University of Massachusetts Dartmouth
Department of Electrical and Computer Engineering

ECE 320 DISCRETE-TIME LINEAR SYSTEMS

Fall 2013

MATLAB **Project 2 Solutions**

Project 5.1: Computing Samples of the DTFT

This project describes how to use Matlab to compute samples of the discrete-time Fourier transform $X(e^{j\omega})$ for a finite-length signal $x[n]$, and then applies this technique to a rectangular pulse and triangle wave form.

(a) The signal is

$$x[n] = u[n] - u[n-11] = \begin{cases} 1, & 0 \le n \le 10 \\ 0, & \text{otherwise.} \end{cases}$$

Substituting this into the DTFT analysis equation, we get

$$\begin{aligned} X(e^{j\omega}) &= \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n} \\ &= \sum_{n=0}^{10} e^{-j\omega n} \\ &= \frac{1-e^{j11\omega}}{1-e^{-j\omega}} \\ &= \frac{e^{-j(11/2)\omega}}{e^{-j\omega/2}} \frac{e^{j(11/2)\omega} - e^{-j(11/2)\omega}}{e^{j\omega/2} - e^{-j\omega/2}} \\ &= e^{-j5\omega} \frac{2j\sin(11\omega/2)}{2j\sin(\omega/2)} \\ &= e^{-j5\omega} \frac{\sin(11\omega/2)}{\sin(\omega/2)} \end{aligned}$$

The commands to define the signal $x[n]$ are

```
x = ones(1,11);
nx = 0:10; % time indices associated with x
```

(b) These commands define the frequencies used, and then define a vector using them from the analytic expression above. The Matlab function `dtftsinc` from the book is used for the magnitude portion.

```
N=100;
k = [0:N-1];
w = 2*pi*k/N;
% find X(ejw) using function based on analytic expression
X = exp(-j*5*w).*dtftsinc(11,w);
```

The commands below make the requested plots, as shown in Figure 1.

```
figure(1)
clf
% plot magnitude of DTFT
```

```
subplot(211)
plot(w/pi,abs(X))
xlabel('Frequency (radians)')
ylabel('Magnitude')
title('Part (b): |X(e^{j\omega})| from Analytic expression')
% plot phase of DTFT
subplot(212)
plot(w/pi,angle(X))
xlabel('Frequency (radians)')
ylabel('Phase (radians)')
title('Project 5.1(b): \angle X(e^{j\omega}) from Analytic expression')
```

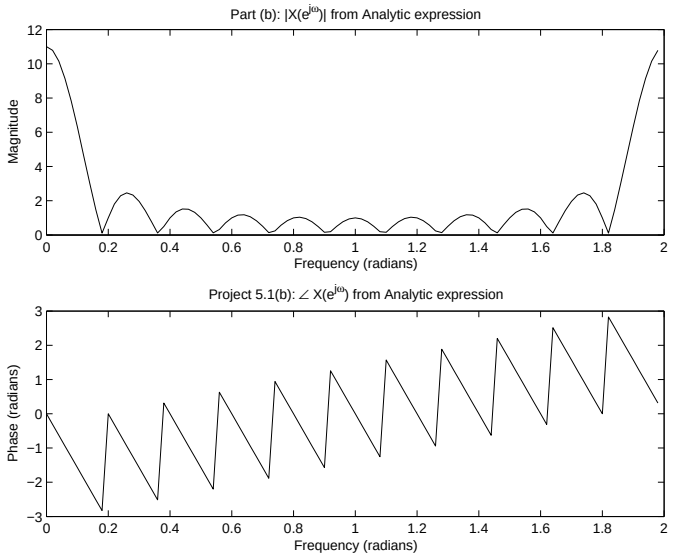

Figure 1: Magnitude and Phase of $X(e^{j\omega})$ for Project 5.1, Part (b)

(c) The following commands compute the samples of $X(e^{j\omega})$ from the time signal $x[n]$, then shift the samples to run from $-\pi$ from π.

```
% use fft to find samples of X(ejw)
X = fft(x,100);
% shift samples for frequency range -pi < w < pi
Xshift = fftshift(X);
% shift frequency samples to match Xshift
wshift = w-pi;
```

The commands below plot the magnitude and phase of the values obtained using `fft`. Figure 2 shows the plots that result. The plots look like those from part (b) if they had been shifted by π based on the periodic repetition, as expected. The new plots are centered on $\omega = 0$.

```
figure(2)
clf
% plot magnitude computed from FFT
subplot(211)
plot(wshift/pi,abs(Xshift));
xlabel('Frequency (radians)')
ylabel('Magnitude')
title('Project 5.1(c): |X(e^{j\omega})| from FFT')
```

```
% plot phase
subplot(212)
plot(wshift/pi,angle(Xshift));
xlabel('Frequency (radians)')
ylabel('Phase (radians)')
title('Project 5.1(c): \angle X(e^{j\omega}) from FFT');
```

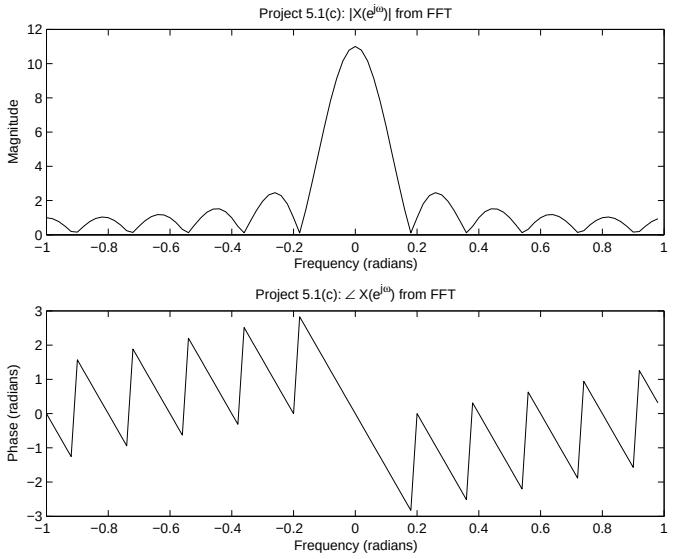

Figure 2: Magnitude and Phase of $X(e^{j\omega})$ obtained using `fft` for Project 5.1, Part (c)

(d) Let $r[n] = x[n+5]$. From the DTFT properties

$$R(e^{j\omega}) = e^{j5\omega}X(e^{j\omega}).$$

Using $X(e^{j\omega})$ from above, we find the analytic expression for the Fourier transform of the shifted sequence is

$$R(e^{j\omega}) = \frac{\sin(11\omega/2)}{\sin(\omega/2)}.$$

From the DTFT properties, we want to use $a = 5$, so

```
a = 5;
Xr = exp(j*w*a).*X;
```

