Lab 1：Introduction

Author	Name & Student ID：王卓扬（12112907）、冯彦捷（12010825）

Introduction

1. Use MATLAB to represent DT Unit Impulse Signal and DT Unit Step Signal.
2. Use MATLAB to represent arbitrary DT Signals.
3. Use MATLAB to apply simple transformations to the DT signals, like time-shifting and time reversing, etc.
4. Use MATLAB to verify the fundamental properties of various systems, like linearity, time variance, stability, causality and invertibility.
5. Use simple linear constant-coefficient difference equations (like the first-order autoregression equations) to describe the DT systems.

Lab results & Analysis：

1.4 Properties of Discrete-Time Systems

Discrete-time systems are often characterized in terms of a number of properties such as linearity, time invariance, stability, causality, and invertibility. It is important to understand how to demonstrate when a system does or does not satisfy a given property. MATLAB can be used to construct counter-examples demonstrating that certain properties are not satisfied. In this exercise, you will obtain practice using MATLAB to construct such counter examples for a variety of systems and properties.

Basic Problems

For these problems, you are told which property a given system does not satisfy, and the input sequence or sequences that demonstrate clearly how the system violates the property. For each system, define MATLAB vectors representing the input(s) and output(s). Then, make plots of these signals, and construct a well reasoned argument explaining how these figures demonstrate that the system fails to satisfy the property in question.

Question (a)

The system $y[n] = \sin((\pi / 2) * x[n])$ is not linear. Use the signals $x1[n] = \delta[n]$ and $x2[n] = 2 * \delta[n]$ to demonstrate how the system violates linearity.

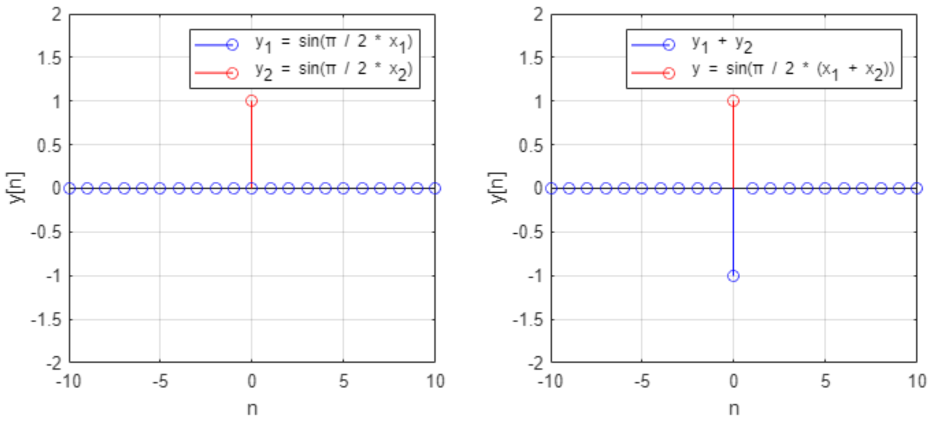

Fig 1.4 (a)

Analysis (a): We find that when the input signal is x1[n] + x2[n], the output signal is not y1[n] + y2[n] (Fig 1.4 (a)), so the system is **not linear**.

Question (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 <= n <= 9, and the output on the interval -6 <= n <= 9, respectively.

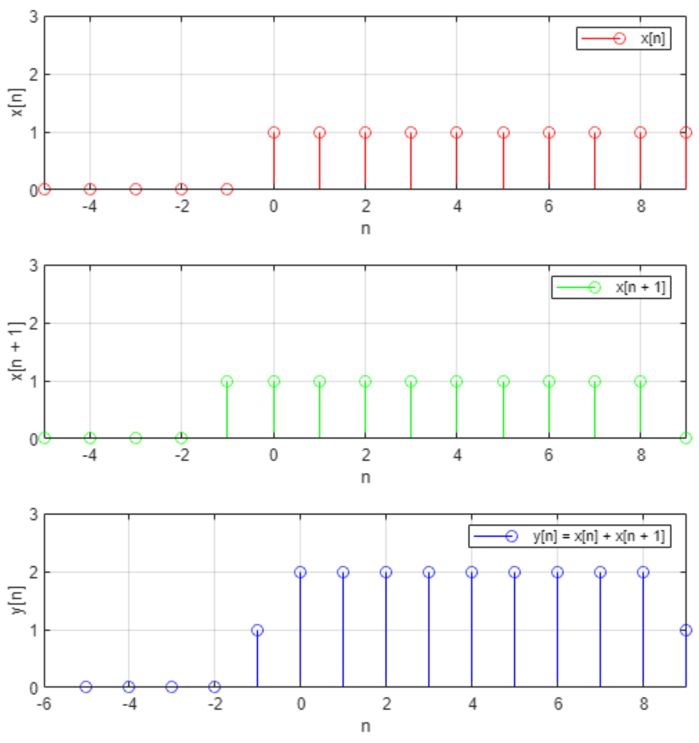

Fig 1.4 (b)

Analysis (b): If the system is causal, y[n] should be always zero when n < 0. However, y[-1] = 1 (Fig 1.4 (b)), that means the system is **not causal**, whose output is partially determined by future inputs.

