Problem 1

Show that for a frame interval of L samples (i.e., time decimation of L samples), $2L$ samples of $|X(nL, k)|$ over $[0, \pi]$ in frequency are required to uniquely recover a sequence $x[n]$. Assume the window length $N_w \geq L/2$. For simplicity, assume N_w and L are even.

Solution.
Given $N_w \geq L/2 \implies \frac{2\pi}{N_w} \leq \frac{4\pi}{L}$, and to recover $x[n]$, we know that $N_w <$ Time decimation Length, is sufficient. By OLA Method, a frequency sampling factor N less than $2\pi/N_w$ would be sufficient for recovering $x[n]$ by avoiding time aliasing.

$$N < \frac{2\pi}{N_w} \leq \frac{4\pi}{L}$$

Number of samples greater than $2L$ would be enough to recover the sequence.

■

Problem 2

Consider modifying the STFT to obtain

$$Y(n,\omega) = X(n,\omega)H(\omega)$$

where the modifying function is given by

$$H(\omega) = e^{jn_0\omega}$$

i.e. a linear-phase modification. Suppose a sequence $y[n]$ is computed by the filter bank summation (FBS) synthesis method:

$$y[n] = \left[\frac{1}{Nw(0)}\right] \sum_{k=0}^{N-1} Y(n,k)e^{j\frac{2\pi}{N}kn}$$

Derive an expression for $y[n]$ in terms of original sequence $x[n]$ and the window $\omega[n]$. Consider two different cases:

1. The length of $\omega[n]$ is less than N
2. The length of $\omega[n]$ greater than or equal to N

Solution.

$$\begin{aligned}
y[n] &= \left[\frac{1}{Nw(0)}\right] \sum_{k=0}^{N-1} Y(n,k)e^{j\frac{2\pi}{N}kn} \\
&= \left[\frac{1}{Nw(0)}\right] \sum_{k=0}^{N-1} (\sum_{m=-\infty}^{\infty} x[m]w[n-m]e^{j\frac{2\pi}{N}k(n_0-m)})e^{j\frac{2\pi}{N}kn} \\
&= \left[\frac{1}{Nw(0)}\right] \sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}kn_0} (\sum_{m=-\infty}^{\infty} x[m]w[n-m]e^{j\frac{2\pi}{N}k(n-m)}) \\
&= \left[\frac{1}{Nw(0)}\right] \sum_{m=-\infty}^{\infty} x[m] (\sum_{k=0}^{N-1} w[n-m]e^{j\frac{2\pi}{N}k(n-m)}e^{j\frac{2\pi}{N}kn_0}) \\
&= \left[\frac{1}{Nw(0)}\right] x[n] * (w[n] \sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}k(n+n_0)}) \\
&= \left[\frac{1}{Nw(0)}\right] x[n] * (w[n] \cdot N \sum_{r=-\infty}^{\infty} \delta[n+n_0-rN])
\end{aligned}$$

Now this is equal to $x[n]$ if

$$w[n] \cdot \sum_{r=-\infty}^{\infty} \delta[n+n_0-rN] = w[0]\delta[n]$$

Case 1. $N_w < N$
The above constraint is already satisfied for such a window whose length is less than to the number of analysis filters N. So $y[n] = x[n]$.

Case 2. $N_w \geq N$
The constraint above won't hold for a general window, but if $w[rN] = 0 \ \forall r \in \mathbb{Z}\backslash\{0\}$, like the sinc function, then the constraint holds.

■

Problem 3

Programming Assignment: In this problem, use the voiced speech waveform *speech2_10k* in the workspace *ex7M1.mat* to design the time-scale modification systems based on OLA and LSE synthesis methods. The speech was sampled at 10000 samples/s.

1. Write a MATLAB function to compute the STFT of the sequence *speech2_10k* using a 30 ms triangular analysis window, created using MATLAB function *triang.m*, at a 15 ms frame interval. Then reconstruct the original waveform from the STFT using OLA approach for synthesis. Ignore tapering end effects of the first and last frames.