We verify that the imaginary part is negligible using `max(imag(Xr))`, which returns `ans = 1.0880e-14`, or just roundoff errors. This confirms we implemented the shift properly. Once that is done, we discard the imaginary part and make the plots. The figures agree well, confirming we implemented the shift correctly.

```
Xr = real(Xr);
figure(3)
clf
subplot(211)
% plot resulting Xr
plot(w/pi,Xr)
xlabel('Frequency (radians)')
ylabel('X(e^{j\omega})')
title('Project 5.1(d): DTFT for x[n+5] from FFT')
% plot result from dtftsinc
```

```
% since Xr should be the same as dtftsinc(11,w)
subplot(212)
plot(w/pi,dtftsinc(11,w))
xlabel('Frequency (radians)')
ylabel('X(e^{j\omega})')
title('Part (d): DTFT for x[n+5] from dtftsinc')
```

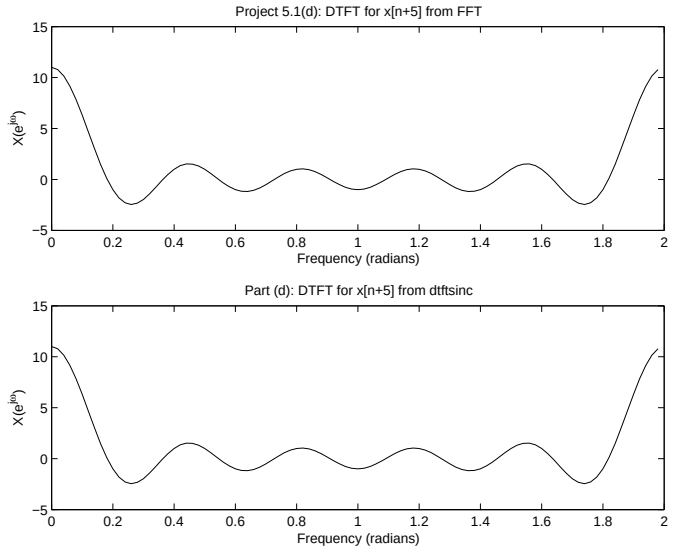

Figure 3: Comparison of analytic and computed samples for the shifted version of $X(e^{j\omega})$ for Project 5.1, Part (d)

(e) This part applies the same technique to a triangular signal. The signal $z[n]$ begins at $n = -5$ and is symmetric about $n = 0$. Since Matlab's `fft` assumes the signal starts at $n = 0$, we need to modify the values of $Z(e^{j\omega})$ from `fft` as if the time signal had been shifted left 5 samples, the same as we did in part (d).

```
% Define the ramp sequence given for z[n]
z = [0 1 2 3 4 5 4 3 2 1 0];
nz = [-5:5];
Z = fft(z,100);
Zr = exp(j*w*5).*Z;
% Verify that the imaginary part of Z is very small before removing it:
max(imag(Zr))
% ans =
%   4.5297e-14
% Again, because z[n] is symmetric about n=0, Zr should be purely real:
Zr = real(Zr) ;
figure(4)
clf
plot(w/pi,Zr)
xlabel('Frequency normalized by \pi (\omega/\pi)')
ylabel('Z(e^{j\omega})')
title('Project 5.1(e): DTFT for z[n]')
```

Note that because $z[n]$ is the convolution of two rectangular pulses the DTFT is like $(\sin(5\omega/2)/\sin(\omega/2))^2$. Based on this, we expect the numerator, and thus $Z(e^{j\omega})$ to be zero from $\omega\pi = 0.4, 0.8, 1.2, and 1.6$, which is what the plots show.

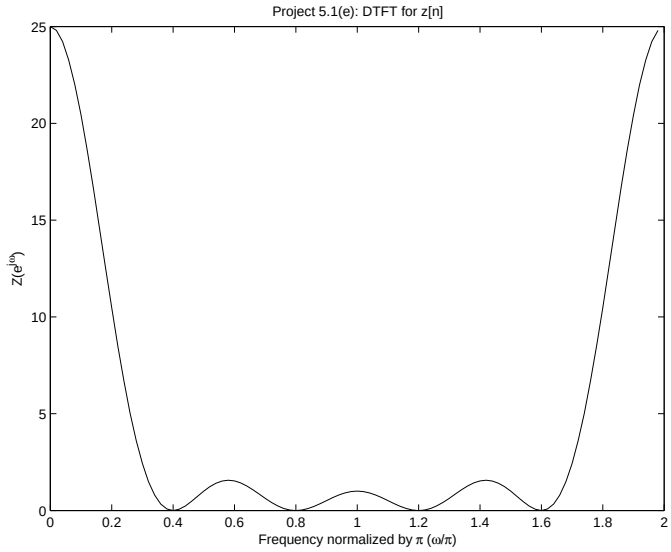

Figure 4: Discrete-time Fourier transform $Z(e^{j\omega})$ of triangular sequence $z[n]$ for Project 5.1, Part (e)

Project 6.3: Filter Design by Transformation

This project investigates techniques for designing FIR filters based on a prototype LPF.

(a) From Table 5.2, the ideal impulse response $h[n]$ for a LPF with cutoff $\pi/5$ is

$$h[n] = \frac{\sin(\pi n/5)}{\pi n}.$$

Figure 5 shows the prototype $h[n]$ given in the top panel and the sinc for the ideal $h_{id}[n]$. The filters do have generally similar forms, but the lower amplitudes for $h[n]$ at the edges look like it has had a window applied to it. Figure 6 plots the frequency response magnitude for this filter. This is in fact a LPF with cutoff $\pi/5$. The commands to define these signals and make the plots are

```
[H,w] = freqz(h,1,1024,'whole');
nh = 0:96;  %time indices for impulse response
hid = (1/5)*sinc((1/5)*(nh-48));

figure(1)
clf
subplot(211)
stem(nh,h)
xlabel('Time (samples)')
ylabel('h[n]')
title('Project 6.3, part (a): Impulse Response')
subplot(212)
stem(nh,hid)
xlabel('Time (samples)')
ylabel('Ideal h[n]')
title('Project 6.3, part (a): Sinc for LPF')

figure(2)
clf
```

```
subplot(311)
plot(w/pi,abs(H))
xlabel('Frequency (\omega/\pi)')
ylabel('|H(e^{j\omega})|');
title('Project 6.3, part (a):  Frequency Response Magnitude')
```

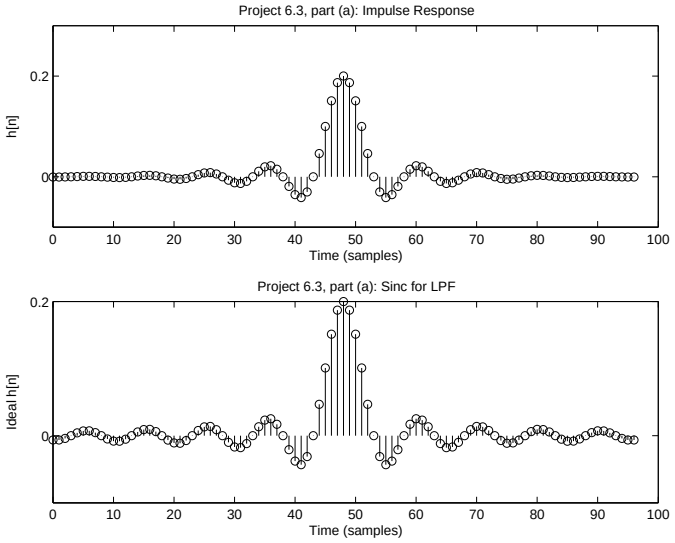

Figure 5: Comparison of $h[n]$ given with sinc for the same cutoff frequency.

