Report of Signals and Systems Lab Assignment 1

Written by HUANG Guanchao, SID 11912309 and GONG Xinrui, SID 11911233.

The source code of this assignment can be retrieved at our GitHub repo.

1.4

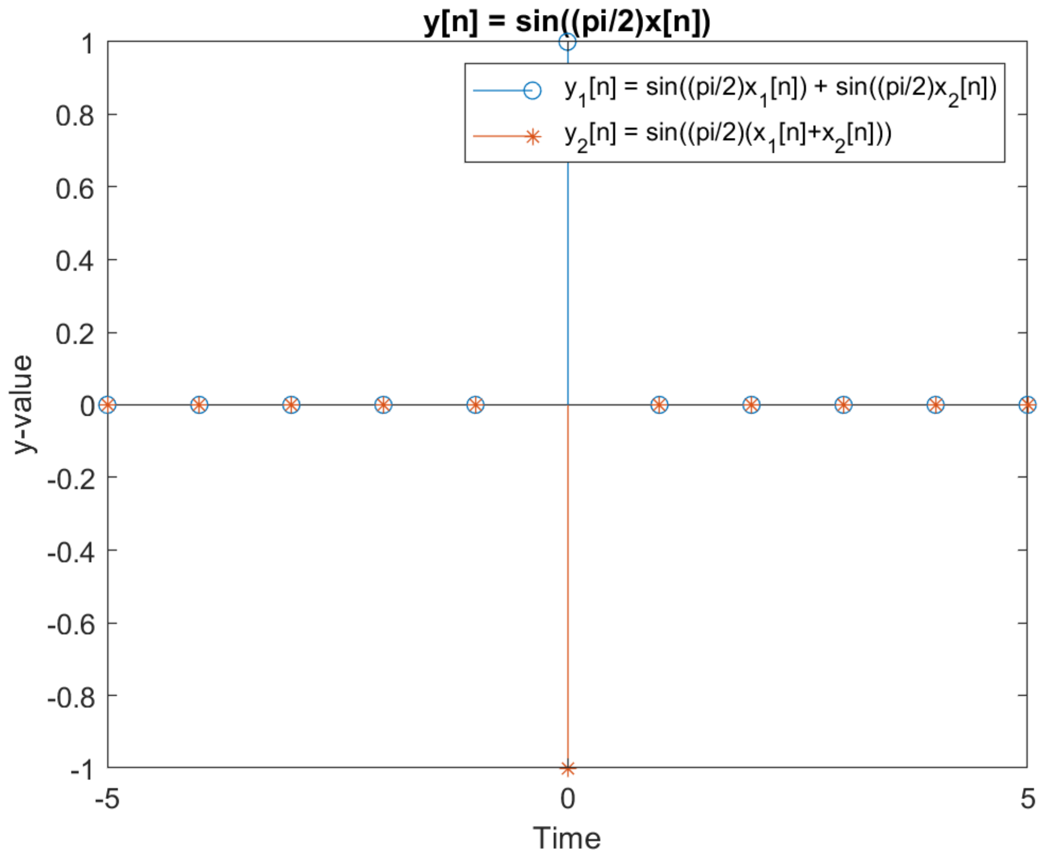
Basic Problem a

> The system $y[n] = \sin\{(\pi/2)x[n]\}$ is not linear. Use the signals $x_1[n] = \delta[n]$ and $x_2[n] = 2\delta[n]$ to demonstrate how the system violates linearity.

A function `L` is defined to generate output signal from a matrix.

Firstly, we generated input impulse signals by defining matrix `x1` and `x2`. Based on this we generated the output signal `y1` and `y2` corresponding to them by calling function `L`.

Then, we produced combined output signal `y3 = y1 + y2`, and combined input signal `x3 = x1 + x2`, as well as its corresponding output signal `y4`. The plot of `y3` and `y4` with respect to `n` is shown below.



In the figure, $y_1[n] = \sin\{(\pi/2)x_1[n]\} + \sin\{(\pi/2)x_2[n]\}$ is not identical with $y_2[n] = \sin\{(\pi/2)(x_1[n] + x_2[n])\}$, hence the system is not linear.

The MATLAB script is shown in the following code block.

```
% define scopes
l = 5;

% Basic Problem a
n = [-l:l];
x1 = [zeros(1, l) 1 zeros(1, l)];
x2 = 2 * x1;
x3 = x1 + x2;

y1 = L(x1);
y2 = L(x2);
Py3 = y1 + y2
y4 = L(x3)

stem(n, y3, 'o')
hold on
stem(n, y4, '*')

title('y[n] = sin((pi/2)x[n])')
xlabel('Time');
ylabel('Output')
legend('y_1[n] = sin((pi/2)x_1[n]) + sin((pi/2)x_2[n])', ...
    'y_2[n] = sin((pi/2)(x_1[n]+x_2[n]))');

% define system
function y = L(x)
    y = sin(pi / 2 * x)
```

