system $y[n]$ to any input signal $x[n]$ can be expressed as the convolution of the input signal with the impulse response:

$$y[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[n-k]$$

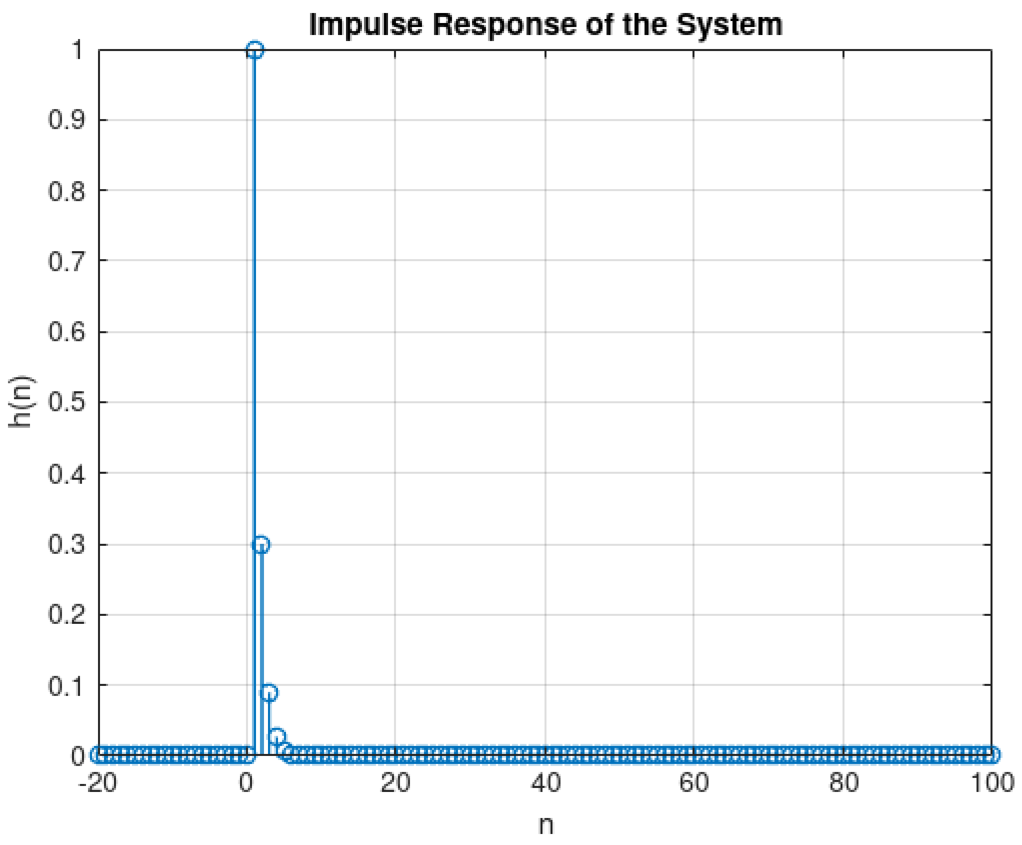
This equation represents the convolution sum, where $h[n-k]$ is the shifted version of the impulse response.

The impulse response fully characterizes an LTI system. Knowing the impulse response allows you to predict the system's output for any given input. The length of the impulse response determines the memory of the system. A system with a finite-length impulse response is called a Finite Impulse Response (FIR) system, while a system with an infinite impulse response is called an Infinite Impulse Response (IIR) system. If the impulse response $h[n]$ is normalized such that $h[0] = 1$, it is referred to as the unit impulse response. The frequency response of a system, which describes how the system responds to different frequencies, can be obtained by taking the Fourier transform of the impulse response.

Write a program to find the impulse response of the system given by $y(n) = x(n) + x(n-1) - 0.7y(n-1)$ at $-20 \leq n \leq 100$.

```
% Define the system parameters
a = 1;   % Coefficient for x(n)
b = 1;   % Coefficient for x(n-1)
c = 0.7; % Coefficient for y(n-1)

% Define the range of n
n = -20:100;
```