Intermediate Problems

For these problems, you will be given a system and a property that the system does not satisfy, but must discover for yourself an input or pair of input signals to base your argument upon. Again, create MATLAB vectors to represent the inputs and outputs of the system and generate appropriate plots with these vectors. Use your plots to make a clear and concise argument about why the system does not satisfy the specified property.

Question (c)

The system y[n] = log(x[n]) is not stable.

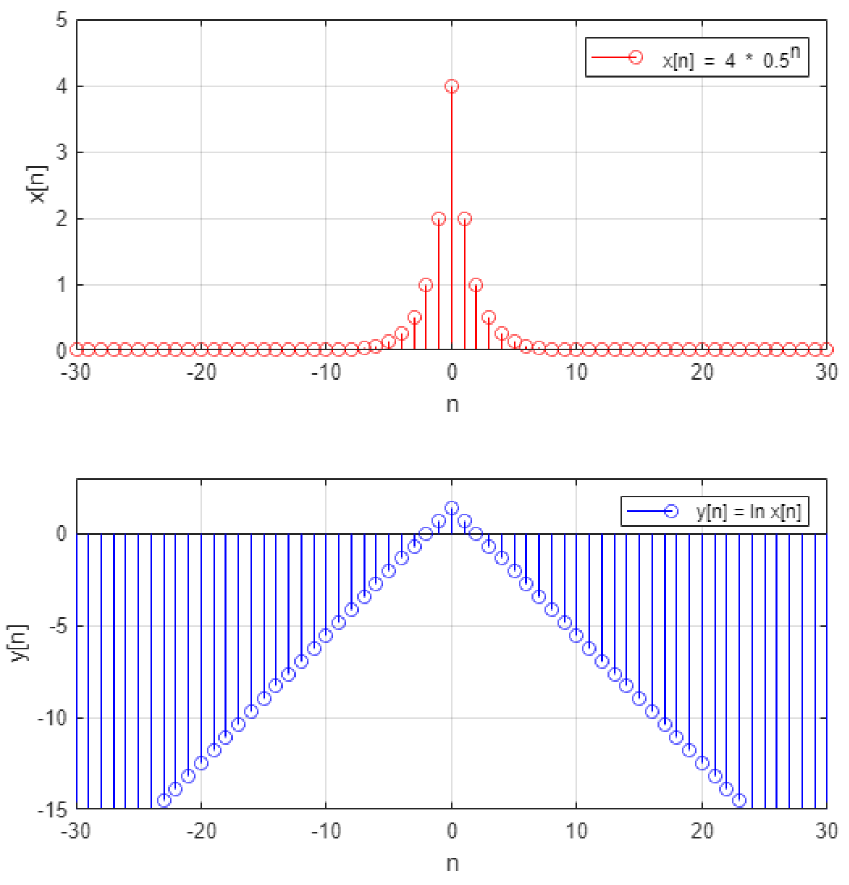

Fig 1.4 (c)

Analysis (c): We use x[n] = 4 * 0.5 ^ n as the input signal, which is bounded. But the output signal y[n] is not bounded (Fig 1.4 (c)). So the system is **not stable**.

Question (d)

The system given in Part (a) is not invertible.

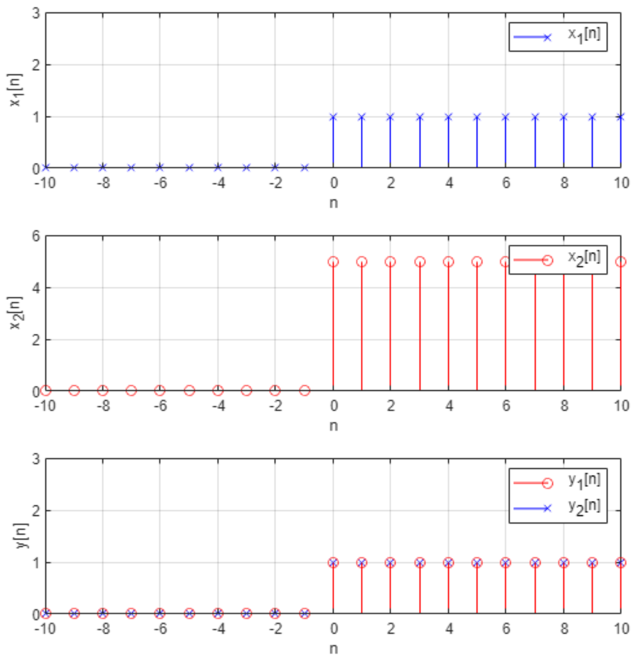

Fig 1.4 (d)

Analysis (d): We take different inputs x1[n] = u[n] and x2[n] = 5u[n], but they produce the same y[n] (Fig 1.4 (d)), which means the system's IO is not one-to-one, that is, not invertible.

Advanced Problems

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.

Question (e)

y[n] = (x[n]) ^ 3.

