72/80. Very good!

Digital Signal Processing Lab Report 2

1st Qiu Kunyuan
Department of EEE.
Southern University of Science and Technology
Shenzhen, PRC
11913019@mail.sustech.edu.cn

***Abstract*—This lab aims to the demonstration of DT(Discrete-Time) systems under the MATLAB® programming environment and exploration of essential characteristics of DT systems using MATLAB. In this lab session, the DT system modeling and analysis method is applied to discretizing CT(Continuous Time) functions, solving basic difference equations, filter building and modeling the stock market. Additionally, a black-box system goes under test and being analyzed in this lab with DT system testing and analysis methods.**

***Index Terms*—discrete-time system, MATLAB, system modeling, difference equation, black-box system**

I. Introduction

A. Discrete-Time System

index n

A DT system is any mapping $S : x \mapsto y$that maps a discrete-time input $x[t]$ to a discrete-time output $y[t]$. Digital computers can and can only store or process DT systems, due to all the quantities and signals expressed in these computers must be digitized and discretized.

Analog systems may process *raw*, referring continuous-time, CT signals directly with infinite sampling frequency, but analog systems are unable to be reconfigured quickly comparing to digital systems, while the parameters of analog systems drifts along with the time and results in the degenerate of accuracy.

Mathematically, the DT systems are mostly expressed in difference equations

$$y[t] = \mathcal{S}(x[t], y[t]) \tag{1}$$

and the notation $\mathcal{S}(x)$ is often used to denote DT systems. Block diagrams are also used to express DT systems, making it possible to obtain an intuitive view of system structure while reserving the mathematical characteristics of the system.

B. Analyzing DT System

A DT system might be linear or nonlinear, time variant or time invariant, causal or noncausal, memoryless or with-memory, active or passive, and stable or unstable. These properties can be written in formulas strictly and verified mathematically by deduction.

Theorem I-C (Property of Linear System)

> *A system is linear, iff the system satisfies both additivity and homogeneity:*
>
> $$\mathcal{S}\left(\sum_i \alpha_i x_i\right) = \alpha_i \sum_i S(x_i) \tag{2}$$

Theorem I-D (Property of Time-Invariant System)

> *A system is time-invariant, if the output of the system only delays for the same time to the delay of the input:*
>
> $$y(t) = \mathcal{S}(x(t)) \Rightarrow y(t - t_0) = \mathcal{S}(x(t - t_0)) \tag{3}$$

A system is called as Linear Time-Invariant(LTI) if the system both satisfies linearity and time-invariance. For LTI system, the output of the DT system is uniquely determined by the impulse response of the system, and can be evaluated closely in forms of *convolution*:

$$y[t] = x[t] * \mathcal{S}[\delta[t]] \tag{4}$$

However, numerical solution of DT system with computer does not require LTI property since solving difference equations in computer are simple cumulative summation, that can be easily realized by loop. In MATLAB realizations of difference equation solver, the built-in cumsum function are frequently used since it adopted BLAS libraries to accelerate the vector operations dramatically.

Although that is true, if the system is not LTI, you couldn't use summation.

II. Background Exercises

A. Example DT Systems

Use DT systems to approximate the CT differentiator and integrator.

a) Problem (i)(ii): For CT differentiator, this system can be approximated by a 1st-order FIR. In difference equation form, the FIR is

$$y[t] = x[t] - x[t - 1] \tag{5}$$

Use MATLAB to demonstrate the approximation by plotting the result of the differentiator:

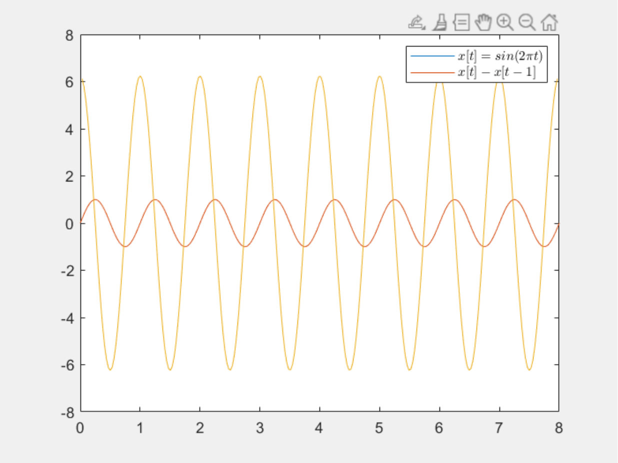


Fig. 1: MATLAB plot of a DT differentiator

For CT integrator, this system can be approximated by a 1st-order IIR. To write the integrator(or accumulator) in closed form, the filter is expressed in IIR form

$$y[t] = y[t-1] \cancel{=}+ x[t] \tag{6}$$

Use MATLAB to demonstrate the approximation by plotting the result of the integrator:

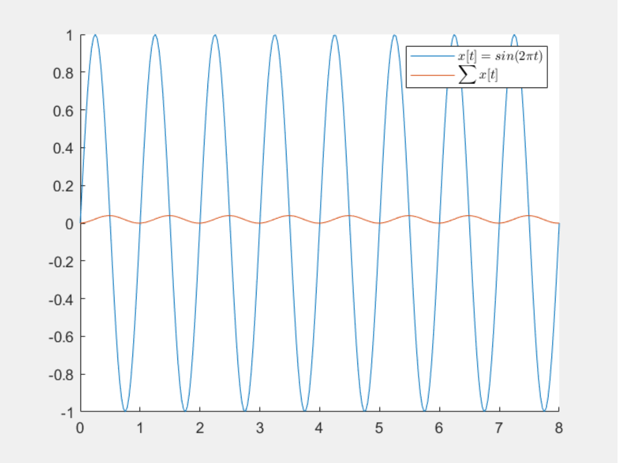


Fig. 2: MATLAB plot of a DT integrator

b) Problem (iii): Diagram of differentiator:

