C0E 472: DIGITAL SIGNAL PROCESSING

# Lab 07: Sampling, Convolution, and FIR Filtering

INDEX NUMBER: 3050420

NAME: HARDY HAWA TUNTEIYA

COURSE: BSc. Biomedical Engineering

Year 4

# INTRODUCTION

The goal of this lab is to learn how to implement FIR filters in MATLAB and then study the response of FIR filters to various signals, including images and speech. The lab focuses on learning how filters can create interesting effects such as blurring and echoes.

# LAB EXCERCISES

## 1. Deconvolution Experiment for 1-D Filters:

**Objective**: Implement an FIR filter, plot the original input signal (that will be altered with the FIR filter) and the output signal. Compare their lengths and analyze the filtering effects.

```
% Deconvolution for 1-D filters
xx = 256 * (rem(0:100, 50) < 10); %signal
bb = [1, -0.9]; % Filter coefficients


% Filtering
w = firfilt(bb, xx); % w[n]=x[n]-0.9x[n-1]


% Plotting
figure;
subplot(2,1,1);
stem(0:100, xx, 'b');
title('Input Signal x[n]');
xlabel('Time Index n'); ylabel('Amplitude');
xlim([0 75]);


subplot(2,1,2);
stem(0:length(w)-1, w, 'filled', 'b');
title('Output Signal w[n]');
xlabel('Time Index n'); ylabel('Amplitude');
xlim([0 75]);


% Display the lengths of the signals
disp(['Length of x[n]: ', num2str(length(xx))]);
disp(['Length of w[n]: ', num2str(length(w))]);
```

# OUTPUT

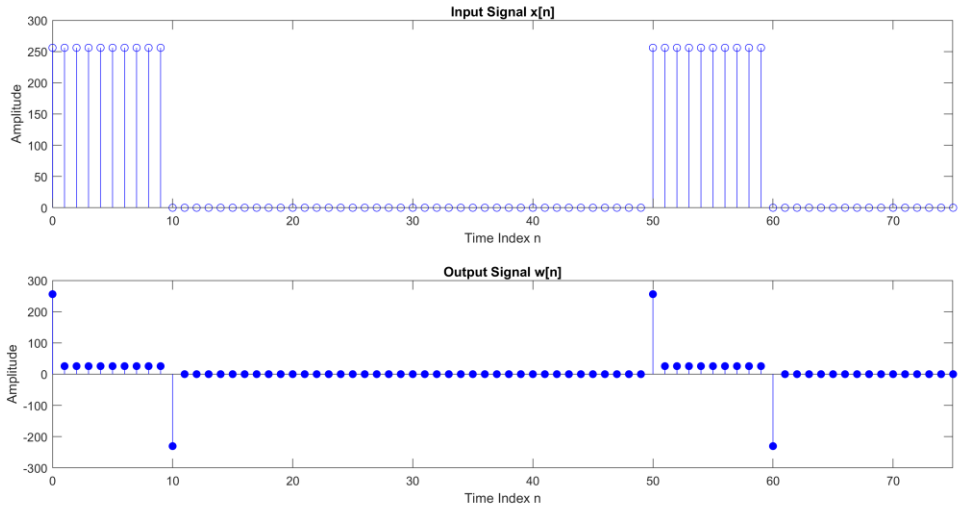



*Figure 1 Deconvolution Experiment for 1-D Filters*

Explanation:

The FIR filter w[n] is a difference equation. The output $w[n]$ appears the way it does because it represents the difference between the current value of the input signal and 0.9 times the previous value. This effectively creates a high-pass filter. Thus, rapid changes in the signal are amplified and similar values are reduced.