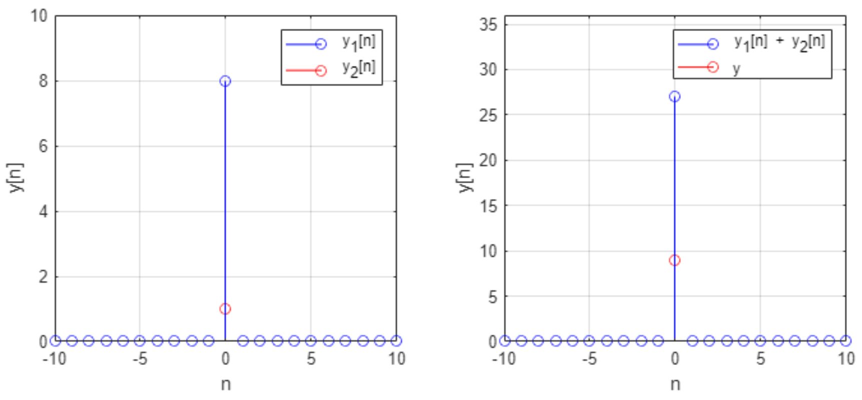

Fig 1.4 (e)

Analysis (e): We take x1[n] = δ[n] and x2[n] = 2δ[n]. And then we find that when the input signal is x1[n] + x2[n], the output signal is not y1[n] + y2[n] (Fig 1.4 (e)), so the system is **not linear**.

Question (f)

y[n] = n * x[n].

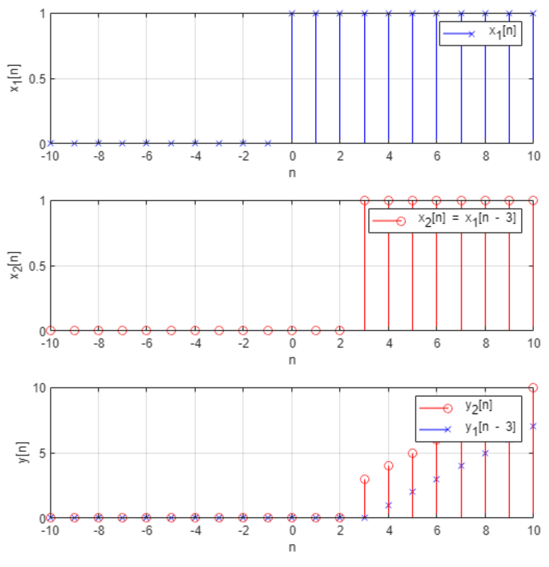

Fig 1.4.1 (f)

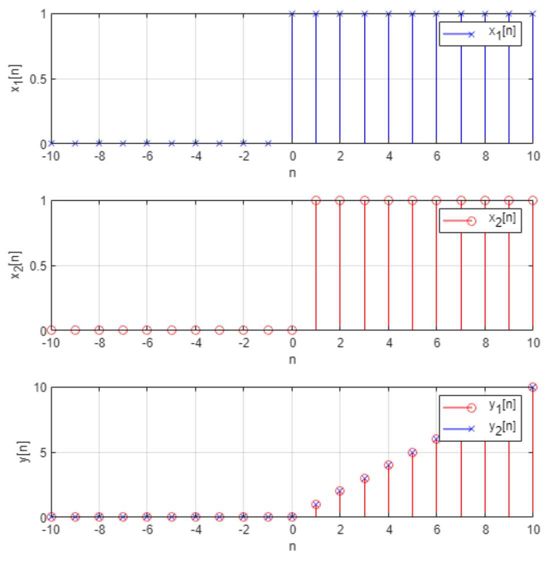

Fig 1.4.2 (f)

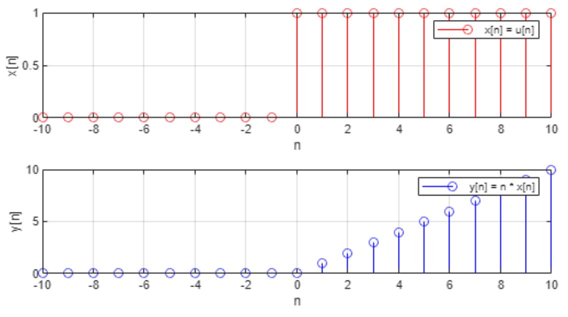

Fig 1.4.3 (f)

Analysis (f):

(1) In Fig 1.4.1 (f), we see when x2[n] = x1[n – 3] = u[n – 3], y2[n] ≠ y1[n – 3], which means that the system is **time-varying**.

(2) In Fig 1.4.2 (f), we can find that when we take different x1[n] = u[n], x2[n] = u[n – 1], the outputs remain the same. So the system is **not invertible**.

(3) In Fig 1.4.3 (f), the input signal x[n] = u[n] is bounded, while the output signal is not bounded, which indicates that the system is **not stable**.

Question (g)

y[n] = x[2 * n].