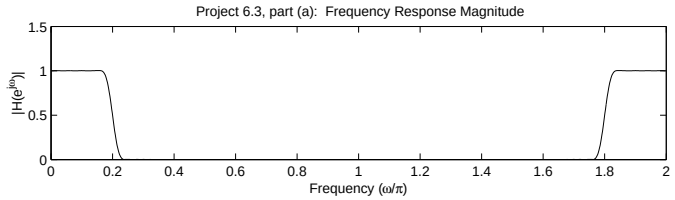

Figure 6: Frequency response magnitude for FIR filter given.

(b) If we shift $H(e^{j\omega})$ by π in ω, it will give the desired $H_{hpf}(e^{j\omega})$. Shifting in frequency corresponds to multiplying by a complex exponential. In this problem, the shift is $\omega_0 = \pi$, so $e^{j\omega_0 n} = e^{j\pi n} = (-1)^n$. Ideally, we want to apply this as though $h[n]$ were still centered at $n = 0$. In fact, since $L = M/2 = 48$ is even here, it doesn't matter in the end, since $(-1)^0 = (-1)^{48} = 1$. The Matlab commands to implement this are

```
L = 48;
h1 = ((-1).^(nh-L)).*h;
H1 = freqz(h1,1,1024,'whole');
wshift = w-pi;

figure(3)
clf
subplot(211)
stem(nh,h1)
xlabel('Time (samples)')
ylabel('h_1[n]')
```

```
title('Project 6.3 part (b): Impulse Response')
subplot(212)
plot(wshift/pi,fftshift(abs(H1)))
xlabel('Frequency (\omega/\pi)')
ylabel('|H_1(e^{j\omega})|');
title('Project 6.3 part(b): Frequency Response Magnitude')
```

Figure 7 plots the impulse response $h_1[n]$ and frequency response magnitude $|H_1(e^{j\omega})|$. As expected, every other sample of the impulse response has its sign changed. The frequency response magnitude is the highpass filter desired.

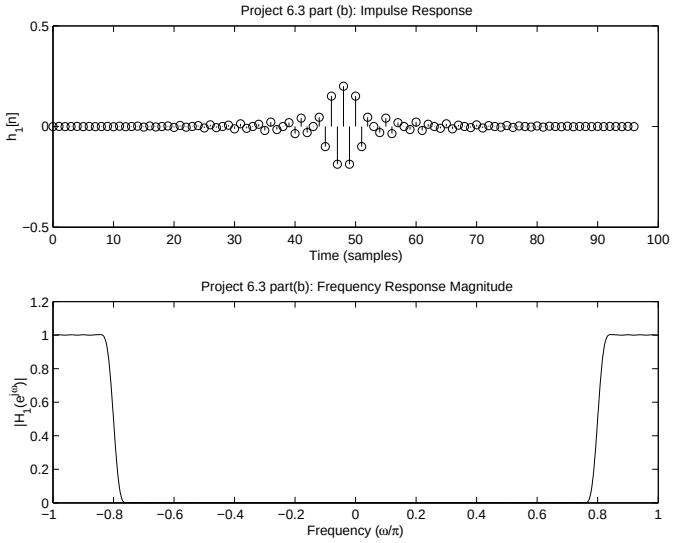

Figure 7: Impulse response and frequency response magnitude for FIR highpass filter.

(c) For this part, we want construct $H_2(e^{j\omega})$ such that it is the sum of the original $H(e^{j\omega})$ shifted by $+\pi/2$ and $-\pi/2$. We can do this by adding the result of multiplying $h[n]$ by $e^{j\pi n/2}$ to the result of multiplying $h[n]$ to $e^{-j\pi n/2}$. This is equivalent to $h[n]2\cos(\pi n/2)$.

The Matlab commands to implement this transformation are

```
h2 = 2*cos(pi*nh/2).*h;
H2 = freqz(h2,1,1024,'whole');
```

Figure 8 plots the impulse response and frequency response magnitude that results. Multiplying by a cosine with this frequency should set every other sample to zero, and every fourth sample has a sign change. The plots in Figure 8 are generated by

```
figure(3)
clf
subplot(211)
stem(nh,h2)
xlabel('Time (samples)')
ylabel('h_2[n]')
title('Project 6.3 part (c): Impulse Response')
subplot(212)
plot(wshift/pi,fftshift(abs(H2)));
xlabel('Frequency (\omega/\pi)')
ylabel('|H_2(e^{j\omega})|');
title('Project 6.3 part (d): Frequency Response Magnitude')
```

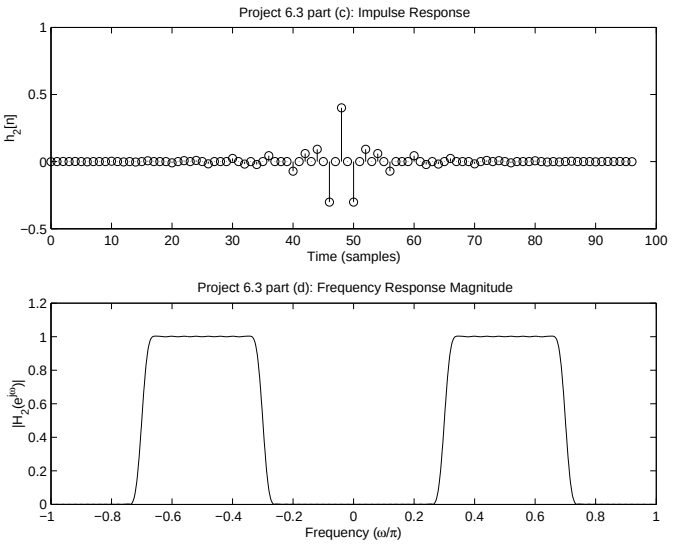

Figure 8: Impulse response and frequency response magnitude for FIR bandpass filter.