Here's how it processes the input signal x[n]:

- The filter starts with an initial condition. Assuming x[−1]= 0
- For each sample n, the filter computes the output w[n] as:
  - For n=1: $w[1] = 256 - 0.9x(0) = 256$
  - For n=2: $w[2] = 256 - 0.9x(256) = 25.6$

(b) Length of the filtered signal:

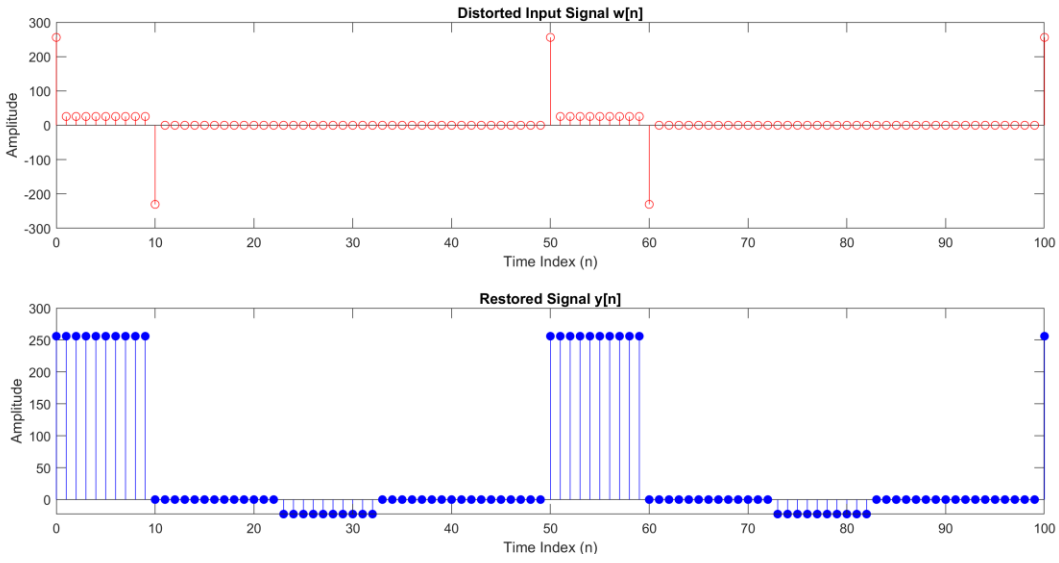
The length of x[n] is determined by the range of n. n goes from 0 to 100, so the length of x[n] is 101 samples. The length of w[n] is 102 samples. This is because the filter includes the value x[n-1] in its operation.

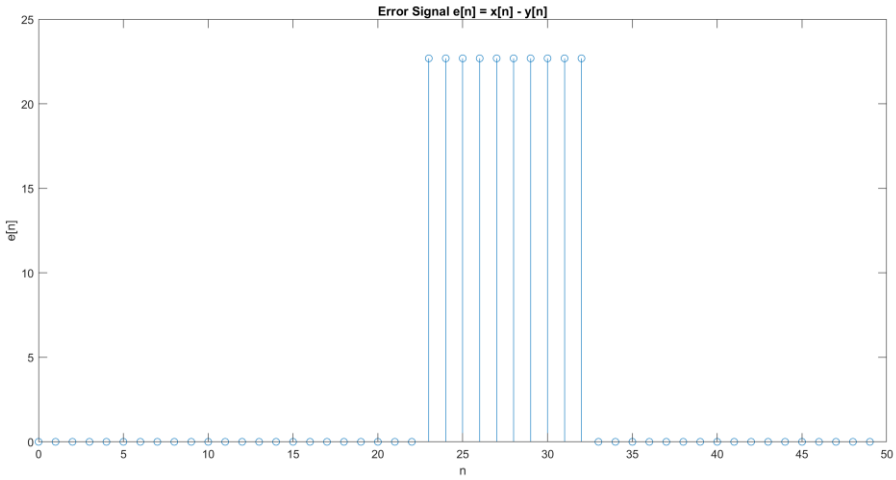
## 1.1 Restoration Filter

- **Objective**: Restore signal distorted by filter.

```
%3.1 RESTORATION
rr = 0.9.^(0:22); % Filter coefficients for restoration filter
zz = firfilt(rr, w);
% Plot the input and output signals
figure;
subplot(2,1,1);
stem(0:length(w)-1, w, 'r');
title('Distorted Input Signal w[n]');
xlabel('Time Index (n)'); ylabel('Amplitude');
xlim([0 100]);

subplot(2,1,2);
stem(0:length(zz)-1, zz, 'filled', 'b');
title('Restored Signal y[n]');
xlabel('Time Index (n)'); ylabel('Amplitude');
xlim([0 100]);

% Compute the error between x[n] and y[n]
error = xx - zz;
% Plot the error signal
figure;
stem(n(1:50), error(1:50));
title('Error Signal e[n] = x[n] - y[n]');
xlabel('n');nylabel('e[n]');
xlim([0 50]);
```