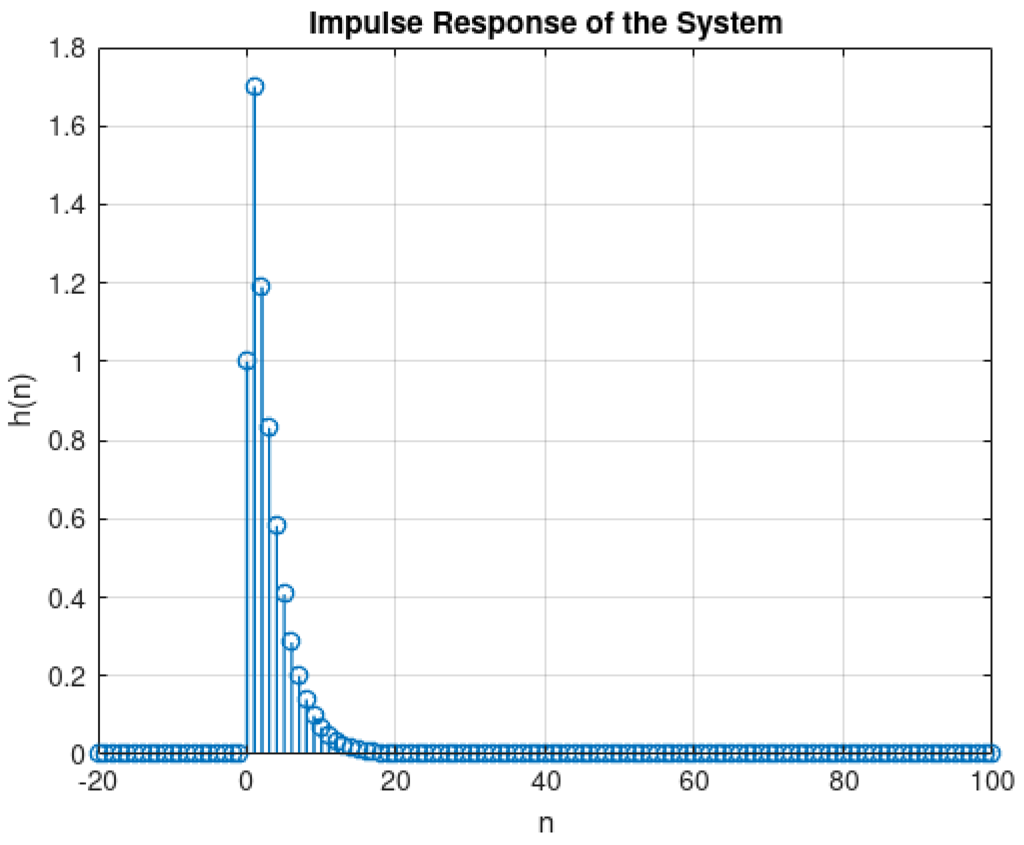
```
% Impulse response calculation
h = zeros(size(n));
h(n == 0) = 1; % Impulse at n = 0

% Calculate the impulse response using the system equation
for i = 2:length(n)
    h(i) = a * h(i-1) + b * (n(i-1) == 0) - c * h(i-1);
end

% Plot the impulse response
stem(n, h, 'LineWidth', 1);
title('Impulse Response of the System');
xlabel('n');
ylabel('h(n)');
grid on;
```



Alternative implementation using filter function.

```
% Define the system coefficients
b = [1, 1];     % Coefficients for x(n) and x(n-1)
a = [1, -0.7];  % Coefficients for y(n-1)

% Impulse response calculation using the filter function
```

```
n = -20:1:100;  % Start the impulse response at n = 0
imp = [zeros(1,20) 1 zeros(1,100)];
impulse_response = filter(b, a, imp);

% Plot the impulse response
stem(n, impulse_response, 'LineWidth', 1);
title('Impulse Response of the System');
xlabel('n');
ylabel('h(n)');
grid on;
```



Step Response

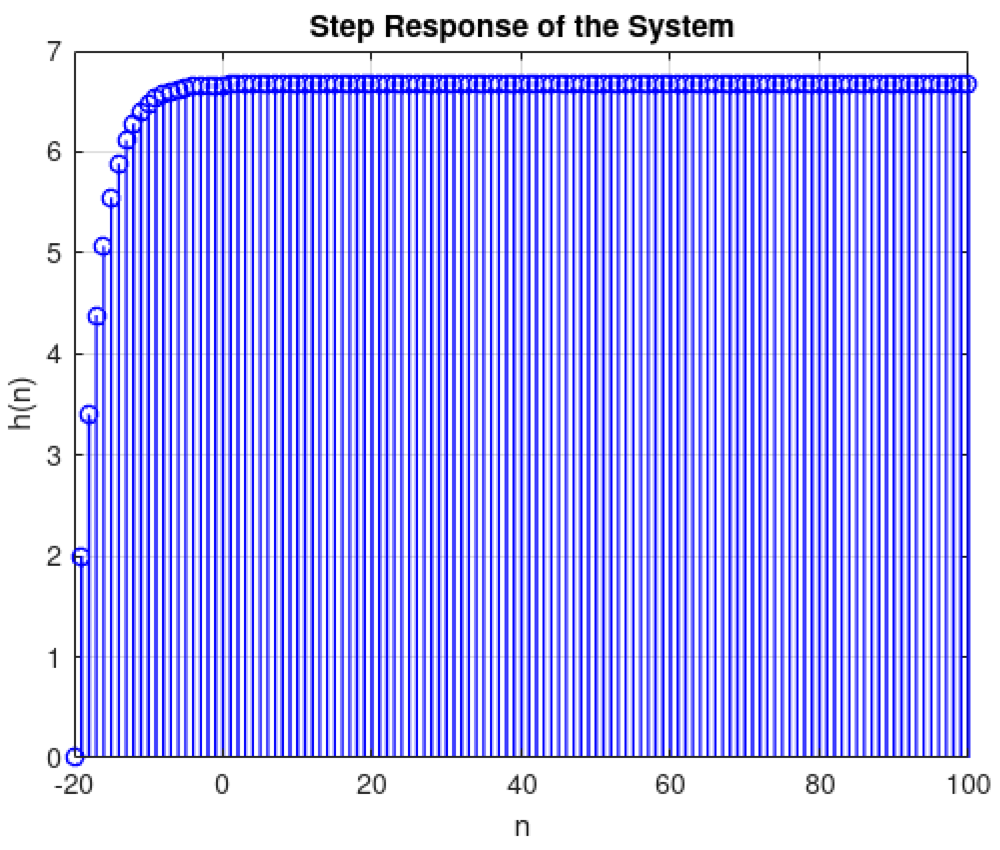
The step response of a discrete-time system is the system's output when it is subjected to a unit step input. Mathematically, it is the response of the system to a signal that is 0 for all time steps less than zero and 1 for all time steps greater than or equal to zero.

