EE323 Digital Signal Processing Lab2 Report:

Yuncheng Tu

Student ID: 12111008

***Abstract*—Discrete-time systems are fundamental tools for data processing. In this experiment, we explore the design and utilization of discrete-time systems represented by difference equations, such as integrators and differentiators. Through practical applications, we delve into the working principles and analyze the characteristics of these systems. By examining their output results, we compare the efficacy of different systems in addressing real-world challenges. This experiment offers valuable insights into the realm of digital signal processing and its practical applications.**

***Index Terms*—DSP, difference equation, impulse response, filter, differentiator, integrator, linearity, time-invarience.**

I. Introduction

DIGITAL signal processing (DSP) stands at the core of modern technology, enabling the manipulation and analysis of digital signals in various applications. At its heart, DSP revolves around discrete-time systems, which take discrete-time signals as input and produce corresponding output signals. These systems are often represented using difference equations or block diagrams, embodying the essence of signal processing.

In electrical engineering, continuous-time signals find their domain in electrical circuits, governed by intricate differential equations. Discrete-time signals, common in digital domains, require the specialized handling of discrete-time systems. These systems are designed to process signals with precision and are instrumental in various domains, from audio equalizers to economic analysis.

Difference equations play a pivotal role in defining the behavior of discrete-time systems. These equations explains the relationship between past and present values of signals, allowing for the transformation of input signals into meaningful output. Digital signal processing leverages both the mathematical elegance of difference equations and the computational power of modern computers, enabling the real-time manipulation and analysis of digital data. Understanding the classification and properties of discrete-time systems, including linear/nonlinear, time-invariant/time-varying, and stable/unstable, is essential to harness the full potential of digital signal processing in diverse applications.

II. Exercises

In many applications of electrical engineering, there is a compelling need to transition from continuous-time systems to discrete-time digital systems. This transition offers advantages such as improved quality and cost-effectiveness, facilitated by the utilization of standardized, high-volume digital processors.

A. Background Exercises

In the realm of signal processing, one fundamental operation is differentiation and integration. Continuous-time differentiators and integrators are expressed as:

1. Differentiator: $y[t] = \frac{d}{dt}x(t)$
2. Integrator:$y(t) = \int_{-\infty}^{t} x(\tau)\, d\tau$

A differentiator approximates the derivative of a continuous-time function, while an integrator computes the integral of the function over time. However, when transitioning to discrete-time systems, we encounter the need for suitable approximations and transformations to maintain accuracy and consistency in signal processing.

1) Example Discrete-time Systems

To approximate these continuous-time functions in a discrete-time system, we consider the discrete equivalents. Assuming a sampling interval denoted as 'T,' we approximate the derivative by computing the slope between two adjacent sampling points. Consequently, the discrete-time differentiator can be formulated as:

$$y[n] = \frac{x[n]-x[n-1]}{T}$$

Similarly, we estimate the integral value at a discrete point by multiplying the function's value at that point by the sampling interval. Thus, the discrete-time integrator is represented as:

$$y[n] = Tx[n] + y[n-1]$$

When the sampling period is equal to 1, these equations simplify to the following closed-form difference equation:

1. Differentiator: $y[n]=x[n]-x[n-1]$
2. Integrator: $y[n]-y[n-1]=x[n]$

The block diagrams of these two systems are as follows (Figure 2.1):

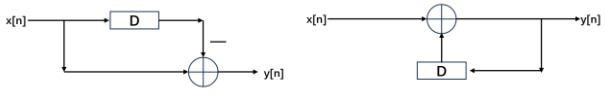

Fig. 1. (from right to left: DT differentiator, DT integrator)

2) Stock Market Example &Moving Average Filter

Digital Signal Processing (DSP) techniques are renowned for their versatility, as they can be applied to a diverse range of signals. Unlike most continuous-time systems, which primarily deal with voltage and current signals, DSP operates on discrete-time signals, essentially sequences of numbers. This fundamental difference enables DSP to find applications across a broad spectrum of fields. To illustrate this, consider the following example.