OUTPUT






### 1.2 Worst-Case Error

```
% Evaluate the worst-case error in the range 0 <= n < 50
worst_case_error = max(abs(error(1:50)));

% Plot the error signal
figure;
stem(n(1:50), error(1:50));
title('Error Signal e[n] = x[n] - y[n]');
xlabel('n');
ylabel('e[n]');
xlim([0 50]);

% Display the worst-case error
disp(['Worst-Case Error: ', num2str(worst_case_error)]);
```

## OUTPUT

Worst-Case Error: 22.6891

b. Analysis of error plot

The worst-case error quantifies the maximum deviation between the restored signal y[n] and the original signal x[n]. A smaller worst-case error indicates a higher quality restoration. Ideally, the error should be as close to zero as possible for a good restoration.

The error should be small enough or close to zero so thatit is not visually distinguishable on the plot.

## 1.3 An Echo Filter

- **Objective**: Implement a filter that adds an echo to an audio signal.

The FIR filter equation given is: $y_1[n] = x_1[n] + r\, x_1[n - P]$

**(a) Determine the Values of r and P**

Sampling Frequency fs=8000 Hz. Desired echo delay = 0.2 seconds

Strength of echo r=0.9

P=fs×time delay=8000×0.2=1600 samples

**(b) Describe the Filter Coefficients and Length**

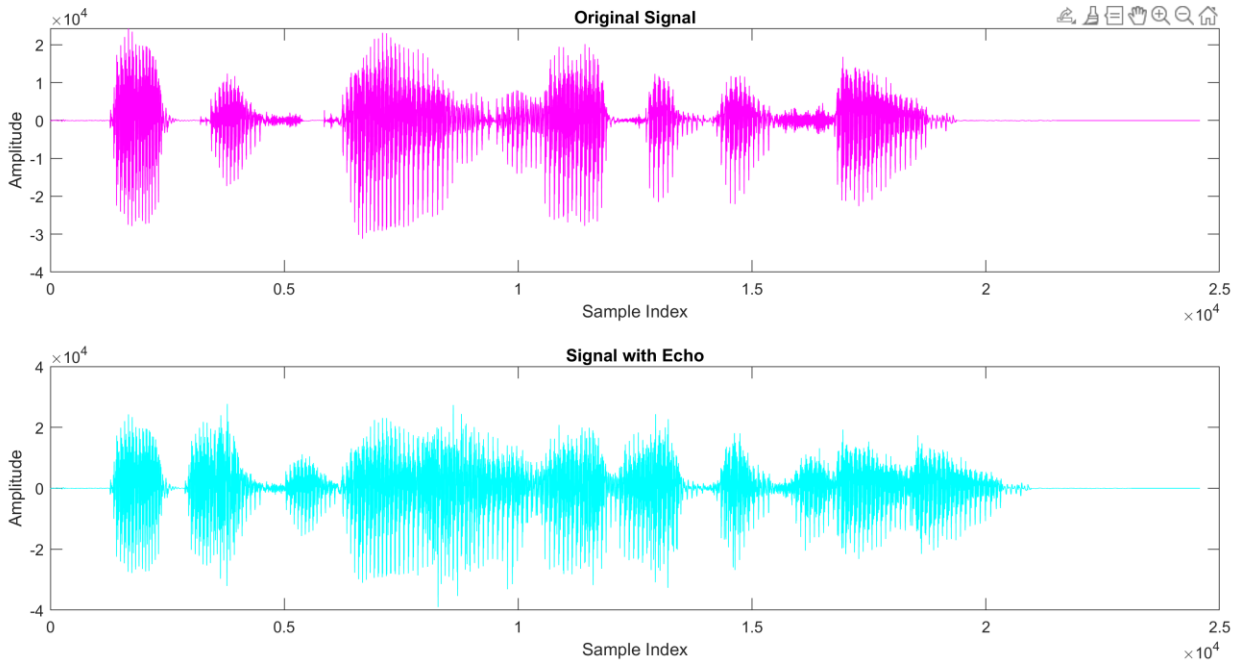
The filter coefficients are [1,0,...,0,r] This means there are P−1 zeros between the first coefficient (1) and the last coefficient (r).