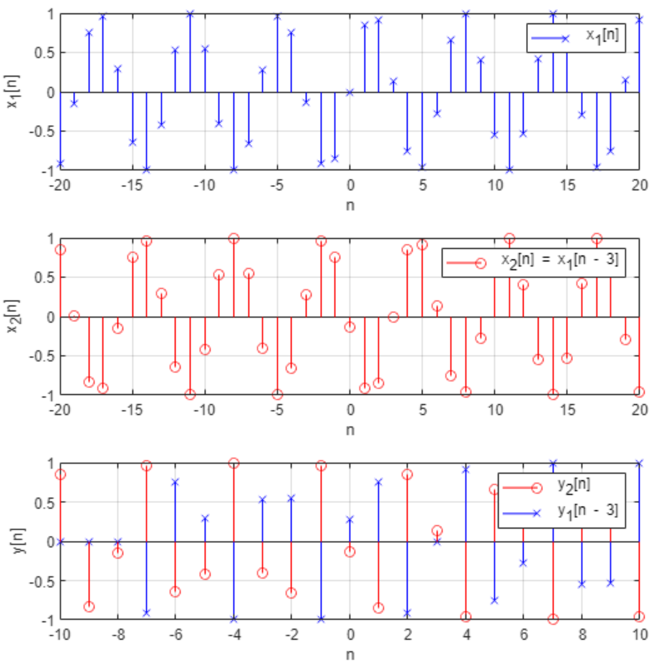

Fig 1.4.1 (g)

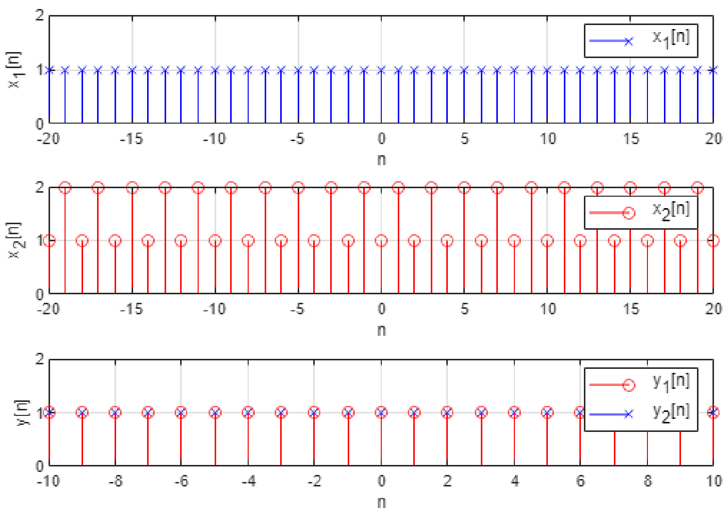

Fig 1.4.2 (g)

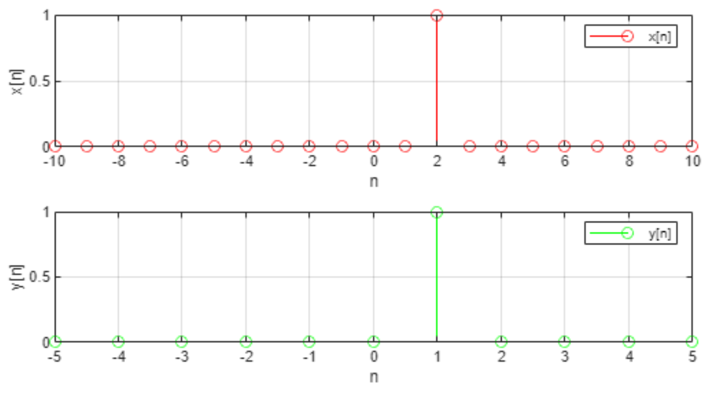

Fig 1.4.3 (g)

Analysis (g):

(1) In Fig 1.4.1 (g), apply a time shift to a sinusoidal signal x1[n] = sin(n) we get x2[n] = x1[n - 3], and we see that y2[n] ≠ y1[n - 3]. So the system is **time-varying**.

(2) In Fig 1.4.2 (g), we can find that when we take different x1[n] and x2[n] as shown in the figure, the outputs remain the same. So the system is **not invertible**.

(3) In Fig 1.4.3 (g), if the system is causal, y[n] should be always zero when n < 2. However, y[1] = 1 (Fig 1.4 (b)), that means the system is **not causal**, whose output is partially determined by future inputs.

1.5 Implementing a First-Order Difference Equation

Discrete-time systems are often implemented with linear constant-coefficient difference equations. Two very simple difference equations are the first-order moving average

y[n] = x[n] + b * x[n - 1], (1.5)

and the first-order autoregression

y[n] = a * y[n - 1] + x[n]. (1.6)

Even these simple systems can be used to model or approximate a number of practical systems. For instance, the first-order autoregression can be used to model a bank account, where y[n] is the balance at time n, x[n] is the deposit or withdrawal at time n, and a = l + r is the compounding due to interest rate r. In this exercise, you will be asked to write a function which implements the first-order autoregression equation. You will then be asked to test and analyze your function on some example systems.

Advanced Problems

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.

