Music 421A
Winter 2011-2012
**Homework #3**
More Windows, Window Design, FIR Filter Design
Due in one week

# Theory Problems

1. (5 pts) Show that the $\cos^p(t)$ window has $p$ leading zeros in the series expansion about its right endpoint.

   **Solution:** Perform the Taylor series expansion of $f(t) = \cos^p(t)$ with respect to $t$ evaluated at $t = t_0 = \pi/2$. The general Taylor series expansion of $f(t)$ about $t = t_0$ is given by

   $$f(t) = f(t_0) + f'(t_0)(t - t_0) + \frac{1}{2!}f''(t_0)(t - t_0)^2 + \frac{1}{3!}f'''(t_0)(t - t_0)^3 + \cdots .$$

   By direct calculation, the $k$th derivative of $f(t)$ is found to be

   $$f^{(k)}(t) = (-1)^{k+1}\frac{p!}{(p-k)!}\cos^{p-k}(t)\sin^{k-1}(t) + g_k(t)$$

   where $g_k(t)$ involves only lower-order derivatives. Since $\cos^{p-k}(\pi/2) = 0$ for $k = 0, 1, \ldots, p-1$, the result is proved. Moreover, for $k = p$, we obtain

   $$f^{(p)}(t) = (-1)^{p+1}p!\,\sin^{p-1}(t) + g_p(t)$$

   for which $f^{(p)}(\pi/2) = (-1)^{p+1}p!\,\sin^{p-1}(\pi/2) \neq 0$.

2. (5 pts) Sketch the window $w$, corresponding to the window transform

   $$W(\omega) \triangleq M\text{asinc}_M(\omega) \triangleq \frac{\sin(M\omega/2)}{\sin(\omega/2)}$$

   where $M$ is an even integer. [Hint: Note that $W$ is not $2\pi$-periodic, but it is $4\pi$-*periodic*. The purest way to deal with this is to define the IDTFT from $-2\pi$ to $2\pi$. Another approach is to use this function to define $W$ on $[-\pi, \pi]$ and define it to be zero outside that interval (the "properly bandlimited" assumption).]

   **Solution:**

   This problem specifies a periodic window transform, continuous in $\omega$, so the appropriate inverse transform is the IDTFT. The tricky part of this problem is recognizing that the function $\text{asinc}_M(\omega)$ is not $2\pi$-periodic, and coming up with something good to do about it.

A quick check shows that the period of the function $\mathrm{asinc}_M(\omega)$ is $4\pi$. Therefore, it makes the most sense to define the inverse DTFT as the integral from $-2\pi$ to $2\pi$ rather than the usual integral from $-\pi$ to $\pi$. Since the inverse DTFT is simply a Fourier series expansion of a periodic spectrum, one should always take the integral over the actual period, just like we always do for Fourier series. When the period doubles in the input to a Fourier series expansion, the harmonics become integer multiples of a *halved* fundamental frequency. Therefore, in the time domain, doubling the integration limits of the inverse DTFT corresponds to *doubling the sampling rate.*

Thus, the window is given by

$$\begin{aligned}
w(n) &= \frac{1}{4\pi}\int_{-2\pi}^{2\pi} W(\omega)\exp(j\omega n)d\omega \triangleq \frac{1}{4\pi}\int_{-2\pi}^{2\pi}\frac{\sin(M\omega/2)}{\sin(\omega/2)}\exp(j\omega n)d\omega \\
&= \frac{1}{4\pi}\int_{-2\pi}^{2\pi}\frac{\exp(j\omega M/2)-\exp(-j\omega M/2)}{\exp(j\omega/2)-\exp(-j\omega/2)}\exp(j\omega n)d\omega \\
&= \frac{1}{4\pi}\int_{-2\pi}^{2\pi}\exp(j\omega(M-1)/2)\frac{1-\exp(-j\omega M)}{1-\exp(-j\omega)}\exp(j\omega n)d\omega \\
&= \frac{1}{4\pi}\int_{-2\pi}^{2\pi}\exp(j\omega[n+(M-1)/2])\sum_{m=0}^{M-1}\exp(-j\omega m)d\omega \\
&= \sum_{m=0}^{M-1}\frac{1}{4\pi}\int_{-2\pi}^{2\pi}\exp(j\omega[n-m+(M-1)/2])d\omega \\
&= \sum_{m=0}^{M-1}\frac{1}{4\pi}\frac{\exp(j2\pi[n-m+(M-1)/2])-\exp(-j2\pi[n-m+(M-1)/2])}{j[n-m+(M-1)/2]} \\
&= \sum_{m=0}^{M-1}\frac{\sin(2\pi[n-m+(M-1)/2])}{2\pi[n-m+(M-1)/2]} \\
&= \sum_{m=0}^{M-1}\mathrm{sinc}[2(n-m)+M-1], \qquad n=\ldots,-\frac{3}{2},-1,-\frac{1}{2},0,\frac{1}{2},1,\frac{3}{2}\ldots
\end{aligned}$$