Length of the filter P+1=1601

**(c) Implement the Echo Filter**

```
% Filter coefficients
r = 0.9;
P = 1600;
b = [1, zeros(1, P-1), r];
% Filter that adds echo
y1 = filter(b, 1, x2);

% Play the original and the echoed signals
sound(x2, 8000);
pause(length(x2)/8000 + 2);

sound(y1, 8000);
% Plot the original and echoed signals for visual inspection
figure;
subplot(2,1,1);
plot(x2, 'm');
title('Original Signal');
xlabel('Sample Index');
ylabel('Amplitude');


subplot(2,1,2);
plot(y1, 'c');
title('Signal with Echo');
xlabel('Sample Index');
ylabel('Amplitude');
```

OUTPUT

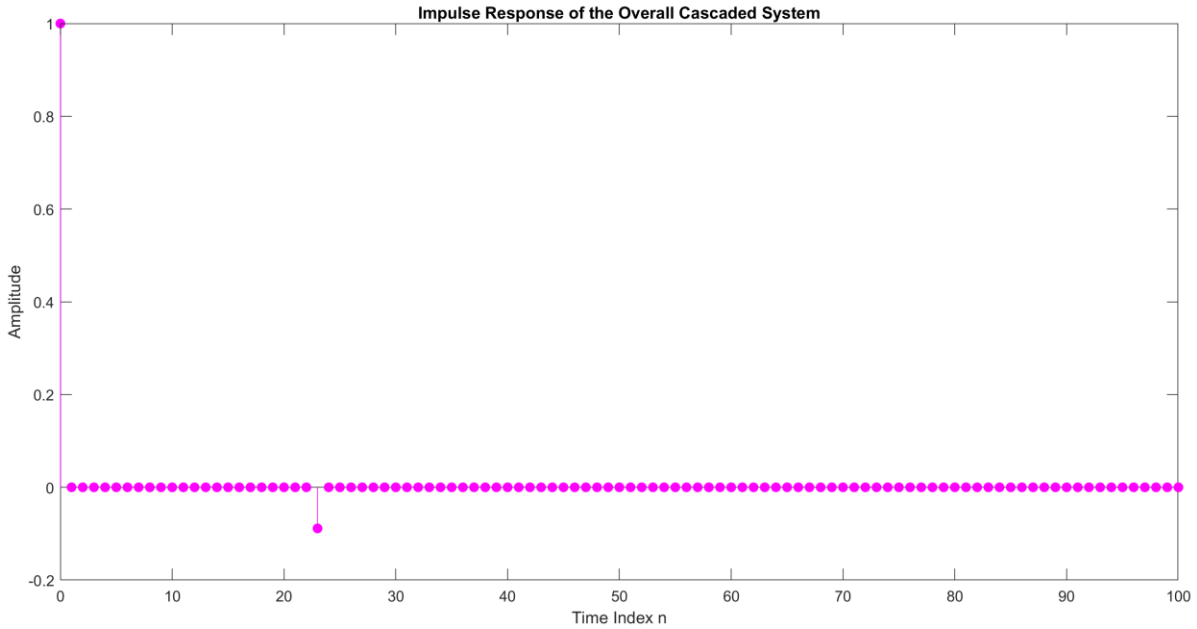



## 2. Cascading Two Systems

- **Objective**: Understand the combined effects of two FIR filters applied sequentially.

```
impulse = [1, zeros(1, 100)];
% Define the coefficients for FIR FILTER-1
q = 0.9;
b1 = [1, -q];
% Define the coefficients for FIR FILTER-2
r = 0.9;
M = 22;
b2 = r.^(0:M);
% Apply FIR FILTER-1
w = filter(b1, 1, impulse);
% Apply FIR FILTER-2
y = filter(b2, 1, w);
% Plot the impulse response of the overall cascaded system
figure;
stem(0:length(y)-1, y, 'filled', 'm');
title('Impulse Response of the Overall Cascaded System');
xlabel('Time Index n');
ylabel('Amplitude');
xlim([0 100]);
```

OUTPUT




For perfect deconvolution, the overall system should behave as an identity system: $y[n]=x[n]$. This implies: $h1[n]*h2[n]=\delta[n]$

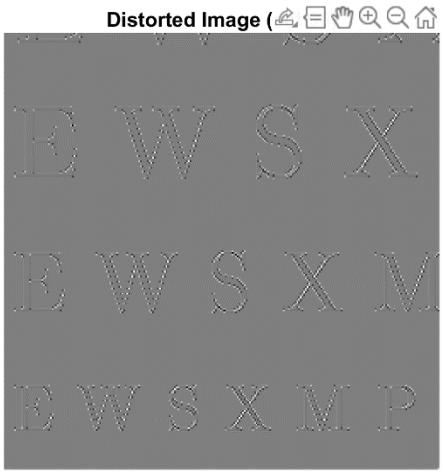
