De La Salle University - Manila

Department of Computer Technology

In Partial Fulfillment of the Course

Digital Signal Processing Laboratory

Term 2, A.Y. 2023 - 2024

Lab 08 - Sampling, Convolution, and FIR Filtering

Submitted by: Group 2

Alamay, Carl Justine S.

Arca, Althea Denisse G.

Submitted to:

Dr. Joel P. Ilao

1. **Contributions**
   1.1. Alamay - Code and Report for sections 3.1 to 3.2.1
   1.2. Arca - Code and Report for sections 3.2.2 to 3.3

2. **Link to Files Used:**
   https://drive.matlab.com/sharing/08cb603e-741c-49e5-84d0-8e6a7ea14c54

3. **Lab Exerciser: FIR filters**
   3.1. Deconvolution Experiment for 1-D Filters
      a. Plot both the input and output waveforms x[n]and w[n] on the same figure
      b. Determine the length of the filtered signal w[n], and explain how its length is related to the length of x[n] and the length of the FIR filter

      3.1.1. Restoration Filter
         a. Obtain the output signal y[n]
         b. Make stem plots of w[n] and y[n]
         c. plot of the error (difference) between x[n] and y[n] over the range 0 n < 50.

      3.1.2. Worst-Case Scenario
         a. Evaluate the worst-case error
         b. Error Plot

      3.1.3. An Echo Filter
         a. Determine the values of r and P
         b. Describe the filter coefficients of this FIR filter, and determine its length.
         c. Implement the echo filter
         d. Implement an echo filter and apply it to your synthesized music (Optional)

   3.2. Cascading Two Systems

      3.2.1. Overall Impulse Response
         a. Plot the impulse response of the overall cascaded system.
         b. Work out the impulse response h[n] of the cascaded system by hand
         c. state the condition on h_1[n] * h_2[n] to achieve perfect deconvolution.

      3.2.2. Distorting and Restoring Images
         a. Load in the image echart.mat
         b. Pick q = 0:9 in FILTER-1 and filter the image echart in both directions
         c. Deconvolve ech90 with FIR FILTER-2,

      3.2.3. A Second Restoration Experiment
         a. deconvolve ech90 with several different FIR filters for FILTER-2.
         b. how large is the worst-case error

   3.3. Filtering a Music Waveform (Extra Credit: 15 points)

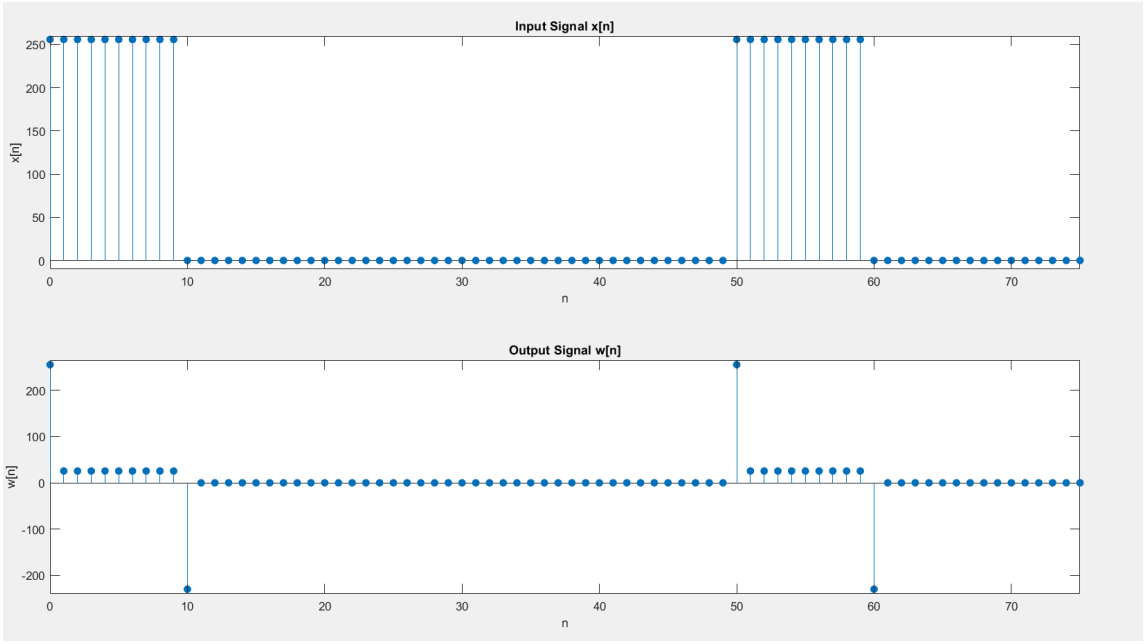
3.1 Deconvolution Experiment for 1-D Filters

```
xx = 256 * (rem(0:100, 50) < 10);

bb = [1, -0.9];
ww = firfilt(bb, xx);

n = 0:100;
n1 = 0:length(ww)-1;

subplot(2, 1, 1);
stem(n, xx, 'filled');
title('Input Signal x[n]');
xlabel('n');
ylabel('x[n]');
axis([0 75 -10 260]);

subplot(2, 1, 2);
stem(n1, ww, 'filled');
title('Output Signal w[n]');
xlabel('n');
ylabel('w[n]');
axis([0 75 min(ww)-10 max(ww)+10]);
```

Code for the deconvolution for 1-D filters.