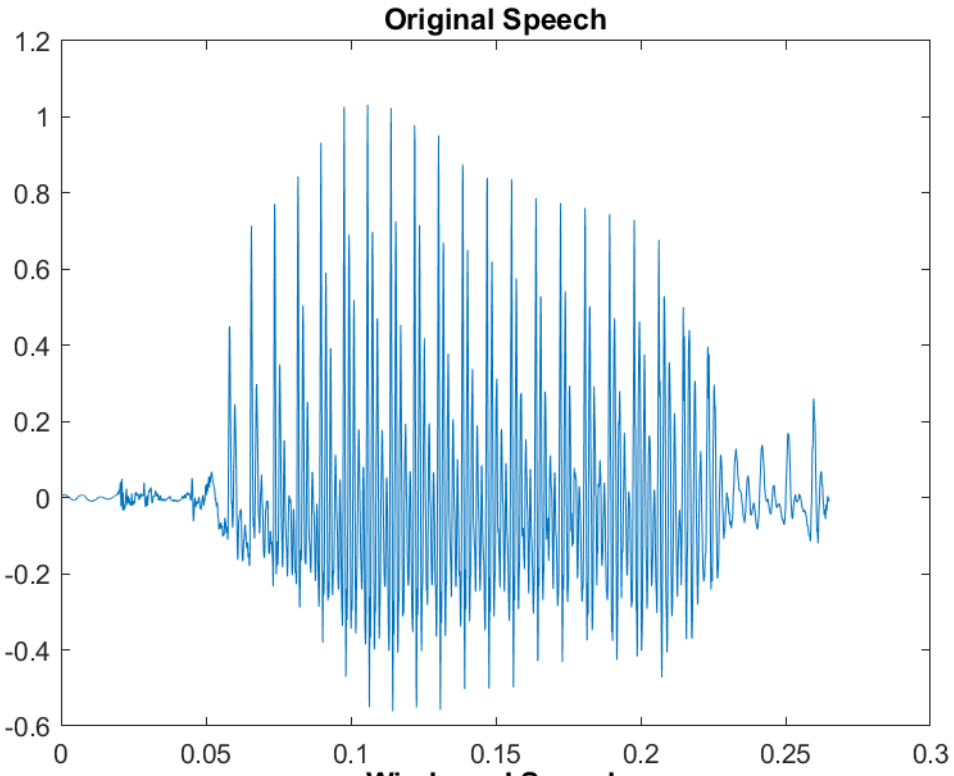
2. Design in MATLAB an OLA-based synthesis method in time-scale expand the speech signal by a factor of two by repeating every frame.

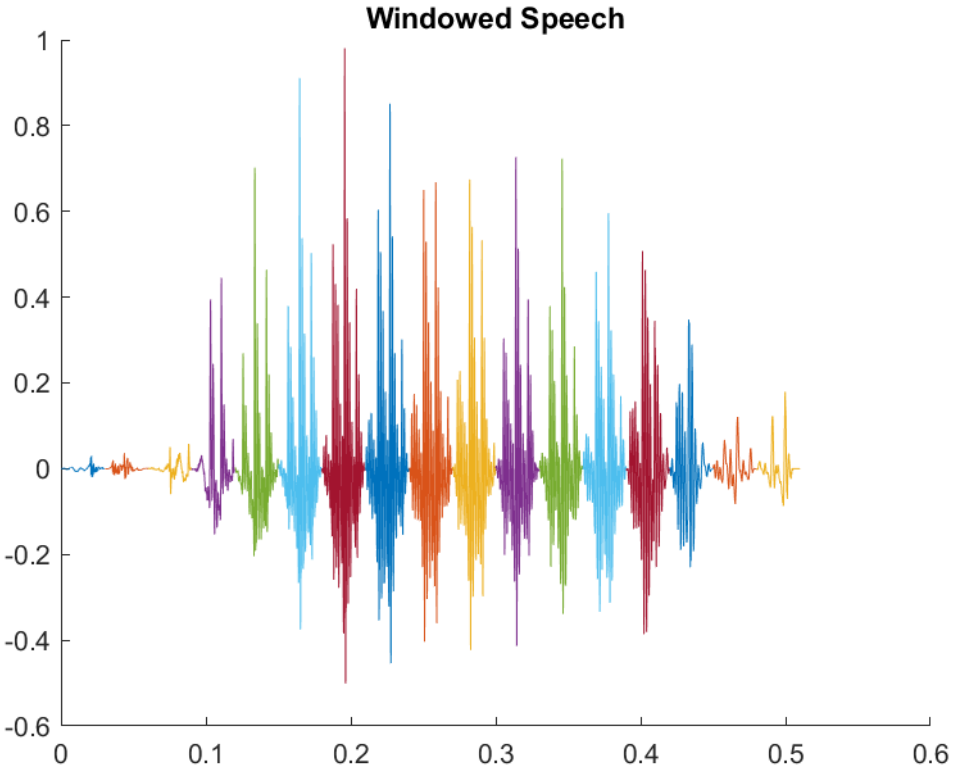
3. Repeat parts (1) and (2) using LSE-based synthesis approach. Compare the time-scaled signals from each approach and comment on the differences.