This means the convolution of h_1[n] and h_2[n] should yield the identity system's impulse response, which is the Kronecker delta function.

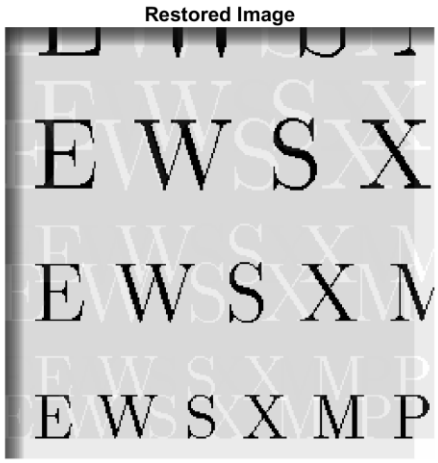
## 2.2 Distorting and Restoring Images

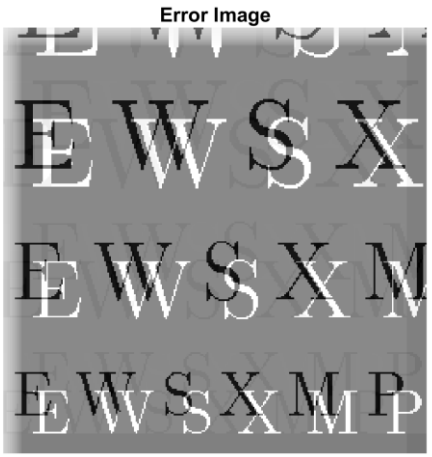
```
% (a) Load the image
load('echart.mat'); % Assuming the image is stored in 'echart' variable
% (b) Apply FIR FILTER-1 in both directions
q = 0.9; h1 = [1, -q];

% Apply the filter horizontally
ech90_h = filter2(h1, echart);
% Apply the filter vertically
ech90 = filter2(h1', ech90_h);

% Display the distorted image
figure;
imshow(ech90, []);
title('Distorted Image (ech90)');

% (c) Deconvolve ech90 with FIR FILTER-2
r = 0.9; M = 22; h2 = r .^ (0:M);
```

```
% Apply the deconvolution filter horizontally
ech_restored_h = filter2(h2, ech90);

% Apply the deconvolution filter vertically
ech_restored = filter2(h2', ech_restored_h);

% Display the restored image
figure;
imshow(ech_restored, []);
title('Restored Image');

% Evaluate the worst-case error
error_image = echart - ech_restored;
worst_case_error = max(abs(error_image(:)));

% Display the error image
figure;
imshow(error_image, []);
title('Error Image');

% Display the worst-case error
disp(['Worst-Case Error: ', num2str(worst_case_error)]);
```

OUTPUT



**Visual Appearance**: The distorted image shows blurring or "ghost" artifacts due to the filtering. The restored image reverses these effects, but some residual artifacts ("ghosts") remain.