The code shows the application of an FIR filter to a discrete-time signal. The input signal undergoes filtering with coefficients [1, -0.9], which highlights the edges of the pulses generated. This is achieved by amplifying the difference between successive samples, resulting in pronounced spikes at the onset and conclusion of each pulse, thereby making the filter particularly useful for detecting changes or edges within the signal.

(a) The input signal graph presents a series of high-amplitude pulses against a background of zeros. These pulses are 10 samples wide and are repeated every 50 samples throughout the signal. Meanwhile, the output signal graph reveals the FIR filter's effect on the input signal. Due to the filter's design, which subtracts 0.9 times the previous value from the current value, the output signal exhibits sharp spikes at the edges of each pulse in the input signal. Between these pulses, the output signal remains close to zero which indicates little to no change in the signal.



Plot of the input signal x[n] and output signal w[n].

3.1 a) Mathematical effect of the filter coefficients in $w[n] = x[n] - 0.9x[n-1]$

$x[n] = \begin{cases} 0\text{-}9 = 256 \\ 10\text{-}99 = 0 \\ 50\text{-}59 = 256 \\ 60\text{-}99 = 0 \\ 100 = 256 \end{cases}$ $w[n] = x[n] - 0.9x[n-1]$

$w[0] = x[0] - 0.9x[-1]$
$256 - 0.9(0) = 256$ (follows the output signal)

$w[1] = x[1] - 0.9x[0]$
$256 - 0.9(256) = 25.6$ (follows the output signal)

@ n value of 1 to 9 have the same output of 25.6.

$w[10] = x[10] - 0.9x[9]$
$0 - 0.9(256) = -230.4$ (follows the output signal)

$w[11] = x[11] - 0.9x[10]$
$0 - 0.9(0) = 0$ (follows the output signal)

☆ n value of 11 to 99 have the same output of 0.

n=50 repeats the process.

$w[n] = \begin{cases} n \,\%\, 50 = 0 & \text{then } 256 \\ 1 \le n \,\%\, 50 \le 9 & \text{then } 25.6 \\ n \,\%\, 50 = 10 & \text{then } -230.4 \\ \text{otherwise} & \text{then } 0 \end{cases}$ piecewise function to define $w[n]$.

Mathematical proof for the effects of the filter coefficients.

The mathematical proof explains why the output signal w[n] looks the way it does; When n values are substituted into the FIR filter's formula, then the resulting values correspond to the amplitude of the specific pulse in the graph after the filter's effects. It also is loyal to the graph generated since every single point on the graph can be calculated using the given formula. And the piecewise function gives the filter coefficients a standardized model to follow.

(b) The length of the filtered signal is the summation of the length of the input signal and the number of coefficients minus 1. The code for this section also uses this formula for calculating the range of x values for the w[n] - *n1 = 0:length(ww)-1*. In the case for the given parameters, since the length of the input signal is 101 and the number of coefficients is 2, then the length of the filtered signal is 101 + 2 - 1 = **102**.