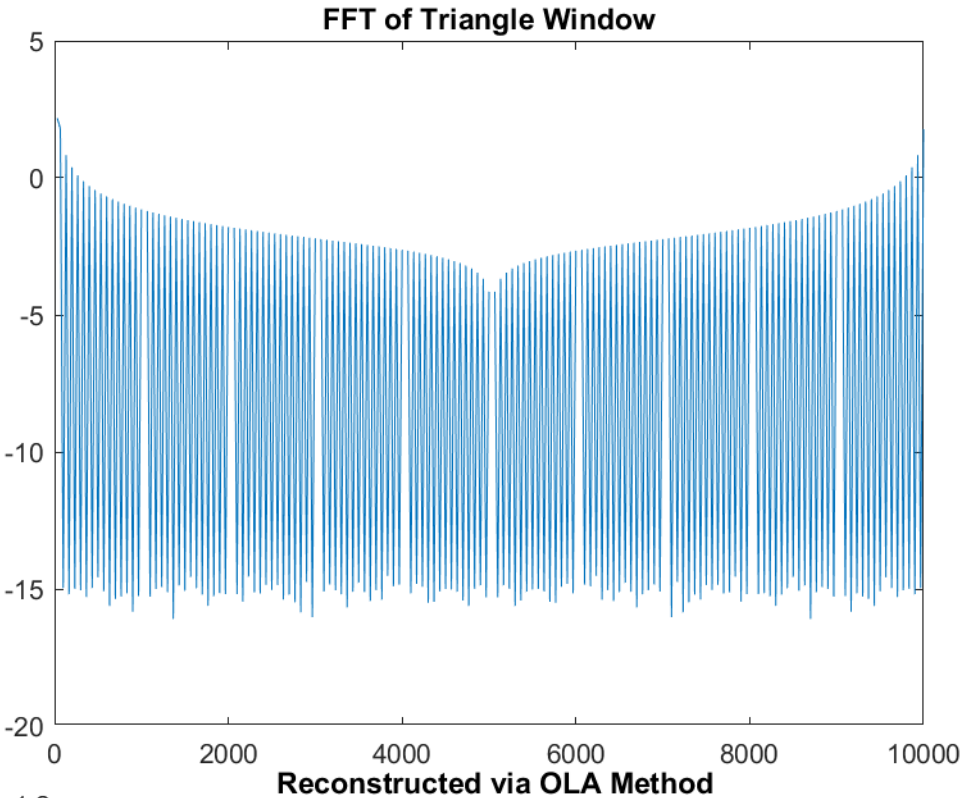
4. Repeat parts (2) and (3) using synchronized OLA and LSE based synthesis approaches. This will require that you estimate a consistent time-instant (e.g. the waveform peak or glottal pulse time) within a glottal cycle in order to synchronize consecutive frames. How does this approach improve your synthesis from parts (2) and (3)?