If the system is described by the difference equation $y[n] = f(x[n])$, where $y[n]$ is the output and $x[n]$ is the input, the step response is obtained by setting $x[n] = 1$ for $n \geq 0$ and $x[n] = 0$ for $n < 0$.

The step response provides information about how the system reacts to sudden changes in input. It is particularly useful in understanding the system's transient behavior, stability, and overall dynamic response. The step response typically consists of two main components: the transient response and the steady-state response. The transient response represents the system's behavior during the initial period of adjustment, while the steady-state response represents the behavior after the system has settled to a stable state.

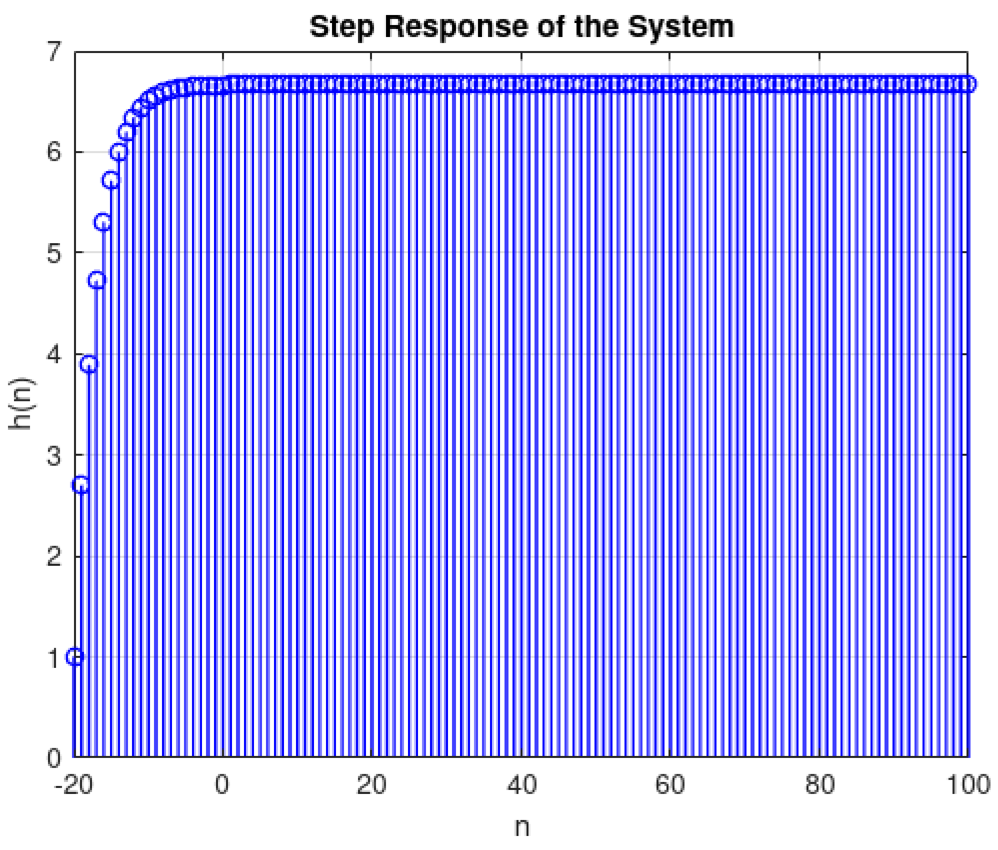
Write a program to find the step response of the system given by $y(n) = x(n) + x(n-1) - 0.7y(n-1)$ at $-20 \le n \le 100$.

```
% Define the system coefficients
a = [1 -0.7];
b = [1 1];

% Define the time vector
n = -20:100;

% Generate a unit step input
u = ones(size(n));

% Initialize the output vector
y = zeros(size(n));

% Compute the step response using the system equation
for i = 2:length(n)
    y(i) = b * [u(i); u(i-1)] - a(2) * y(i-1);
end

% Plot the step response using stem for discrete data
figure;
stem(n, y, 'b', 'LineWidth', 1);
title('Step Response of the System');
xlabel('n');
ylabel('h(n)');
grid on;
```



Alternative implementation

```
% Define the system coefficients
b = [1 1];      % Numerator coefficients
a = [1 -0.7];   % Denominator coefficients

% Create a unit step input
n = -20:100;
u = ones(size(n));

% Compute the step response using the filter function
y = filter(b, a, u);

% Plot the step response
figure;
stem(n, y, 'b', 'LineWidth', 1);
title('Step Response of the System');
xlabel('n');
ylabel('h(n)');
grid on;
```

