**MARUF USMAN**

**COMPUTER ENGINEERING**

**3041120**

**4.1**

**a.** function beta = calculate\_beta(fb, L, fs)

% Calculate unscaled frequency response

n = 0:L-1;

h = cos(2\*pi\*fb\*n/fs);

% Calculate frequency response

[H, ~] = freqz(h, 1, 1024, fs);

% Find maximum magnitude

max\_magnitude = max(abs(H));

% Set beta as reciprocal of max magnitude

beta = 1 / max\_magnitude;

end

**b.** function hh = dtmfdesign(fb, L, fs)

fb = fb(:);

% Number of filters (should be 8 for DTMF)

num\_filters = length(fb);

% Initialize the output matrix

hh = zeros(L, num\_filters);

% Generate time vector

n = (0:L-1)';

% Design each bandpass filter

for i = 1:num\_filters

% Generate the unscaled impulse response

h = cos(2\*pi\*fb(i)\*n/fs);

% Calculate the frequency response

[H, ~] = freqz(h, 1, 1024, fs);

% Find the maximum magnitude of the frequency response

max\_magnitude = max(abs(H));

% Calculate beta to scale the response

beta = 1 / max\_magnitude;

% Scale the impulse response and store in hh

hh(:,i) = beta \* h;

end

% Each BPF is now scaled so that its frequency response has a

% maximum magnitude equal to one.

end

**c.** % DTMF frequencies

low\_freq = [697 770 852 941];

high\_freq = [1209 1336 1477 1633];

fb = [low\_freq high\_freq];

fs = 8000; % Common sampling rate for telephone systems

% Test for L = 40

L\_40 = 40;

hh\_40 = dtmfdesign(fb, L\_40, fs);

% Test for L = 80

L\_80 = 80;

hh\_80 = dtmfdesign(fb, L\_80, fs);

function analyze\_filters(hh, fs, fb)

[num\_samples, num\_filters] = size(hh);

f = linspace(0, fs/2, 1000);

for i = 1:num\_filters

H = freqz(hh(:,i), 1, f, fs);

figure;

plot(f, 20\*log10(abs(H)));

title(['Filter for ', num2str(fb(i)), ' Hz']);

xlabel('Frequency (Hz)');

ylabel('Magnitude (dB)');

grid on;

xlim([0 fs/2]);

ylim([-60 5]);

end

end

% Analyze both cases

analyze\_filters(hh\_40, fs, fb);

analyze\_filters(hh\_80, fs, fb);

**d.** % DTMF frequencies

fb = [697, 770, 852, 941, 1209, 1336, 1477, 1633];

L = 40;

fs = 8000;

% Generate the filters

hh = dtmfdesign(fb, L, fs);

% Calculate frequency response

nfft = 1024;

f = linspace(0, fs/2, nfft/2 + 1);

H = zeros(nfft/2 + 1, 8);

for i = 1:8

[h, w] = freqz(hh(:,i), 1, nfft, fs);

H(:,i) = abs(h(1:nfft/2 + 1));

end

% Plot

figure('Position', [100, 100, 800, 600]);

plot(f, 20\*log10(H));

hold on;

% Add vertical lines for DTMF frequencies

for i = 1:length(fb)

line([fb(i), fb(i)], ylim, 'Color', 'r', 'LineStyle', '--');

end

% Customize plot

title('Magnitude Frequency Response of DTMF Bandpass Filters (L = 40)');

xlabel('Frequency (Hz)');

ylabel('Magnitude (dB)');

grid on;

xlim([0, fs/2]);

ylim([-60, 5]);

legend({'697 Hz', '770 Hz', '852 Hz', '941 Hz', '1209 Hz', '1336 Hz', '1477 Hz', '1633 Hz'}, 'Location', 'bestoutside');