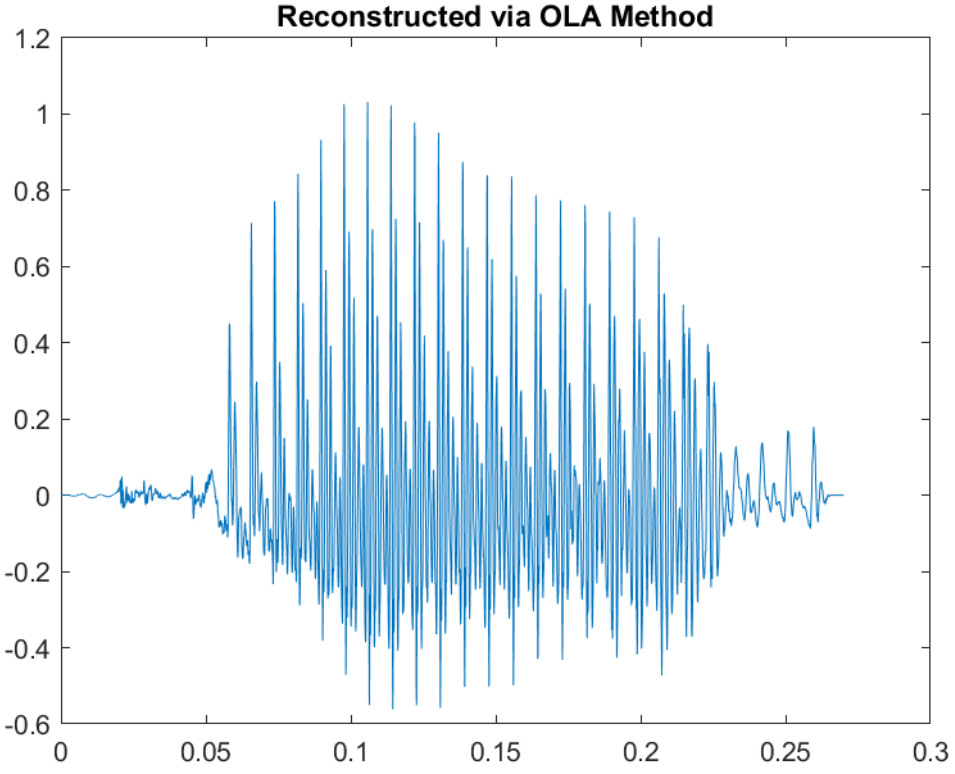
5. Repeat parts (1)-(4) with a speech utterance from your own voice recording. For part (4), consider manually marking voiced and unvoiced regions and invoke pitch synchronise during the voiced regions.

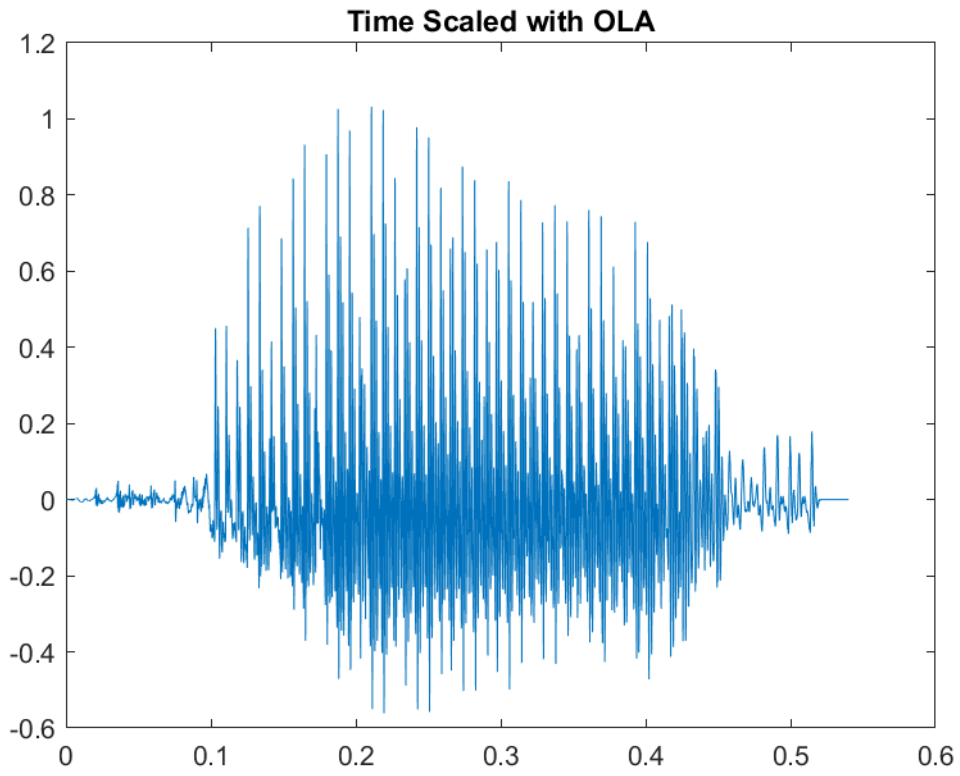
Solution.
OLA Method means Overlapping and Adding, simple as that. For re-synthesis of speech from framed signals, scaling and then OLA led to the reconstruction with spectral contents consistent enough. The original equation needed all frames' contribution to be considered for a particular sample, but we're assuming 0 value of a window outside it's frame, so the result.
While the time scaling was done by duplicating the frames and again applying OLA method. LSE Method for reconstruction wasn't different from OLA much because of the assumption of 0 valued window outside the frame. ■