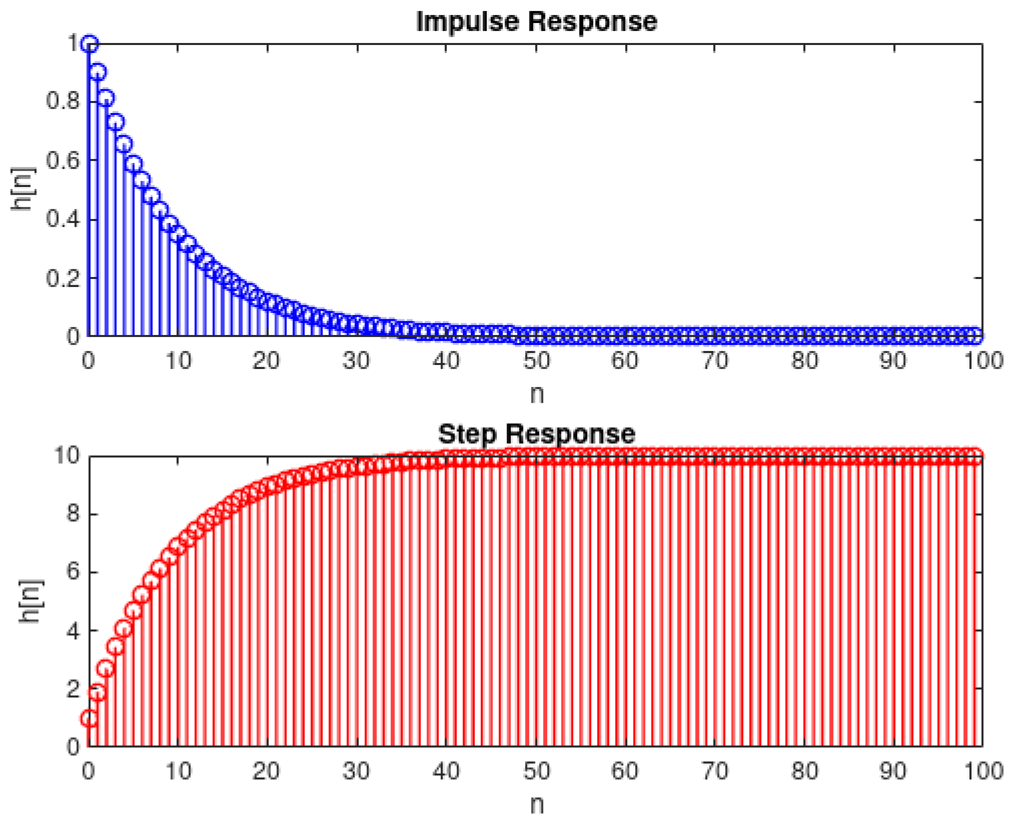


Exercises

Write a program to find the impulse response and step response of the system $y[n] = x[n] - 0.9y[n-1]$ for $0 \leq n \leq 100$.

```
% Define the system coefficients
b = [1];   % Numerator coefficients (coefficients of x[n])
a = [1, -0.9];  % Denominator coefficients (coefficients of y[n-1])

% Impulse response
impulse_response = filter(b, a, [1, zeros(1, 100)]);  % Impulse input [1, 0,
0, ..., 0]

% Step response
step_response = filter(b, a, ones(1, 100));  % Step input [1, 1, 1, ..., 1]

% Plotting
n = 0:99;  % Time indices
subplot(2, 1, 1);
stem(n, impulse_response(1:100), 'b', 'LineWidth', 1);
title('Impulse Response');
xlabel('n');
ylabel('h[n]');

subplot(2, 1, 2);
```

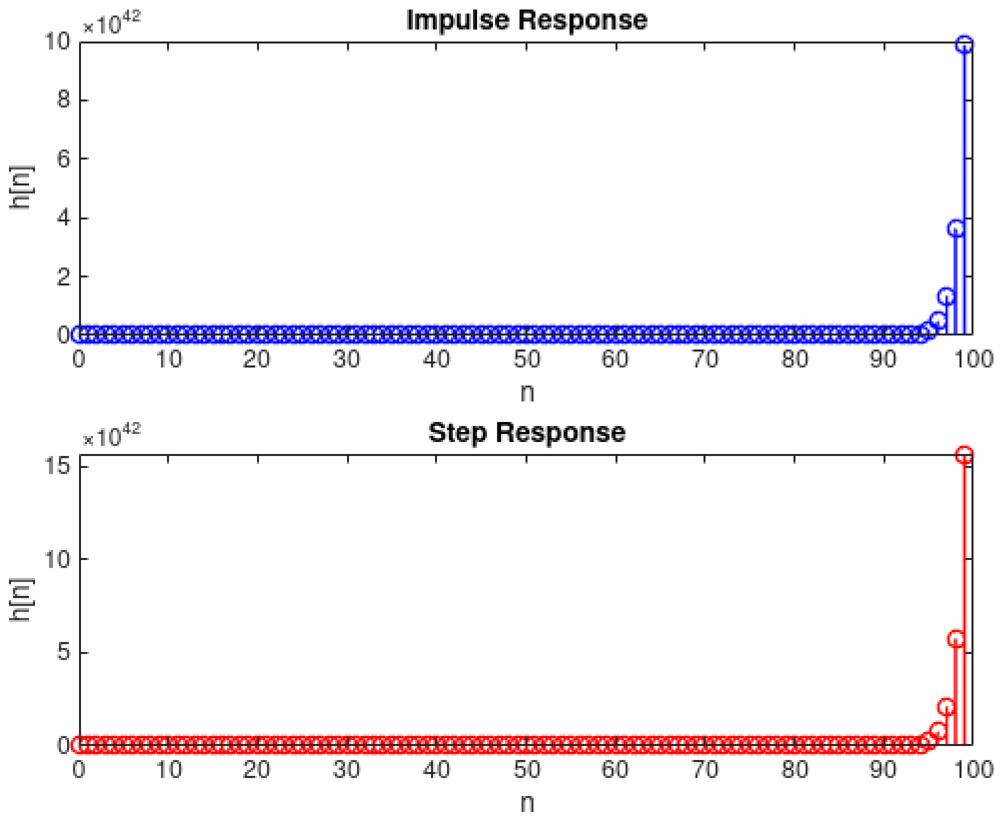
```
stem(n, step_response(1:100), 'r', 'LineWidth', 1);
title('Step Response');
xlabel('n');
ylabel('h[n]');
```