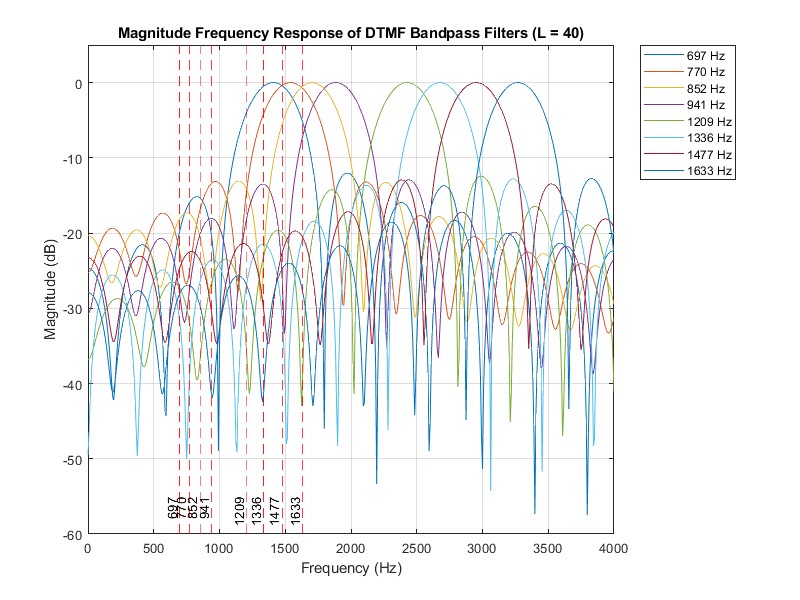
% Add text labels for DTMF frequencies

for i = 1:length(fb)

text(fb(i), -55, num2str(fb(i)), 'Rotation', 90, 'VerticalAlignment', 'bottom', 'HorizontalAlignment', 'right');

end

hold off;

****

**e.** % DTMF frequencies

fb = [697, 770, 852, 941, 1209, 1336, 1477, 1633];

L = 80; % Changed from 40 to 80

fs = 8000;

% Generate the filters

hh = dtmfdesign(fb, L, fs);

% Calculate frequency response

nfft = 1024;

f = linspace(0, fs/2, nfft/2 + 1);

H = zeros(nfft/2 + 1, 8);

for i = 1:8

[h, w] = freqz(hh(:,i), 1, nfft, fs);

H(:,i) = abs(h(1:nfft/2 + 1));

end

% Plot

figure('Position', [100, 100, 800, 600]);

plot(f, 20\*log10(H));

hold on;

% Add vertical lines for DTMF frequencies

for i = 1:length(fb)

line([fb(i), fb(i)], ylim, 'Color', 'r', 'LineStyle', '--');

end

% Customize plot

title('Magnitude Frequency Response of DTMF Bandpass Filters (L = 80)');

xlabel('Frequency (Hz)');

ylabel('Magnitude (dB)');

grid on;

xlim([0, fs/2]);

ylim([-60, 5]);

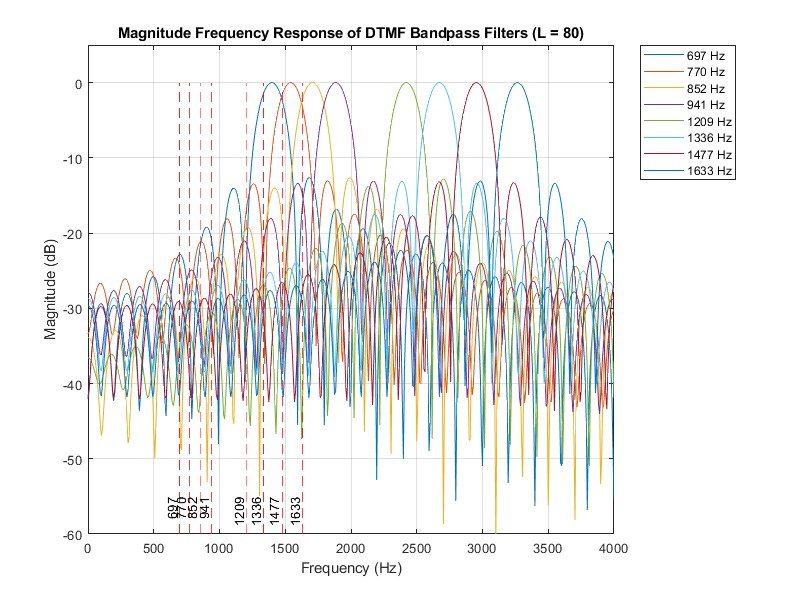
legend({'697 Hz', '770 Hz', '852 Hz', '941 Hz', '1209 Hz', '1336 Hz', '1477 Hz', '1633 Hz'}, 'Location', 'bestoutside');

for i = 1:length(fb)

text(fb(i), -55, num2str(fb(i)), 'Rotation', 90, 'VerticalAlignment', 'bottom', 'HorizontalAlignment', 'right');

end

hold off;