3.1.1 Restoration Filter

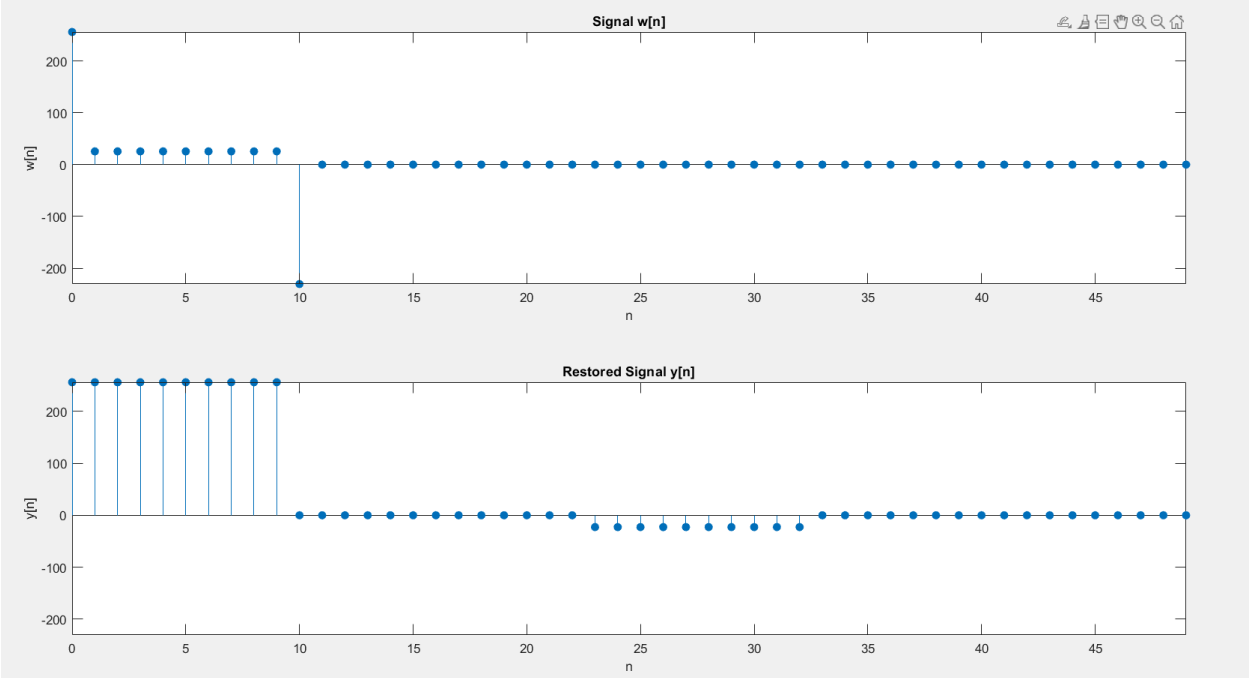
```
bb = ones(1,23);
for M = 0:22
   bb(M+1)=(0.9^M);
end
yy = firfilt(bb, ww);
```

(a) Code to process the output signal y[n].

```
n1 = 0:length(ww)-1;
n2 = 0:length(yy)-1;

figure;
subplot(2, 1, 1);
stem(n1, ww, 'filled');
title('Signal w[n]');
xlabel('n');
ylabel('w[n]');
axis([0 49 min(ww) max(ww)]);

subplot(2, 1, 2);
stem(n2, yy, 'filled');
title('Restored Signal y[n]');
xlabel('n');
ylabel('y[n]');
axis([0 49 min(ww) max(ww)]);
```

Code to plot Signal w[n] and the Restored Signal y[n].

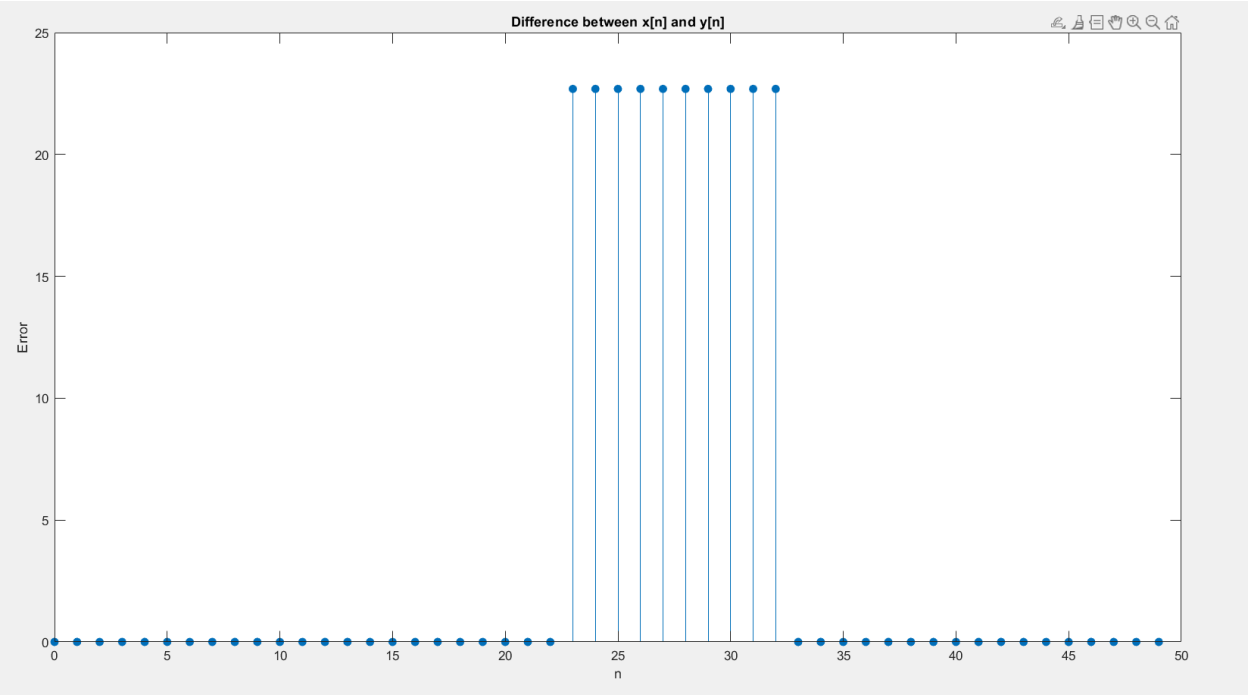
The code shows the application of the restoration filter where the exponentially decaying coefficients are applied to a signal that needs to be restored. By applying the restoration filter with its coefficients, its purpose is to counteract or reverse the effects of the initial FIR filter, thereby attempting to bring the signal closer to its original form.