Original Speech
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
0
0.05
0.1
0.15
0.2
0.25
0.3


Windowed Speech
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
0
0.1
0.2
0.3
0.4
0.5
0.6


FFT of Triangle Window
5
0
-5
-10
-15
-20
0
2000
4000
6000
8000
10000


Reconstructed via OLA Method
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
0
0.05
0.1
0.15
0.2
0.25
0.3



MATLAB Code

```
file = load('./Data_assignment_4/ex7M1.mat');
speech = file.speech2_10k;
Fs = 10000;
range = (1:length(speech))/Fs;
plot(range, speech);
title('Original Speech')

window = triang(0.03*Fs);
window = window';
shift = 0.015*Fs;

w_speech = zeros(1, length(window)*ceil(length(speech)/length(
    window)));
w_speech(1:length(speech)) = speech;
speech = w_speech;

num_shifts = ceil((length(speech) - length(window))/shift);
length(window) + num_shifts*shift;

num_windows = num_shifts+1;
windowed_speech = zeros(num_windows, length(window));
```

```
21 for i=0:num_windows-1
22     windowed_speech(i+1, :) = speech(i*shift+1:(i*shift)+length(
       window)).*window;
23 end
24
25 range_win = (1:length(window))/Fs;
26 figure;
27 hold on;
28 for i=1:num_windows
29     plot(range_win+(i-1)*0.03, windowed_speech(i, :));
30 %     title('Window ', i);
31 end
32 title('Windowed Speech')
33 hold off;
34
35 W = fft(window);
36 f_range = Fs*(1:length(window))/length(window);
37 plot(f_range, log10(abs(W)));
38 title('FFT of Triangle Window')
39
40 % Now reconstructing the original signal from the windowed signals
41 % Assuming 2700 samples again,
42 c_speech = zeros(1, length(speech));
43 for i=0:num_windows-1
44     c_speech(i*shift+1:(i*shift)+length(window)) = c_speech(i*
       shift+1:(i*shift)+length(window)) + windowed_speech(i+1, :)
        * shift/W(1);
45 end
46 plot((1:length(speech))/Fs, c_speech);
47 title('Reconstructed via OLA Method')
48
49 % Now reconstruction through LSE method
50 c_lse_speech = zeros(1, length(speech));
51 for i=0:num_windows-1
52     c_lse_speech(i*shift+1:(i*shift)+length(window)) =
       c_lse_speech(i*shift+1:(i*shift)+length(window)) +
       windowed_speech(i+1, :) / W(1);
53 end
54 plot((1:length(speech))/Fs, c_speech);
55
56 % Time expanding via OLA Method
57 n = 2;
58 c_double_speech = zeros(1, length(speech)*n);
59 for i=0:num_windows-1
60     for j=0:1
```

```
61      c_double_speech(((2*i+j)*shift+1) : (((2*i+j)*shift)+length(
            window))) = c_double_speech(((2*i+j)*shift+1) : (((2*i+j)*
            shift)+length(window))) + windowed_speech(i+1, :)*shift/W
            (1);
62      end
63  end
64  plot((1:length(speech)*n)/Fs, c_double_speech);
```