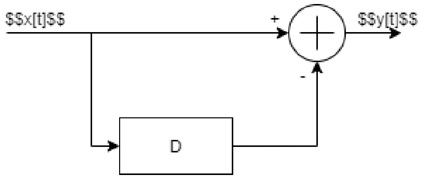


Fig. 3: Block diagram of differentiator

Diagram of integrator:

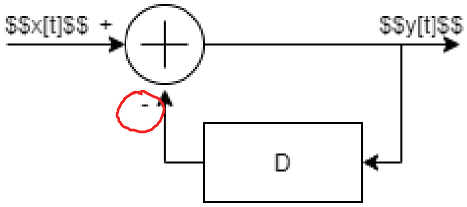


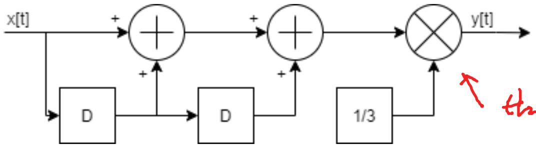
Fig. 4: Block diagram of integrator

B. Stock Market Example

a) Method 2.3: Difference equation:

$$y[t] = \frac{1}{3}(x[t] + x[t-1] + x[t-2]) \tag{7}$$

The diagram of this system is Scaling uses



this for modulation

Fig. 5: Block diagram of system (2.3)

Use MATLAB to evaluate the impulse response:

```
b1 = [1, 1, 1] * 1/3;
a1 = 1;
t = 0:10;
d1 = impseq(t(1), 0, t(end));
y1 = filter(b1, a1, d1);
fig = figure(1);
stem(t, y1);
legend('Method 2.3');
```

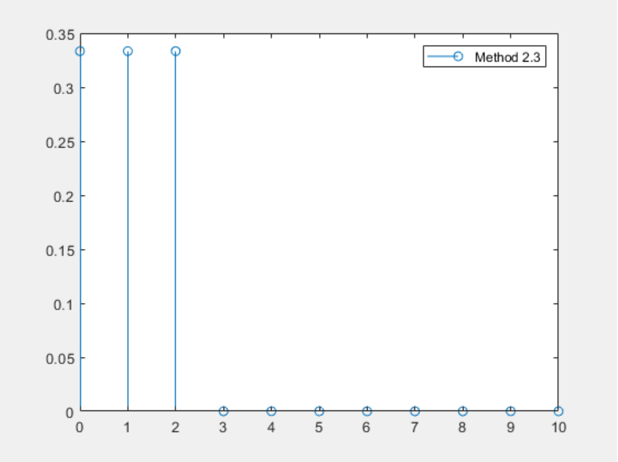


Fig. 6: Impulse response of method 2.3

b) Method 2.4: Difference equation of the system:

$$y[t] - 0.8y[t-1] = 0.2x[t] \tag{8}$$

The diagram of the system is

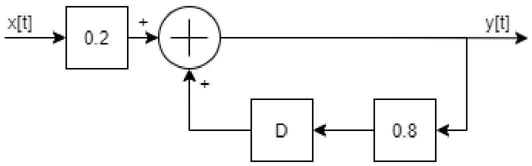


Fig. 7: Block diagram of system (2.4)

Use MATLAB to evaluate the impulse response. The evaluating code is identical to the code used in Problem 2.3, while the only difference is the different coefficients:

```
    b2 = 1 * 0.2;
    a2 = [1, -0.8];
    t = 0:15;
```

Plotting result:

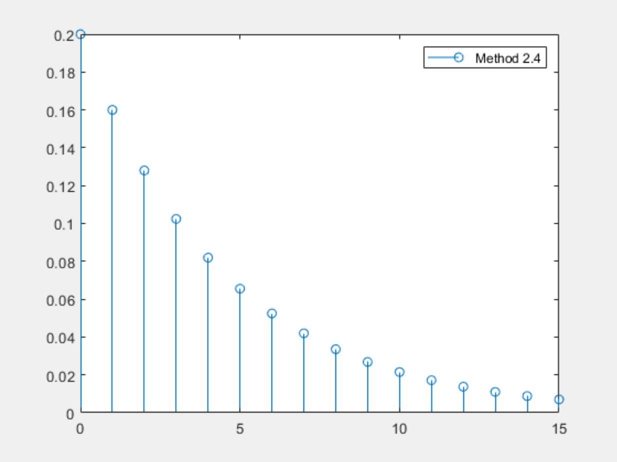


Fig. 8: Impulse response of method 2.4

c) Method 2.5: Difference equation of the system:

$$y[t] - y[t-1] = \frac{1}{3}x[t] - \frac{1}{3}x[t-3] \tag{9}$$

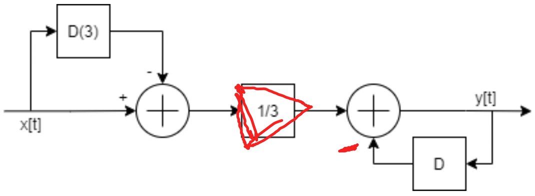


Fig. 9: Block diagram of system (2.5)

Use MATLAB to evaluate the impulse response.

```
    b3 = [1, 0, 0, -1] * 1/3;
    a3 = [1, -1];
    t = 0:10;
```