**f.** function analyze\_dtmf\_filters(L)

% DTMF frequencies

fb = [697, 770, 852, 941, 1209, 1336, 1477, 1633];

fs = 8000;

% Generate filters

hh = dtmfdesign(fb, L, fs);

% Analyze frequency response

nfft = 8192; % Increased for better resolution

f = linspace(0, fs/2, nfft/2 + 1);

H = zeros(nfft/2 + 1, 8);

figure('Position', [100, 100, 1200, 800]);

for i = 1:8

[h, w] = freqz(hh(:,i), 1, nfft, fs);

H(:,i) = abs(h(1:nfft/2 + 1));

% Plot frequency response

subplot(2, 4, i);

plot(f, 20\*log10(H(:,i)));

hold on;

% Add lines for passband and stopband

yline(-3, 'r--', 'Passband');

yline(-12, 'g--', 'Stopband');

% Mark DTMF frequencies

for j = 1:8

xline(fb(j), 'k:');

if i == j

text(fb(j), 0, '\downarrow', 'HorizontalAlignment', 'center', 'VerticalAlignment', 'bottom');

end

end

title(['Filter for ' num2str(fb(i)) ' Hz']);

xlabel('Frequency (Hz)');

ylabel('Magnitude (dB)');

grid on;

xlim([600 1700]); % Zoom to DTMF frequency range

ylim([-40 5]);

end

% Check specifications

passband\_ok = zeros(1, 8);

stopband\_ok = zeros(1, 8);

for i = 1:8

passband\_ok(i) = sum(H(:,i) > 0.707 & f' ~= fb(i)) == 0;

stopband\_ok(i) = sum(H(:,i) > 0.25 & abs(f' - fb(i)) > 20) == 0;

end

fprintf('Filter length L = %d\n', L);

fprintf('Passband criteria met: %d/8\n', sum(passband\_ok));

fprintf('Stopband criteria met: %d/8\n', sum(stopband\_ok));

if sum(passband\_ok) < 8 || sum(stopband\_ok) < 8

fprintf('Problematic filters:\n');

for i = 1:8

if ~passband\_ok(i) || ~stopband\_ok(i)

fprintf(' %d Hz\n', fb(i));

end

end

end

end

for L = [80, 100, 120, 140, 160]

analyze\_dtmf\_filters(L);

end

The frequency response shows how each filter passes one DTMF component while rejecting others. As L increases, the main lobe becomes narrower, and the side lobes become smaller, improving selectivity.

For smaller L values, some filters may have passbands wide enough to include adjacent DTMF frequencies. As L increases, the passbands narrow, eventually meeting our specification of including only one DTMF frequency.

The filters with center frequencies closest together are the most challenging to design. In the DTMF system, these are the lower frequency pairs, particularly 697 Hz and 770 Hz. These filters require a larger L to achieve sufficient separation.

**4.2**

**a.** function sc = dtmfscore(xx, hh)

%DTMFSCORE

% usage: sc = dtmfscore(xx, hh)

% returns a score based on the max amplitude of the filtered output

% xx = input DTMF tone

% hh = impulse response of ONE bandpass filter

% The signal detection is done by filtering xx with a length-L

% BPF, hh, and then finding the maximum amplitude of the output.

% The score is either 1 or 0.

% sc = 1 if max(|y[n]|) is greater than, or equal to, 0.59

% sc = 0 if max(|y[n]|) is less than 0.59

xx = xx\*(2/max(abs(xx))); %--Scale the input x[n] to the range [-2,+2]

% Filter the input signal

y = conv(xx, hh);

% Find the maximum amplitude of the filtered output

max\_amp = max(abs(y));

% Determine the score

if max\_amp >= 0.59

sc = 1;

else

sc = 0;

end

end

**b.** Check the code above (% Determine the score)

**c.**  Check the code above ( xx = xx\*(2/max(abs(xx))); )

**d.** Explaining the maximum value of the magnitude response H(e^jω):

The maximum value of |H(e^jω)| must be equal to one for each filter due to the following reasons:

Normalization: Having |H(e^jω)| = 1 at the center frequency ensures that the filter's gain is consistent across all DTMF frequencies. This makes the scoring threshold (0.59) applicable to all filters.