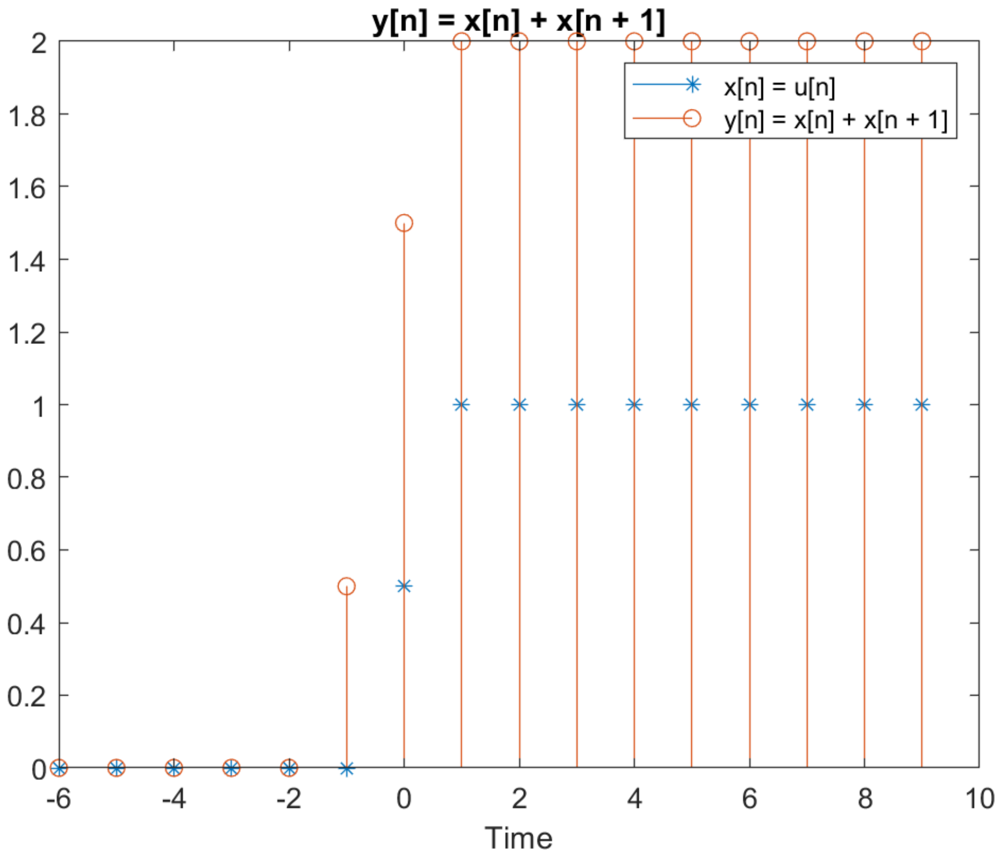
```
end
```

Basic Problem b

> The system $y[n] = x[n] + x[n+1]$ is not causal. Use the signal $x[n] = u[n]$ to demonstrate this. Define the MATLAB vectors x and y to represent the input on the interval $-5 \leq n \leq 9$, and the output on the interval $-6 \leq n \leq 9$, respectively.

Firstly, integer sequence `n` is generated for representing time.

We produced input step signal $x[n]$ and its shift $x[n+1]$ by defining matrix `x0` and `x1` then the output `y`. The plot is shown below.



The excitation starts at time $n = 0$, however, before that at time $n = -1$, the output $y[n]$ is non-zero. Hence, the system is not causal.

The MATLAB script is shown in the following code block.

```
% Basic Problem b
n = [-6:9];
x = [zeros(1, 6) 1/2 ones(1, 9)]
x0 = [zeros(1, 6) 1/2 ones(1, 9)];
x1 = [zeros(1, 5) 1/2 ones(1, 10)];
y = x0 + x1;

stem(n, x, '*');
hold on
stem(n, y);

title('y[n] = x[n] + x[n + 1]')
xlabel('Time')
```

```
legend('x[n] = u[n]', 'y[n] = x[n] + x[n + 1]')
```

Intermediate Problem c

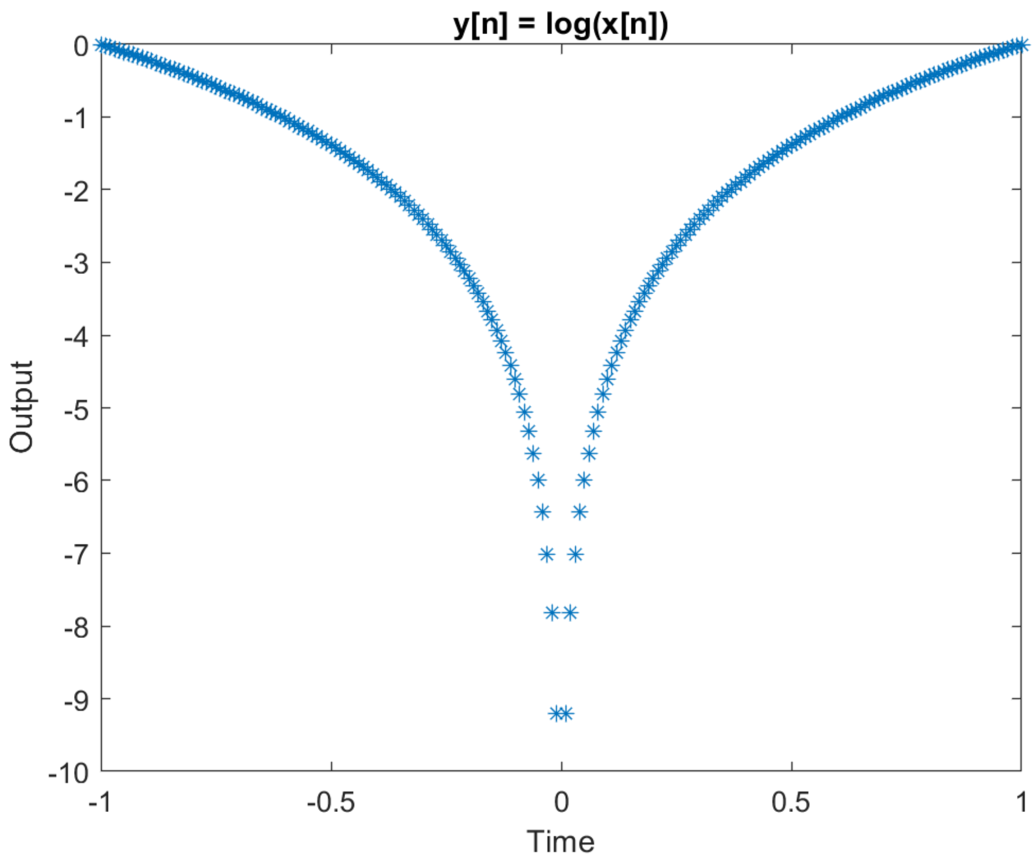
Firstly, sequence `n` is generated for representing time. Then by matrix operation, we defined input signal `x` to be $x[n] = n^2$.