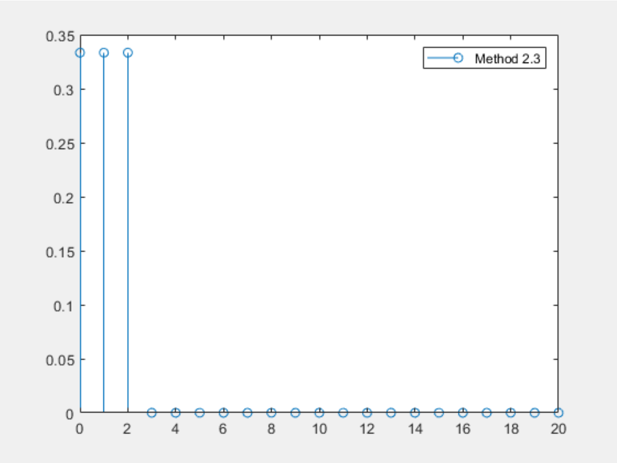


Fig. 10: Impulse response of method 2.4

d) Moving Averages: For the methods (2.3), the output $y[t]$ is the mean value of the nearest inputs, and the window length of the input is 3. Therefore, this method is *moving average*.

For the method (2.5), use Z-transform to simplify the IIR filter:

$$\begin{aligned} Y[z] &= z^{-1}Y[z] + \frac{x[z]}{3} - z^{-3}\frac{x[z]}{3} \\ &= \frac{1-z^{-3}}{3(1-z^{-1})}X[z] \\ &= \frac{1}{3}(1+z^{-1}+z^{-2})X[z] \end{aligned} \tag{10}$$

Therefore in time domain, the system is

$$y[t] = \frac{1}{3}(x[t] + x[t-1] + x[t-2]), \tag{11}$$

, which is a moving average.

III. Example DT systems

MATLAB code for signal processing:

```
    x1=signal;
    dx1 = [x1(1), diff(x1)];
    %differentiator
    ix1 = cumsum(x1);
    %integrator
```

(a) For system (1)

$$x_1[t] = \delta[t] - \delta[t-5] \tag{12}$$

The differentiation result are two impulse pairs and the integration result is a box function.

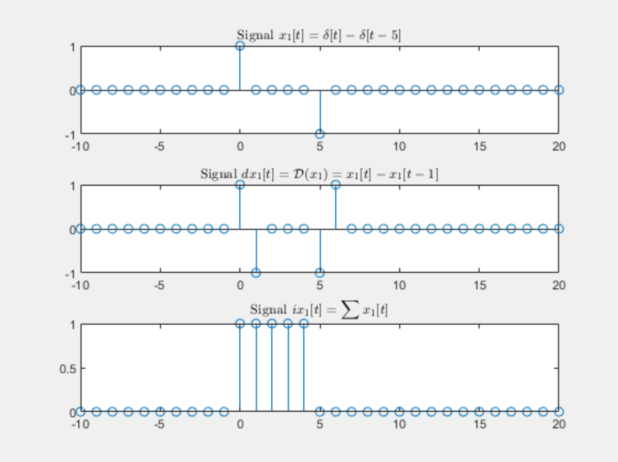


Fig. 11: Difference and integration result of signal $x_1[t]$

(b) For system (2)

$$x_2[t] = u[t] - u[t-(N+1)],\ N = 10 \tag{13}$$

the differentiation result are 2 impulses and the integration result is a ramp function connected by a Heaviside step function.

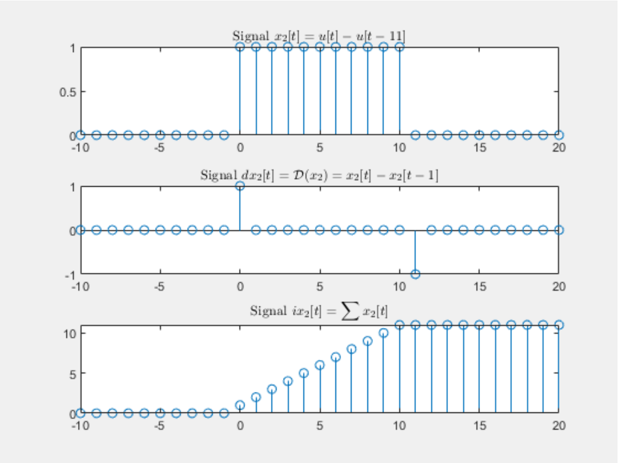


Fig. 12: Difference and integration result of signal $x_2[t]$

-1

Discussion of the stability.

IV. Difference Equations

a) Case 1: The system $\mathcal{S}_1$ is a differentiator. Therefore, its system diagram is

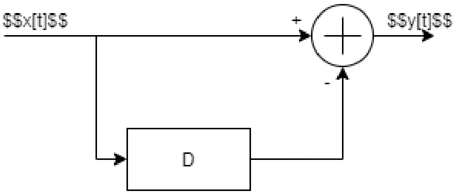


Fig. 13: System diagram of $\mathcal{S}_1$

The impulse response of the system is an impulse pair, plotted by MATLAB and shown below:

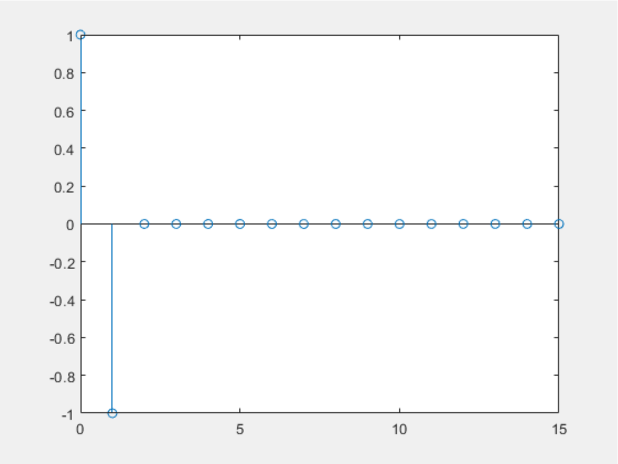


Fig. 14: Impulse response of system $\mathcal{S}_1$

b) Case 2: The system $\mathcal{S}_2$ is a 1st-order IIR filter. Its system diagram is

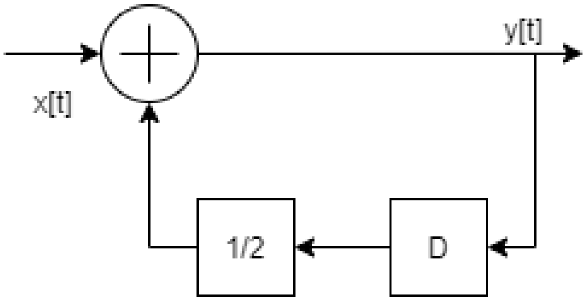


Fig. 15: System diagram of $\mathcal{S}_2$

The impulse response of this filter is an attenuating power function. Use MATLAB to plot the response:

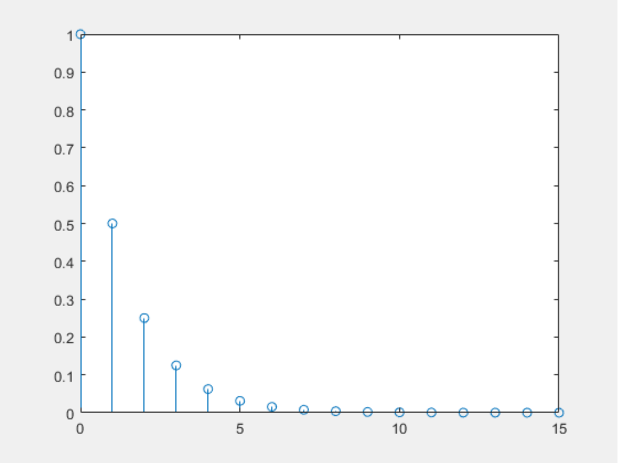


Fig. 16: Impulse response of system $\mathcal{S}_2$

mathematic derivation -2

c) Case 3: The system $\mathcal{S}_3$ is the serial connection of system $\mathcal{S}_1$ and $\mathcal{S}_2$. Therefore, this system is a general digital filter containing one FIR loop and one IIR loop.

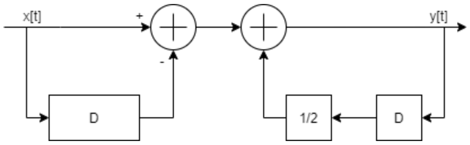


Fig. 17: System diagram of system $\mathcal{S}_3$

Use MATLAB to evaluate and plot the impulse of this system:

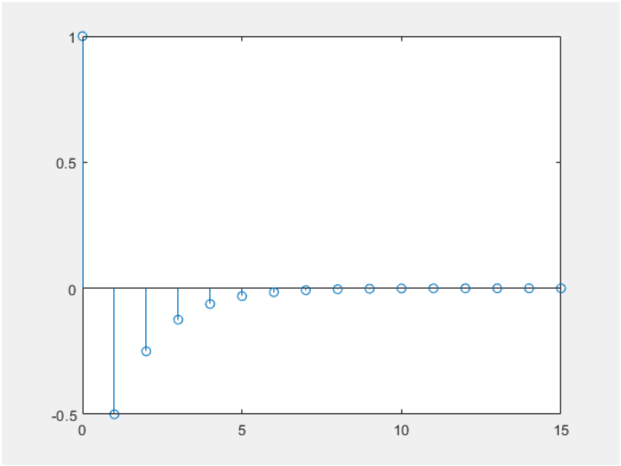


Fig. 18: Impulse response of system $\mathcal{S}_3$

d) Case 4: The system $\mathcal{S}_4$ is the serial connection of system $\mathcal{S}_2$ and $\mathcal{S}_1$. Its structure is nearly identical to system $\mathcal{S}_3$:

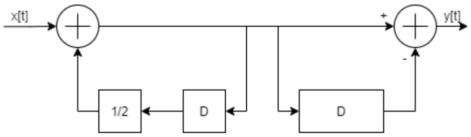


Fig. 19: System diagram of system $\mathcal{S}_4$

Use MATLAB to plot the impulse of this system:

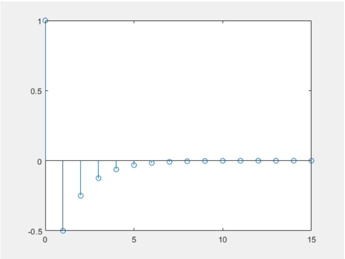


Fig. 20: Impulse Response of system $\mathcal{S}_4$

Compare this response with the response of the previous system, it's easy to find that the response of system $\mathcal{S}_2(\mathcal{S}_1)$ is identical as $\mathcal{S}_1(\mathcal{S}_2)$. This is evident to the commutativity of two serial LTI systems.

e) Case 5: The system $\mathcal{S}_5$ is the addition of the filter coefficients of system $\mathcal{S}_2$ and $\mathcal{S}_1$. The structure of this new system is completely different to system $\mathcal{S}_1$ or $mathcalS_2$, therefore its response is consequently different.