Write a program to find the impulse response and step response of the system $y[n] = exp(x[n])$ for $0 \leq n \leq 100$.

```
% Define the time indices
n = 0:99;

% Impulse response
impulse_response = exp(n);

% Step response
step_response = cumsum(impulse_response);

% Plotting
subplot(2, 1, 1);
stem(n, impulse_response, 'b', 'LineWidth', 1);
title('Impulse Response');
xlabel('n');
ylabel('h[n]');
```

```
subplot(2, 1, 2);
stem(n, step_response, 'r', 'LineWidth', 1);
title('Step Response');
xlabel('n');
ylabel('h[n]');
```



Linearity

A system is considered linear if it satisfies two important properties: *superposition and homogeneity*.

Superposition:

A system is linear if the response to the sum of two (or more) input signals is equal to the sum of the individual responses to each input signal.

If $y_1(t)$ is the response of the system to input $x_1(t)$, and $y_2(t)$ is the response to input $x_2(t)$, then the system is linear if:

$$[a \cdot x_1(t) + b \cdot x_2(.)] = a \cdot y_1(t) + b \cdot y_2(t)$$

In simpler terms, the effect of multiple inputs is additive in a linear system.

Homogeneity (Scaling):

A system is linear if the response to a scaled version of an input signal is equal to the scaled version of the response to the original input signal. If $y(t)$ is the response of the system to input $x(t)$, then the system is linear if:

$[a \cdot x(t)] = a \cdot y(t)$

A system is linear if it satisfies both superposition and homogeneity simultaneously. These properties together ensure that the response of the system exhibits a consistent and predictable behavior in the presence of different inputs. Linearity facilitates the analysis and understanding of systems. It allows engineers and scientists to break down complex systems into simpler parts and analyze their responses individually.

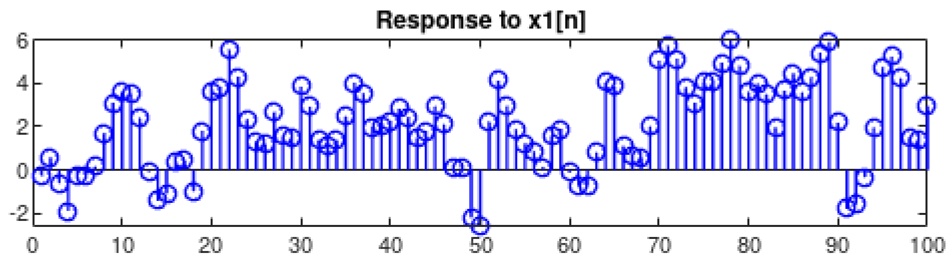
A system is considered nonlinear if it fails to satisfy either the superposition or homogeneity property. Nonlinear systems can exhibit more complex and diverse behaviors, making their analysis and control challenging.

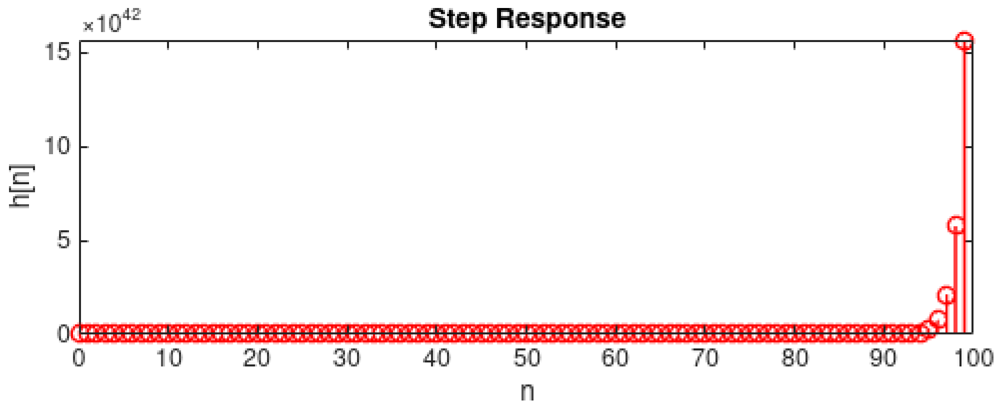
Check linearity of the system $y(n) = x(n) + x(n-1) - 0.7y(n-1)$

```
% Define the system coefficients
b = [1, 1];     % Numerator coefficients (coefficients of x[n] and x[n-1])
a = [1, -0.7];  % Denominator coefficients (coefficients of y[n-1])

% Generate random inputs
x1 = randn(1, 100);  % First random input
x2 = randn(1, 100);  % Second random input

% Generate the corresponding outputs
y1 = filter(b, a, x1);
y2 = filter(b, a, x2);

% Apply the system to the sum of inputs
y_sum = filter(b, a, x1 + x2);

% Check superposition property
if isequal(y_sum, y1 + y2)
    disp('The system is linear.');
else
    disp('The system is not linear.');
end
```