(d) The desired frequency response in this part is

$$H_3(e^{j\omega}) = 1 - H_2(e^{j\omega}).$$

Taking the inverse Fourier transform gives

$$h_3[n] = \delta[n] - h_2[n].$$

This subtraction should happen so that $h_3[n]$ is still symmetric. This means that the impulse should have 48 zeros, a one, then 48 more zeros. This transformation is implemented by

```
h3 = [zeros(1,48) 1 zeros(1,48)]-h2;
H3 = freqz(h3,1,1024,'whole');
```

The frequency response and impulse response are shown in Figure 9. The frequency response magnitude is the desired bandstop filter. These plots were generated by

```
figure(4)
clf
subplot(211)
stem(nh,h3)
xlabel('Time (samples)')
ylabel('h_3[n]')
title('Project 6.3 part (d): Impulse Response')
subplot(212)
plot(wshift/pi,fftshift(abs(H3)))
xlabel('Frequency (\omega/\pi)')
ylabel('|H_3(e^{j\omega})|');
title('Project 6.3 part (d): Frequency Response Magnitude')
```

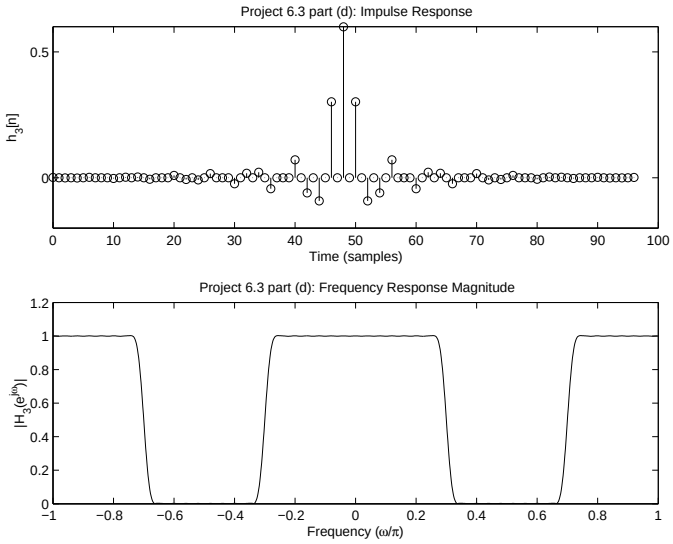

Figure 9: Impulse response and frequency response magnitude for FIR bandstop filter.

Speech Filtering Project

This project illustrates how filters can remove noise and annoying tones from speech signals, making it easier to understand them.

(a) We begin by reading in the signal, and then plotting its spectrum, both on a linear and dB scale. From listening to the signal, we can tell that the speech is the lowest frequencies, and the noise is high frequency. From listening, the noise is louder than the speech, which matches what we see in the plots. It appears that the noise is from roughly $\omega = 0.4\pi$ and up.

A tone is basically a sine or cosine, so should have two peaks at $\pm\omega$. Also, from listening, we know that the tone is louder than the speech or noise. The peaks around $\pm\omega = 0.2\pi$ are the tone. Figures 10 and 11 show the spectrum for the noisy speech signal, made with the following commands

```
% read in noisy speech data
load group0.mat

% Set Nfft to be the next largest power of 2 from length of y
Nfft = 2^ceil(log2(length(y)));
Y = fft(y,Nfft);
omega = (0:(Nfft-1))*(2*pi/Nfft);
omegaplot = omega-pi;

figure(1)
clf
plot(omegaplot/pi,fftshift(abs(Y)));
xlabel('Frequency (\omega/\pi)')
ylabel('Y(e^{j\omega})')
title('Project 3, part a: Noisy signal spectrum')

% same plot on dB scale
figure(2)
clf
plot(omegaplot/pi,fftshift(20*log10(abs(Y))));
xlabel('Frequency (\omega/\pi)')
ylabel('Y(e^{j\omega}) (dB)')
title('Project 3, part a: Noisy signal spectrum')
text(0.2,75,'Tone')
text(0,35,'Speech')
text(0.6,40,'Noise')
```

(b) We can find the frequency response for the notch filter by substituting into the discrete-time Fourier transform analysis equation

$$\begin{aligned} H_{notch}(e^{j\omega}) &= \sum_{n=-\infty}^{\infty} h_{notch}[n]e^{-j\omega n} \\ &= 1 - 2\cos(\omega_n)e^{-j\omega} + e^{-j2\omega} \\ &= \left(1 - e^{j\omega_n}e^{-j\omega}\right)\left(1 - e^{-j\omega_n}e^{-j\omega}\right) \end{aligned}$$

From the factored version, we can see that $H_{notch}(e^{j\omega})$ will be zero for $\omega = \pm\omega_n$.

(c) To remove the annoying tone, we want to put the notch at the frequency of the tone. For our signal, this is $\omega_n = 0.2\pi$. We can then define this 3-point FIR filter. Here, it's normalized so that it adds to 1 so the DC gain is one. That isn't strictly necessary, but is convenient.

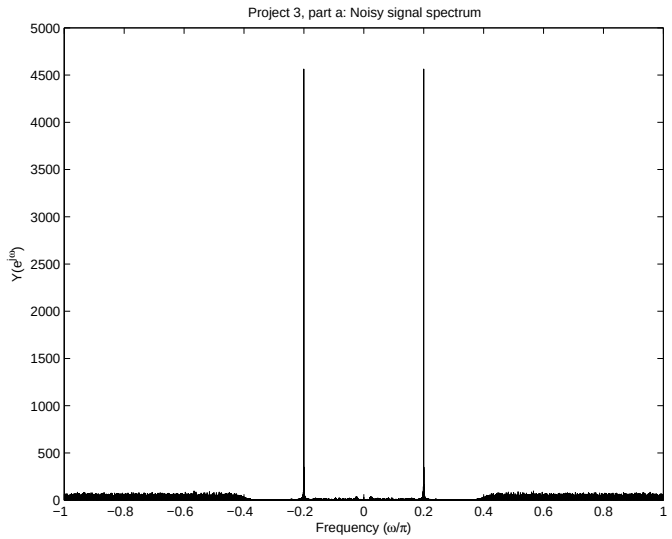

Figure 10: Noisy speech on a linear amplitude scale. Compared to the tone and noise, the actual speech is essentially invisible.

```
omegan = 0.2*pi;
hnotch = (1/(2-2*cos(omegan)))*[1 -2*cos(omegan) 1];
Hnotch = fft(hnotch,Nfft);
```

Figure 12 shows the frequency response from above. As we expect from our analytic result, the notch filter is 0 ($-\infty$ dB) at $\omega_n = \pm 0.2\pi$. It is not ideal that the levels of the high frequency components are amplified by this filter. In this particular project, this is not a major concern since the high frequencies will be removed in the second half of the project. This plot was created with

```
figure(3)
clf
plot(omegaplot/pi,fftshift(20*log10(abs(Hnotch))));
xlabel('Frequency (\omega/\pi)')
ylabel('Y(e^{j\omega}) (dB)')
title('Project 3, part c: Notch filter frequency response.')
```

