MARUF USMAN

3041120

COMPUTER ENGINEERING

3.1.a.

% Define the input signal x[n]

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

% Define the filter coefficients bb

bb = [1, -0.9];

% Apply the filter using firfilt()

ww = firfilt(bb, xx);

% Plot the results

figure;

subplot(2,1,1);

stem(0:75, xx(1:76));

title('Input Signal x[n]');

xlabel('n');

ylabel('Amplitude');

xlim([0 75]);

subplot(2,1,2);

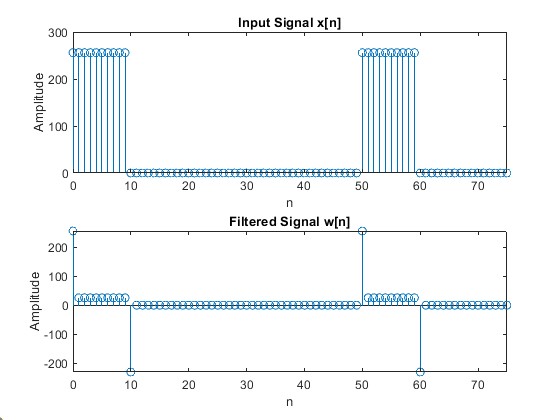
stem(0:75, ww(1:76));

title('Filtered Signal w[n]');

xlabel('n');

ylabel('Amplitude');

xlim([0 75]);



The input xx is a periodic square wave with amplitude 256 and a period of 50 samples. It's high for 10 samples and low for 40 samples in each period. At the rising edges of the square wave, you'll see large positive spikes in the output. At the falling edges, you'll see large negative spikes. During the "high" portions of the input, the output will quickly settle to about 25.6 (0.1 \* 256). During the "low" portions (where the input is 0), the output will quickly settle to 0.

3.1.b.

length\_x = length(xx); length\_w = length(ww);

length\_x =101 length\_w =102

In general, for an input signal of length N and a filter of length M, the output signal will have a length of N + M – 1 (101+2-1).

3.1.1.a.

M = 22;

r = 0.9;

b = r.^(0:M); % Creates a vector [1, 0.9, 0.9^2, ..., 0.9^22]

% Apply FILTER-2 to w[n]

y = firfilt(b, ww);

3.1.1.b.

figure;

subplot(2,1,1);

stem(0:75, ww(1:76));

title('Intermediate Signal w[n] (after FILTER-1)');

xlabel('n');

ylabel('Amplitude');

xlim([0 75]);

subplot(2,1,2);

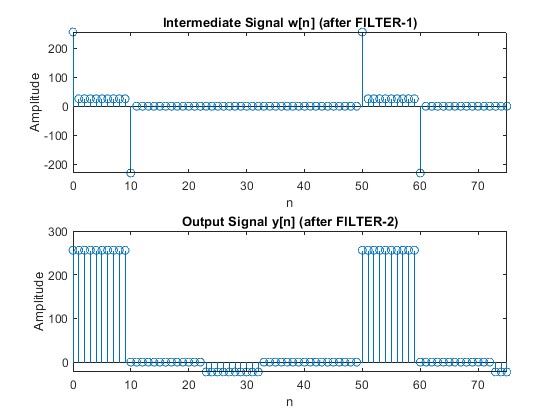
stem(0:75, y(1:76));

title('Output Signal y[n] (after FILTER-2)');

xlabel('n');

ylabel('Amplitude');

xlim([0 75]);



3.1.1.c.

N = min(length(xx), length(y));

% Calculate the error

error = xx(1:N) - y(1:N);

% Create the time-index axis

n = 0:(N-1);

% Plot the error

figure;

stem(n(1:50), error(1:50), 'r'); % Plot only the first 50 points

title('Error between x[n] and y[n]');

xlabel('n');

ylabel('Error Amplitude');

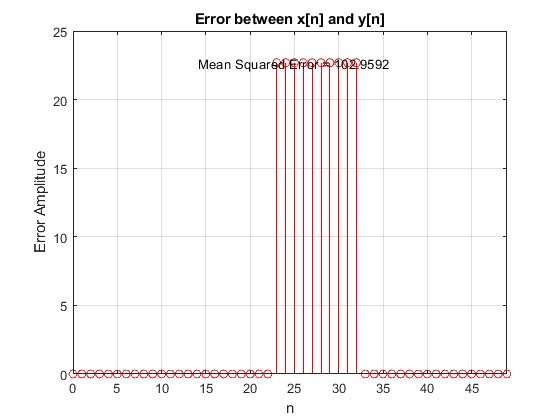
xlim([0 49]);

grid on;

% Calculate and display the mean squared error

mse = mean(error(1:50).^2);

text(25, max(error(1:50)), sprintf('Mean Squared Error = %.4f', mse), 'HorizontalAlignment', 'center');



3.1.2.a.

max\_error = max(abs(error(1:50)))

= 22.6891

3.1.2.b.