While plotting the figure, output `y` is printed out through terminal as

```
>> run("c:\Users\Guanc\Documents\GitHub\SigSys-lab\src\A1\A1_c.m")

...

Columns 97 through 112

   -6.4378   -7.0131   -7.8240   -9.2103      -Inf   -9.2103   -7.8240   -7.0131
  -6.4378   -5.9915   -5.6268   -5.3185   -5.0515   -4.8159   -4.6052   -4.4145

...

>>
```

In which there is `-Inf` output.

The plot is shown below.



The system is not stable, because when we give a bounded input the output approaches infinity at time $n = 0$, which contradicts the definition of stability.

The MATLAB script is shown in the following code block.

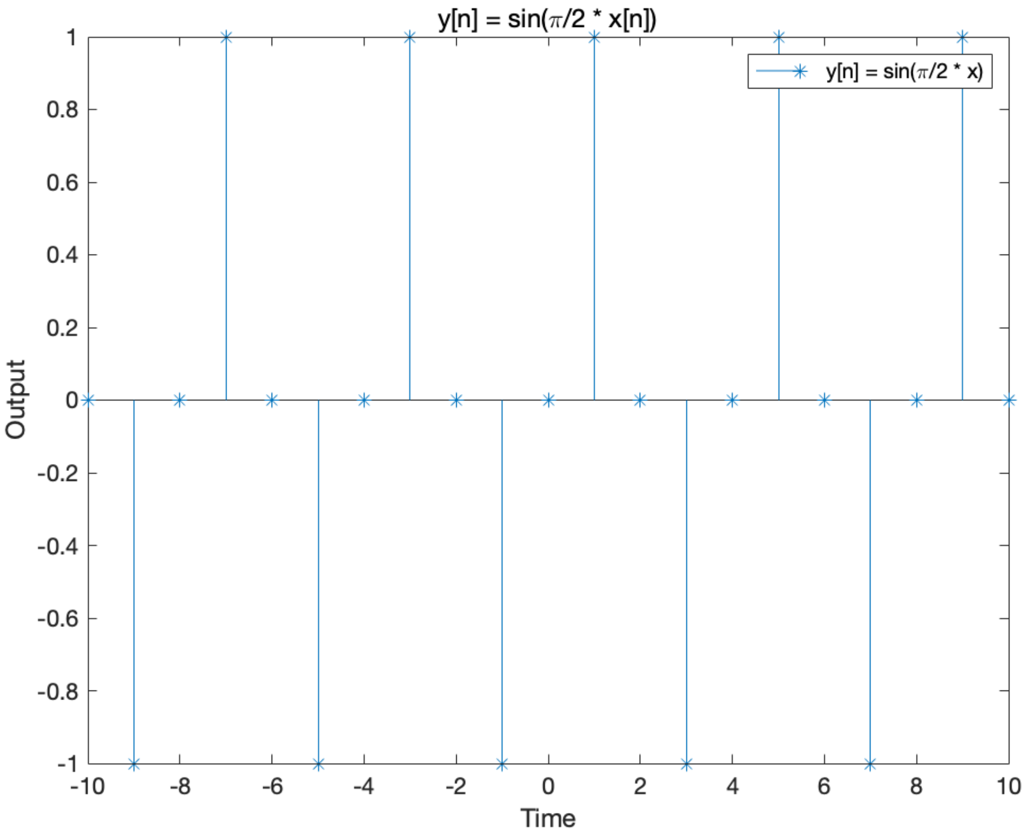
```
n = [-1 : 0.01 : 1];
x = n.^2;
y = log(x)
plot(n,y,'*')

title('y[n] = log(x[n])');
xlabel('Time');
ylabel('Output');
```

Intermediate Problem d

Sequence `n` is generated for representing time. By matrix operation, we defined input signal `x[n] = n`.Output signal is defined as `y[n] = sin(pi/2*x[n])`.
With different input `x[n]`, output `y[n]` only has two values, the result is shown below.

```
...

y =

  列 1 至 8

   -0.0000   -1.0000    0.0000    1.0000   -0.0000   -1.0000    0.0000    1.0000

  列 9 至 16

   -0.0000   -1.0000         0    1.0000    0.0000   -1.0000   -0.0000    1.0000

  列 17 至 21

    0.0000   -1.0000   -0.0000    1.0000    0.0000

...
```

The plot is shown below.



The system is not invertible, since with different inputs from -10 to 10, the output is always 0 or 1, which contradicts the definition of invertibility.

The MATLAB script is shown below.

```
l = 10;

n = -l:l;
x = n;

y = sin((pi / 2) * x);

stem(n, y, '*');
title('y[n] = sin(pi/2 * x[n])');
xlabel('Time');
ylabel('Output');
legend('y[n] = sin(pi/2 * x)');
```