(b) Plot of the original signal with its restored form.

```
yy1 = yy(1:length(xx));
error = xx - yy1;

figure;
stem(0:49, error(1:50), 'filled');
title('Difference between x[n] and y[n]');
xlabel('n');
ylabel('Error');
```

Code to showcase the error difference between y[n] and x[n].



(c) Plot of the Error Difference between x[n] and y[n].

The error difference between x[n] and y[n] shows how effective the restoration filter has been able to counteract or reverse the effects of the initial FIR filter. The error values between x[n] and y[n] are consistently 22.6891 for n values from 23 to 32, indicating a specific, uniform discrepancy between the original signal and the restored signal over this range of samples. This uniform error suggests that the restoration filter is unable to perfectly correct the signal alterations within this segment. The error magnitude of 22.6891 reflects the difference in amplitude between the expected signal values and the actual signal values for these particular samples.

3.1.2 Worse-Case Error

```
difference = abs(xx(1:50) - yy(1:50));
worsecase = max(difference);

disp(num2str(worsecase));
```

Code to calculate the worse-case error difference of x[n] and y[n].

(a) Using the max() function on the absolute value of the difference between the 0 and 50th range of x[n] and y[n], we get a value of **22.6891** - The same value which appears as a range of spikes on the graph showing the difference between the two vectors.

(b) The error plot and worst-case error indicate the effectiveness of the signal restoration process. A smaller worst-case error suggests a closer approximation of the original signal. If the error plot shows consistently low values, it implies that the restoration filter successfully recovers the original signal with minimal changes. For the relation between the error plot and the worse case error value, we can see that the error value is present between the n values of 23 to 34. This means that those specific points on the graph have rather poor restoration quality as compared to other points on the graph.

3.1.3 An Echo Filter

$$y_1[n] = x_1[n] + r\,x_1[n - P]$$

FIR Filter for an echo.

(a) r seems to represent the strength of the echo is 90%, or **0.9**, and P seems to be the delay in samples meaning it's 8000 Hz multiplied by 0.2, which equals **1600 Hz**.

(b) The FIR filter will have a coefficient of 1 followed by the delayed signal's values of constant 0s until P. This means that the length of the filter coefficients of the FIR filter is P + 1, which means it's **1601**.

```
load labdat.mat;

fs = 8000;
r = 0.9;
P = 1600;

b = zeros(1, P+1);
b(1) = 1;
b(P+1) = r;

y = firfilt(b, x2);
soundsc(y, fs);
```

(c) Code to produce x2 from labdat.mat with an echo.

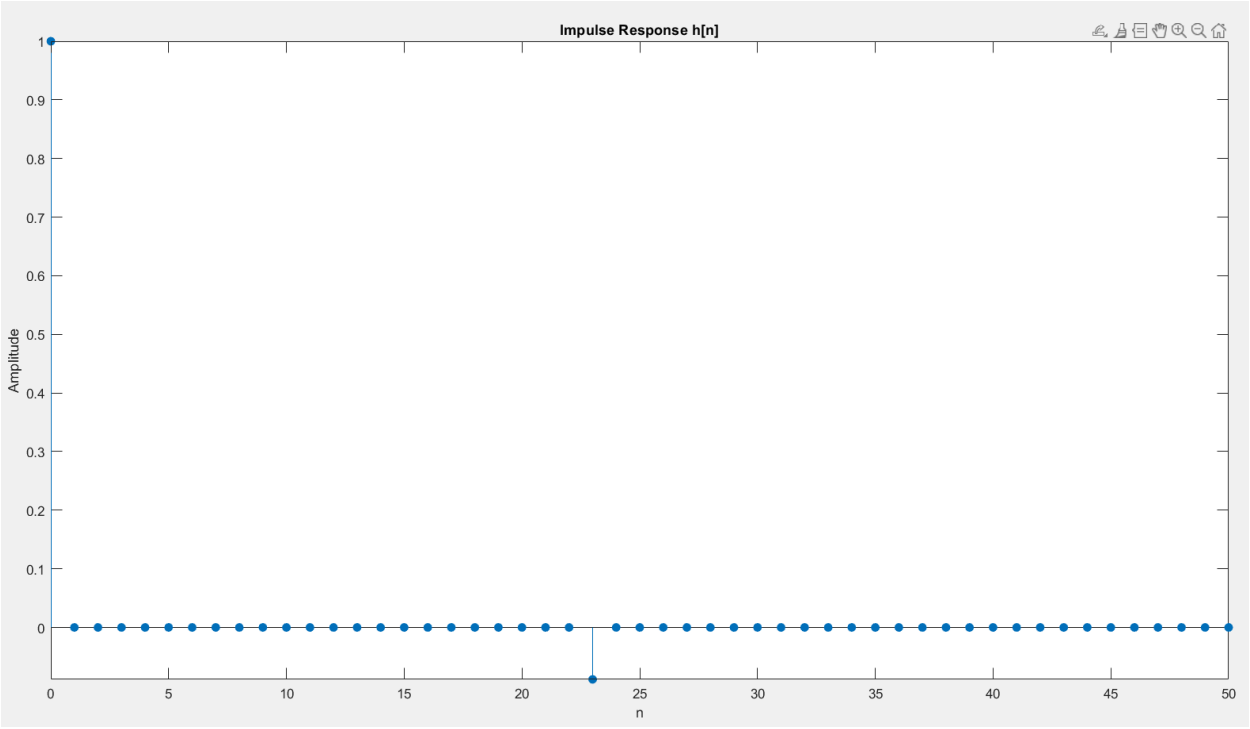
3.2.1 Overall Impulse Response

```
q = 0.9;
h1 = [1, -q];

bb = zeros(1,23);
for M = 0:22
   bb(M+1) = (0.9^M);
end

impulse = [1, zeros(1, 100)];
w = firfilt(h1, impulse);
yy = firfilt(bb, w);

figure;
stem(0:length(yy)-1, yy, 'filled');
title('Impulse Response h[n]');
xlabel('n');
ylabel('Amplitude');
axis([0 50 min(yy) max(yy)]);
```