Recall that $\mathrm{sinc}(x)$ is zero at all integers except 0. Therefore, each term in the sum over $m$ is be zero except at $m = n + (M-1)/2$ where that term is 1. Since $m$ ranges over the integers between 0 and $M-1$, and since $M$ is even, the $M$ values of $n$ for which such an $m$ exists under the sum are $-(M-1)/2$ to $(M-1)/2$, excluding the integers in that interval. Hence, the window function is given by

$$\begin{aligned}
w(n) &= \sum_{m=0}^{M-1}\mathrm{sinc}[2(n-m)+M-1], \qquad n=\ldots,-\frac{3}{2},-1,-\frac{1}{2},0,\frac{1}{2},1,\frac{3}{2}\ldots \\
&= \begin{cases} 1, & n=-\frac{M-1}{2},-\frac{M-3}{2},\ldots,-\frac{3}{2},-\frac{1}{2},\frac{1}{2},\frac{3}{2},\ldots,\frac{M-3}{2},\frac{M-1}{2} \\ 0, & \text{otherwise} \end{cases}
\end{aligned}$$

Thus, the given window transform asinc$_M$ is the DTFT of the $M$-sample pulse centered at time 0, regardless of whether $M$ is odd or even. For even $M$ it means time 0 is midway between two unit samples at times $\pm 1/2$.

Here is a derivation by former classmate Hoi Wong:

We already know the formulation of rectangular windows, where $w_R[n]$ is defined in the text and we renamed the length of the original window to $P$:

$$W_R(\omega) = \frac{\sin(P\frac{\omega}{2})}{\sin(\frac{\omega}{2})}$$

Since $M$ is even, we define $P \in \mathbb{Z}^+$ s.t. $M = 2P$. WLOG, and using SASP's definition of $w_R[n]$, we also require $P$ to be odd for the moment. (Other cases where $M$ has to be even (because $P = 2^{p+1}q : p \in \mathbb{Z}^+, q \in 2\mathbb{Z}^+ - 1$) will be explained after the math.)

$$W(\omega) = \frac{\sin(2P\frac{\omega}{2})}{\sin(\frac{\omega}{2})}$$

Using the double-angle formula $\sin(2\alpha) = 2\cos(\alpha)\sin(\alpha)$,

$$\begin{aligned} W(\omega) &= 2\cos\left(P\frac{\omega}{2}\right)\frac{\sin(P\frac{\omega}{2})}{\sin(\frac{\omega}{2})} \\ &= 2\cos\left(P\frac{\omega}{2}\right)W_R(\omega) \end{aligned}$$

Expressing cosine as complex exponentials,

$$W(\omega) = (e^{j\omega\frac{P}{2}} + e^{-j\omega\frac{P}{2}})W_R(\omega)$$

or in the time domain,

$$w[n] = \left(\delta\left[n - \frac{P}{2}\right] + \delta\left[n + \frac{P}{2}\right]\right) * w_R[n].$$

There are two ways to explain what happens when $M$ is even:

1) [More intuitive explanation] From HW 1 solution, if $M$ is even for rectangular window, the samples will occur at non-integer (half) indicies; but if $M$ is even, $P/2$ has to be integer and hence are the delays/advances caused by the unit impulse terms. So, the result will end up being the same as the case when $M$ is odd.