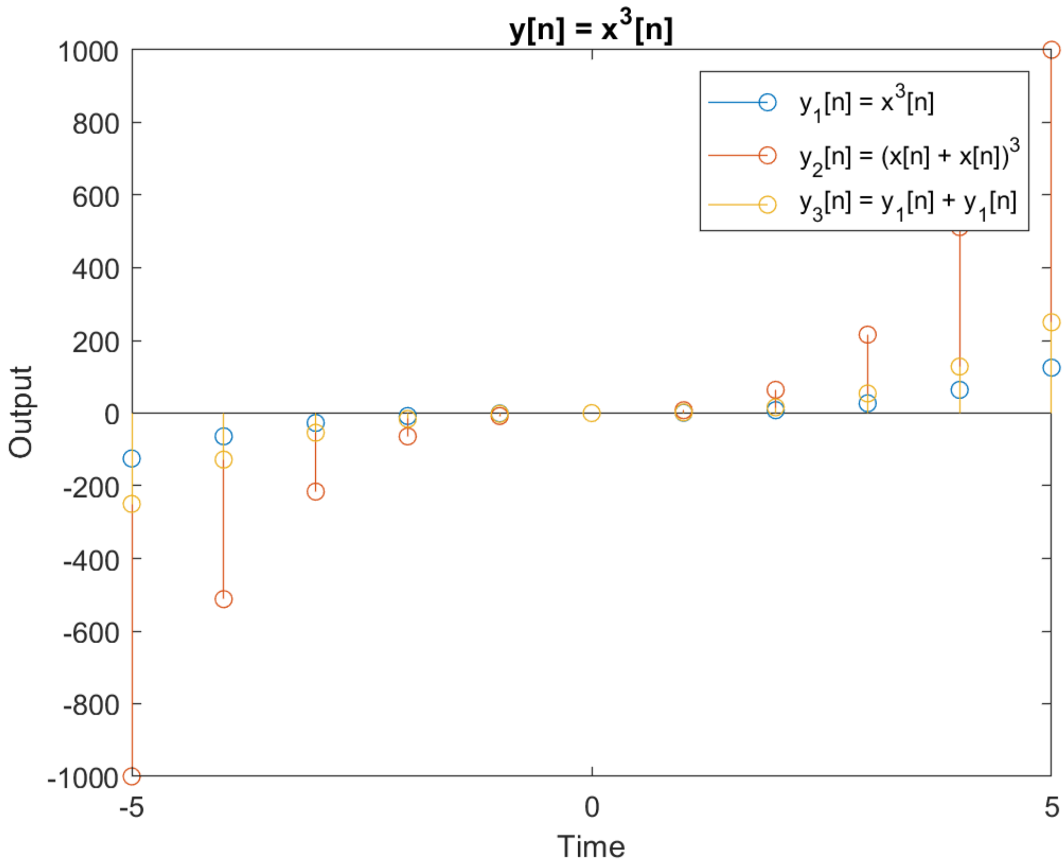
Advanced Problems e, f, g

> For each of the following systems, state whether or not the system is linear, time-invariant, causal, stable, and invertible. For each property you claim the system does not possess, construct a counter-argument using MATLAB to demonstrate how the system violates the property in question.
>
> (e). $y[n] = x^3[n]$.

Signals are defined as follows.

Signal	Variable Name	Expression
input signal $x[n]$	`x`	$x[n] = n$
output signal $y_1[n]$	`y1`	$y_1[n] = x^3[n]$
output signal $y_2[n]$	`y2`	$y_2[n] = (x[n] + x[n])^3$
output signal $y_3[n]$	`y3`	$y_3[n] = y_1[n] + y_1[n] = 2x^3[n]$

The plot is shown below.



According to the plot, $2x^3[n] \neq (2x[n])^3$, hence the system is nonlinear.

The system is time-invariant, causal, stable and invertible.

The MATLAB script is shown in the following code block.

```
% define scope
l = 5;

% 1.4 Advanced Problem e
n = [-l:l];
x = n;

y1 = L(x);
y2 = L(x + x)
y3 = y1 + y1

stem(n, y1);
hold on
stem(n, y2);
hold on
```

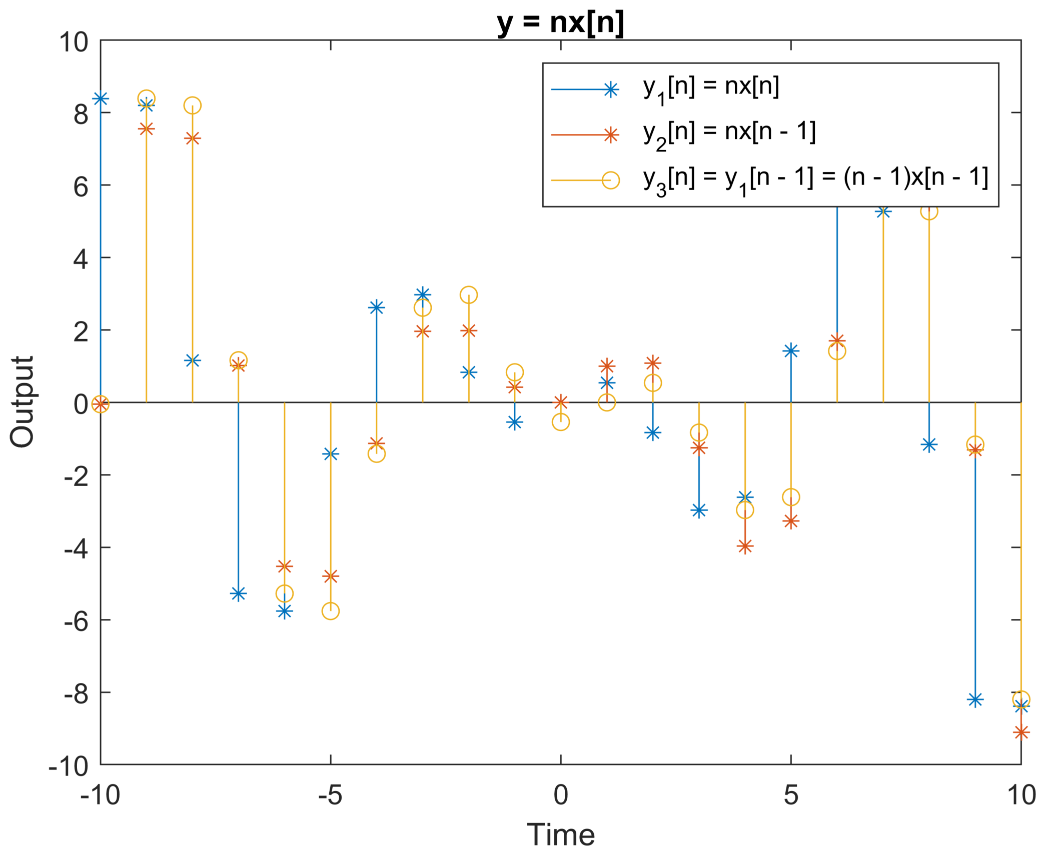
```
stem(n, y3);

title('y[n] = x^3[n]');
xlabel('Time');
ylabel('Output');
legend('y_1[n] = x^3[n]', ...
    'y_2[n] = (x[n] + x[n])^3', ...
    'y_3[n] = y_1[n] + y_1[n]')

% define system
function y = L(x)
    y = x.^3;
end
```

> (f). $y[n] = nx[n]$.