Code to implement the concept of cascading two systems.

In the code above, an impulse signal, consisting of a followed by one hundred 0s, is processed through the first FIR filter defined by the coefficients [1, -0.9], which effectively applies a simple difference operation emphasizing the change between consecutive samples. This filtered signal is then passed through a second FIR filter with exponentially decaying coefficients, generated by raising 0.9 to the power of 0 through 22, which consequently creates a restoration effect on the signal.

The resulting graph visualizes the impulse response of the cascaded system, which shows the combined result of both FIR filters on a single vector. An initial spike at n = 0 shows the direct response to the impulse, followed by zeroes until a negative spike appears at n = 23. This reflects the characteristics of the second FIR filter's exponentially decaying coefficients.



(a) Plot of the impulse response h[n].

3.2.1 b)

$h[n]$: FIR filter-1 * FIR filter-2

FIR filter-1 : $w[n] = x[n] - 0.9x[n-1]$

: $h_1[z] = 1 - 0.9z^{-1}$

FIR filter-2 : $\sum_{k=0}^{M} r^k w[n-k]$

: $h_2[z] = \sum_{k=0}^{22} (0.9^k z^{-k})$

$h[n] = (1 - 0.9z^{-1})\left((0.9^0 z^0) + (0.9^1 z^{-1}) + (0.9^2 z^{-2}) + (0.9^3 z^{-3}) + \ldots + (0.9^{22} z^{-22})\right)$

$= (1 + 0.9^1 z^{-1} + 0.9^2 z^{-2} + \ldots + 0.9^{22} z^{-22}) - (0.9^1 z^{-1} - 0.9^2 z^{-2} - \ldots - 0.9^{22} z^{-22} - 0.9^{23} z^{-23})$

$h[n] = 1 - 0.9^{23} z^{-23}$

$\boxed{h[n] = [1, -0.0486]}$

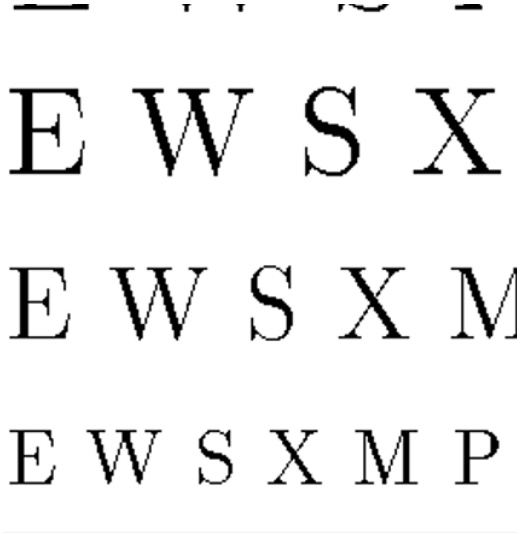
(b) Proof for the impulse response h[n].