2) [More mathematical explanation] Let $P = 2P_0 = 2(2P_1) = 2(2(2P_2)) = ...$ until all factors of two was brought out of the $P_{p-1}$ term, and call the remaining odd term $M$ (so the math for the odd-length rectangular window holds). Apply the resulting expression derived above recursively. Here is the first iteration as an example:

$$w_R[n] = \left(\delta\left[n - \frac{P_0}{2}\right] + \delta\left[n + \frac{P_0}{2}\right]\right) * w_{R_0}[n]$$

where $w_{R_0}$ is half (can be non-integer) as long as of $w_R$, and so on. By the end of the day, you will end up with a bunch of unit impulse terms convolved together:

$$\left(\delta\left[n-\frac{P}{2}\right]+\delta\left[n+\frac{P}{2}\right]\right)*\left(\delta\left[n-\frac{P_0}{2}\right]+\delta\left[n+\frac{P_0}{2}\right]\right)*...*(\delta\left[n-M\right]+\delta\left[n+M\right])*w_{R_{p-1}}[n]$$

Since we are dividing the length by 2 for each iteration, the first $p-1$ terms (grouped by parentheses) are going to be integer shifts. Therefore, only the last grouped term $(\delta[n-M]+\delta[n+M])$ gives half-indicies shifts. From the width of the $w_{R_{p-1}}[n]$ and all the magical placement by the scattered impulses (due to convolving a bunch of impulse pairs together), you will indeed get the same answer.

To sum up, if $M$ is odd, the unit impulse terms are responsible for the half-shifts; if $M$ is odd, the rectangular window is responsible for having samples at half indicies. You get the same answer in both cases.

*(For students following the 'double angle' approach above: if they get the odd $M$ case right and recognize (derivation not necessary) that even $M$ will give the same result, they should get full credit for this problem.)*

Another approach to this problem is to *introduce the assumption* that the given window transform pertains only to the frequency interval $(-\pi,\pi]$ and that it is *zero elsewhere* on $(-\infty,\infty)$. In this case, the answer is given by the *inverse Fourier transform,* evaluated (i.e., sampled at) integer time values:

$$\begin{aligned}
w(t) &= \frac{1}{2\pi}\int_{-\infty}^{\infty} W(\omega)\exp\left(j\omega t\right)d\omega \triangleq \frac{1}{2\pi}\int_{-\pi}^{\pi} W(\omega)\exp\left(j\omega t\right)d\omega \\
&\triangleq \frac{1}{2\pi}\int_{-\pi}^{\pi}\frac{\sin(M\omega/2)}{\sin(\omega/2)}\exp\left(j\omega t\right)d\omega \\
&= \ldots \text{ same steps as before } \ldots \\
&= \sum_{m=0}^{M-1}\text{sinc}[t-m+(M-1)/2], \qquad t\in\mathbf{R}
\end{aligned}$$

The window is then defined as $w(n)$, i.e., $w(t)$ sampled on the integers. In this case, the answer may look a little surprising.

Note that in this bandlimited formulation, the superposition of sinc functions above gives the unique, bandlimited interpolation of the time-domain window samples, and we can define the sampling instants anywhere we wish, not just on the integers. For this problem, the convenient choice of sampling instants would be $t_n = n + 1/2$. Such a choice of sampling times at odd multiples of $1/2$ allows one to sketch the ideal $M$-sample pulse consisting of $M$ unit samples at times $n = -M/2, -M/2 + 1, \ldots, -1, 0, 1, \ldots, M/2 - 1$, as in the first solution given for this problem.

3. (5 pts) **System Identification Using the Poisson Window**
Suppose now that we are attempting to measure and model a *loudspeaker* impulse response $h(t)$.

We want to model the speaker impulse response using a one-pole filter:

$$y(n) = b_0 x(n) + a_0 y(n-1)$$

where, $b_0$, and $a_0$ are our design parameters we will be fitting.

Suppose we record the speaker impulse response with sampling rate $f_s = 50,000$, and it rings for over 2 minutes. We have limited memory in which to load the file for our model fitting, and can only use 2 seconds of data.

(a) (3 pts) Explain how the Poisson window can effectively suppress the ringing of the speaker response and allow us to fit a one-pole model to a shorter windowed recording, with a post-processing step to correct the fitted pole so that the window has no effect in the end.

**Solution:** To suppress the ringing of the of the speaker response and allow the model fitting on only 2 seconds of the data, the Poisson window can be applied prior to fitting. The model and be fit on the windowed data (with a scaled unit circle), and then post-processed to undo the effect of the window.

More specifically, by applying a Poisson or exponential window, the ringing is damped by a factor $r = e^{-\frac{\alpha}{(M-1)/2}}$, resulting in a transfer function $H(\frac{z}{r})$. This is exactly the scaling factor that needs that to be applied in reverse to compensate for the window.