For proving the system is not time-invariant,

- First, defining integer sequence `n1` for representing time.
- Then, we generated input signal as `x1 = cos(n1)`, and correspondingly the output signal `y1 = n1 .* x1`, which indicates $y_1[n] = n\cos n$.
- Then, by defining `n2 = n1 - 1`, we generated time-shifted input signal `x2 = cos(n2)`, correspondingly `y2 = n1 .* x2`, which represents $y_2[n] = n\cos(n-1)$.
- Finally, generate time-shifted output signal $y_3[n] = y_1[n-1] = (n-1)\cos(n-1)$, by defining matrix `y3 = n2 .* x2`.

The plot is shown below.



According to the plot, the graph of $y_2[n]$ an $y_3[n]$ are not identical, hence the system is not time-invariant.

Define time interval as `n4 = double((intmax - 10):intmax)`, hence the integers in the interval are sufficiently large. Then define the input signal as `x4 = n4 .^ 33`, and output signal as `y4 = n4 .* x4`. The three matrices are printed out in the terminal as is shown in the code block below

```
...

n4 =

   1.0e+09 *

    2.1475    2.1475    2.1475    2.1475    2.1475    2.1475    2.1475    2.1475
   2.1475    2.1475    2.1475


x4 =

  1.0e+307 *

    8.9885    8.9885    8.9885    8.9885    8.9885    8.9885    8.9885    8.9885
   8.9885    8.9885    8.9885


y4 =

   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf

...
```

Apparently, the input signal is bounded, but there appears `Inf` in the output, hence the system is not stable.

Define integer sequence `n5 = -1:1;`. Then define input signal `x5 = 1 ./ n5`, which is $x_5[n] = 1/n$. And correspondingly output signal `y5 = n5 .* x5`. The three matrices are printed out in the terminal as shown below.

```
...

x5 =

  Columns 1 through 16

   -0.1000   -0.1111   -0.1250   -0.1429   -0.1667   -0.2000   -0.2500   -0.3333
  -0.5000   -1.0000       Inf    1.0000    0.5000    0.3333    0.2500    0.2000

  Columns 17 through 21

    0.1667    0.1429    0.1250    0.1111    0.1000


y5 =

     1     1     1     1     1     1     1     1     1     1   NaN     1     1
  1     1     1     1     1     1     1     1


>>
```

Obviously, for varied input signal, the output signal is always `1`, hence the system is non-invertible.

The system is linear and causal.

The MATLAB script is shown in code block below.

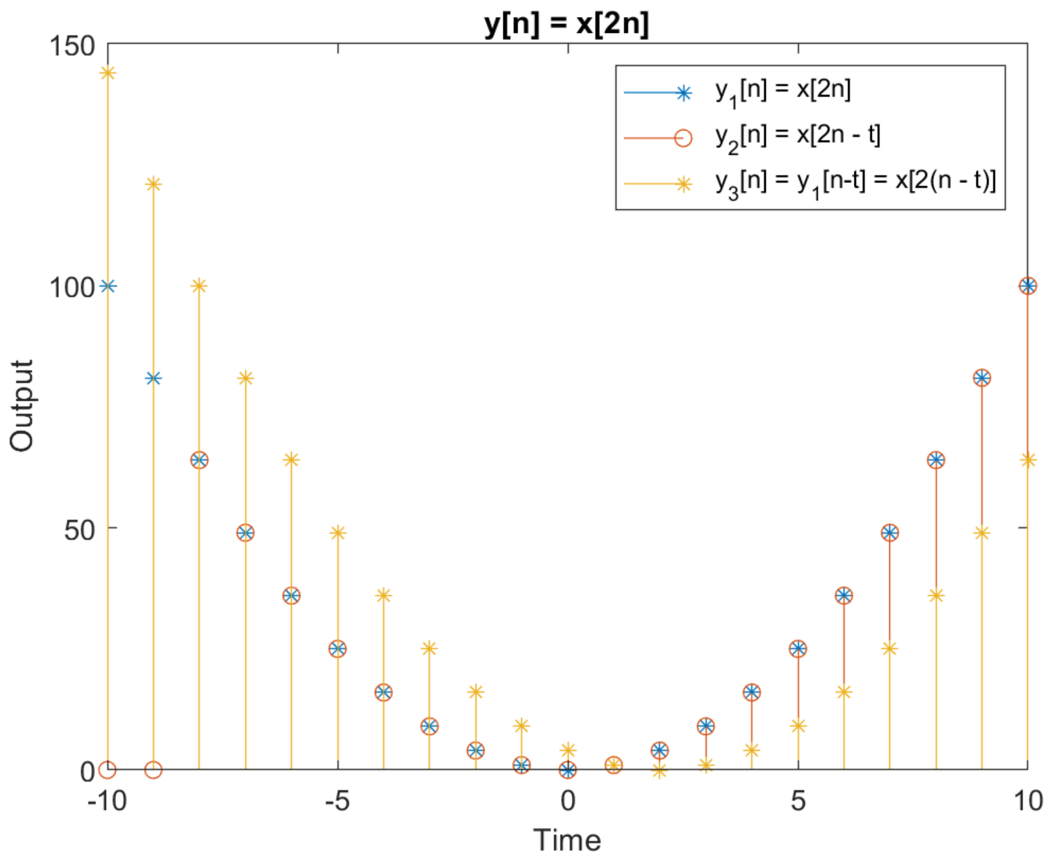
```
clf
% define scope
l = 10;

% time-invariance
% the original function:x[n] = cos(n) and y = n*x[n]
n1 = -l:l;
x1 = cos(n1);
y1 = n1 .* x1

% let the input signal have a time shift:
% x2[n] = x1[n-1]
n2 = n1 - 1;
x2 = cos(n2);
y2 = n1 .* x2

% let the original output signal have a time shift:
% y3[n] = y1[n-1]
y3 = n2 .* x2

figure(1)
stem(n1, y1, '*')
hold on
stem(n1, y2, '*')
hold on
stem(n1, y3, 'o')

title('y = nx[n]')
xlabel('Time')
ylabel('Output')
legend('y_1[n] = nx[n]', ...
    'y_2[n] = nx[n - 1]', ...
    'y_3[n] = y_1[n - 1] = (n - 1)x[n - 1]')

% stability
n4 = double((intmax - 10):intmax)
x4 = n4.^33
y4 = n4 .* x4

% invertibility
n5 = -l:l;
x5 = 1 ./ n5
y5 = n5 .* x5

figure(2)
stem(n5, x5, '')
hold on
stem(n5, y5, '')

title('y = nx[n]')
xlabel('Time')
```

```
ylabel('Output')
legend('x[n] =  1/n', ...
    'y[n] = nx[n] = 1')
```

> (g). $y[n] = x[2n]$.