3.2.2 Distorting and Restoring Images

(a) Load in the image echart.mat

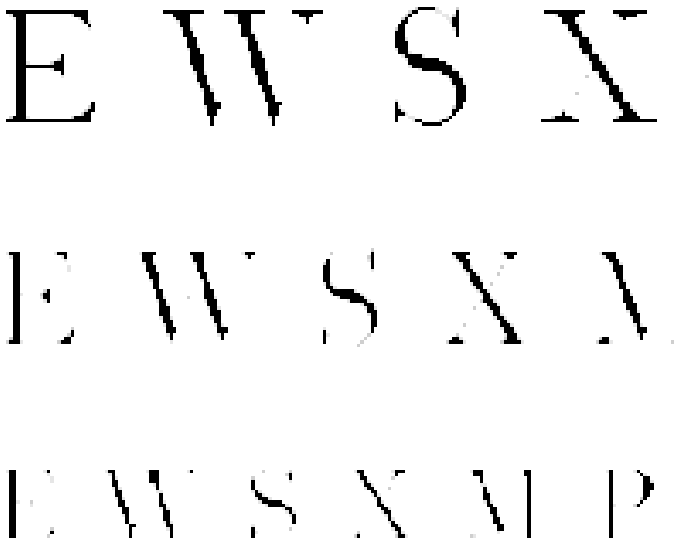
```
% Load the image echart.mat

load echart.mat;
```

Generates a matrix called echart.



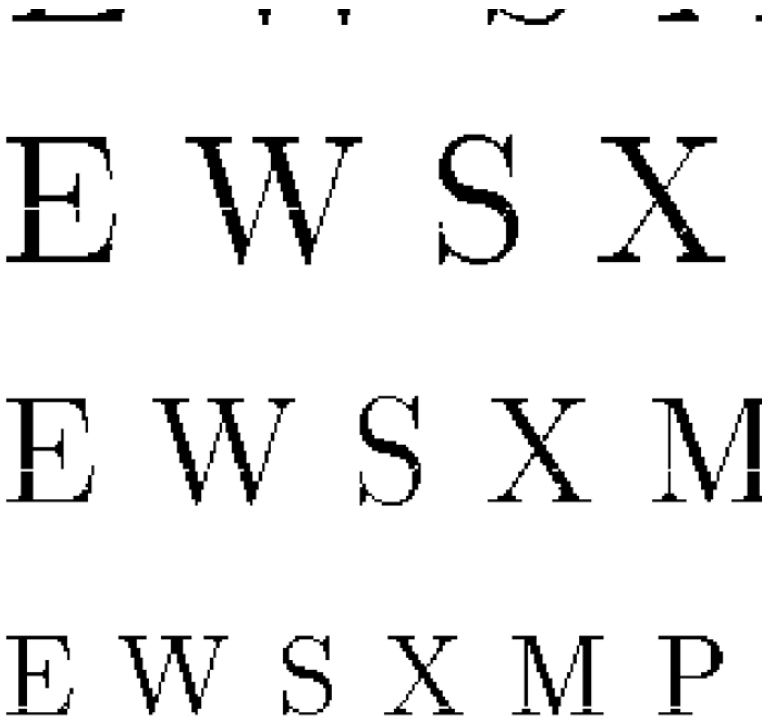
(b) Pick q = 0:9 in FILTER-1 and filter the image echart in both directions

```
q = 0.9;

bdiffh = [1, q];

% Apply FILTER-1 horizontally

ech90_horizontal = conv2(echart, bdiffh);

% Apply FILTER-1 vertically to the result of the horizontal filtering

ech90 = conv2(ech90_horizontal, bdiffh');
```



(c) Deconvolve ech90 with FIR FILTER-2,

```
r = -0.9;

M = 22;

brestore = r.^(0:M);

% Deconvolve with FILTER-2

ech_restored = conv2(ech90, brestore);

ech_brestored = conv2(ech_restored, brestore);
```



The Visual appearance of the output does not look very clear and has visible white lines which makes the deconvoluted image have a “ghostly appearance”. Our mathematical understanding on the cascade filtering process is that the deconvoluted image is obtained by filtering the image first with the FIR_Filter1 and then the resulting image is filtered with the FIR_filter2 and with these the ghostly appearance of the image could be explained as the result of the ringing artifacts from the deconvolution process. Since the deconvolution filter is the reverse effect of the convolutional effect of the first filter although it does not work perfectly and using which causes the signal energy leak into the nearby pixels which appears as “ghosts” in the output image and based on the previous calculations the size of these ghosts or echoes could be determined as it depends on the parameters of the filters and the characteristics of the image as well by comparing the previous outputs and experimenting on their parameters, we believe that the ghosts will get noticeable or bigger in size in images with sharp edges and higher contrasts. We are also able to calculate the worst-case error through the parameters of the two filters, the noise level in the image, and the original image’s characteristics.Relative to black and white, the worst-case error would be equal to the amplitude of the ringing artifacts which could be calculated by the absolute value of the difference between the two filters. |ringing_artifacts