**"Ghosts"**: These artifacts appear because the deconvolution is not perfect. They are more noticeable in regions with sharp transitions (e.g., black-to-white).

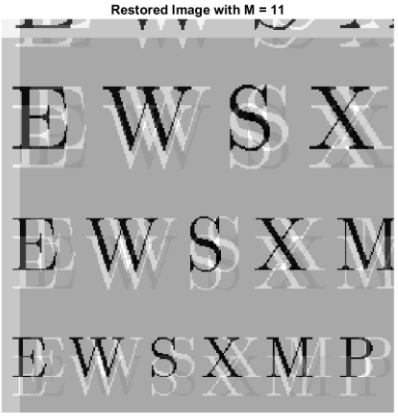
**Worst-Case Error**: For high-quality restoration, this error should be minimal, ideally below the threshold of human visual perception.

## 3.3 A Second Restoration Experiment

```
% Define FIR FILTER-1 coefficients
q = 0.9;
b1 = [1, -q];


% Define the filter lengths for FIR FILTER-2
M_values = [11, 22, 33];
r = 0.9;


% Preallocate for results
best_m = 0;
best_restored_image = [];
best_error = Inf;


for M = M_values
    % Define FIR FILTER-2 coefficients
    b2 = r.^(0:M);
    % Apply FIR FILTER-2 along the horizontal direction
    y_horizontal_filtered = filter(b2, 1, ech90, [], 2);


    % Apply FIR FILTER-2 along the vertical direction
    y = filter(b2, 1, y_horizontal_filtered, [], 1);
    % Display the restored image
    figure;
    imshow(y, []);
    title(['Restored Image with M = ', num2str(M)]);

    % Calculate the error
    error = echart - y;


    % Evaluate the worst-case error
    worst_case_error = max(abs(error(:)));
```

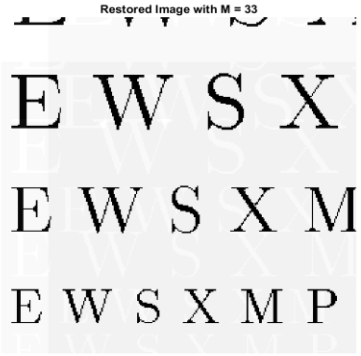
```
    disp(['Worst-Case Error for M = ', num2str(M), ': ',
num2str(worst_case_error)]);

    % Find the best result
    if worst_case_error < best_error
        best_error = worst_case_error;
        best_m = M;
        best_restored_image = y;
    end
end
% Display the best restored image
figure;
imshow(best_restored_image, []);
title(['Best Restored Image with M = ', num2str(best_m)]);
```

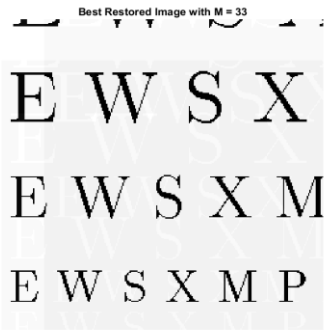
OUTPUT




Worst-Case Error for M = 11: 144.0391


Worst-Case Error for M = 22: 25.01




Worst-Case Error for M = 33: 14.1845

**Visual Appearance**: By comparing the restored images for different values of MMM, you can determine which filter produces the best restoration. The best result is the one with the minimal visual artifacts and the smallest worst-case error.

**Worst-Case Error in Gray Levels**: This metric helps in understanding how significant the residual artifacts are. If the error is less than 1 gray level, it might not be perceptible to the human eye.