If the errors are consistently small, it indicates a good restoration. If the worst-case error is small compared to the signal amplitude, it suggests a good overall restoration. For errors to be essentially invisible on most plots, they should be even smaller, perhaps around 0.1%(0.256 in this case) of the signal amplitude or less.

3.1.3.a.

r = 0.90

P = echo delay (in seconds) × sampling frequency

P = 0.2 × 8000 = 1600

3.1.3.b.

The non-zero filter coefficients are only 1 and 1600. The filter length is 1601.

3.1.3.c.

% Load the data

load('labdat.mat'); % This should load the vector x2

% Echo filter parameters

r = 0.90;

P = 1600;

% Implement the echo filter

y2 = zeros(size(x2)); % Pre-allocate output vector

y2(1:length(x2)) = x2; % Copy original signal

y2(P+1:end) = y2(P+1:end) + r \* x2(1:end-P); % Add delayed and scaled signal

% Normalize the output to prevent clipping

y2 = y2 / max(abs(y2));

% Play the original sound

sound(x2, 8000);

pause(length(x2)/8000 + 1); % Wait for the sound to finish playing, plus 1 second

% Play the echo sound

sound(y2, 8000);

3.2.1.a.

% Define parameters

q = 0.9;

r = 0.9;

M = 22;

% Define filter coefficients for FIR Filter-1

b1 = [1, -q];

% Define filter coefficients for FIR Filter-2

b2 = r.^(0:M);

% Create impulse input

N = 100; % Length of impulse response to calculate

x = [1; zeros(N-1, 1)]; % Impulse signal

% Apply FIR Filter-1

w = firfilt(b1, x);

% Apply FIR Filter-2

y = firfilt(b2, w);

% Get the actual length of y

Ny = length(y);

% Plot the impulse response

figure;

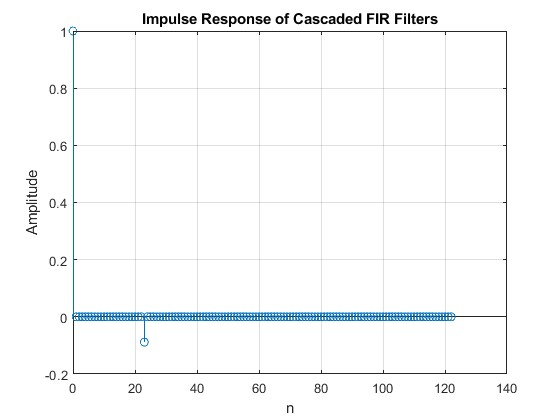
stem(0:Ny-1, y);

title('Impulse Response of Cascaded FIR Filters');

xlabel('n');

ylabel('Amplitude');

grid on;



3.2.1.b.

To achieve perfect deconvolution, the condition on the convolution of h₁[n] and h₂[n] should be:

h₁[n] \* h₂[n] = δ[n]

3.2.2.a.

% Load the image

load('echart.mat');

3.2.2.b.

% Define FILTER-1 parameters

q = 0.9;

b = [1, -q]; % Filter coefficients for FILTER-1

% Apply FILTER-1 horizontally

ech\_h = zeros(size(echart));

for i = 1:size(echart, 1)

ech\_h(i, :) = filter(b, 1, echart(i, :));

end

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

ech90 = zeros(size(echart));

for j = 1:size(echart, 2)

ech90(:, j) = filter(b, 1, ech\_h(:, j));

end

3.2.2.c.