Sinusoid gain: In a DTMF tone, both sinusoids experience a known gain (or attenuation) through the bandpass filter. If |H(e^jω)| = 1 at the center frequency, then:

The matching sinusoid passes through at full strength (gain ≈ 1)

The non-matching sinusoid is attenuated (gain < 1)

Predictable output: With |H(e^jω)| = 1, the amplitude of the output can be predicted if we know both the frequency response and the input amplitude. For a perfectly matching sinusoid:

output\_amplitude ≈ input\_amplitude \* |H(e^jω)| ≈ input\_amplitude \* 1

Threshold validity: The 0.59 threshold is based on this scaling. It assumes that a strong match will produce an output close to the input amplitude, while a weak match will be significantly attenuated.

By maintaining |H(e^jω)| = 1 for each filter, we ensure that the scoring rule is consistently applicable across all DTMF frequencies, making the detection process reliable and the threshold meaningful.

**e.** function sc = dtmfscore(xx, hh)

xx = xx\*(2/max(abs(xx))); %--Scale the input x[n] to the range [-2,+2]

% Filter the input signal

y = conv(xx, hh);

% Find the maximum amplitude of the filtered output

max\_amp = max(abs(y));

% Determine the score

if max\_amp >= 0.59

sc = 1;

else

sc = 0;

end

if nargout == 0 % Only plot if no output argument is requested

figure;

plot(1:min(500, length(y)), y(1:min(500, length(y))));

title('Filtered Output');

xlabel('Sample');

ylabel('Amplitude');

grid on;

end

end

**4.3.** function keys = dtmfrun(xx, L, fs)

%DTMFRUN keys = dtmfrun(xx,L,fs)

% returns the list of key names found in xx.

% keys = array of characters, i.e., the decoded key names

% xx = DTMF waveform

% L = filter length

% fs = sampling freq

center\_freqs = [697 770 852 941 1209 1336 1477 1633]; %<==FILL IN THE CODE HERE

hh = dtmfdesign(center\_freqs, L, fs);

% hh = L by 8 MATRIX of all the filters. Each column contains the

% impulse response of one BPF (bandpass filter)

[nstart, nstop] = dtmfcut(xx, fs); %<--Find the beginning and end of tone bursts

keys = [];

for kk=1:length(nstart)

x\_seg = xx(nstart(kk):nstop(kk)); %<--Extract one DTMF tone

%<=======================FILL IN THE CODE HERE

scores = zeros(1, 8);

for i = 1:8

scores(i) = dtmfscore(x\_seg, hh(:,i));

end

row\_freqs = scores(1:4);

col\_freqs = scores(5:8);

if sum(row\_freqs) == 1 && sum(col\_freqs) == 1

row\_index = find(row\_freqs);

col\_index = find(col\_freqs) - 4;

keypad = ['123'; '456'; '789'; '\*0#'];

keys = [keys keypad(row\_index, col\_index)];

else

keys = [keys '?']; % Error indicator

end

end

end

**4.4.**

% Set up parameters

fs = 8000;

L = 80; % Adjust this value if needed

% Test with the example from Fig. 8

tk = ['A','B','C','D','\*','#','0','1','2','3','4','5','6','7','8','9'];

xx = dtmfdial(tk, fs);

% soundsc(xx, fs) % Uncomment to play the sound

keys = dtmfrun(xx, L, fs);

disp('Example test result:');

disp(keys);

% Test with the given phone number

test\_number = '407\*89132#BADC';

xx\_test = dtmfdial(test\_number, fs);

keys\_test = dtmfrun(xx\_test, L, fs);

disp('Phone number test result:');

disp(keys\_test);

% Create spectrogram

figure;

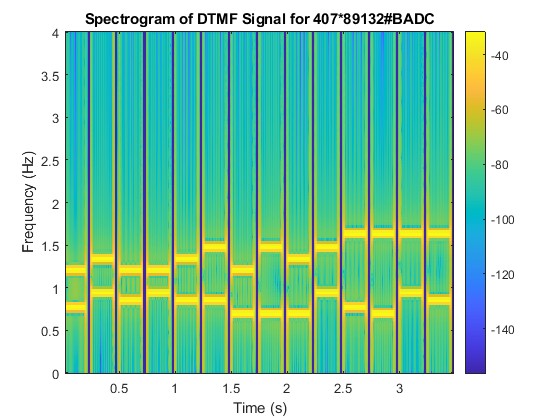
spectrogram(xx\_test, 256, 250, [], fs, 'yaxis');

title('Spectrogram of DTMF Signal for 407\*89132#BADC');

xlabel('Time (s)');

ylabel('Frequency (Hz)');

colorbar;



**4.5.** xx is not provided.