Firstly, integer sequence `n` is generated for representing time. Then by matrix operation, we defined input signal `x` to be $x[n] = n^2$.

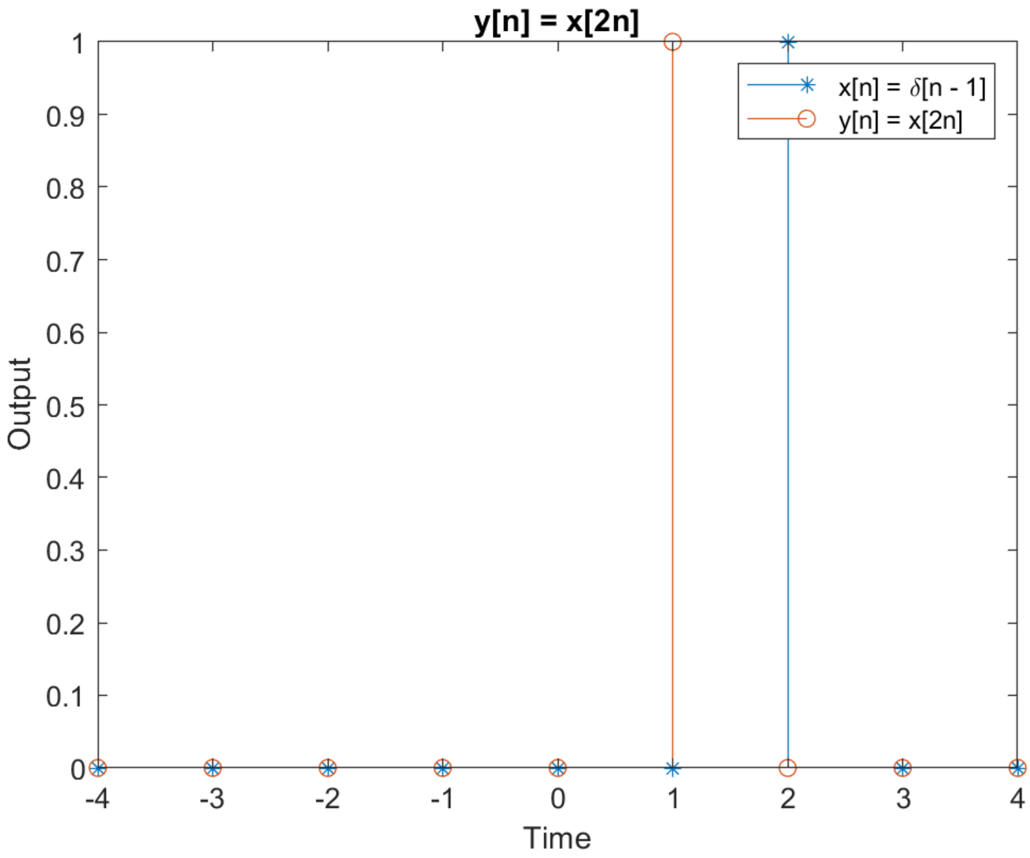
The plot is shown below.



According to the figure, $y_2[n] = x[2n - t]$ is not identical with $y_3[n] = y_1[n - t] = x[2(n - t)]$, hence the system is time-variant.

Define input signal $x[n] = \delta[n - 2]$, which is an impulse at time $n = 2$, by defining matrix.

The plot is shown below.



According to the figure, the impulse input is at time $n = 2$, however, there is response at time $n = 1$, which is before the excitation, hence the system is not causal.

The system is linear, stable and invertible.

The MATLAB script for proving the time-variance is shown in the code block below.

```
% define scope
l = 10;

% 1.4 advanced problem g
% time-invariance

n = [-l:l]
t = 2
y1 = L(n)
nt = n - t;
y2 = [zeros(1:t) y1(1, t + 1:2 * l + 1)]
y3 = L(nt)


stem(n, y1, '*');
hold on
stem(n, y2);
hold on
stem(n, y3, '*');

...

%def function
function y = L(x)
    y = x.^2
```

```
end
```

The MATLAB script for proving the system is not causal is shown in the code block below.

```
% causality
l = 4;
n = [-l:l];
% define unit impulse input at n = 2
x1 = [zeros(1, l + 2) 1 zeros(1, l - 2)]

nt = 1:2:2 * l + 1
y4 = [zeros(1, l / 2) x1(nt) zeros(1, l / 2)]

stem(n, x1, '*')
hold on
stem(n, y4)
```

1.5

> $$y[n] = ay[n-1] + x[n]. \tag{1.6}$$

Advanced Problem a

> Write a function `y = diffeqn(a, x, yn1)` which computes the output $y[n]$ of the causal system determined by Eq. (1.6). The input vector `x` contains $x[n]$ for $0 \leq n \leq N-1$ and `yn1` supplies the value of $y[-1]$. The output vector `y` contains y[n] for $0 \leq n \leq N-1$. The first line of your M-file should read
>
> ```
> function y = diffeqn(a, x, yn1)
> ```
>
> Hint: Note that $y[-1]$ is necessary for computing $y[0]$, which is the first step of the autoregression. Use a `for` loop in your M-file to compute $y[n]$ for successively larger values of n, starting with n = 0.