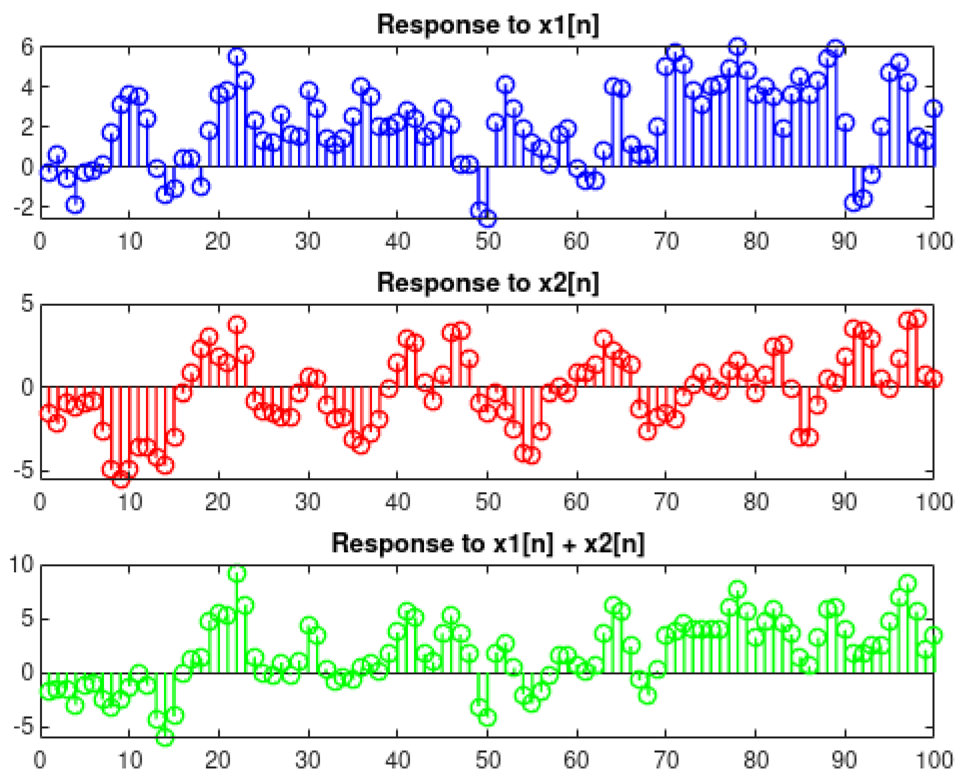
```
The system is not linear.
```

```

% Plotting for illustration
subplot(3, 1, 1);
stem(y1, 'b', 'LineWidth', 1);
title('Response to x1[n]');
```





```
subplot(3, 1, 2);
stem(y2, 'r', 'LineWidth', 1);
title('Response to x2[n]');

subplot(3, 1, 3);
stem(y_sum, 'g', 'LineWidth', 1);
title('Response to x1[n] + x2[n]');
```



Exercise

Repeat the check for $y[n] = x[n] - 0.9y[n-1]$ and $y[n] = exp(x[n])$

```
%YOUR ANSWER CODE HERE
%%%%%% Checking for y[n] = x[n] - 0.9y[n-1] %%%%%%
% Define the system coefficients
b = [1];     % Numerator coefficients (coefficients of x[n])
a = [1, -0.9];  % Denominator coefficients (coefficients of y[n] and y[n-1])

% Generate random inputs
x1 = randn(1, 100);  % First random input
x2 = randn(1, 100);  % Second random input

% Generate the corresponding outputs
y1 = filter(b, a, x1);
y2 = filter(b, a, x2);

% Apply the system to the sum of inputs
y_sum = filter(b, a, x1 + x2);

% Check superposition property
if isequal(y_sum, y1 + y2)
    disp('The system is linear.');
else
```

```
    disp('The system is not linear.');
end
%% Ans: The system is not linear


%%%%%% Checking for y[n] = exp(x[n]) %%%%%%
g1 = exp(x1);
g2 = exp(x2);
g3 = exp(x1 + x2);
if isequal(g3, g1 + g2)
    disp('The system is linear.');
else
    disp('The system is not linear.');
end
%% Ans: The system is not linear
```

Causality

A causal system is one where the output at any given time depends only on the current and past values of the input and the past values of the output. In other words, the system does not use future input values or information in its computations. A system is considered causal if the output at any time n is determined solely by the past and present values of the input and output. For a discrete-time system, causality is expressed as:

$$y[n] = f(x[n], x[n-1], x[n-2], \ldots, y[n-1], y[n-2], \ldots)$$

A system is non-causal if it requires future values of the input to compute the current output. Non-causal systems are not physically realizable as they would require information from the future, which is generally not available. The impulse response of a causal system is zero for negative time indices (discrete-time) or before time $t = 0$ (continuous-time). The impulse response represents the system's behavior when subjected to an impulse input, and causality ensures that the system's response only begins after the impulse is applied.