Imagine a stockbroker who seeks to determine whether the average value of a particular stock is trending upwards or downwards. To accomplish this, the stockbroker needs to filter out the daily fluctuations in the stock's value. In this context, three methods for computing the average are recommended, as follows:

method 2.3: avgvalue[today] =
(value[today]+value[yesterday]+value[2 days ago])/3

method 2.4: avgvalue[today] =
0.8 * avgvalue[yesterday] + 0.2 * (value[today])

method 2.5: avgvalue[today] =
avgvalue[yesterday] + (value[today]-(value[3 days ago])/3

Then comes the difference equation:

method 2.3: $y[n] = (x[n] + x[n-1] + x[n-2])/3$

method 2.4: $y[n] - 0.8 * y[n-1] = 0.2 * x[n]$

method 2.5: $y[n] - y[n-1] = (x[n] - x[n-3])/3$

The key feature shared by methods 2.3 and 2.5 is that they are known as "moving averages.", which is used in smoothing random variation in data.

In Method 2.3, the average is computed by taking the sum of the stock values for today, yesterday, and two days ago, and then dividing by 3. The window of data considered for the average is "moving" each day, as it shifts forward in time, incorporating the most recent data while discarding the oldest value.

In Method 2.5, the average is updated daily by taking the previous day's average and adding the difference between today's stock value and the stock value three days ago. This method also uses a "moving" window, as it adapts the weighting of recent data, giving more emphasis to the most recent information.

Then we draw impulse response diagrams of each system:

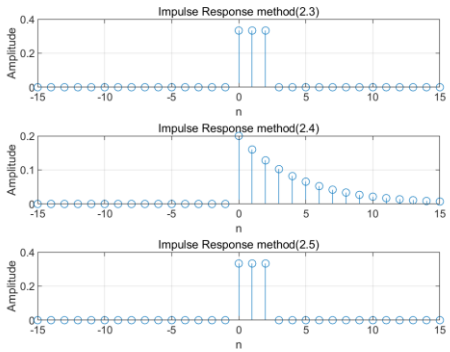

Fig. 2. Impulse response of three methods

We can observe that moving-average filters 2.3 and 2.5 share the same impulse response. We can demonstrate mathematically that 2.5 is a variation of 2.3, and they are fundamentally identical. The following is the mathematical proof:

$$y[n] = \frac{1}{M}\sum_{l=0}^{m-1} x\,[n-l]$$
$$= \frac{1}{M}\left(\sum_{l=1}^{M} x\,[n-l] + x[n] - x[n-M]\right)$$
$$= \frac{1}{n}\left(\sum_{l=0}^{M-1} X\,[n-L-1] + x[n] - x[n-M]\right)$$
$$= y[n-1] + \frac{1}{M}(x[n] - x[n-M])$$

Therefore, both Method 2.3 and Method 2.5 are known as "moving averages" because they involve the systematic updating of an average value over time, with the goal of extracting the underlying trend by attenuating the effects of short-term fluctuations. If there is no bias in measurements, an improved estimate of noise data is obtained by simply increasing M.

The block diagrams are as follows:

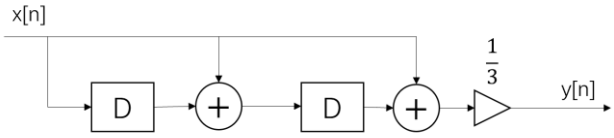

Fig. 3. Method 2.3

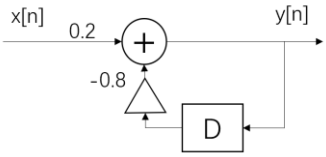

Fig. 4. Method 2.4

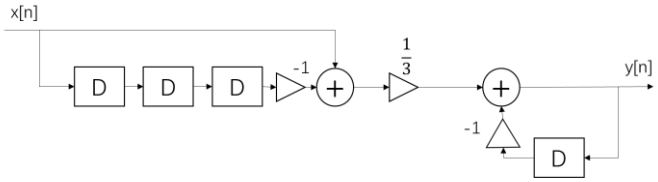

Fig. 5. Method 2.5

B. 2.3Example Discrete-Time Systems

In this analysis, we employed the differentiator and integrator systems implemented in the previously defined functions to process two distinct input signals, namely, the delta signal (δ[n] - δ[n - 5]) and the unit step signal u[n] - u[n - (N + 1)] for (N = 10). The outcome of these processes was then visualized in paired figures, providing an insightful perspective on their behavior(Figure 3).