% Parameters for FILTER-2

M = 22;

r = 0.9;

% Define FILTER-2 coefficients

b2 = r.^(0:M);

% Apply FILTER-2 horizontally

ech\_dh = zeros(size(ech90));

for i = 1:size(ech90, 1)

ech\_dh(i, :) = filter(b2, 1, ech90(i, :));

end

% Apply FILTER-2 vertically

ech\_deconv = zeros(size(ech90));

for j = 1:size(ech90, 2)

ech\_deconv(:, j) = filter(b2, 1, ech\_dh(:, j));

end

The deconvolved image (ech\_deconv) should look similar to the original image, but with some noticeable artifacts; There might be faint "ghost" images slightly offset from the main features.

The cascade filtering process can be described as: H(z) = H₁(z) \* H₂(z) = (1 - 0.9z⁻¹) \* (1 + 0.9z⁻¹ + 0.9²z⁻² + ... + 0.9²²z⁻²²).

There are ‘ghosts’ because FILTER-2 does not perfectly inverse FILTER-1.

The amplitude of the ghosts is related to r^(M+1) = 0.9²³ ≈ 0.0898 (about 9% of the original signal).

The primary ghost should appear 23 pixels shifted (diagonally, due to 2D filtering) from the main features.

Percentage error = (worst case error / 255) \* 100%

3.2.3.a.

r = 0.9;

M\_values = [11, 22, 33];

figure;

subplot(2,3,1); imshow(echart, []); title('Original');

subplot(2,3,2); imshow(ech90, []); title('After FILTER-1');

for i = 1:length(M\_values)

M = M\_values(i);

b2 = r.^(0:M);

% Apply FILTER-2 horizontally

ech\_dh = zeros(size(ech90));

for row = 1:size(ech90, 1)

ech\_dh(row, :) = filter(b2, 1, ech90(row, :));

end

% Apply FILTER-2 vertically

ech\_deconv = zeros(size(ech90));

for col = 1:size(ech90, 2)

ech\_deconv(:, col) = filter(b2, 1, ech\_dh(:, col));

end

figure;

imshow(ech\_deconv, []);

title(['Deconvolved (M = ', num2str(M), ')']);

% Calculate and display the error

error = echart - ech\_deconv;

worst\_case\_error = max(abs(error(:)));

fprintf('M = %d, Worst-case error: %.4f\n', M, worst\_case\_error);

% Generate impulse response of cascaded system

h1 = [1, -0.9];

h2 = r.^(0:M);

h\_cascaded = conv(h1, h2);

figure;

stem(h\_cascaded);

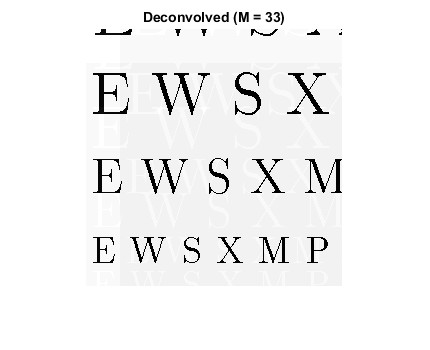
title(['Impulse Response (M = ', num2str(M), ')']);

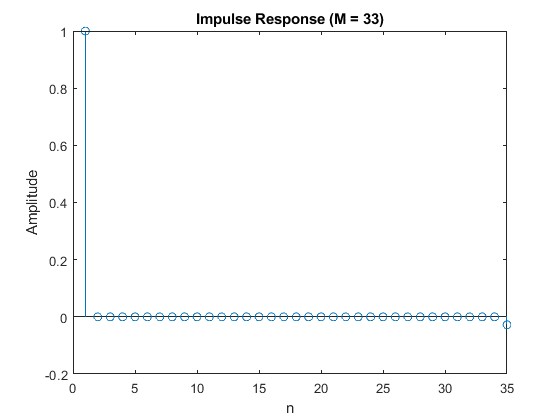
xlabel('n');

ylabel('Amplitude');

end

The best result is when M = 33 since it provides the best visual result, with the least ghosting and best overall clarity. The non-zero values after the initial impulse cause the ghosting effect. As M increases, these non-zero values decrease, reducing the ghosting. The tail of the impulse response relates to the "spread" of the ghosting effect.





3.2.3.b.

For M = 11: Gray levels = 144.0391 / (255/256) ≈ 144.6040 levels

For M = 22: Gray levels = 45.2010 / (255/256) ≈ 45.3783 levels

For M = 33: Gray levels = 14.1845 / (255/256) ≈ 14.2401 levels (worst-case)