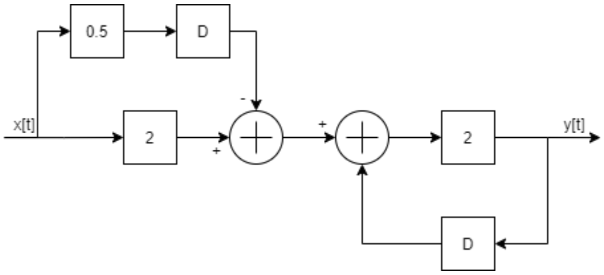
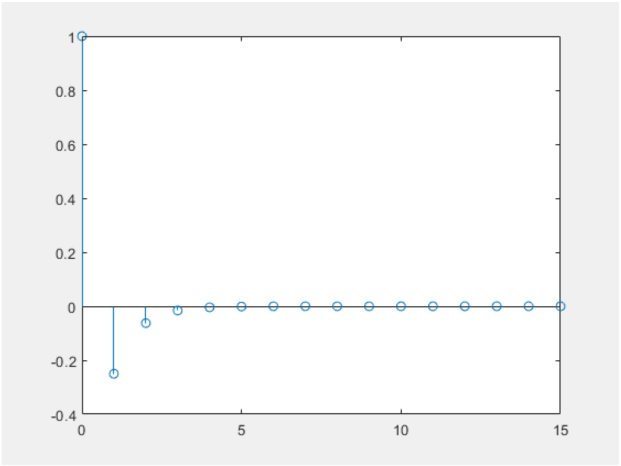


Fig. 21: System diagram of system $\mathcal{S}_5$

Use MATLAB to plot the impulse of this system:



—1

Fig. 22: Impulse response of system $\mathcal{S}_5$ They actually are equivalent.

This response is different to the response of paralleled system composed by $\mathcal{S}_1$ and $\mathcal{S}_2$, for the system analyzed above is addition of the filter coefficients of the two components.

Your result in Fig. 22 is not correct, even following coe. addition

V. Audio Filtering

Use the audioread()(which replaces the auread() in newer versions of MATLAB) command to load the audio file into MATLAB, and use sound() command to play this audio.

From the coefficient of the filter, it's easy to infer that the system $\mathcal{S}_1$ is a differentiator that blocks the low-frequency components of the input signal. The frequency response of $\mathcal{S}_1$, as Fig.23 shows, is a 1st-order low-pass filter. And the system $\mathcal{S}_2$, with the frequency response shown in Fig.24, is a low-pass filter that blocks the high-frequency components of the input signal.

Therefore, we can anticipate the effect of the two filters. The output of the filter $\mathcal{S}_1$ hears should be sharper than input due to the decreased low frequency components, while the output of filter $\mathcal{S}_2$ should be more mellow than input.

The frequency response of the two filters are evident to the anticipations above:

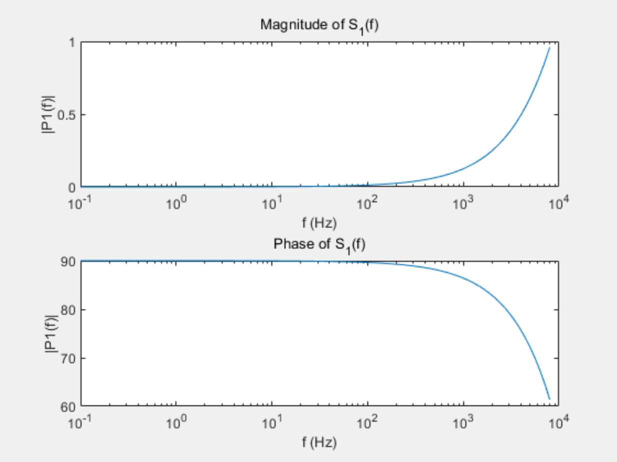


Fig. 23: Frequency response of $\mathcal{S}_1$

Digital filter, the response should normalize the sampling frequency to 2π. Not covered by lecture yet.