(b) (2 pts) We want to estimate the model using the 2 seconds of data with a Poisson window T60 time of the same length (2 seconds). What is the $\alpha$ parameter of the Poisson window to acheive this specification? Recall that the Poisson window is defined as

$w_P(n) = w_R(n) e^{-\alpha \frac{|n|}{\frac{M-1}{2}}}$.

**Solution:**

The time constant $\tau$ is first computed via

$\tau = \mathrm{T}_{60} / \ln(1000)$.

$\alpha$ can then be computed via

$\alpha = .5(\mathrm{M}-1)\mathrm{T}/\tau = .5(100000-1)\ln(1000)(1/50000)/(\mathrm{T}_{60}) = 3.4538$

where M is the number of samples of the window and T is the sampling period.

4. (5 pts) Let $W(\omega_a) = T_M(\alpha\omega_a)$, $\omega_a \in (-\infty, \infty)$, be a continuous spectrum, where $T_M$ denotes the $M$th Chebyshev polynomial, and $\alpha$ is any constant scale factor. Prove that the corresponding time-domain signal $w(t)$ has finite support, and find the range of $t$ specifying that support.

**Solution:** $W(\omega_a) = 1$ when M=1. $W(\omega_a) = \alpha\omega_a$ when M=2. Using the recursive relation of Chebychev polynomial, $T_M(\alpha\omega_a) = 2\alpha\omega_a T_{M-1}(\alpha\omega_a) - T_{M-2}(\alpha\omega_a)$, $W(\omega_a)$

can be expressed as a polynomial of $\omega_a$.
From Fourier Transform, $w_0(t) = \delta(t)$ and the M-th order polynonimal of $j\omega$ correponds to the M-th order derivative (including lower order derivatives). Thus, the time-domain signal $w_M(t)$ can be expressed as the M-th order deriavtive of $\delta(t)$. As a result, the $w_M(t)$ has finite support at the range of $t = 0$.

5. (5 pts) Let $W(\omega_d) = T_M[\alpha \cos(\omega_d/2)]$, $\omega_d \in (-\pi, \pi)$, be the continuous spectrum (DTFT) of a discrete-time signal $w(n)$, where $T_M$ again denotes the $M$th Chebyshev polynomial, and $\alpha$ is any constant scale factor. Prove that the corresponding time-domain signal $w(n)$ has finite support, and find the integers $n$ specifying that support.

   **Solution:** If $M$ is odd, $n = -M/2, \ldots, M/2$. If $M$ is even, I expect (but have not proved) that finite-support no longer occurs, because the finite-support window lies "between samples". Up to 2 extra points for delving into this better than others.

# Lab Problems

1. (5 pts) Find out where the Chebyshev window "breaks down" in Matlab. Let the length be fixed at $M = 31$, and try various ripple specifications until there is an obvious error in the window obtained. Describe the source of the failure when the ripple specification is (a) too large and (b) too low. The following Matlab code can be used as a starting point:

```
N = 8192;
M = 31;
w = chebwin(M,rip);
W = fft(w,N);
% normalize and clip the window transform in dB:
Wdb = 20*log10(max(abs(W)/max(abs(W)),...
                    10^(-rip*1.5/20)));
f = [0:N-1]/N - 0.5;
plot(f,fftshift(Wdb)); grid;
axis([-0.5 0.5 -rip*1.5 0]);
hold on;
plot([f(1),f(N)],[-rip,-rip],'--k');
text(f(1)+0.02,-rip+rip*1.5/20,...
      sprintf('rip = %0.1f dB',rip));
xlabel('Normalized Frequency (cycles/sample)');
ylabel('Magnitude (dB)');
title(sprintf('Length %d Chebyshev window',M));
```

   **Solution:** (5 pts) When the ripple specification is too large (about 350 dB or so), i.e., the desired sidelobe ripple is too far down, finite precision causes the sidelobe level