Question (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 <= n <= N - 1 and yn1 supplies the value of y[-1]. The output vector y contains y[n] for 0 <= n <= N - 1. The first line of your M-file should read *function y = diffeqn(a, x, yn1)*

Here’s the code:

```
% diffeqn.m
function y = diffeqn(a, x, yn1)
    y = a * yn1 + x(1);
    for k = x(2 : end) y = [y, a * y(end) + k]; end
end
```

Code 1.5.1 (a)

And a recursive version:

```
% diffeqn.m - Recursive version
function y = diffeqn(a, x, yn1)
    if length(x) == 1
        y = a * yn1 + x(end);
    else
        y = diffeqn(a, x(1 : end - 1), yn1);
        y = [y, a * y(end) + x(end)];
    end
end
```

Code 1.5.2 (a)

Analysis (a): None.

Question (b)

Assume that a = 1, y[-1] = 0, and that we are only interested in the output over the interval 0 <= n <= 30. Use your function to compute the response due to x1[n] = δ[n] and x2[n] = u[n], the unit impulse and unit step, respectively. Plot each response using *stem*.

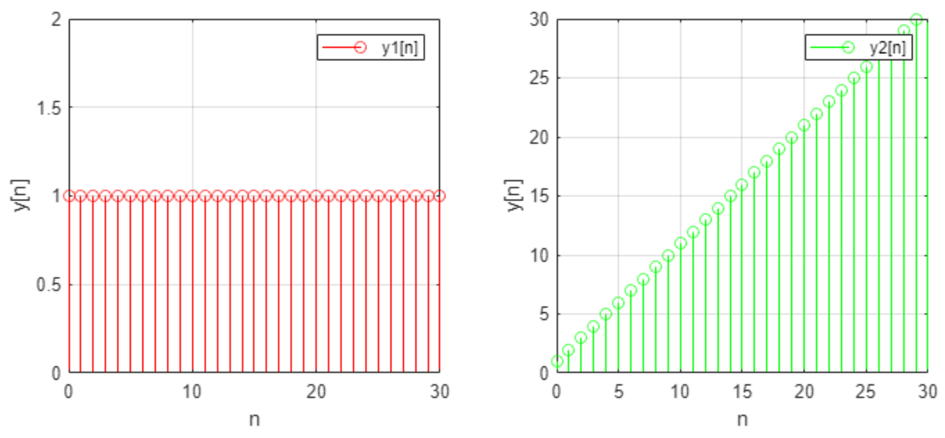

Fig 1.5 (b)

Analysis (b): In Fig 1.5 (b), the left is the response of the input x1[n] = δ[n] and the right is the response of the input x2[n] = u[n].

Question (c)

Assume again that a = 1, but that y[-1] = -1. Use your function to compute y[n] over 0 <= n <= 30 when the inputs are x1[n] = u[n] and x2[n] = 2 * u[n]. Define the outputs produced by the two signals to be y1[n] and y2[n], respectively. Use *stem* to display both outputs. Use *stem* to plot (2 * y1[n] - y2[n]). Given that Eq. (1.6) is a linear difference equation, why isn't this difference identically zero?

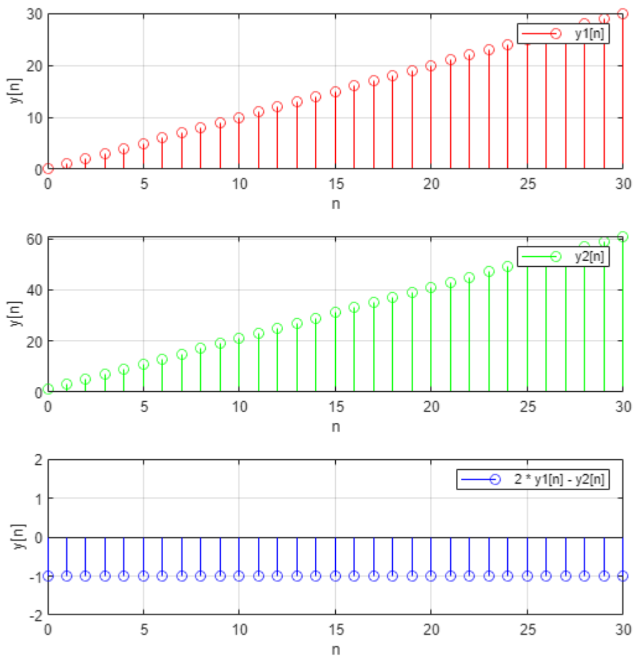

Fig 1.5 (c)

Analysis (c): The results are shown in Fig 1.5 (c). Why isn't this difference identically zero? Because the constant a is not zero. So **y1[n] * 2 – y2[n] = 2ay1[n – 1] – ay2[n – 1] = y[–1] = -1**.

Question (d)

The causal systems described by 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 <= n <= 30. Assuming both y[-1] = 0 and y[-1] = 1 / 2, compute the two output signals y[n] for 0 <= n <= 30. Use *stem* to display both responses. How do they differ?