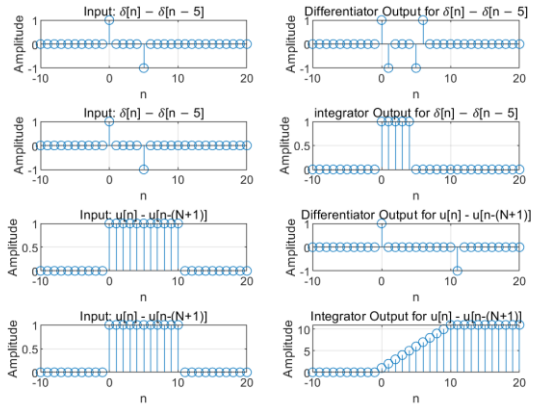

Fig. 6. Outputs signals of differentiator and integrator with two given signals as input

The differentiator system exhibited stability in its response. This stability is evident through its difference equation:

$y[n] = x[n] - x[n-1]$. In this formulation, each element of the output is the result of subtracting the previous element from the current element of the input signal. Consequently, when the input signal is bounded, the output signal is similarly bounded. This stability is corroborated by the results presented in the figures, which consistently reflect well-behaved responses.

In contrast, the integrator system demonstrated instability. The expression $y[n] = y[n-1] + x[n]$ in a discrete-time integrator signifies that the current output element is determined by summing the last output element with the present input element. This implies that each output element is the cumulative sum of all preceding input elements up to the current time. Consequently, when the input signal is bounded, there is no guarantee that the elements in the output signal will remain bounded. This instability is illustrated by the figures, which portray erratic behavior in response to the input signals.

C. 2.4Difference Equations

In this section, we delve into the effects of two discrete-time filters, denoted as S1 and S2. These filters are defined by distinct difference equations:

$$S1:\ y[n] = x[n] - x[n-1]$$

$$S2:\ y[n] = \frac{y[n-1]}{2} + x[n]$$

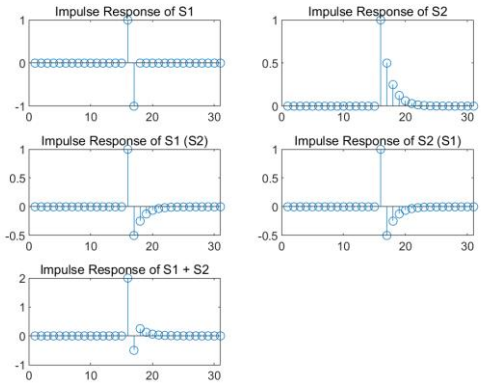

Fig. 7. Impulse response of these 5 systems

By visual inspection, it is readily apparent that the impulse response of $S1(S2)$ and $S2(S1)$ are identical. Through rigorous mathematical calculation, we confirm that they are equal to the convolution result of the impulse responses of $S1$ and $S2$, denoted as $S1 \otimes S2$

A visual analysis of the impulse response of the system $S1 + S2$ demonstrates that it equals the sum of the impulse responses of $S1$and $S2$. In other words, $S = S1 + S2$. This observation asserts that for systems in parallel relationships, the impulse response of the entire system is the summation of the impulse responses of each sub-system.

The block diagrams are as follows:

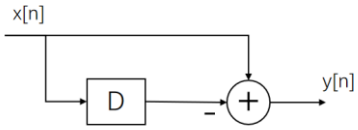

Fig. 8. $S1$: $y[n] = x[n] - x[n-1]$

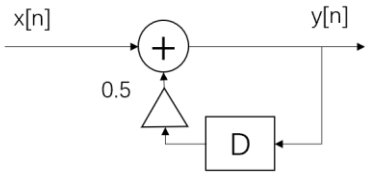

Fig. 9. $S2$: $y[n] = (y[n-1])/2 + x[n]$

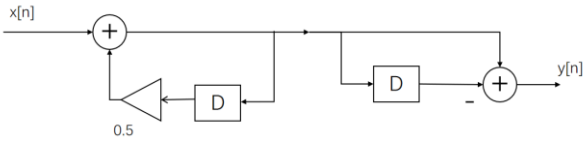

Fig. 10. S1(S2)