(d) Now, we apply the notch filter to our noisy speech signal to get a new signal $r[n]$. When we save it and listen, the tone is gone, which is a great improvement. The spectrum in Figure 13 shows the peaks at $\pm 0.2\pi$ are now gone.

```
r = filter(hnotch,1,y);
wavwrite(r/max(abs(r)),10000,16,'/tmp/proj2r.wav')
R = fft(r,Nfft);

figure(4)
clf
plot(omegaplot/pi,fftshift(20*log10(abs(R))))
xlabel('Frequency (\omega/\pi)')
ylabel('R(e^{j\omega}) (dB)')
title('Project 3, part d: Spectrum with tone removed')
```

(e) From our plots in part (a), it appears that an ideal LPF with a cutoff of 0.4π should remove most of the high frequency noise, but leave the speech. I designed this filter first with `fir1`, and processed the speech. This removed a lot of the high frequency noise and made the

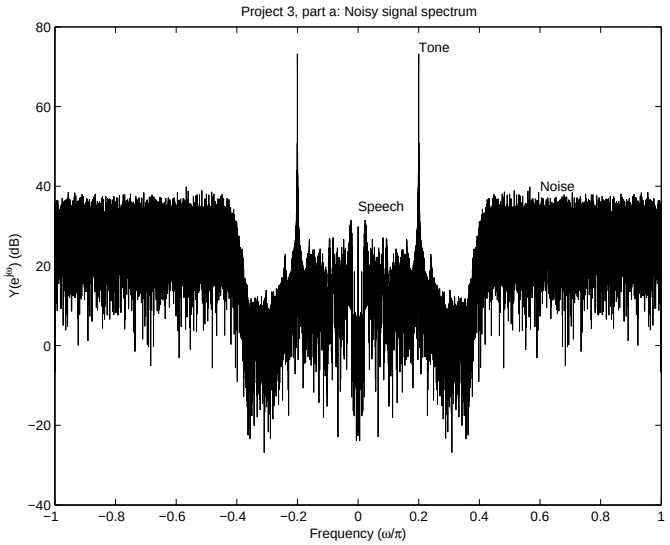

Figure 11: Noisy speech on a dB scale. This makes the different regions very clear. The speech is at frequencies under 0.4π, the noise is above 0.4π and the tones are at 0.2π.

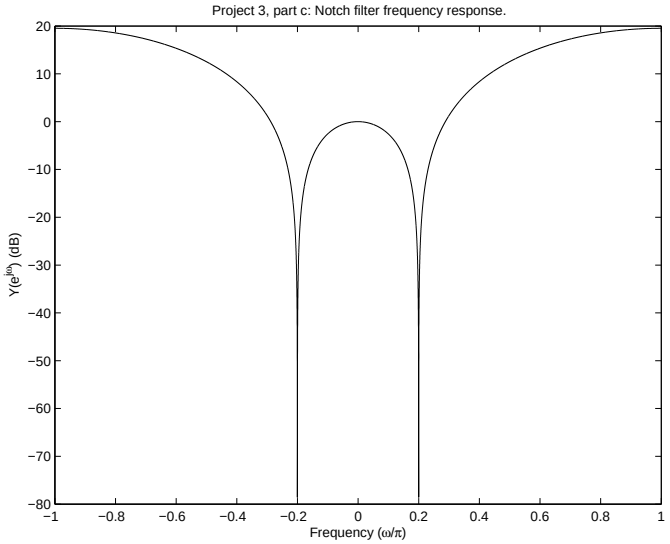

Figure 12: Frequency response for the notch filter.

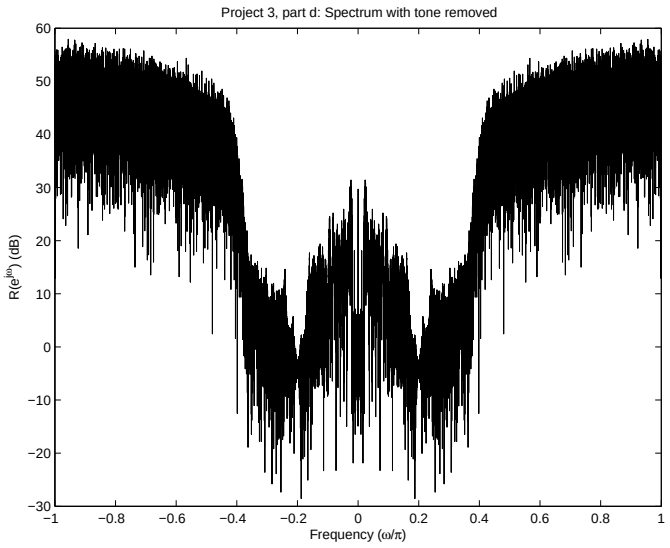

Figure 13: Spectrum of signal $r[n]$ after applying notch filter to remove tones

speech easier to understand. I experimented a little bit with lower cutoff frequencies, but they didn't really seem to help the speech intelligibility much, so I stuck with $\omega_c = 0.4\pi$. Figure 14 plots the frequency response that results. The Matlab commands to find $h[n]$ and the frequency frequency response are given below:

```
Nfilter = 100;
hlpf = fir1(Nfilter,0.4);
Hlpf = fft(hlpf,Nfft);

figure(5)
clf
plot(omegaplot/pi,fftshift(20*log10(abs(Hlpf))));
xlabel('Frequency (\omega/\pi)')
ylabel('Hlpf(e^{j\omega}) (dB)')
title('Project 3, part c: Lowpass filter frequency response')
```

(f) For this part, we apply the filter from part (e) to the speech signal $r[n]$ from part (d) which had the tone removed. Figure 15 shows that the resulting spectrum $S(e^{j\omega})$ has the high frequency noise above 0.4π greatly reduced. Listening to the wav file, it is much easier to understand now. The high frequency noise is not completely removed, but it is much lower, almost 50–60 dB lower than the spectrum of $R(e^{j\omega})$ from part (d). We can now hear clearly that the speaker is saying "He has the bluest eyes" for my test data. (You may have a different test sentence.)

```
s = filter(hlpf,1,r);
wavwrite(s/max(abs(s)),10000,16,'/tmp/proj2s.wav')
S = fft(s,Nfft);

figure(6)
clf
plot(omegaplot/pi,fftshift(20*log10(abs(S))))
xlabel('Frequency (\omega/\pi)')
ylabel('S(e^{j\omega}) (dB)')
title('Project 3, part f: Spectrum with tone and noise removed')
```