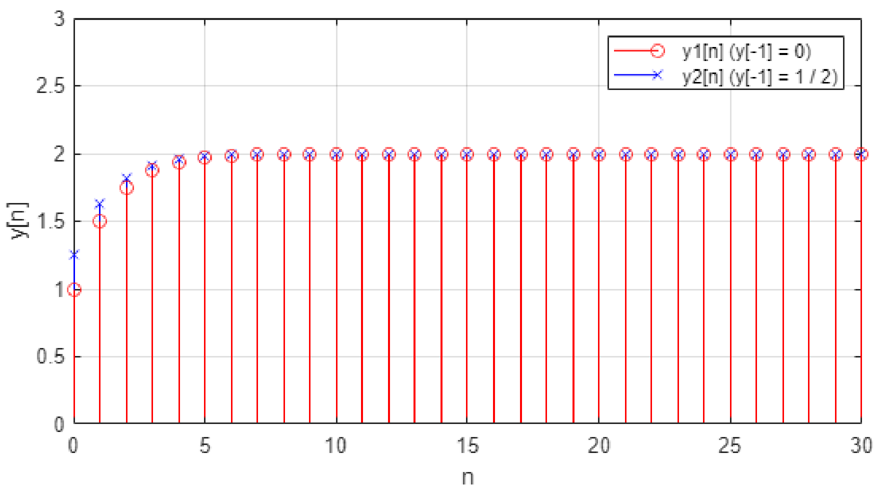

Fig 1.5 (d)

Analysis (d): Let a = 1 / 2, x[n] = u[n]. The curves of the responses when y[-1] = 0 and y[-1] = 1 / 2, start from their respective y[-1] and both converge to 2 at n → infinity. They differ in the start points.

Note: Please indicate meaning of the symbols in all expressions. Please indicate the coordinate and unit in all figures.

Experience

(1) It is better to define a vector n (or other name) to describe the domains of the DT signals because the vectors always start from the index 1.

(2) The order of command *stems* is opposite to the order of legends.

(3) Don't forget to add an *end* at the end of the body of *for*, *if*, *etc*.

(4) It is better to package the frequently-used codes into individual functions.

(5) Templates for plotting and basic signal transforming can be developed in the future.

Score	

Appendix

Fundamental Signal Generators

```
function delta = Impulse(n) % DT Unit Impulse Signal
    delta = zeros(1, length(n));
    delta(n == 0) = 1;
end
```

```
function u = Step(n) % DT Unit Step Signal
    u = zeros(1, length(n)) + (n >= 0);
end
```

1.4 (a)

```
clear;
l = -10;
r = 10;
n = l : r;
System_a = @(x) sin((pi / 2) * x);

x1 = Impulse(n);
x2 = Impulse(n) * 2;
y1 = System_a(x1);
y2 = System_a(x2);
y = System_a(x1 + x2);

figure(1);
set(gcf, 'position', [0 0 800 320]);
```

```
d = [l, r];

subplot(1, 2, 1);
stem(n, y1, 'r'), stem(n, y2, 'b');
legend('y_1 = sin(\pi / 2 * x_1)', 'y_2 = sin(\pi / 2 * x_2)');
hold on, grid on, xlim(d), ylim([-2 2]), ylabel('y[n]'), xlabel('n');

subplot(1, 2, 2);
stem(n, y1 + y2, 'r'), stem(n, y, 'b');
legend('y_1 + y_2', 'y = sin(\pi / 2 * (x_1 + x_2))');
hold on, grid on, xlim(d), ylim([-2 2]), ylabel('y[n]'), xlabel('n');
```

1.4 (b)

```
clear;
l = -5;
r = 9;
n = l : r;
System_b = @(x) x + circshift(x, -1);

x = Step(n);
y = System_b(x);

figure(2);
set(gcf, 'position', [0 0 600 600]);
d = [l, r];
D = [l - 1, r];

subplot(3, 1, 1);
stem(n, x, 'r');
legend('x[n]');
grid on, xlim(d), ylim([0 3]), ylabel('x[n]'), xlabel('n');

subplot(3, 1, 2);
stem(n, circshift(x, -1), 'g');
legend('x[n + 1]');
grid on, xlim(d), ylim([0 3]), ylabel('x[n + 1]'), xlabel('n');

subplot(3, 1, 3);
stem(n, y, 'b');
legend('y[n] = x[n] + x[n + 1]');
grid on, xlim(D), ylim([0 3]), ylabel('y[n]'), xlabel('n');
```

1.4 (c)

```
clear;
```

```
l = -30;
r = 30;
n = l : r;
System_c = @(x) log(x);

x = 4 * 0.5 .^ abs(n);
y = System_c(x);

figure(3);
set(gcf, 'position', [0 0 600 600]);
d = [l, r];

subplot(2, 1, 1);
stem(n, x, 'r');
legend('x[n] = 4 * 0.5^n');
grid on, xlim(d), ylim([0 5]), ylabel('x[n]'), xlabel('n');

subplot(2, 1, 2);
stem(n, y, 'b');
legend('y[n] = ln x[n]');
grid on, xlim(d), ylim([-15 3]), ylabel('y[n]'), xlabel('n');
```