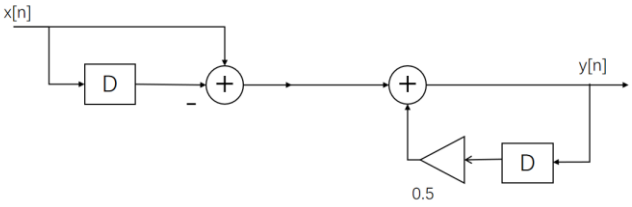

Fig. 11. S2(S1)

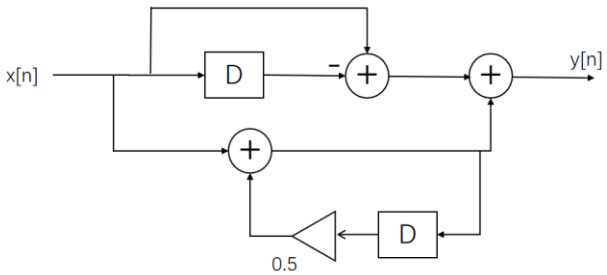

Fig. 12. S1+S2

III. Audio Filtering

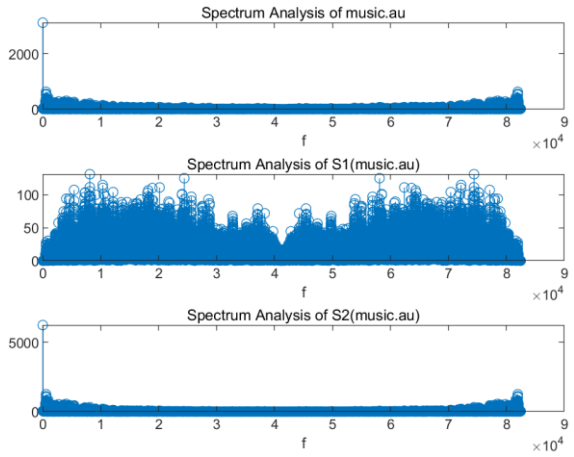

Fig. 13.

When the audio signal is processed by the S1 system, it boosts high-frequency sounds, as seen in the spectrogram, where the intensity tends to focus towards the center. On the other hand, the S2 system has the opposite effect, spreading the signal towards the extremes. Due to display issues, high-frequency parts are visible at both ends, with low-frequency components concentrated in the center. Using auditory perception, we found that S1 makes the sound sharper and higher in pitch, while S2 gives it a deeper tone. Furthermore, when the initial signal goes through S1 and S2 three times, these effects become even more noticeable in the spectrogram and the sound

IV. Inverse Systems

We need to find out the inverse system of S2, which is S3. $i.e.\ \delta[n] = S3(S2(x))$.

The system $y = S_3$can be described by the difference equation: $y[n] = ax[n] + bx[n-1]$. The following is the mathematical calculation:

For system $S2$,we have $y2[n] = \frac{y2[n-1]}{2} + x2[n]$

Then we need to find out the output of $y3[n] = S3(S2)$ is exactly the input $x2[n]$, which makes S3 an inverse system. So we have:

$$y3[n] = a\left(\frac{1}{2}y2[n-1] + x2[n]\right) + by2[n-1]$$
$$= \left(\frac{a}{2} + b\right)y2[n-1] + ax2[n]$$

Since $y3[n] = x2[n]$, then we have

$$\frac{a}{2} + b = 0$$
$$a = 1$$

Therefore, a=1 and b=-0.5.

The difference equation of S3 becomes

$$y[n] = x[n] - 0.5x[n-1]$$

Then we can draw the block diagram of S3 system.

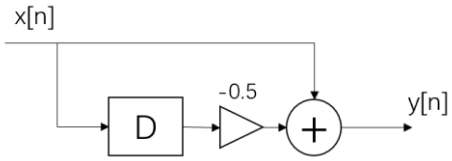

Fig. 14. block diagram of S3 system

We can also plot the impulse response of system $S3$ and $S3(S2)$.