$$ringing\ artifacts\ = |FIRFILTER_1(1)\ -\ FIRFILTER_2(1)|$$

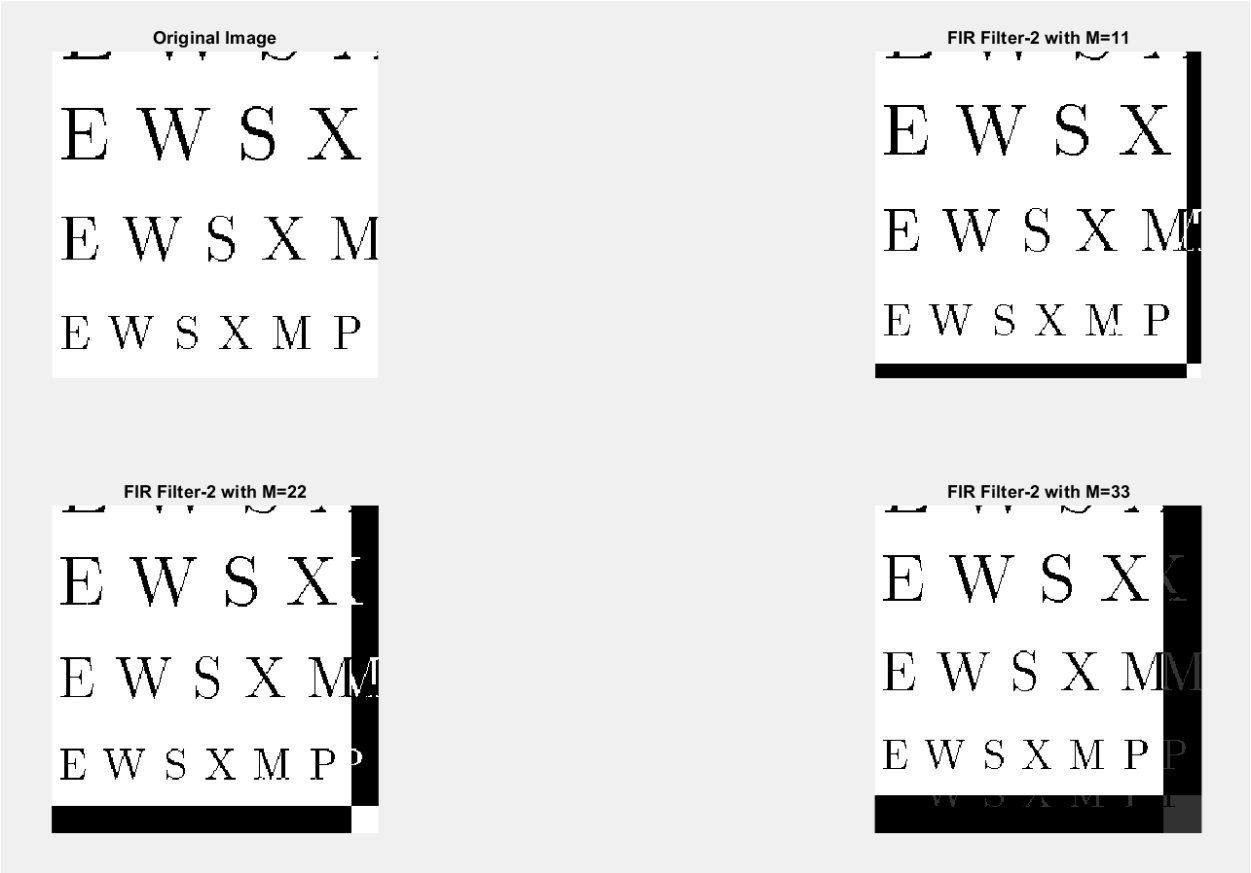
```
% Evaluate worst-case error

min_height = min(size(echart, 1), size(ech_brestored, 1));

min_width = min(size(echart, 2), size(ech_brestored, 2));
```

```
worst_case_error = max(abs(echart(1:min_height, 1:min_width) -
ech_brestored(1:min_height, 1:min_width)), [], 'all');

fprintf('Worst-case error: %f\n', worst_case_error);
```

3.2.3 A Second Restoration Experiment

```
load echart.mat

aa = [1 -0.9];
ech90 = conv2(echart, aa);
ech90 = conv2(ech90, aa');

figure;
subplot(2, 2, 1);
imshow(echart);
title('Original Image');

bb = 0.9.^(0:11);
M1 = conv2(ech90, bb);
M1 = conv2(M1, bb');
subplot(2, 2, 2);
imshow(M1);
title('FIR Filter-2 with M=11');

bb = 0.9.^(0:22);
M2 = conv2(ech90, bb);
M2 = conv2(M2, bb');
subplot(2, 2, 3);
imshow(M2);
title('FIR Filter-2 with M=22');

bb = 0.9.^(0:33);
M3 = conv2(ech90, bb);
M3 = conv2(M3, bb');
subplot(2, 2, 4);
imshow(M3);
title('FIR Filter-2 with M=33');
```

Code to generate images with different iterations of FIR filter-2.