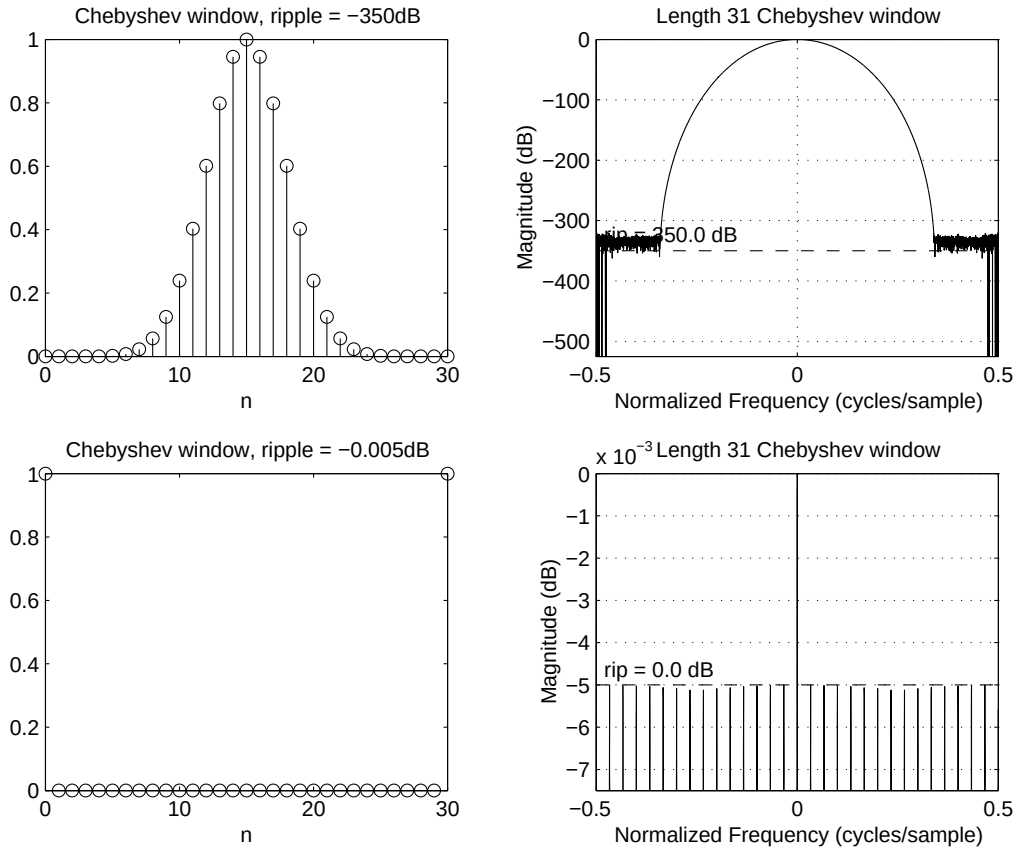



Figure 1: The Chebyshev window at its breakdown

to exceed specifications and not be equiripple, as shown in Figure 1 (top right). For full credit, a description of what happens visually is required. Insightful analysis of precision effects may result in extra credit.

When the ripple specification approaches 0 dB, (about 0.005 dB or so), i.e., the desired sidelobe ripple approaches the main-lobe height, the nature of the breakdown is shown in Figure 1 (bottom). The time-domain window approaches two impulses located at the window endpoints, and this interferes with standard normalization practice in the time domain. At least two answers are correct here, and possibly more: (1) when the endpoint impulses exceed the midpoint height, it is no longer possible to normalize the window to maximum amplitude 1 in the time domain by normalizing its midpoint to amplitude 1. (2) If normalization by midpoint amplitude is all that is required (ignoring the ever increasing endpoint impulses), this too becomes impractical as the midpoint amplitude approaches zero. Extra credit is given for showing theoretically that the midpoint amplitude approaches zero, thus defeating the standard normalization practice.

2. (10 pts) For $\omega_c T = \pi/2$, design a length $M = 100$ real FIR lowpass filter using

the window method. Plot the amplitude response (dB gain versus frequency) for the following windows:

- Hamming
- Hann
- Blackman
- Kaiser with $\beta = 10$

Explain why the stopband rejection of the lowpass filter (designed by a window) is so different from the side-lobe level of the window itself.

**Solution:**
The plots of the frequency responses are shown in Figure 2. The sidelobe levels in this problem are smoother and lower than the previous one. This is because the sidelobes of the window spectrum gets smoothed out by the convolution process in the frequency domain with the ideal lowpass filter frequency response which is a rectangular function. This may be counterintuitive, but we recall that sidelobes alternate positive and negative, thus when we sum over more of them under the ideal rect function, we get closer to zero. (This frequency domain convolution is due to multiplication in time domain of the ideal impulse response by the window.)