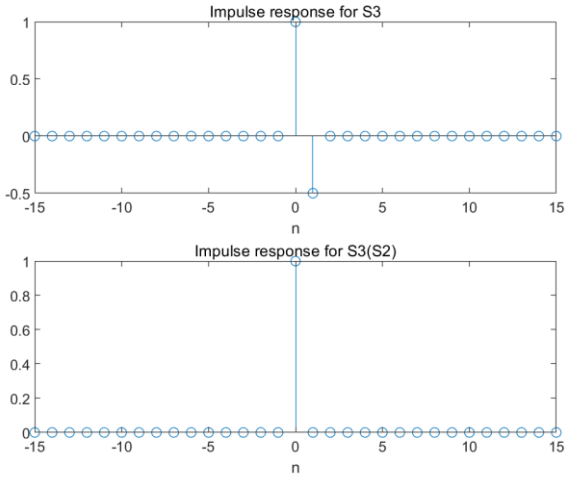

Fig. 15. impulse response of system S3 and S3(S2)

V. System Tests

Now we need to test which system is non-linear, and which system is time-varying.

A. *Linearity test*

In this section, we conduct a systematic examination of the linearity properties of three distinct discrete-time systems, namely, bbox4, bbox5, and bbox6. The primary objective is to evaluate whether these systems exhibit linear behavior, a crucial aspect in understanding their response to input signals.

The core of the linearity test involved investigating whether each system adhered to the principle of linearity, as expressed by the equation:

$$bbox(ax1[n] + bx2[n])$$
$$= a * bbox(x1[n]) + b * bbox(x2[n])$$

To validate the linearity, two random input signals, x1[n] and x2[n], were generated using the random function 'rand,' ensuring that different input signals were used in each test run.

In this context, two input signals, x1[n] and x2[n], were chosen randomly, ensuring variability with each test run. Subsequently, the systems' responses to the linear combination

$$4 * x1[n] + 5 * x2[n]$$

were compared with the linear combination of their responses to individual input signals.

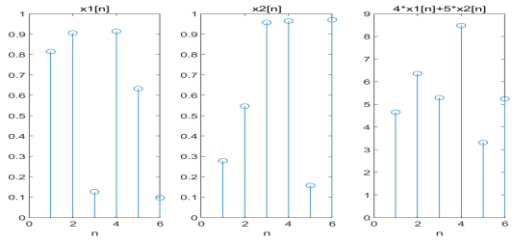

Fig.16. random input for linearity test

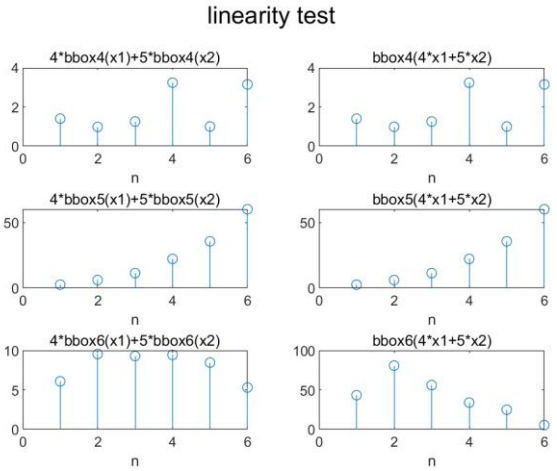

Fig. 17. Results of linearity test

The results obtained from the linearity test for bbox4 and bbox5 systems exhibited consistent behavior.

In contrast, the test results for bbox6 was found to be non-linear. Specifically, the response of bbox6 to the linear combination $4 * x1[n] + 5 * x2[n]$ did not match the linear combination of its responses to the individual input signals.

B. Time invariance test

In this section, we conducted a comprehensive examination to assess the time invariance properties of three discrete-time systems: bbox4, bbox5, and bbox6.

Testing Methodology: The time invariance test was executed in three distinct steps:

1) Step 1: Input signal shift and System Response Calculation

The input signal x[n] was initially constructed with specific characteristics, including a time window of length N for subsequent time-shifting.

A time-shifted version of the input signal, x[n + N], was generated by shifting x[n] by N samples to the left.

The shifted input signal, x[n + N], was passed through each of the three systems (bbox4, bbox5, and bbox6) to calculate the corresponding response, denoted as y1[n].