The function is shown in the code block below.

```
function y = diffeqn(a, x, yn1)
    N = length(x)
    y(1) = a * yn1 + x(1)

    for index = 2:N
        y(index) = a * y(index - 1) + x(index)
    end

end
```

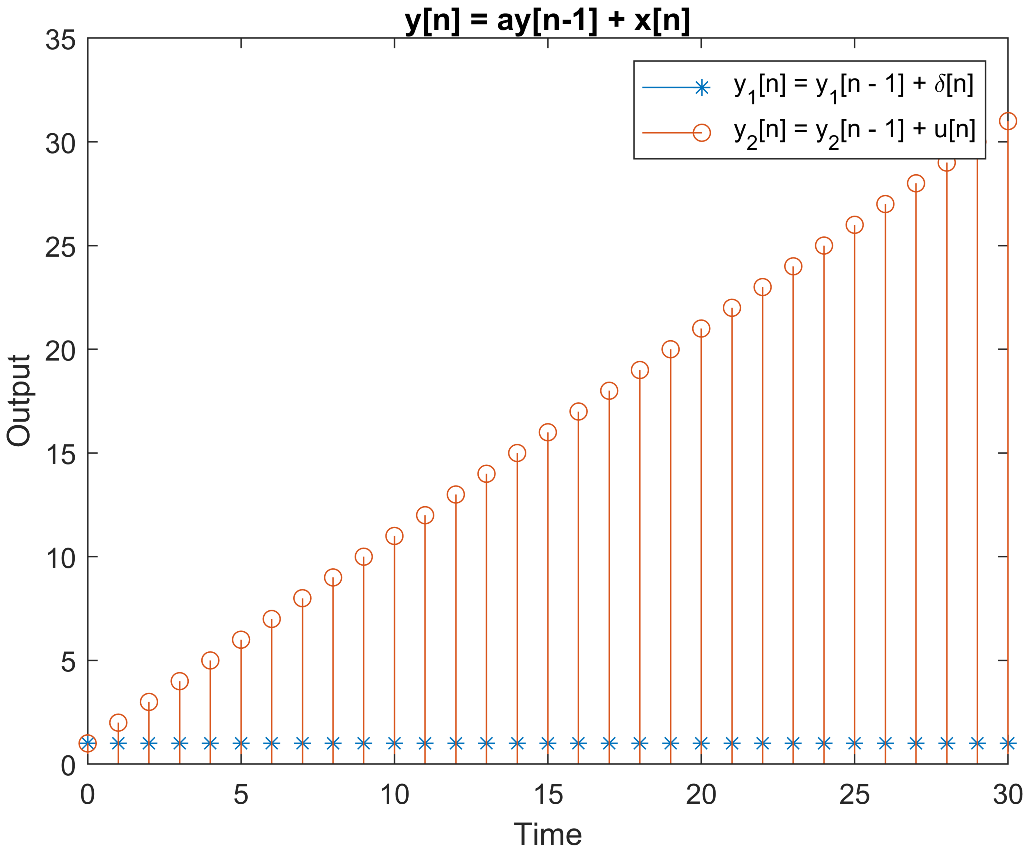
Here's the interpretation of this function.

1. read in the length of the input function as `N`.
2. assign value to `y(1)`.
3. using a `for` loop to assign values to following elements in the output.

Advanced Problem b

> Assume that $a = 1$, $y[-1] = 0$, and that we are only interested in the output over the interval $0 \le n \le 30$. Use your function to compute the response due to $x_1[n] = u[n]$ and $x_2[n] = u[n]$, the unit impulse and unit step, respectively. Plot each response using `stem`.

Using the function define in problem a to plot $y[n]$, the figure is shown below.



The MATLAB script is shown in the code block below.

```
a = 1;
n = 0:1:30;
x1 = [1 zeros(1, 30)];
x2 = ones(1, 31);

y1 = diffeqn(a, x1, yn1)
y2 = diffeqn(a, x2, yn1)

stem(n, y1, '*')
hold on
stem(n, y2)
```