1.4 (d)

```
clear;
l = -10;
r = 10;
n = l : r;
System_d = @(x) sin((pi / 2) * x);

x1 = Step(n);
x2 = Step(n) * 5;
y1 = System_d(x1);
y2 = System_d(x2);

figure(4);
set(gcf, 'position', [0 0 600 600]);
d = [l, r];

subplot(3, 1, 1);
stem(n, x1, 'bx');
legend('x_1[n]');
grid on, xlim(d), ylim([0 3]), ylabel('x_1[n]'), xlabel('n');

subplot(3, 1, 2);
```

```
stem(n, x2, 'r');
legend('x_2[n]');
grid on, xlim(d), ylim([0 6]), ylabel('x_2[n]'), xlabel('n');

subplot(3, 1, 3);
stem(n, y1, 'bx'), stem(n, y2, 'r');
legend('y_1[n]', 'y_2[n]');
hold on, grid on, xlim(d), ylim([0 3]), ylabel('y[n]'), xlabel('n');
```

1.4 (e)

```
clear;
l = -10;
r = 10;
n = l : r;
System_e = @(x) x .^ 3;

x1 = Impulse(n);
x2 = Impulse(n) * 2;
y1 = System_e(x1);
y2 = System_e(x2);
y = System_e(x1 + x2);

figure(5);
set(gcf, 'position', [0 0 800 320]);
d = [l, r];

subplot(1, 2, 1);
stem(n, y1, 'r'), stem(n, y2, 'b');
legend('y_1[n]', 'y_2[n]');
hold on, grid on, xlim(d), ylim([0 10]), ylabel('y[n]'), xlabel('n');

subplot(1, 2, 2);
stem(n, y1 + y2, 'r'), stem(n, y, 'b');
legend('y_1[n] + y_2[n]', 'y');
hold on, grid on, xlim(d), ylim([0 36]), ylabel('y[n]'), xlabel('n');
```

1.4 (f) Time-varying

```
clear;
l = -10;
r = 10;
offset = -3;
n = l : r;
System_f = @(x) n .* x;
```

```
x1 = Step(n);
x2 = Step((l + offset) : (r + offset));
y1 = System_f(x1);
y2 = System_f(x2);

figure(6);
set(gcf, 'position', [0 0 600 600]);
d = [l, r];

subplot(3, 1, 1);
stem(n, x1, 'bx');
legend('x_1[n]');
grid on, xlim(d), ylim([0 1]), ylabel('x_1[n]'), xlabel('n');

subplot(3, 1, 2);
stem(n, x2, 'r');
legend('x_2[n] = x_1[n - 3]');
grid on, xlim(d), ylim([0 1]), ylabel('x_2[n]'), xlabel('n');

subplot(3, 1, 3);
stem(n, [zeros(1, abs(offset)), y1(1 : end + offset)], 'bx'), stem(n, y2,
'r');
legend('y_2[n]', 'y_1[n - 3]');
hold on, grid on, xlim(d), ylim([0 10]), ylabel('y[n]'), xlabel('n');
```

1.4 (f) Non-invertible

```
clear;
l = -10;
r = 10;
n = l : r;
System_f = @(x) n .* x;

x1 = Step(n);
x2 = Step((l - 1) : (r - 1));
y1 = System_f(x1);
y2 = System_f(x2);

figure(7);
set(gcf, 'position', [0 0 600 600]);
d = [l, r];

subplot(3, 1, 1);
stem(n, x1, 'bx');
legend('x_1[n]');
```

```
grid on, xlim(d), ylim([0 1]), ylabel('x_1[n]'), xlabel('n');

subplot(3, 1, 2);
stem(n, x2, 'r');
legend('x_2[n]');
grid on, xlim(d), ylim([0 1]), ylabel('x_2[n]'), xlabel('n');

subplot(3, 1, 3);
stem(n, y1, 'bx'), stem(n, y2, 'r');
legend('y_1[n]', 'y_2[n]');
hold on, grid on, xlim(d), ylim([0 10]), ylabel('y[n]'), xlabel('n');
```

1.4 (f) Non-stable

```
clear;
l = -10;
r = 10;
n = l : r;
System_f = @(x) n .* x;

x = Step(n);
y = System_f(x);

figure(8);
set(gcf, 'position', [0 0 600 300]);
d = [l, r];

subplot(2, 1, 1);
stem(n, x, 'r');
legend('x[n] = u[n]');
grid on, xlim(d), ylim([0 1]), ylabel('x[n]'), xlabel('n');

subplot(2, 1, 2);
stem(n, y, 'b');
legend('y[n] = n * x[n]');
grid on, xlim(d), ylim([0 10]), ylabel('y[n]'), xlabel('n');
```