Corresponding code:

```
%The input signal x[n] is first time-shifted and
then passed through the system to obtain y1[n]
N=7;
x=[zeros(1,N),ones(1,N),zeros(1,N)];%input,right
shift or left shift<=N samples
x_shift=[ones(1,N),zeros(1,2*N)]%shift input
signal x by N samples to the left
y4_x_shift=bbox4(x_shift);%y1[n] for bbox4
y5_x_shift=bbox5(x_shift);%y1[n] for bbox5
y6_x_shift=bbox6(x_shift);%y1[n] for bbox6
```

2) Step2: Calculation of y[n + N]

The unshifted input signal x[n] was passed through the same systems to calculate the response, y[n].

The calculated responses y[n] were subsequently shifted by N samples to the left, yielding y[n + N].

3) Step3: Time invariance test

Both the time-shifted responses y[n + N] (from Step TWO) and the y1[n] responses (from Step ONE) were compared for each system.

The comparison involved visualizing and scrutinizing these responses over a range of discrete time indices.

4) Results and Analysis:

The results of the time invariance test were visually represented for each system, namely, bbox4, bbox5, and bbox6. The test outcomes were analyzed based on the comparison between the time-shifted responses, y[n + N], and the y1[n] responses.

5) Final Analysis:

Bbox4 and Bbox6: For bbox4 and bbox6, the time invariance test demonstrated that the systems satisfied the time invariance principle. The responses y[n + N] were observed to be consistent with the y1[n] responses, confirming that these systems exhibit time invariance, as expected.

Bbox5: In contrast, bbox5 yielded distinct results. The time invariance test revealed that bbox5 did not adhere to the principle of time invariance. The responses y[n + N] for bbox5 did not align with the y1[n] response.

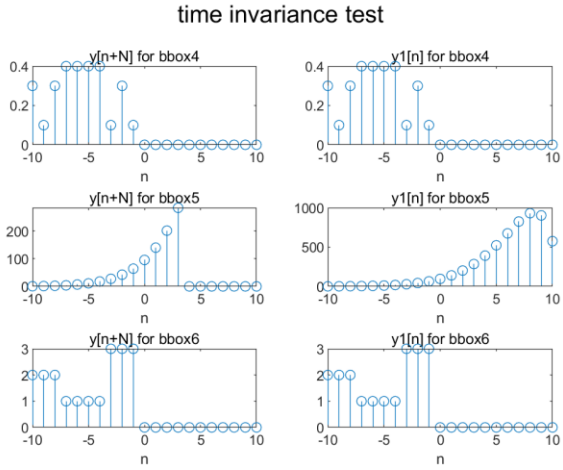

Fig. 18. Results of time-invariance test

VI. Stock Market Example

In this section, we examined the application of two distinct filters, specifically Filters (2.4) and (2.5), to smooth daily stock market exchange rates, leveraging the vector "rate" from the "stockrates.mat" file. The analysis aimed to understand the behavior and performance of these filters in terms of advantages, disadvantages, and the initialization of their outputs.

A. Filter Implementation:

Two custom filter functions were devised to implement Filters (2.4) and (2.5) as follows:

*1) Filter (2.4): y2[n] = 0.8 * y2[n - 1] + 0.2 * x[n]*

In this case, the output filter was initialized by setting the value of avgvalue(yesterday) to 0. The filter was implemented using a for loop to iterate through the input data.

```
function y=filter_2_4(x)
% y[n]=0.8y[n-1]+0.2x[n]
N=length(x);
y=zeros(1,N+1);
%initialize the value of avgvalue(yesterday)
for n=2:N+1
y(n)=0.8*y(n-1)+0.2*x(n-1);
end
y=y(2:N+1);
end
```

2) Filter (2.5): y3[n] = y3[n - 1] + (x[n] - x[n - 3]) / 3

For this filter, the initial values of the "value" vector were set to 0 for the days prior to data collection. The filter implementation involved a for loop and utilized a three-day window of historical data for calculation.

```
function y=filter_2_5(x)
N=length(x);
y=zeros(1,N+1);
x=[zeros(1,3) x];
%set the initial values of the "value" vector to 0
for n=2:N+1
    y(n)=y(n-1)+(1/3)*(x(n+2)-x(n-1));
end
y=y(2:N+1);
end
```