Check causality of the system $y(n) = x(n) + x(n-1) - 0.7y(n-1)$

```
% System coefficients
b = [1, 1];   % Coefficients of x(n) and x(n-1)
a = [1, -0.7]; % Coefficient of y(n-1)

% Number of samples
N = 100;

% Generate random input signal x(n)
x = rand(1, N);

% Apply the system to the entire input signal
y_full = filter(b, a, [1, x, zeros(1, N-1)]);

% Apply the system to the input signal appended with zeros for future values
y_truncated = filter(b, a, [1, x, zeros(1, N)]);
```

```
% Check causality
isCausal = isequal(y_full, y_truncated);

% Display result
if isCausal
    disp('The system is causal.');
else
    disp('The system is not causal.');
end
```

```
The system is not causal.
```

Exercise

Repeat the check for $y[n] = x[n] - 0.9y[n-1]$ and $y[n] = exp(x[n])$

```
%YOUR ANSWER CODE HERE
%%%%%% Checking for y[n] = x[n] - 0.9y[n-1] %%%%%%
% Define the system coefficients
b = [1];     % Numerator coefficients (coefficients of x[n])
a = [1, -0.9];  % Denominator coefficients (coefficients of y[n] and y[n-1])

% Number of samples
N = 100;

% Generate random input signal x(n)
x = rand(1, N);

% Apply the system to the entire input signal
y_full = filter(b, a, [1, x, zeros(1, N-1)]);

% Apply the system to the input signal appended with zeros for future values
y_truncated = filter(b, a, [1, x, zeros(1, N)]);

% Check causality
isCausal = isequal(y_full, y_truncated);

% Display result
if isCausal
    disp('The system is causal.');
else
    disp('The system is not causal.');
end
%% Ans: The system is not causal


%%%%%% Checking for y[n] = exp(x[n]) %%%%%%
% Define the system

% Generate some example input values for x[n]
```

```
n = 0:10;
x = sin(n); % Example input signal

% Compute the output y[n]
y = exp(x);

% Check if the system is causal
is_causal = all(y(2:end) == exp(x(2:end)));

if is_causal
    disp('The system is causal.');
else
    disp('The system is not causal.');
end
%% Ans: The system is causal
```

Stability

Stability refers to the ability of a system to maintain a bounded response in the presence of various inputs.

There are two primary types of stability: **BIBO Stability** (Bounded Input, Bounded Output) and **Asymptotic Stability**.

BIBO Stability:

A system is BIBO stable if, for every bounded input signal, the system's output remains bounded. This can be expressed as:

If $|x[n]| < M$ for all n, where M is a finite constant, then $|y[n]| < N$ for all n,

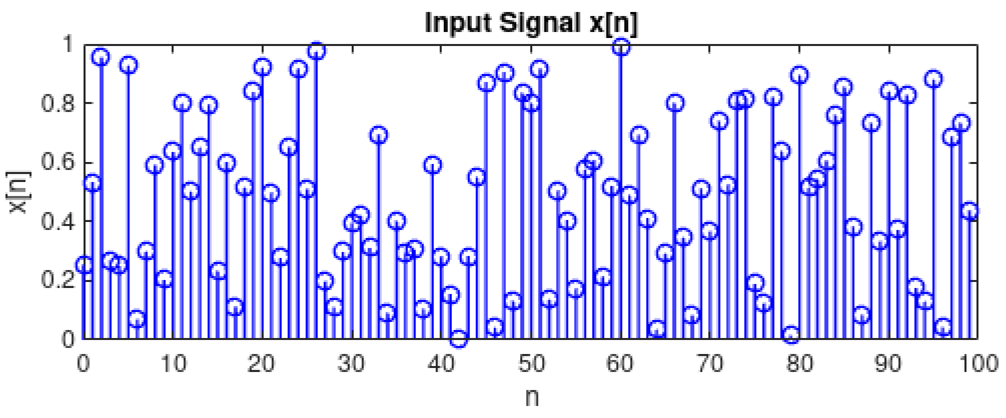
where N is another finite constant. BIBO stability ensures that the system's response to any finite and bounded input does not grow unbounded. It is a desirable property for systems to avoid excessive responses.

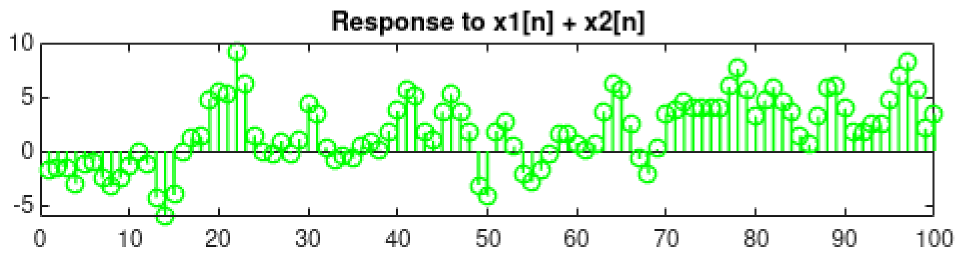
For a linear time-invariant (LTI) discrete-time system, BIBO stability and internal stability are closely related. If a system is internally stable, it is generally BIBO stable. However, the reverse may not always be true. BIBO stability is concerned with the input-output relationship, while internal stability is concerned with the stability of the system's internal dynamics.