1.4 (g) Time-varying

```
clear;
l = -10;
r = 10;
offset = -3;
n = l : r;
N = (l * 2) : (r * 2);
O = (l * 2 + offset) : (r * 2 + offset);
```

```
System_g = @(x) x(mod(N, 2) == 0);

x1 = sin(N);
x2 = sin(O);
y1 = System_g(x1);
y2 = System_g(x2);

figure(9);
set(gcf, 'position', [0 0 600 600]);
S = [l * 2 + offset, r * 2 + offset];
D = [l * 2, r * 2];
d = [l, r];

subplot(3, 1, 1);
stem(N, x1, 'bx');
legend('x_1[n]');
grid on, xlim(D), ylim([-1 1]), ylabel('x_1[n]'), xlabel('n');

subplot(3, 1, 2);
stem(N, x2, 'r');
legend('x_2[n] = x_1[n - 3]');
grid on, xlim(D), ylim([-1 1]), ylabel('x_2[n]'), xlabel('n');

subplot(3, 1, 3);
stem(n, [zeros(1, abs(offset)), y1(1 : end + offset)], 'bx'), stem(n, y2,
'r');
legend('y_2[n]', 'y_1[n - 3]');
hold on, grid on, xlim(d), ylim([-1 1]), ylabel('y[n]'), xlabel('n');
```

1.4 (g) Non-invertible

```
clear;
l = -10;
r = 10;
n = l : r;
N = (l * 2) : (r * 2);
System_g = @(x) x(mod(N, 2) == 0);

x1 = ones(1, length(N));
x2 = ones(1, length(N)) + (mod(N, 2) == 1);
y1 = System_g(x1);
y2 = System_g(x2);

figure(10);
set(gcf, 'position', [0 0 600 400]);
```

```
D = [l * 2, r * 2];
d = [l, r];

subplot(3, 1, 1);
stem(N, x1, 'bx');
legend('x_1[n]');
grid on, xlim(D), ylim([0 2]), ylabel('x_1[n]'), xlabel('n');

subplot(3, 1, 2);
stem(N, x2, 'r');
legend('x_2[n]');
grid on, xlim(D), ylim([0 2]), ylabel('x_2[n]'), xlabel('n');

subplot(3, 1, 3);
stem(n, y1, 'bx'), stem(n, y2, 'r');
legend('y_1[n]', 'y_2[n]');
hold on, grid on, xlim(d), ylim([0 2]), ylabel('y[n]'), xlabel('n');
```

1.4 (g) Non-causal

```
clear;
l = -5;
r = 5;
n = l : r;
N = (l * 2) : (r * 2);
O = (l * 2 - 2) : (r * 2 - 2);
System_g = @(x) x(mod(N, 2) == 0);

x = Impulse(O);
y = System_g(x);

figure(11);
set(gcf, 'position', [0 0 600 300]);
D = [l * 2, r * 2];
d = [l, r];

subplot(2, 1, 1);
stem(N, x, 'r');
legend('x[n]');
grid on, xlim(D), ylim([0 1]), ylabel('x[n]'), xlabel('n');

subplot(2, 1, 2);
stem(n, y, 'g');
legend('y[n]');
grid on, xlim(d), ylim([0 1]), ylabel('y[n]'), xlabel('n');
```

1.5 (b)

```
clear;

l = 0;
r = 30;
n = l : r;
a = 1;
yn1 = 0;

x1 = Impulse(n);
x2 = Step(n);
y1 = diffeqn(a, x1, yn1);
y2 = diffeqn(a, x2, yn1);

figure(1);
set(gcf, 'position', [0 0 800 320]);
d = [l, r];

subplot(1, 2, 1);
stem(n, y1, 'r');
legend('y1[n]');
grid on, xlim(d), ylim([0 2]), ylabel('y[n]'), xlabel('n');

subplot(1, 2, 2);
stem(n, y2, 'g');
legend('y2[n]');
grid on, xlim(d), ylim([0 30]), ylabel('y[n]'), xlabel('n');
```

1.5 (c)

```
clear;

l = 0;
r = 30;
n = l : r;
a = 1;
yn1 = -1;

x1 = Step(n);
x2 = Step(n) * 2;
y1 = diffeqn(a, x1, yn1);
y2 = diffeqn(a, x2, yn1);

figure(2);
```

```
set(gcf, 'position', [0 0 600 600]);
d = [l, r];

subplot(3, 1, 1);
stem(n, y1, 'r');
legend('y1[n]');
grid on, xlim(d), ylim([0 30]), ylabel('y[n]'), xlabel('n');

subplot(3, 1, 2);
stem(n, y2, 'g');
legend('y2[n]');
grid on, xlim(d), ylim([0 61]), ylabel('y[n]'), xlabel('n');

subplot(3, 1, 3);
stem(n, 2 * y1 - y2, 'b');
legend('2 * y1[n] - y2[n]');
grid on, xlim(d), ylim([-2 2]), ylabel('y[n]'), xlabel('n');
```

1.5 (d)

```
clear;

l = 0;
r = 30;
n = l : r;
a = 1 / 2;

x1 = Step(n);
x2 = Step(n);
y1 = diffeqn(a, x1, 0);
y2 = diffeqn(a, x2, 1 / 2);

figure(3);
set(gcf, 'position', [0 0 640 320]);
d = [l, r];

stem(n, y2, 'bx'), stem(n, y1, 'r');
legend('y1[n] (y[-1] = 0)', 'y2[n] (y[-1] = 1 / 2)');
hold on, grid on, xlim(d), ylim([0 3]), ylabel('y[n]'), xlabel('n');
```