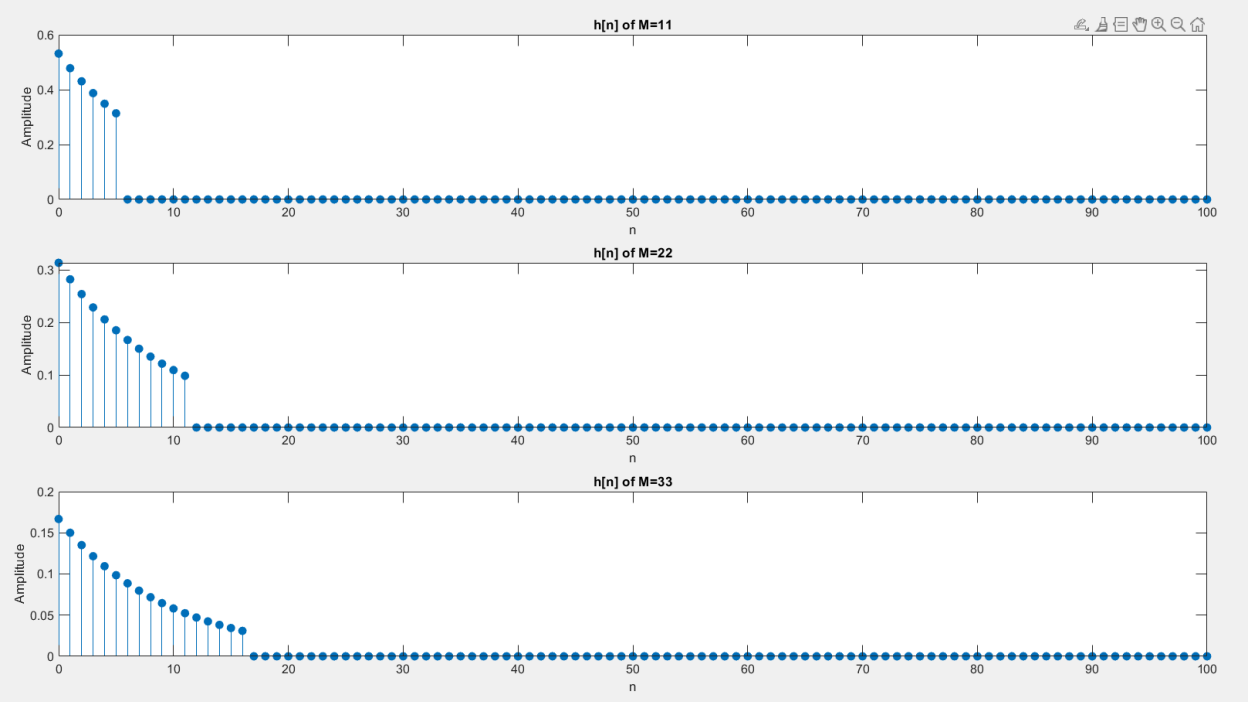
The different resulting images of FIR Filter-2.

(a) By observation of the resulting images, M=33 seems to produce the best image because it significantly reduces the visibility of "ghosts" during the filtering process thereby minimizing the errors or distractions from the deconvolved image. On the other hand, the use of M=22 and 11 seem to be not optimal because it not only makes "ghosts" apparent but also introduces white marking within some of the letters, which is far off from the original image. M=33 being the best option all has something to do with the diminishing impact of the negative final coefficient which is a result from the convolution process of the system's filter coefficients.

This can be further demonstrated by plotting their impulse response and comparing them side-by-side. M=33 generally has a lower impulse response as compared to the other M values, which means that it's more suitable for the convolution process because it preserves fine details while also enhancing the image's resolution and sharpness. A higher impulse response, on the other hand, prioritizes smoothing which is not important for filtering the relevant image.

```
Ms = [11, 22, 33];

figure;
for i = 1:length(Ms)
   bb = 0.9.^(0:Ms(i));
   impulse = [1, zeros(1, 100)];

   h = conv(impulse, bb, 'same');

   subplot(length(Ms), 1, i);
   stem(0:length(h)-1, h, 'filled');
   title(sprintf('h[n] of M=%d', Ms(i)));
   xlabel('n');
   ylabel('Amplitude');
end
```

Code to generate the impulse response of the different M values.



Different impulse responses of FIR filter-2.

```
% M=11
difference1 = abs(M1(1:257, 1:256) - echart);
worstcase1 = max(difference1, [], "all");

% M=22
difference2 = abs(M2(1:257, 1:256) - echart);
worstcase2 = max(difference2, [], "all");

% M=33
difference3 = abs(M3(1:257, 1:256) - echart);
worstcase3 = max(difference3, [], "all");

disp(worstcase1);
disp(worstcase2);
disp(worstcase3);
```

Code to calculate the worst-case errors of different M values.

(b) By using the max() function on the absolute value of the difference between the filtered images and the original within 256x256 pixels, we are able to get the worst-case errors between the 3 different M values. And these are:

M=11 has a worst-case error value of 144.0391,
M=22 has a worst-case error value of 45.2010,
M=33 has a worst-case error value of 14.1845.

In the context of the application of FIR filter-2 with different M values being applied to a single image, if the worst-case error values are close to or less than one gray scale level then it might appear too obscure or just flat out unnoticeable.