Relevant code:

```
M = 100;
N = 8*M;
n = -M/2+1:M/2;
hid=sin(pi*0.5*n)./(pi*0.5*n+eps);
hid(n==0) = 1; % due to sinc exception

w=hamming(M)';
hw=hid.*w;
Hw=fft(hw,N);
figure(3);
plot([0:1/N:0.5-1/N],20*log10(max(0.000001,abs(Hw(1:N/2))/max(abs(Hw)))),'b--');
grid
figure(2);
subplot(411);
plot([0:1/N:0.5-1/N],20*log10(max(0.000001,abs(Hw(1:N/2))/max(abs(Hw)))),'b');
grid

w=hanning(M)';
hw=hid.*w;
Hw=fft(hw,N);
figure(3);
hold on;
```

```
plot([0:1/N:0.5-1/N],20*log10(max(0.000001,abs(Hw(1:N/2))/max(abs(Hw)))),'k');
grid
xlabel('normalized frequency');ylabel('Normalized gain(dB)');
figure(2);
subplot(412);
plot([0:1/N:0.5-1/N],20*log10(max(0.000001,abs(Hw(1:N/2))/max(abs(Hw)))),'b');
grid

w=blackman(M)';
hw=hid.*w;
Hw=fft(hw,N);
figure(3);hold on;
plot([0:1/N:0.5-1/N],20*log10(max(0.000001,abs(Hw(1:N/2))/max(abs(Hw)))),'r-.');
grid
xlabel('normalized frequency');ylabel('Normalized gain(dB)');
hold on;
figure(2);
subplot(413);
plot([0:1/N:0.5-1/N],20*log10(max(0.000001,abs(Hw(1:N/2))/max(abs(Hw)))),'b');
grid

% Kaiser
w=kaiser(M,10)';
hw=hid.*w;
Hw=fft(hw,N);
figure(3);
hold on;
plot([0:1/N:0.5-1/N],20*log10(max(0.000001,abs(Hw(1:N/2))/max(abs(Hw)))),'g:');
grid
legend('hamming','hann','blackman','kaiser');
hold off
figure(2);
subplot(414);
plot([0:1/N:0.5-1/N],20*log10(max(0.000001,abs(Hw(1:N/2))/max(abs(Hw)))),'b');
grid
```

3. Design a length $M = 51$ Chebyshev window with 40 dB of sidelobe attenuation using

   - `chebwin`
   - `firpm` (formerly called `remez`)
   - `linprog` [Hints: $M = 2L+1$, where $L = 25$. Use a frequency grid length $K$ which is much larger than $L$, such as in the hundreds.]

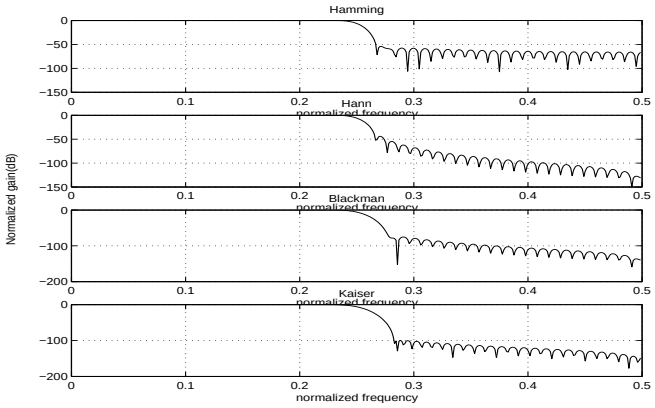



Figure 2: The magnitude frequency responses of Hamming, Hann, Blackman and Kaiser(10) window filter design.

in matlab. Normalize each window such that the main lobe of the window transform has peak magnitude 0 dB. For `firpm` and `linprog`, set the normalized transition bandwidth to 0.068 (where 0 is dc and 1 is half the sampling rate, is as commonly used in matlab). Using `cputime`, measure the average compute-time for ten iterations of each of the three methods above.

(a) (5 pts) Report the three average compute times. Divide the two longer compute-times by the smallest compute time and report those two speed ratios.