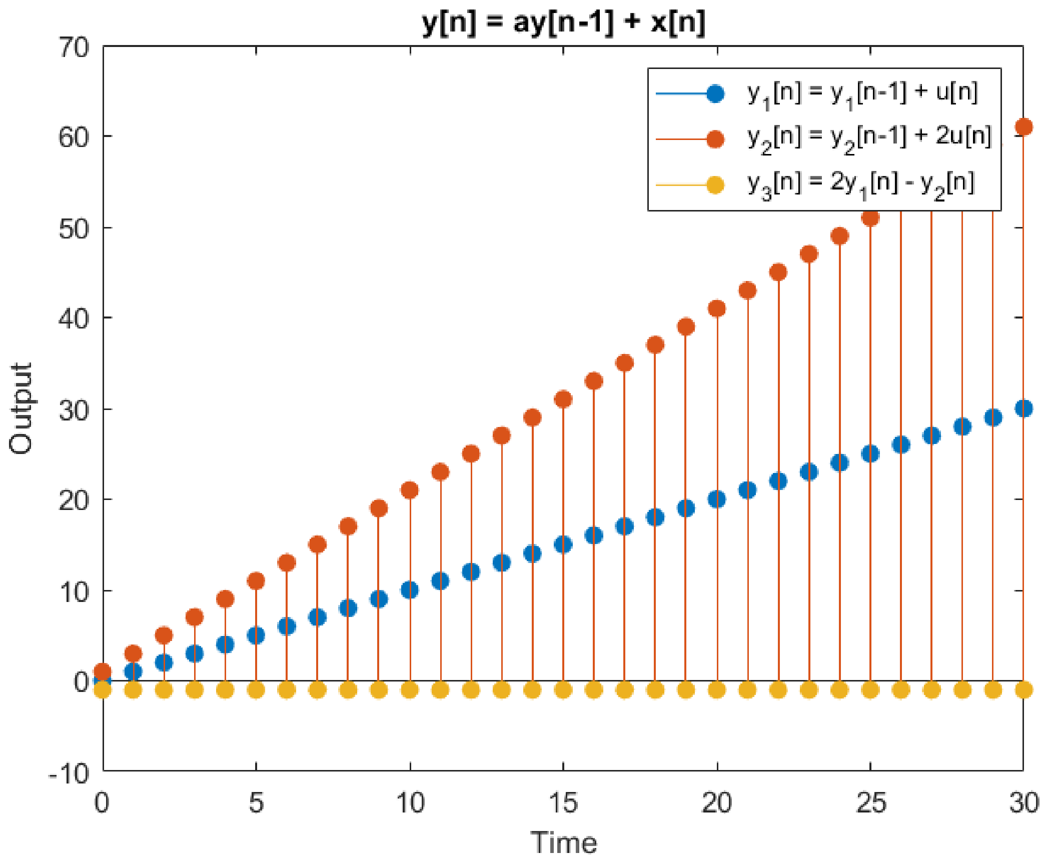
Advanced Problem c

> Assume again that $a = 1$, but that $y[-1] = -1$. Use your function to compute $y[n]$ over $0 \le n \le 30$ when the inputs are $x_1[n] = u[n]$ and $x_2[n] = 2u[n]$. Define the outputs produced by the two signals to be $y_1[n]$ and $y_2[n]$, respectively. Use `stem` to display both outputs. Use stem to plot $(2y_1[n] - y_2[n])$. Given that $\mathrm{Eq.}\ (1.6)$ is a linear difference equation, why isn't this difference identically zero?

The MATLAB script is shown below.

```
a = 1;
yn1 = -1;
x1 = ones(1, 31);
x2 = 2 * x1;

y1 = diffeqn(a, x1, yn1)
y2 = diffeqn(a, x2, yn1)
y3 = y1 .* 2 - y2

stem(n, y1, '')
hold on
stem(n, y2, '')
hold on
stem(n, y3, '')
```

The plot is shown below.



The expression of difference can be transferred into

$$\begin{aligned}2y_1[n]-y_2[n] =&2y_1[n-1]+2u[n]-y_2[n-1]-2u[n]\\ =&2y_1[n-1]-y_2[n-1]\\ =&\cdots\\ =&2y_1[-1]-y_2[-1]\\ =&-1\end{aligned}$$

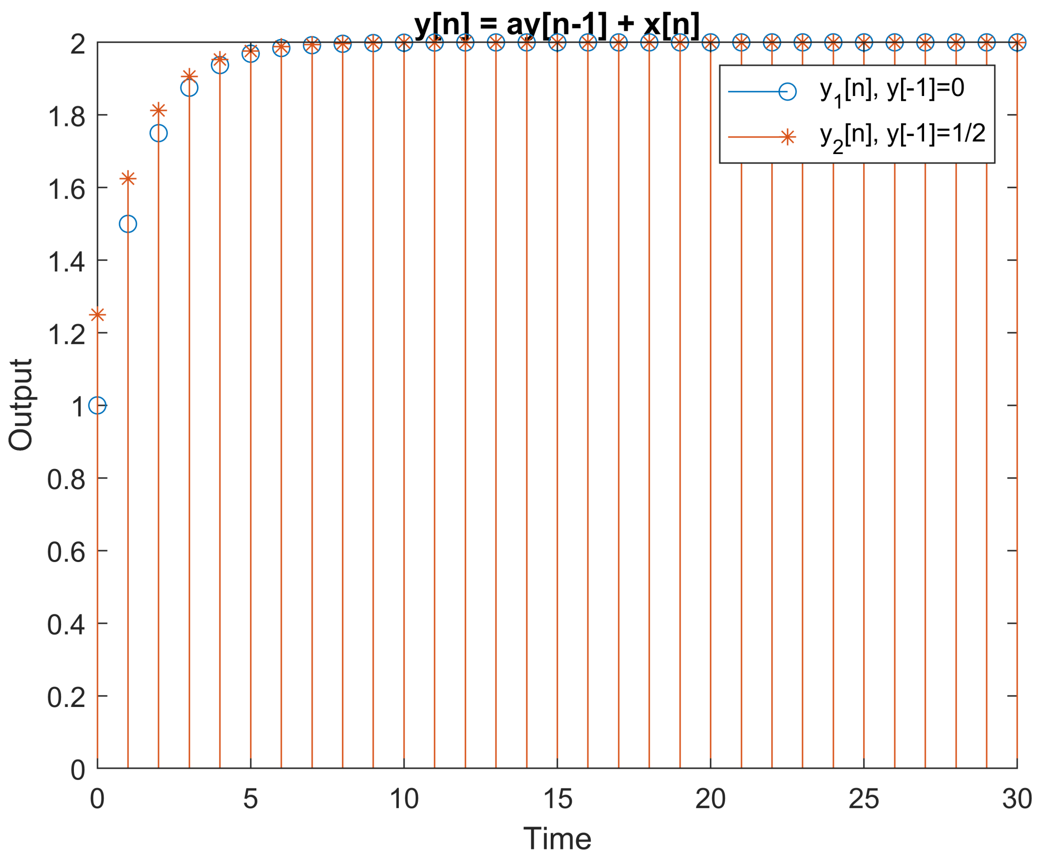
Hence, the signal $y_3[n]=2y_1[n]-y_2[n]$ is a non-zero constant.

Advanced Problem d

> The causal systems described by $\mathrm{Eq.}\ (1.6)$ are BIBO (bounded-input bounded-output) stable whenever $|a| < 1$. A property of these stable systems is that the effect of the initial condition becomes insignificant for sufficiently large n. Assume $a = 1/2$ and that `x` contains $x[n] = u[n]$ for $0 \le n \le 30$. Assuming both $y[-1] = 0$ and $y[-1] = 1/2$, compute the two output signals $y[n]$ for $0 \le n \le 30$. Use `stem` to display both responses. How do they differ?

The MATLAB script is shown in the code block below.

```
a = 1/2;
yn1_1 = 0;
yn1_2 = 1/2;

x = ones(1, 31);

y1 = diffeqn(a, x, yn1_1)
y2 = diffeqn(a, x, yn1_2)

stem(n, y1)
hold on
stem(n, y2, '*')
```



According to the plot, the magnitude of signal $y_1[n]$ is less than $y_2[n]$ at the beginning, and the two signals both approaches the same value.