Check causality of the system $y(n) = x(n) + x(n-1) - 0.7y(n-1)$

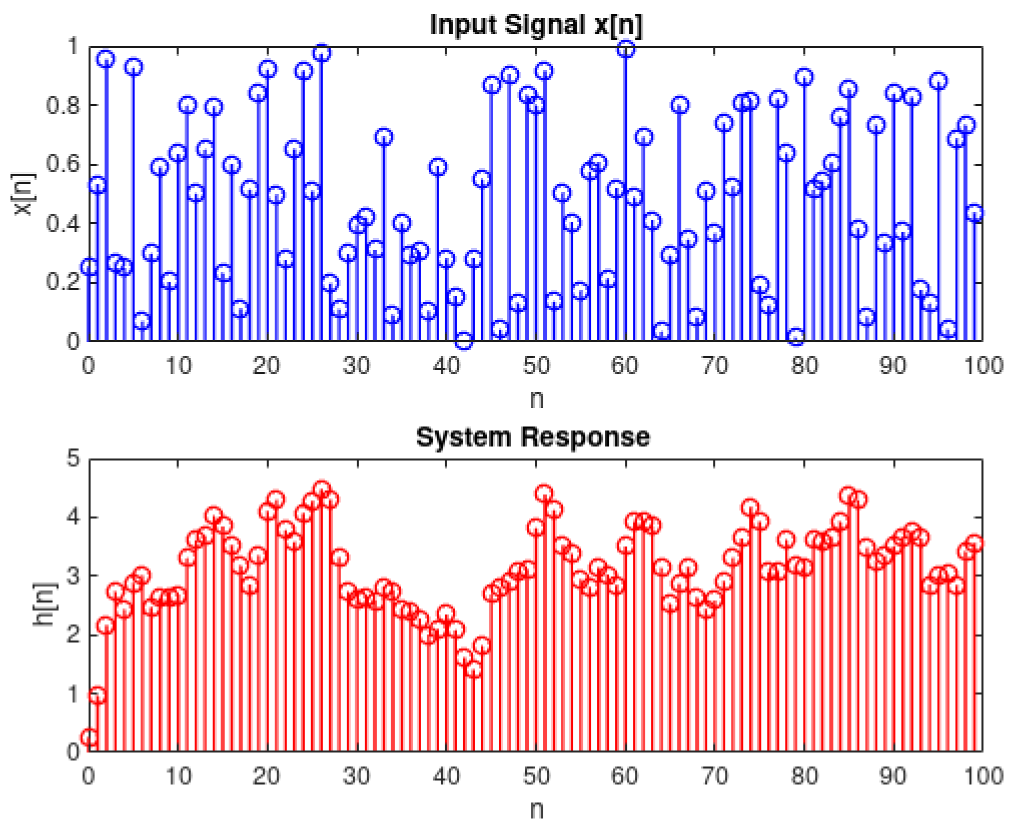
```
% Define the system coefficients
b = [1, 1];   % Numerator coefficients (coefficients of x[n] and x[n-1])
a = [1, -0.7];  % Denominator coefficients (coefficients of y[n-1])

% Generate a random input signal
n = 0:99;  % Time indices
x = rand(size(n));  % Random input signal

% Simulate the system response
y = filter(b, a, x);
```

```
% Plotting
subplot(2, 1, 1);
stem(n, x, 'b', 'LineWidth', 1);
title('Input Signal x[n]');
xlabel('n');
ylabel('x[n]');
```





```
subplot(2, 1, 2);
stem(n, y, 'r', 'LineWidth', 1);
title('System Response ');
xlabel('n');
ylabel('h[n]');
```



```
% Calculate and display poles and zeros
[z, p, k] = tf2zpk(b, a);
disp(z);
```

```
    -1
```

```
disp(p);
```

```
    0.7000
```

```

% Check stability based on pole locations
if all(abs(p) < 1)
    disp('The system is stable.');
else
    disp('The system is unstable.');
end
```

```
The system is stable.
```

Exercise

Repeat the check for $y[n] = x[n] - 0.9y[n-1]$ and $y[n] = exp(x[n])$

```
%YOUR ANSWER CODE HERE
%%%%%% Checking for y[n] = x[n] - 0.9y[n-1] %%%%%%
% Define the system coefficients
```

```
b = [1];     % Numerator coefficients (coefficients of x[n])
a = [1, -0.9];  % Denominator coefficients (coefficients of y[n] and y[n-1])
% Generate a random input signal
n = 0:99;  % Time indices
x = rand(size(n));  % Random input signal

% Simulate the system response
y = filter(b, a, x);

% Plotting
subplot(2, 1, 1);
stem(n, x, 'b', 'LineWidth', 1);
title('Input Signal x[n]');
xlabel('n');
ylabel('x[n]');
subplot(2, 1, 2);
stem(n, y, 'r', 'LineWidth', 1);
title('System Response ');
xlabel('n');
ylabel('h[n]');

% Calculate and display poles and zeros
[z, p, k] = tf2zpk(b, a);
disp(z);
disp(p);

% Check stability based on pole locations
if all(abs(p) < 1)
    disp('The system is stable.');
else
    disp('The system is unstable.');
end
%% Ans: The system is not stable




%%%%%% Checking for y[n] = exp(x[n]) %%%%%%
n = -10:10;
x = sin(n); % Example input signal, can be replaced with any signal

% Compute the output y[n]
y = exp(x);

% Check if the system is stable
is_stable = all(isfinite(y)); % Check if all output values are finite

if is_stable
    disp('The system is stable.');
else
```

```
    disp('The system is not stable.');
end
%% Ans: The system is stable
```