**Solution:** The following code shows the required matlab operations:

```
M=51;
rip=40;
fsb=.068;
Ni = 10;

t1 = zeros(Ni,1);
for i=1:Ni
  t1i = cputime;
  w1 = chebwin(M,rip);
  t1(i) = cputime - t1i;
end
w1=w1/sum(w1);

t2 = zeros(Ni,1);
for i=1:Ni
  t2i = cputime;
  w2 = firpm(M-1,[0 0 fsb 1],[1 1 0 0])';
  t2(i) = cputime - t2i;
end
```

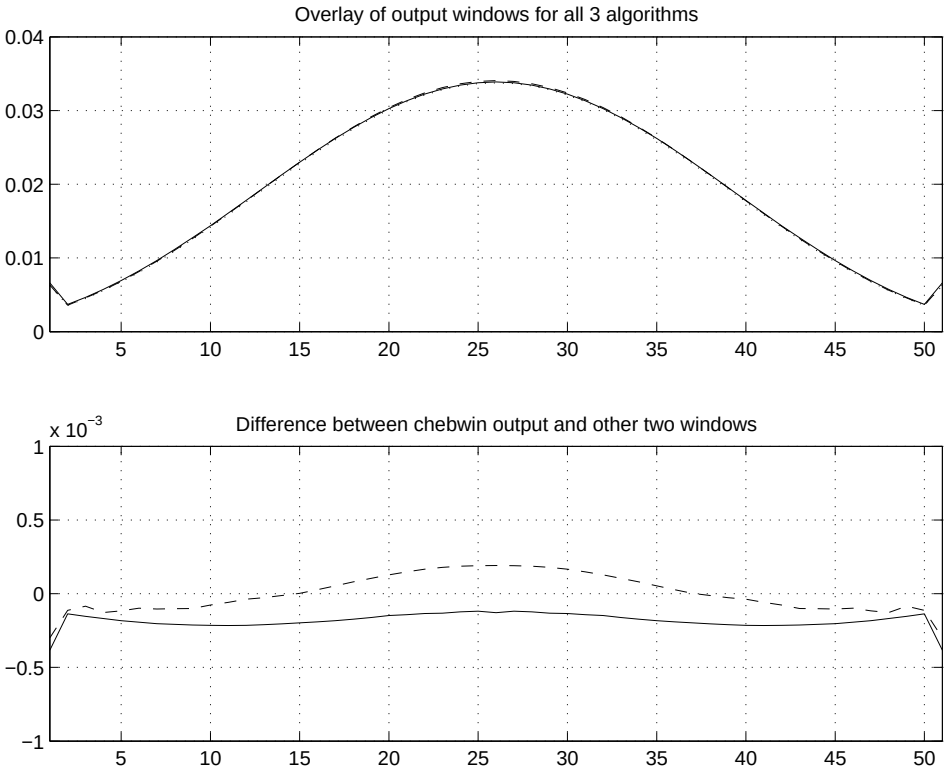
```
L = (M-1)/2;
i = [fsb:.005:1];
Asb = zeros(2*length(i),L+2);
for j=1:length(i)
  wi = pi*i(j);
  Asb(2*j-1,1) = 1;
  Asb(2*j,1) = -1;
  for l=1:L
   Asb(2*j-1,l+1) = 2*cos(wi*l);
   Asb(2*j,l+1) = -1*Asb(2*j-1,l+1);
  end
  Asb(2*j-1,L+2) = -1;
  Asb(2*j,L+2) = -1;
end

f = [zeros(L+1,1);1]';
A = [-1*eye(L+1) zeros(L+1,1);Asb];
b = zeros(size(A,1),1);
Aeq = [1 2*ones(1,L) 0];
beq = 1;

t3 = zeros(Ni,1);
for i=1:Ni
  t3i = cputime;
  x = linprog(f,A,b,Aeq,beq);
  w3 = [flipud(x(2:L+1));x(1:L+1)];
  t3(i) = cputime-t3i;
end

figure;subplot(211);plot(w1);hold on;
plot(w2,':');plot(w3,'--');grid on;axis([1 51 0 .04]);
title('Overlay of output windows for all 3 algorithms');
subplot(212);plot(w2-w1);hold on;plot(w3-w1,'--');axis([1 51 -.001 .001]);grid
title('Difference between chebwin output and other two windows');

[H1 W1] = freqz(w1,1,51*5);
[H2 W2] = freqz(w2,1,51*5);
[H3 W3] = freqz(w3,1,51*5);
figure;plot(W1,20*log10(abs(H1)));hold on;
plot(W2,20*log10(abs(H2)),':');plot(W3,20*log10(abs(H3)),'--');grid on;
title('Frequency domain overaly of all 3 windows');
```




**Solution:** (a) Times will depend on the computer used, but in one run, the values were .001, .0370, and 3.1 seconds, respectively. The ratios were undefined for `chebwin` (fastest time registers as 0), 1.33 for `firpm`, and 1.0097 and 1.0065 for `linprog`

(b) (5 pts) Plot an overlay of the three windows in the time domain. Repeat with the chebwin case subtracted out.