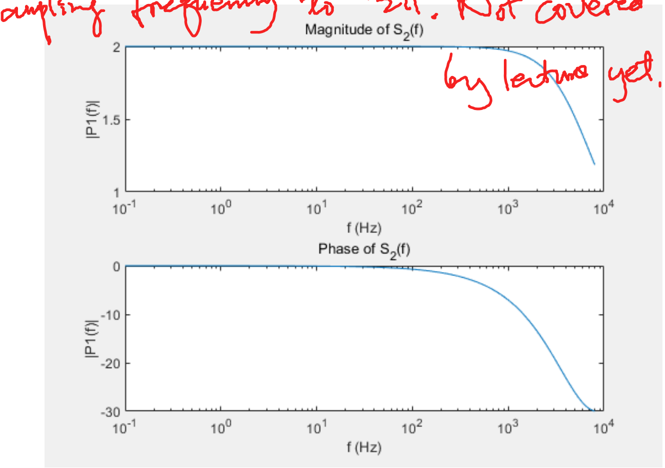


Fig. 24: Frequency response of $\mathcal{S}_2$

Use the MATLAB code shown below to evaluate the output of the two filters and plot the frequency spectrum of the output.

```
clear;close;clc;

[y, Fs] = audioread("music.au");
el = floor(length(y) / 2) * 2;
f0 = Fs * (0:(el / 2)) / el;

fy = fftSingle(y);
fy(1) = 0;

s1 = [[1, -1]; [1, 0]];
s2 = [[1, 0]; [1, -0.5]];
y1 = filter(s1(1, :), s1(2, :), y);
y2 = filter(s2(1, :), s2(2, :), y);
fy1 = fftSingle(y1);
fy1(1) = 0;
fy2 = fftSingle(y2);
fy2(1) = 0;

%Plotting the fft output;

%Bode Plot
w = logspace(-1, log10(Fs), 200);
sys1 = tf(s1(1, :), s1(2, :), 1 / Fs);
[mag, phase, wout] = bode(sys1, w);
mag1 = mag(1, :);
phase1 = phase(1, :);
%Plot the Bode plot of System S1;

sys2 = tf(s2(1, :), s2(2, :), 1 / Fs);
[mag, phase, wout] = bode(sys2, w);
mag2 = mag(1, :);
phase2 = phase(1, :);
%Plot the Bode plot of System S2;

function fout = fftSingle(y)
    el = floor(length(y) / 2) * 2;
    fy = abs(fft(y) / length(y));
    fy = fy(1:el);
    fy = fy(1:(el / 2 + 1));
    fy(2:(end - 1)) = 2 * fy(2:(end - 1));
    fout = fy;
end
```

To compare the output and input of each filter, the output and input are stacked in the same plot, and there are difference plots showing the subtraction of output and input.

The output of filter $\mathcal{S}_1$ shown in Fig.25 meets the anticipation from the coefficients of $\mathcal{S}_1$:

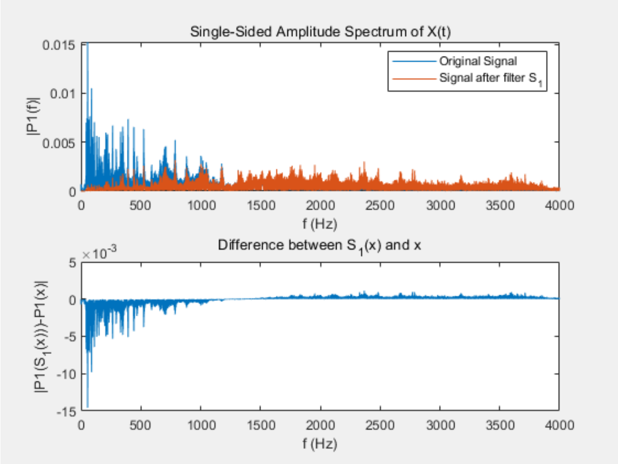
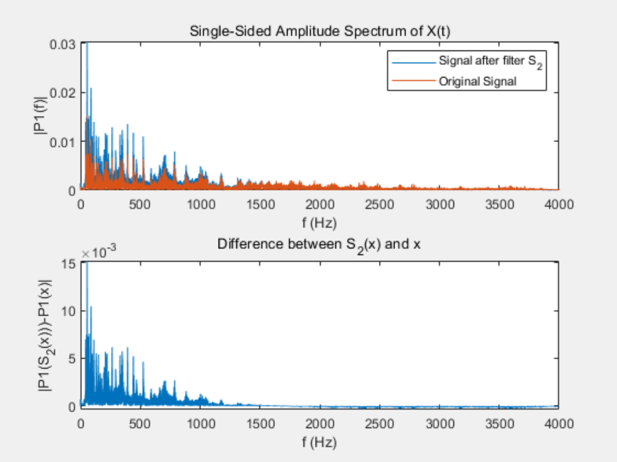


Fig. 25: Output spectrum of $\mathcal{S}_1(x(t))$

And the output of filter $\mathcal{S}_2$ shown in Fig.26 confirms the anticipation using the coefficients of $\mathcal{S}_2$:



+1

Fig. 26: Output spectrum of $\mathcal{S}_2(x(t))$

VI. Inverse System

The definition of *inverse system* is

Theorem VI-A (Inverse System)

A system $\mathcal{A}$ is called the inverse system of the given system $\mathcal{S}$, if and only if

$$\mathcal{A}(\mathcal{S}(x(t))) = x(t) \tag{14}$$

Use the z-Transformation of the impulse signal to acquire the inverse system $\mathcal{S}^{-1}(t)$ of the given system $\mathcal{S}(t)$:

$$\begin{aligned} \mathcal{A}(\mathcal{S}(x(t))) &= x(t) \\ \Leftrightarrow x(t) * \mathcal{A}(t) * \mathcal{S}(t) &= x(t) \\ \Leftrightarrow \mathcal{A}(z)\mathcal{S}(z) &= 1 \end{aligned} \tag{15}$$

Therefore, the inverse system in Z-domain can be easily obtained:

Theorem VI-B (Obtain the Inverse System)

The inverse system $\mathcal{S}^{-1}(z)$ of a given transfer function $\mathcal{S}(z)$ in Z-domain is:

$$\mathcal{S}^{-1}(z) = \frac{1}{\mathcal{S}(z)} \tag{16}$$

For the system

$$\mathcal{S}_2(t): \;\; y[t] = \frac{y[t-1]}{2} + x[t] \tag{17}$$

, its inverse system in Z-domain is

$$\begin{aligned} \mathcal{S}_2^{-1}(z) &= \frac{1}{\mathcal{S}_2(z)} \\ &= \left(\frac{z}{z-0.5}\right)^{-1} \\ &= 1 - \frac{1}{2z} \end{aligned} \tag{18}$$

, and thus the inverse system in time domain is

$$\mathcal{S}_2^{-1}(t): \;\; y[t] = x[t] - \frac{x[t-1]}{2} \tag{19}$$

Use MATLAB to verify the correctness of this inversion:

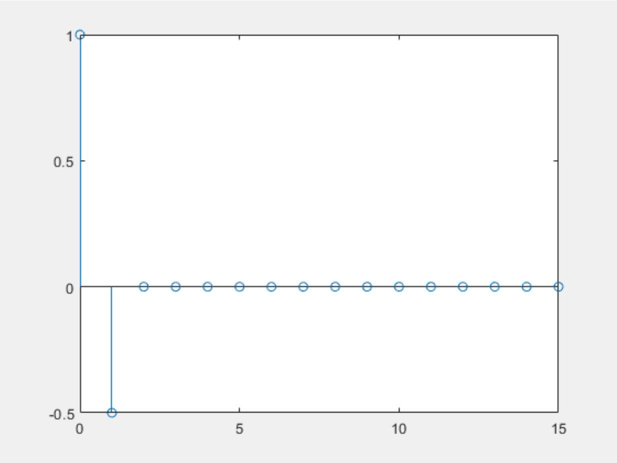


Fig. 27: Response of inverse system $\mathcal{S}_3$

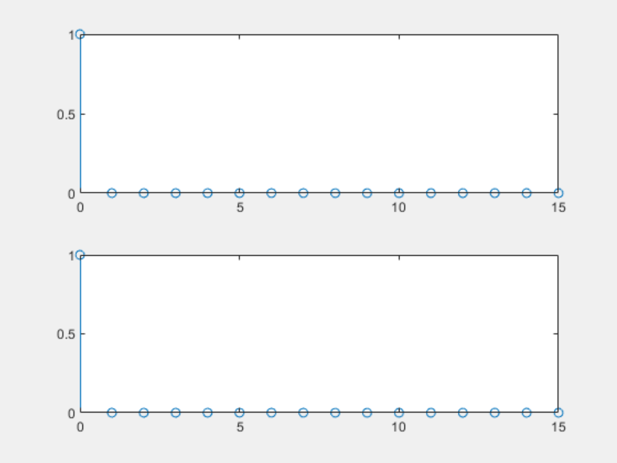


Fig. 28: Impulse signal after $\mathcal{S}_2$ and its inverse system

As Fig.28 shows, the impulse signal is recovered after passing the inverse system $\mathcal{S}_2^{-1}$.

VII. Black Box Model Identification

The justification of linearity and time-invariance of any given system can be easily utilized under MATLAB environment with its native support to function handles and vectorized signal operations. In another word, the theorems of linearity(I-C) and time-invariance(I-D) can be translated into MATLAB code with intuition.

```
function ret=detNL(ufunc,tl1,tl2,tsL)
    %Boolean function justifies linearity of
    arbitary function @ufunc(x)
    ts1=rand([1,tsL]);
    ts2=rand([1,tsL]);
    test=ufunc(ts1*tl1+ts2*tl2)-ufunc(ts1)*tl1-
    ufunc(ts2)*tl2;
    ret=(max(abs(test))<=1e-5)*1;
end
```

```
function ret=detTV(ufunc,dt,paddle,tsL)
    %Boolean function justifies time-invariance of
     arbitary function @ufunc(x)
    ts1=[zeros(1,paddle),rand([1,tsL]),zeros(1,
    paddle)];
    ts2=circshift(ts1,dt);
    out1=ufunc(ts1);
    out2=ufunc(ts2);
    test=out2-circshift(out1,dt);
    ret=(max(abs(test))<=1e-5)*1;
end
```

circular shift ?
should use
linear shift.

-1

The output of the two determining functions above indicates whether the unknown function @ufunc(x), where means the black-box function bboxN(x), satisfies linearity and time-invariance or not. A for-loop is adopted to apply the two determining functions to a list containing the handle of unknown functions that are demanded for determining:

```
    func_unkn={@bbox1,@bbox2,@bbox3};
    isnl=[];
    istv=[];
    %Global Variables
    for f=func_unkn
        a=detNL(f{1},randi(12),randi(12),120);
        b=detTV(f{1},randi(20),40,120);
        isnl=[isnl,a];
        istv=[istv,b];
    end
```

For the linearity, the justification output is

```
isnl  =
         1       1       0
```

indicating the non-linear function is bbox3(x). For the time-invariance, the output is

```
istv  =
         0       1       1
```

indicating the time-variant function is bbox1(x).

VIII. Stock Market Example

a) Filter 2.4: The filter () function in MATLAB automatically sets the initial conditions to zero if not providing further information of initial conditions by passing argument to this function. Therefore, applying the filter(2.4), which is illustrated in Fig.7, to the stock market data.

```
    f24 = [[0.2, 0]; [1, -0.8]];
    y1 = filter(f24(1, :), f24(2, :), rate);
```

Plot the output of the filter below:

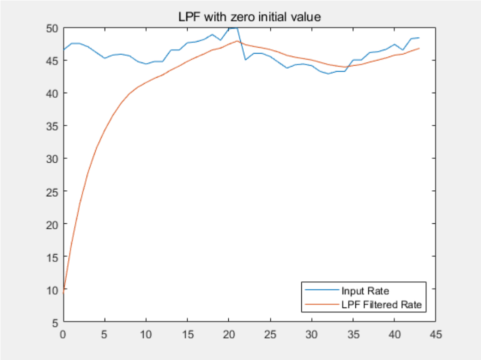


Fig. 29: 1st-order LPF filtering result with $y[-1] = 0$

It's quite clear in the figure that the zero-initial output of the filter deviates dramatically at the initial, even though the output tracks up to the input smoothly along with the growth of time vector. This deviance shows more clear on the residue/smoothness plot Fig.31.