B. Advantages and Disadvantages:

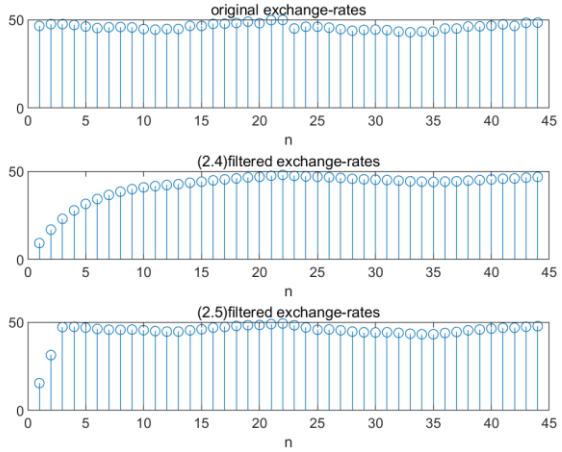

Fig. 19.

1) Filter (2.4):

Advantages: As time progresses, this filter exhibits gradual convergence toward a stable constant. Consequently, the average level line becomes notably smoother.

Disadvantages: A notable drawback of this filter is its extended convergence time, which typically spans 10 to 15 days, representing a lengthy waiting period before stable results are achieved.

2) Filter (2.5):

Advantages: Filter (2.5) maintains an almost consistently flat average value line, minimizing waiting times for stable results.

Disadvantages: In comparison to Filter (2.4), Filter (2.5) demonstrates a higher degree of fluctuations in the average value line, resulting in less stable filtered outcomes.

C. Better Methods for Initializing Filter Outputs

1) Exponential Moving Average (EMA) Initialization:

Initialize the filter outputs using an exponential moving average of the historical data. EMA provides more weight to recent data points and can offer a more responsive initialization compared to a simple mean or zero values.

2) Statistical Measures:

Utilize statistical measures such as the median or mode of the historical data as the initial filter output. These measures can be less sensitive to outliers and extreme values, providing a robust initialization.

3) Dynamic Initialization:

Implement a dynamic initialization method that adapts to the data's characteristics. For example, you could calculate the initial values based on the data's variance, skewness, or other statistical properties.

VII. Conclusion

In this practical exploration, various discrete systems, including differentiators and integrators, were meticulously constructed through mathematical formulations. By subjecting these systems to a range of input signals and exploring their combinations, critical insights into the input-output relationships of discrete systems were revealed. The experiment illuminated the fundamental characteristics of linear time-invariant systems, emphasizing their additivity and commutativity. Additionally, the creation of inverse systems, involving the interchange of x and y variables, unveiled the expression of the inverse system. Three black-box discrete systems were then subjected to comprehensive testing, leading to the identification of bbox4 as a time-variant system and bbox6 as a nonlinear system. Armed with a robust theoretical framework, practical applications emerged, showcasing the noise reduction capabilities of discrete systems in audio data processing. Moreover, the experiment underscored how differentiators amplify high frequencies while attenuating low frequencies, while integrators exhibited the opposite effect..

VIII. PARTIAL PYTHON CODES

2.3 Example Discrete-Time Systems

```
n = -10:20;
delta_signal=(n==0)-(n==5);%input signal of δ[n] – δ[n – 5]
```

```
N = 10;
u_signal = (n >= 0) - (n >= N + 1);%input signal of u[n] - u[n - (N + 1)] with N = 10
% Apply the differentiator and integrator to the input signals
y1_diff = differentiator(delta_signal);
y2_diff = differentiator(u_signal);
y1_int = integrator(delta_signal);
y2_int = integrator(u_signal);


% Create a single figure with subplots to visualize the results
figure;


% Subplot 1: Input delta[n] - delta[n-5]
subplot(4, 2, 1);
stem(n, delta_signal);
title('Input: δ[n] - δ[n - 5]');
xlabel('n');ylabel('Amplitude');grid on;


% Subplot 2: Differentiator Output for delta[n] - delta[n-5]
subplot(4, 2, 2);
stem(n, y1_diff);
title('Differentiator Output for δ[n] - δ[n - 5]');
xlabel('n');ylabel('Amplitude');grid on;


% Subplot 3: Input delta[n] - delta[n-5]
subplot(4, 2, 3);
stem(n, delta_signal);
title('Input: δ[n] - δ[n - 5]');
xlabel('n');ylabel('Amplitude');grid on;


% Subplot 4: integrator Output for delta[n] - delta[n-5]
subplot(4, 2, 4);
stem(n, y1_int);
title('integrator Output for δ[n] - δ[n - 5]');
xlabel('n');ylabel('Amplitude');grid on;


% Subplot 5: Input u[n] - u[n-(N+1)]
subplot(4, 2, 5);
stem(n, u_signal);
title('Input: u[n] - u[n-(N+1)]');
xlabel('n');ylabel('Amplitude');grid on;


% Subplot 6: Differentiator Output for u[n] - u[n-(N+1)]
subplot(4, 2, 6);
stem(n, y2_diff);
title('Differentiator Output for u[n] - u[n-(N+1)]');
xlabel('n');ylabel('Amplitude');grid on;
% Subplot 7: Input u[n] - u[n-(N+1)]
subplot(4, 2, 7);
stem(n, u_signal);
title('Input: u[n] - u[n-(N+1)]');
xlabel('n');ylabel('Amplitude');grid on;


% Subplot 8: Integrator Output for u[n] - u[n-(N+1)]
subplot(4, 2, 8);
stem(n, y2_int);
title('Integrator Output for u[n] - u[n-(N+1)]');
xlabel('n');ylabel('Amplitude');grid on;
```

