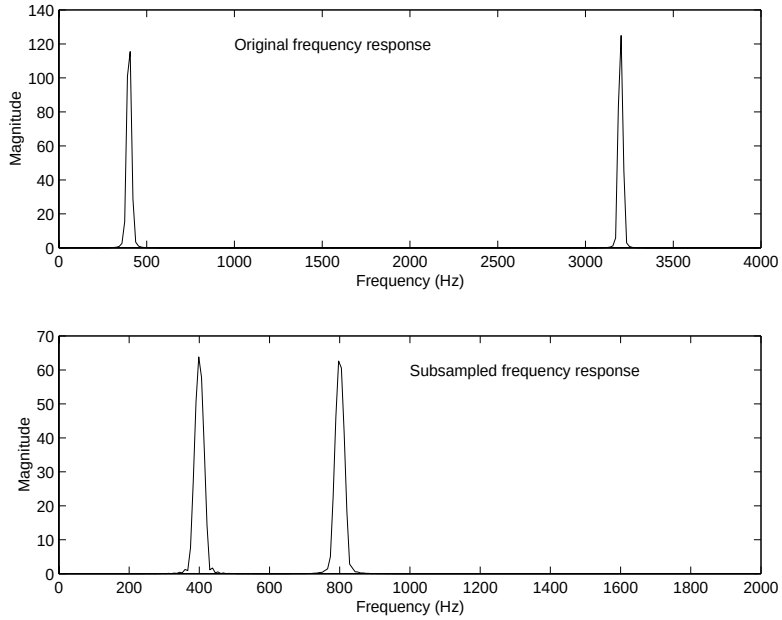
1. The meat of this part is as follows:

```
fs=8000; ts=1/fs;                          % time vector
t=0:ts:ts*511;

a=sin(0.8*pi*t*fs)+sin(0.1*pi*t*fs);       % input signal
b=a(1:2:length(a));                        % subsample
[ha,wa]=freqz(a.*hann(512),1,256,fs);      % compute frequency content
[hb,wb]=freqz(b.*hann(256),1,256,fs/2);
```

with results:



The first plot shows two components, one at 400 Hz (0.1π), the other at 3200 Hz (0.8π). After subsampling, the Nyquist frequency drops to 2000 Hz. The 400 Hz signal is unchanged. The 3200 Hz signal, however, is above the Nyquist frequency after subsampling, and therefore aliases down to $4000 - 3200 = 800$ Hz.

2. The ideal compensation filter response can be computed as follows:

```
w=0:pi/128:255*pi/128;                     % ideal response
fr=(w(1:129)+eps)./ ...
   (2*sin((w(1:129)+eps)/2));              % eps prevents trouble at DC
fr(130:256)=conj(fr(128:-1:2));            % conjugate mirror
ir_ideal=fftshift(real(ifft(fr)));         % ideal impulse response
```

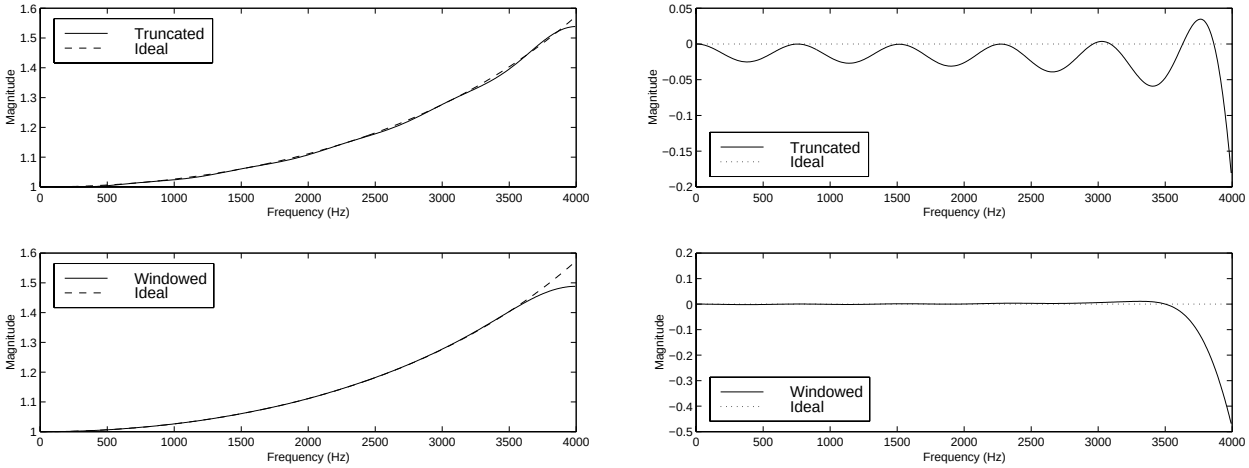
We generate the ideal frequency response `fr` by first applying the ideal formula from the notes over values of `w` from 0 to π, and then conjugate mirror this to form the response to slightly less than 2π. A real time-domain function has a Fourier transform that is conjugate symmetric; by the uniqueness of the Fourier transform, a conjugate symmetric function will have a real inverse Fourier transform (note that we don't really need to conjugate here because the frequency response is real; in general, however, we do). We compute the impulse response by taking the `ifft` and performing an `fftshift` of the real part. Note that introducing a delay in the frequency response by multiplying each element by a factor of the form $e^{-j\omega n}$, where n is the delay in samples, causes the computed impulse response to shift in time; one can use this to avoid using `fftshift`. Note

also that *n* does not have to be an integer; despite the discrete nature of time in a digital system, a filter with *any* time delay can be synthesized. 'Analog freaks' seem unable to grasp this.

We now compute the impulse responses of our two approximate filters:

```
ir_trunc=ir_ideal(119:139);                % truncated impulse response
ir_trunc=ir_trunc/sum(ir_trunc);           % set DC gain to unity
ir_win=ir_trunc.*hamming(21)';             % ditto for windowed filter
ir_win=ir_win/sum(ir_win);
```

We set the DC gain to 1 on these filters by dividing by the sum of the coefficients (recall that the DC gain of a filter is given by the sum of its coefficients). This is not necessary, but avoids any constant gain offset that causes the dB plot to look rather strange. The results for the frequency responses and deviations in dB from the ideal of the two filters are shown next.



Both filters fit the ideal response reasonably well, though they deviate at the edge of the band. The truncated filter shows the expected Gibbs-related ripples throughout the passband, while the windowed filter has a much smoother response. However, the windowed filter has greater error at the band edge. This stems from the fact that its effective length is less than the truncated filter, despite having the same number of coefficients, because the coefficients far from the middle of the filter are attenuated by the windowing and therefore contribute less to the response. This is the price of windowing. (Another way of saying this is that the truncation filter has the lowest MSE.) However, increasing the length of the windowed filter improves its accuracy, whereas increasing the length of the truncated filter will not reduce the Gibbs ripples.

See the ftp site *pepperoni.ece.utexas.edu* in the directory */pub/Assignments*, or the Web site `http://anchovy.ece.utexas.edu/members/tom/ee381k`, for the complete code listing.