b) Filter 2.5: With the moving-average(MA) filtering there is no requirement of the initial conditions given. The filter output tracks up to the input quickly from zero and keeps a accurate tracking, for the MA filter is a FIR and it produces sharper response to its input. The MATLAB code is identical to the 1st-order IIR LPF above but the filter coefficients:

```
    f25 = [[1, 0, 0, -1] * 1/3; [1, -1, 0, 0]];
    y2 = filter(f25(1, :), f25(2, :), rate);
```

Plot the output of the filter below:

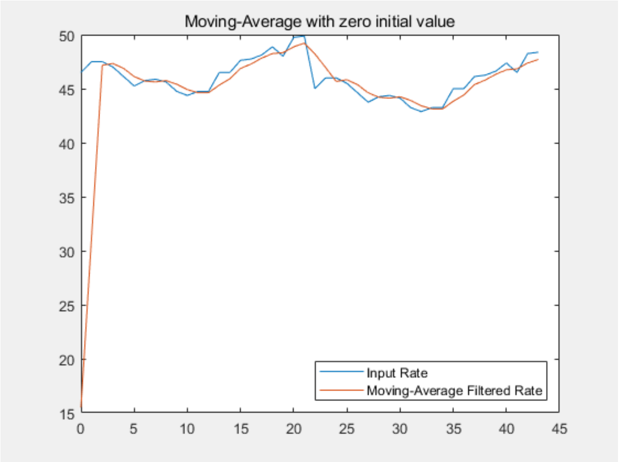


Fig. 30: 3 Point Moving-Average filtering result

The MA filter is sharper in tracking, but the smoothness of its output is worse than the result of 1st-order IIR filtering. This defect exposes more on the residue/smoothness plot Fig.31.

c) Performance Comparation: Use two criteria, the residue and 1st-order difference, to evaluate the accuracy and output smoothness of the two filters.

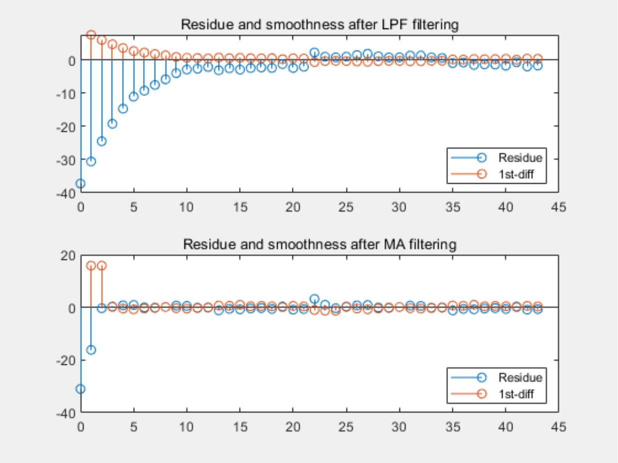


Fig. 31: Residue/Smoothness Plot

The residue is difference between filter output and filter input, smaller residue indicates faster tracking and better performance, while smaller 1st-order difference means smoother output. It's obvious on the plot that the filter (2.4) boasts its better smoothness than filter (2.5), however the bad tracking speed of filter (2.4) make this filter much less applicant.

d) Initialing Filter (2.4): Initialize the filter (2.4) is equivalent to obtaining the zero-input response of the filter at 0^+. This is a essential problem in signal processing and system analyzing and can be solved by integration crossing $t = 0$:

$$y(0^+) = y(0^-) + \mathcal{S}(x_{t<0}) \qquad (20)$$

In discrete systems, this response can be simply calculated by solving the difference of the system with provided inputs and outputs before $t = 0$:

+1

$$\sum_{-\infty}^{-1}\alpha_k y[k]+\alpha_0 y[0^+] = \sum_{-\infty}^{0}\beta_k x[k] \Rightarrow y[0_+] = \mathcal{S}(x_{t<0})+\sum_{-\infty}^{-1}\alpha_k y[k] \qquad (21)$$

There is an existing function filtic () in MATLAB to perform this calculation. By intuition, the initial value at $t = 0$ can be selected as the initial condition at $t = -1$ in best preservation of continuity. Substitute the evaluated IC(Initial Condition) into the filter and then obtain the output with better IC:

```
    f24 = [[0.2, 0]; [1, -0.8]];
    zi1 = filtic(f24(1, :), f24(2, :), rate(1));
    y1zi = filter(f24(1, :), f24(2, :), rate, zi1)
    ;
```

The poor tracking at the beginning of output is greatly improved, which is clearly shown on the input-output plot and the residue/difference plot Fig.32: This filter enjoys good tracking accuracy similar to filter (2.5), while preserving its better smoothness.

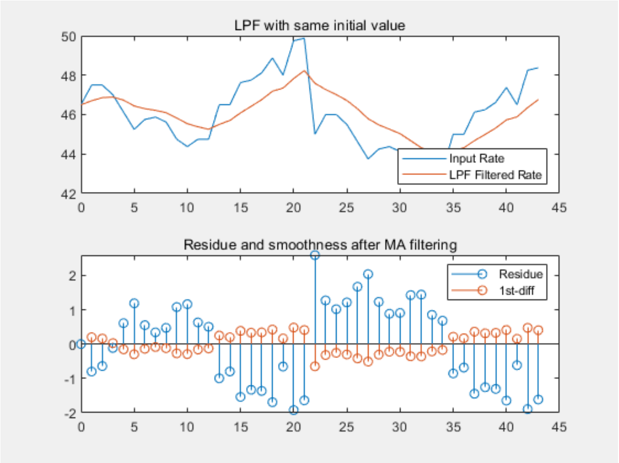


Fig. 32: MA filtering result with better IC

No summary. It supposes to describe one of the most significant outcomes you achieved in this lab
-5