2.4 Difference Equation

```
n=-15:15;
impulse = (n==0); % Impulse signal δ[n]


% Calculate impulse responses for each system
impulse_response_S1 = S1(impulse);
impulse_response_S2 = S2(impulse);
impulse_response_S1_S2 = S1(impulse_response_S2);
impulse_response_S2_S1 = S2(impulse_response_S1);
impulse_response_S1_plus_S2 = S1(impulse) + S2(impulse);


% Plot the impulse responses
figure
subplot(3, 2, 1);
stem(impulse_response_S1);
title('Impulse Response of S1');


subplot(3, 2, 2);
stem(impulse_response_S2);
title('Impulse Response of S2');


subplot(3, 2, 3);
stem(impulse_response_S1_S2);
title('Impulse Response of S1 (S2)');


subplot(3, 2, 4);
stem(impulse_response_S2_S1);
title('Impulse Response of S2 (S1)');
```

```
subplot(3, 2, 5);
stem(impulse_response_S1_plus_S2);
title('Impulse Response of S1 + S2');
```

2.5 Audio Filtering

```
audio=audioread('music.au');
audio_S1=S1(audio);
audio_S2=S2(audio);
%sound(audio)
%sound(audio_S1)
sound(audio_S2);
figure
subplot(3,1,1),stem(audio)
subplot(3,1,2),stem(audio_S1)
subplot(3,1,3),stem(audio_S2)
```

```
figure
subplot(3,1,1),stem(abs(fft(audio))),title('Spectrum
Analysis of music.au');xlabel('f')
subplot(3,1,2),stem(abs(fft(audio_S1))),title('Spectrum
Analysis of S1(music.au)');xlabel('f')
subplot(3,1,3),stem(abs(fft(audio_S2))),title('Spectrum
Analysis of S2(music.au)');xlabel('f')
```

2.6 Inverse Systems

we can use MATLAB to solve for the values of a and b using symbolic algebra.

S2: y2(n) = y2(n - 1) / 2 + x2(n);

S3: $y3[n] = ax3[n] + bx3[n-1]$

when input signal is an impulss signal ,impulse input signal==output of S3(S2(impulse))

```
% Create an input impulse signal
n=-15:15;
impulse = (n==0)
syms a b


% Solve the equations
eq1 = impulse(16)
==a*impulse_response_S2(16)+b*impulse_response_S2(15)
eq2 = impulse(17)
==a*impulse_response_S2(17)+b*impulse_response_S2(16)


% Solve for a and b
solution = solve([eq1, eq2], [a, b]);
```

```
% Extract the values of a and b
a_value = solution.a
b_value = solution.b
% Compute the impulse response of S3
impulse_response_S3 = S3(impulse);


% Compute the impulse response of S3(S2(x))
impulse_response_S3_S2 = S3(impulse_response_S2);


figure
subplot(2,1,1),stem(n,impulse_response_S3),xlabel('n'),ti
tle('Impulse response for S3');
subplot(2,1,2),stem(n,impulse_response_S3_S2),xlabel('n')
,title('Impulse response for S3(S2)');
```

References