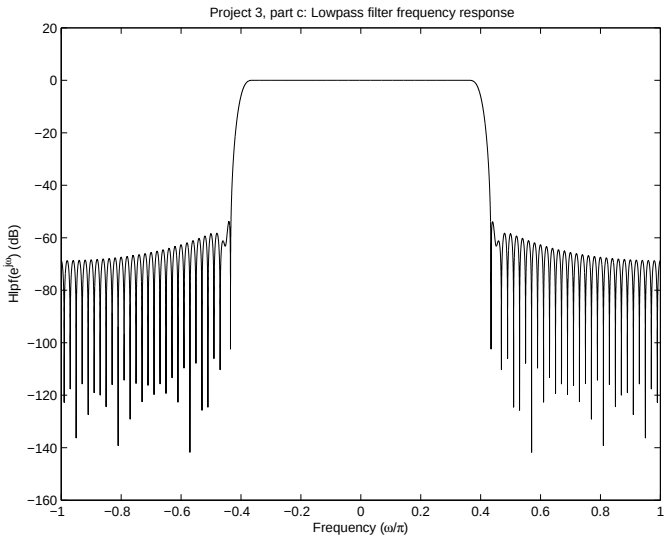

Figure 14: Frequency response for the lowpass filter designed in part (e).

(g) Our processing in parts (d) and (f) is implementing a series or cascade combination of two FIR LTI filters. First, we used $y[n]$ as the input for $h_{notch}[n]$ to get the output $r[n]$ without the tone, then $r[n]$ was the input to our LPF with $h_{lpf}[n]$ to get $s[n]$. We know from Chapter 2 that if we have the series combination of two LTI systems, the overall impulse response is the convolution of the individual impulse responses. If we convolve $h_{notch}[n] * h_{lpf}[n]$ we will get a new impulse response $h_{combo}[n]$ which will remove the tone and the high frequency noise all at once. The following commands implement this convolution, compute and plot the frequency response magnitude $|H_{combo}(e^{j\omega})|$, and then plot the signal spectrum for the output of this new filter when $y[n]$ is the input. As we can see in Figure 16, the new filter's frequency response has a notch at 0.2π and strongly attenuates all the frequencies above 0.4π, as we desire. Figure 17 shows the output spectrum on a dB scale, which as we hope sounds just like the output of the cascade. Listening to the resulting `.wav` file shows they sound the same as well.

```
hcombo = conv(hnotch,hlpf);
Hcombo = fft(hcombo,Nfft);
s2 = filter(hcombo,1,y);
wavwrite(s2/max(abs(s2)),10000,16,'/tmp/proj2s2.wav')
S2 = fft(s2,Nfft);

figure(7)
clf
plot(omegaplot/pi,fftshift(20*log10(abs(Hcombo))));
xlabel('Frequency (\omega/\pi)')
ylabel('Hcombo(e^{j\omega}) (dB)')
title('Project 3, part g: Combined filter frequency response')

figure(8)
clf
plot(omegaplot/pi,fftshift(20*log10(abs(S2))))
xlabel('Frequency (\omega/\pi)')
ylabel('S2(e^{j\omega}) (dB)')
title('Project 3, part h: Output of combo filter')
```

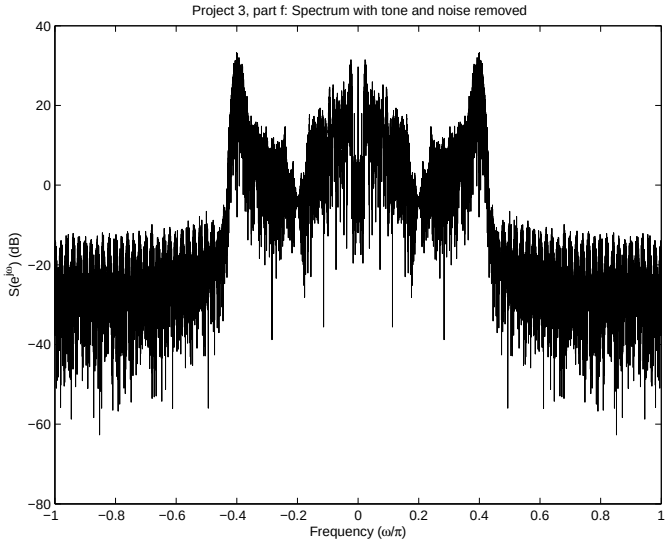

Figure 15: Spectrum $S(e^{j\omega})$ after filtering to remove the tone and noise.

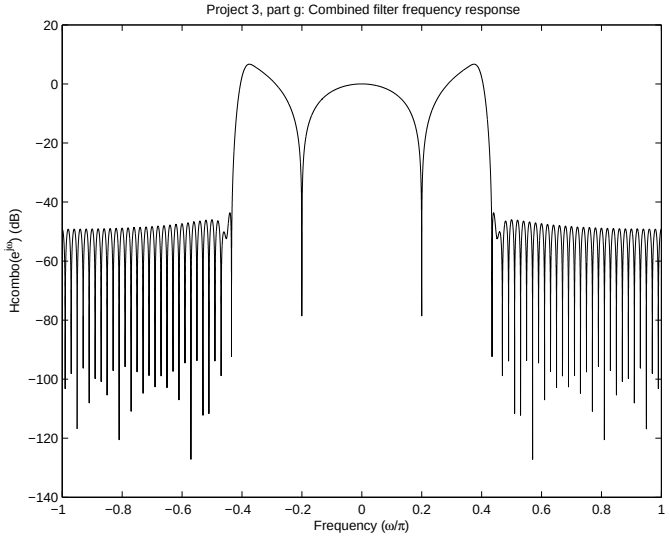

Figure 16: Frequency response $H_{combo}(e^{j\omega})$ for combined notch and lowpass filter.