**Solution:**

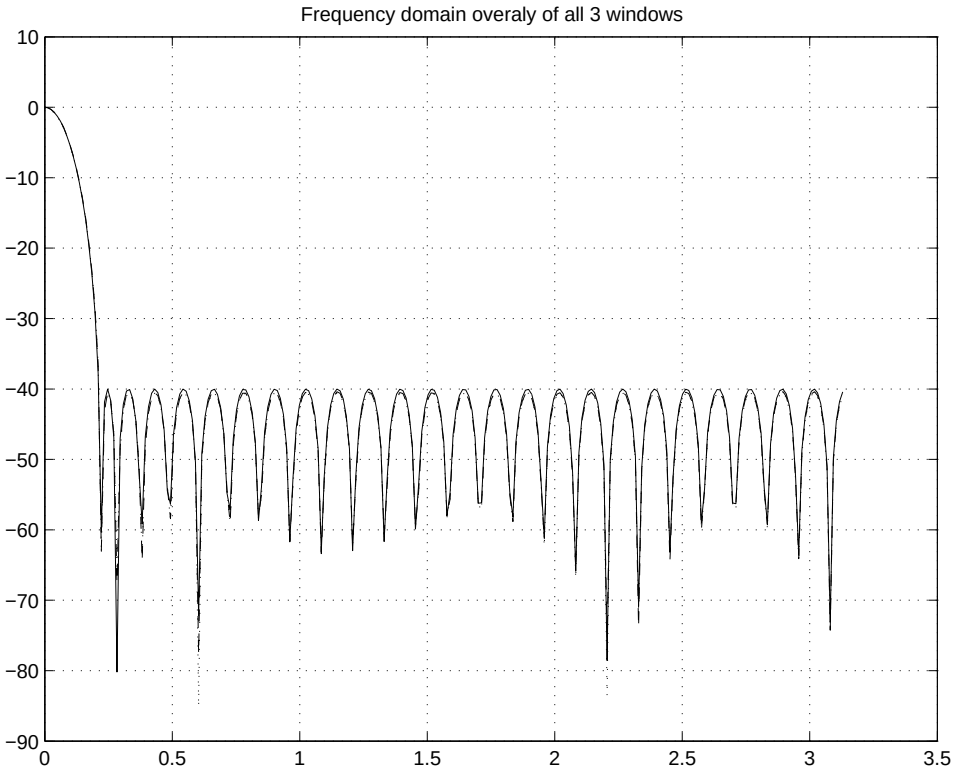
(c) (5 pts) Plot an overlay of the window magnitude spectra in the frequency domain. Use at least a factor of five zero-padding.

**Solution:**

(d) (2 pts) Which method is the most accurate and why do you think that is? Describe any numerical considerations you can see.

**Solution:** We can check the required equal-ripple property and form differences with the `chebwin` case. (We expect it case to be most accurate since it is known in closed form).

(e) **Optional:** For each method, find the order (to within 10%) above which numerical problems become significant. [It is probably easiest to inspect the magnitude spectra to detect numerical troubles.] [To obtain an accuracy of 10%, one can increase the order by 10% each trial, or double it each trial, followed by 10%




increments over the last interval, etc.]

**Solution:**

Due to computer variation, the answers are not unique. You could check the window shape or spectrum to see whether there is obvious degradation. Example answer:

For chebwin:
M = 551 ok, M = 601 the window shape is degraded.
For firpm:
M = 231 ok, M = 251 the window shape is degraded.
For linprog:
M = 141 ok, M = 151 the window shape is degraded.

Note that this question originally used a length 255 window, but the matlab function `linprog` was numerically unable to handle this optimization. But, for those who want to try, it can be solved using the disciplined convex optimization toolbox `cvx`, which should be installed on CCRMA machines. Build the matrices the exact same way, but instead of calling `linprog`, use the following code:

```
f=f';
cvx_begin
variable x(L+2)
```

```
minimize f*x
subject to
A*x<=b
Aeq*x==beq
cvx_end
```

This code will solve the constrained minimization and return the right half of the correct window.