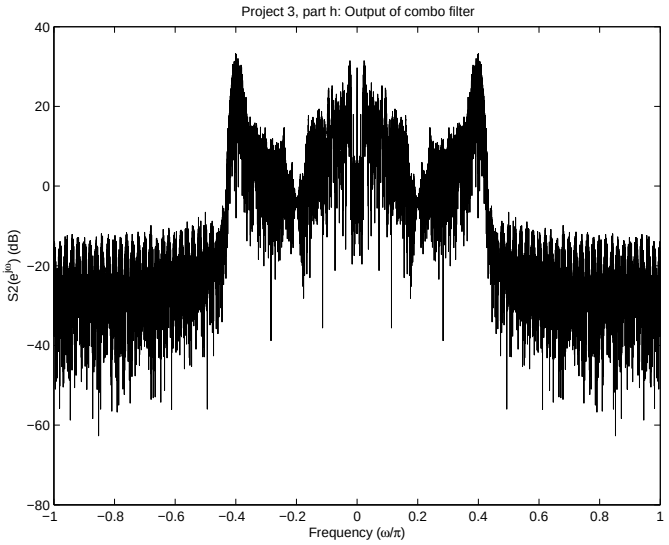

Figure 17: Spectrum $S_2(e^{j\omega})$ after filtering $y[n]$ with the combination filter.

```
% Project 5.1
% ECE 320 Fall 2013
% John Buck

finalplots = 1;

% Part (a)
% See attached sheet for analytic result.

% define signal
x = ones(1,11);
nx = 0:10; % time indices associated with x

% Part (b)

N=100;
k = [0:N-1];
w = 2*pi*k/N;

% find X(ejw) using function based on analytic expression
X = exp(-j*5*w).*dtftsinc(11,w);

figure(1)
clf
% plot magnitude of DTFT
subplot(211)
plot(w/pi,abs(X))
xlabel('Frequency (radians)')
ylabel('Magnitude')
title('Part (b): |X(e^{j\omega})| from Analytic expression')
% plot phase of DTFT
subplot(212)
plot(w/pi,angle(X))
xlabel('Frequency (radians)')
ylabel('Phase (radians)')
title('Project 5.1(b): \angle X(e^{j\omega}) from Analytic expression')

if finalplots
  print -deps proj51b.eps
end

% Part (c)

% use fft to find samples of X(ejw)
X = fft(x,100);
% shift samples for frequency range -pi < w < pi
Xshift = fftshift(X);

% shift frequency samples to match Xshift
wshift = w-pi;

figure(2)
clf
% plot magnitude computed from FFT
subplot(211)
plot(wshift/pi,abs(Xshift));
xlabel('Frequency (radians)')
ylabel('Magnitude')
title('Project 5.1(c): |X(e^{j\omega})| from FFT')
% plot phase
subplot(212)
plot(wshift/pi,angle(Xshift));
xlabel('Frequency (radians)')
ylabel('Phase (radians)')
title('Project 5.1(c): \angle X(e^{j\omega}) from FFT');

if finalplots
  print -deps proj51c.eps
end

% This plot looks just like the plot from part (b) except
% the frequency axis has been shifted by pi, which is what
% we expected.  The plot is now centered about omega = 0.

% Part (d)

a = 5;
Xr = exp(j*w*a).*X;

% check to verify imaginary part of Xr is small
max(imag(Xr))
% this returns
%
% ans =
%    1.0880e-14
% so we are fine.  The imaginary part is very very small.

% discard imaginary part
Xr = real(Xr);

figure(3)
clf
subplot(211)
% plot resulting Xr
plot(w/pi,Xr)
xlabel('Frequency (radians)')
ylabel('X(e^{j\omega})')
title('Project 5.1(d): DTFT for x[n+5] from FFT')
% plot result from dtftsinc
% since Xr should be the same as dtftsinc(11,w)
subplot(212)
plot(w/pi,dtftsinc(11,w))
xlabel('Frequency (radians)')
ylabel('X(e^{j\omega})')
title('Part (d): DTFT for x[n+5] from dtftsinc')

if finalplots
  print -deps proj51d.eps
end

% As one more check, find the maximum error between Xr and dtftsinc:
max(abs(Xr-dtftsinc(11,w)))

% this returns:
% ans =
%   5.3291e-14
% which is just roundoff error, so we get the same answer analytically
% and computationally.

% Part (e)

% Define the ramp sequence given for z[n]

z = [0 1 2 3 4 5 4 3 2 1 0];

% Because fft assumes the first point is the n=0 sample, this sequence
% is really z[n-5], so we will need to add a phase term to correct this
% after computing the fft.

Z = fft(z,100);
```

```
Zr = exp(j*w*5).*Z;

% Verify that the imaginary part of Z is very small before removing it:
max(imag(Zr))
% ans =
%   4.5297e-14

% Again, because z[n] is symmetric about n=0, Zr should be purely real:
Zr = real(Zr) ;

% Here, I plot the frequency normalized by pi, which is often helpful to
% see whether zero crossings and such fall at the correct fractions of pi.

figure(4)
clf
plot(w/pi,Zr)
xlabel('Frequency normalized by \pi (\omega/\pi)')
ylabel('Z(e^{j\omega})')
title('Project 5.1(e): DTFT for z[n]')

if finalplots
  print -deps proj51e.eps
end

% Note that because z[n] is the convolution of two rectangular pulses
% the DTFT is like [sin(5w/2) / sin(w/2)].^2.

% The zeros fall at omega/pi = 0.4, 0.8, 1.2, and 1.6 as we would predict.
```

```
% CESS Project 6.3 Parts (a)-(d)
% ECE 320 Fall 2013
% John Buck

finalplots = 0;

% load prototype filter
load protoh

% part (a)
[H,w] = freqz(h,1,1024,'whole');
nh = 0:96;  %time indices for impulse response
hid = (1/5)*sinc((1/5)*(nh-48));

figure(1)
clf
subplot(211)
stem(nh,h)
xlabel('Time (samples)')
ylabel('h[n]')
title('Project 6.3, part (a): Impulse Response')
subplot(212)
stem(nh,hid)
xlabel('Time (samples)')
ylabel('Ideal h[n]')
title('Project 6.3, part (a): Sinc for LPF')

if finalplots
  print -deps proj63ahn.eps
end

figure(2)
clf
subplot(311)
plot(w/pi,abs(H))
xlabel('Frequency (\omega/\pi)')
ylabel('|H(e^{j\omega})|');
title('Project 6.3, part (a):  Frequency Response Magnitude')

% This frequency response is a good approximation to the ideal
% specification.  It is a lowpass filter with a cutoff of pi/5.

if finalplots
  print -deps proj63aHejw.eps
end

% The attached notes show analytically that the ideal impulse
% response h_{id}[n] = sin((pi/5)*n)/(pi*n).  The h[n] plotted
% above is a truncation of this response, shifted by 48 samples, so
% that it is causal.

% part (b)
% If we shift H(ejw) by pi in frequency, we will get the desired
% H_{hpf}(ejw).  Shifting in frequency is the same as multiplying
% by exp(j*omega_0*n) in time.  For this problem, we want omega_0 =
% pi, so exp(j*omega_0*n) = (-1)^n.  We can define

h1 = ((-1).^nh).*h;
H1 = freqz(h1,1,1024,'whole');
wshift = w-pi;

figure(2)
clf
subplot(211)
stem(nh,h1)
xlabel('Time (samples)')
ylabel('h_1[n]')
title('Project 6.3 part (b): Impulse Response')
subplot(212)
plot(wshift/pi,fftshift(abs(H1)))
xlabel('Frequency (\omega/\pi)')
ylabel('|H_1(e^{j\omega})|');
title('Project 6.3 part(b): Frequency Response Magnitude')

if finalplots
  print -deps  proj63b.eps
end

% As we expect, h1[n] is the same as h[n] with the sign of every
% other sample changed.  We can see that we get the desired
% frequency response in the plot, with the highpass filter
% frequency response shown.

% part (c)

% For this part, we want construct H2(ejw) such that it is the sum
% of the original H(ejw) shifted by +pi/2 and -pi/2.  We can do
% this by adding the result of multiplying h[n] by exp(j*pi*n/2)
% to the result of adding h[n] to exp(-j*pi*n/2).   This is
% equivalent to h[n]cos(pi*n/2).

h2 = 2*cos(pi*nh/2).*h;  % note factor of 2 needed to keep
                         % amplitude of filter at 1
H2 = freqz(h2,1,1024,'whole');


figure(3)
clf
subplot(211)
stem(nh,h2)
xlabel('Time (samples)')
ylabel('h_2[n]')
title('Project 6.3 part (c): Impulse Response')
subplot(212)
plot(wshift/pi,fftshift(abs(H2)));
xlabel('Frequency (\omega/\pi)')
ylabel('|H_2(e^{j\omega})|');
title('Project 6.3 part (d): Frequency Response Magnitude')

if finalplots
  print -deps  proj63c.eps
end

% Every other sample of h2[n] is zero, and every fourth sample has
% it sign changed from h[n].

% part (d)

% Here the desired frequency response magnitude is |H3(ejw)| = 1-|H2(ejw)|.  In
% the time domain, we need to subtract h2[n] from an impulse at the
% correct sample, which is the point of symmetry.  Thus, h3[n] =
% delta[n-48] - h2[n].

h3 = [zeros(1,48) 1 zeros(1,48)]-h2;
H3 = freqz(h3,1,1024,'whole');

figure(4)
clf
subplot(211)
stem(nh,h3)
```

```
xlabel('Time (samples)')
ylabel('h_3[n]')
title('Project 6.3 part (d): Impulse Response')
subplot(212)
plot(wshift/pi,fftshift(abs(H3)))
xlabel('Frequency (\omega/\pi)')
ylabel('|H_3(e^{j\omega})|');
title('Project 6.3 part (d): Frequency Response Magnitude')

if finalplots
  print -deps  proj63d.eps
end
```

```
% new project
% John Buck
% ECE 320 Fall 2013

FINALPLOTS = 1;

% read in group project
load group0.mat

% Set Nfft to be the next largest power of 2 from length of y
Nfft = 2^ceil(log2(length(y)));

Y = fft(y,Nfft);
omega = (0:(Nfft-1))*(2*pi/Nfft);
omegaplot = omega-pi;

figure(1)
clf
plot(omegaplot/pi,fftshift(abs(Y)));
xlabel('Frequency (\omega/\pi)')
ylabel('Y(e^{j\omega})')
title('Project 3, part a: Noisy signal spectrum')

if FINALPLOTS
  print -deps Proj3Yejw.eps
end

% same plot on dB scale
figure(2)
clf
plot(omegaplot/pi,fftshift(20*log10(abs(Y))));
xlabel('Frequency (\omega/\pi)')
ylabel('Y(e^{j\omega}) (dB)')
title('Project 3, part a: Noisy signal spectrum')
text(0.2,75,'Tone')
text(0,35,'Speech')
text(0.6,40,'Noise')

if FINALPLOTS
  print -deps Proj3Yejwdb.eps
end

% Looks like the speech falls mainly below |omega| \leq 0.4 \pi,
% with a tone right at 0.2pi.

% Part (b): Notch filter

% Define the notch filter impulse response
omegan = 0.2*pi;
hnotch = (1/(2-2*cos(omegan)))*[1 -2*cos(omegan) 1];

% note ok that it's not flat outside

% Compute frequency repsonse of notch filter
Hnotch = fft(hnotch,Nfft);

figure(3)
clf
plot(omegaplot/pi,fftshift(20*log10(abs(Hnotch))));
xlabel('Frequency (\omega/\pi)')
ylabel('Y(e^{j\omega}) (dB)')
title('Project 3, part c: Notch filter frequency response.')

if FINALPLOTS
  print -deps Proj3Hejwnotch.eps
end

% part (d)

r = filter(hnotch,1,y);

wavwrite(r/max(abs(r)),10000,16,'/tmp/proj2r.wav')

R = fft(r,Nfft);

figure(4)
clf
plot(omegaplot/pi,fftshift(20*log10(abs(R))))
xlabel('Frequency (\omega/\pi)')
ylabel('R(e^{j\omega}) (dB)')
title('Project 3, part d: Spectrum with tone removed')

if FINALPLOTS
  print -deps Proj3Rejw.eps
end

% part (e)

% need lpf to cutoff at 0.4pi
Nfilter = 100;
omegalpf = 0.30;
hlpf = fir1(Nfilter,0.4);

% Compute frequency repsonse of notch filter
Hlpf = fft(hlpf,Nfft);

figure(5)
clf
plot(omegaplot/pi,fftshift(20*log10(abs(Hlpf))));
xlabel('Frequency (\omega/\pi)')
ylabel('Hlpf(e^{j\omega}) (dB)')
title('Project 3, part c: Lowpass filter frequency response')

if FINALPLOTS
  print -deps Proj3Hlpf.eps
end

% part (f)

% remove noise with lpf
s = filter(hlpf,1,r);

wavwrite(s/max(abs(s)),10000,16,'/tmp/proj2s.wav')

S = fft(s,Nfft);

figure(6)
clf
plot(omegaplot/pi,fftshift(20*log10(abs(S))))
xlabel('Frequency (\omega/\pi)')
ylabel('S(e^{j\omega}) (dB)')
title('Project 3, part f: Spectrum with tone and noise removed')

if FINALPLOTS
  print -deps Proj3Sejw.eps
end

% part (g)
% combine the filters by convolving them
```

```
hcombo = conv(hnotch,hlpf);

% Compute frequency repsonse of notch filter
Hcombo = fft(hcombo,Nfft);

figure(7)
clf
plot(omegaplot/pi,fftshift(20*log10(abs(Hcombo))));
xlabel('Frequency (\omega/\pi)')
ylabel('Hcombo(e^{j\omega}) (dB)')
title('Project 3, part g: Combined filter frequency response')

if FINALPLOTS
  print -deps Proj3Hcombo.eps
end

s2 = filter(hcombo,1,y);

wavwrite(s2/max(abs(s2)),10000,16,'/tmp/proj2s2.wav')


S2 = fft(s2,Nfft);

figure(8)
clf
plot(omegaplot/pi,fftshift(20*log10(abs(S2))))
xlabel('Frequency (\omega/\pi)')
ylabel('S2(e^{j\omega}) (dB)')
title('Project 3, part h: Output of combo filter')

if FINALPLOTS
  print -deps Proj3S2ejw.eps
end
```