Problem 4

Programming Assignment: In this MATLAB exercise, use the workspace *ex7M2.mat*, as well as the function *uniform_bank.m* to explore the conditions for the recovery of a sequence using the filter bank summation (FBS) method.

1. There are four different analysis windows (or "analysis filters") in the workspace *ex7M2.mat filter1, filter2, filter3*, and *filter4*. Using MATLAB command subplot, plot all the filters. Note that *filter4* is simply a shifted version of *filter3*.

2. The function uniform_bank.m creates a filter bank $hk[n]$. The output of the function is a 2-D array of modulated filter impulse response, but without the demodulation term $e^{-j\frac{2\pi}{N}kn}$ (as shown in Figure 7.5 in book Discrete Time Speech Signal Processing by Thomas F. Quatieri). Also, the impulse responses are real because the complex conjugate response pairs have been combined. Note that the first and last filter do not correspond to complex conjugate pairs. Explain why.

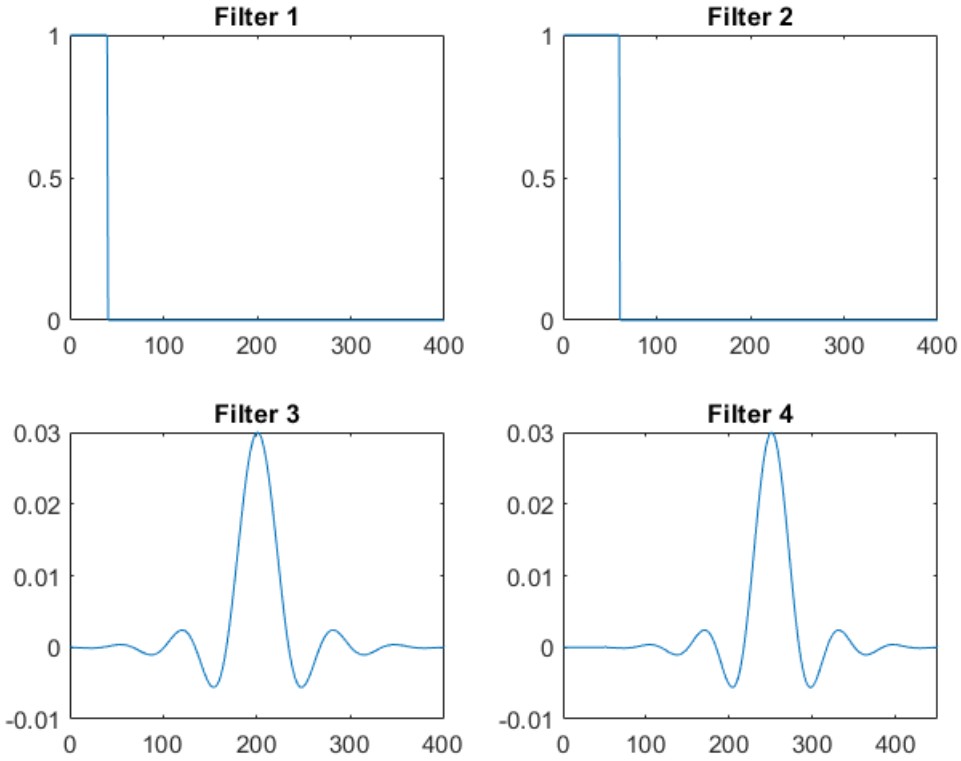
3. Do a help command on uniform_bank.m and run the function with a 250 Hz spacing between filters, the analysis filter filter1, and a plot factor of 100 (scaling the output for a good display). The function plots the frequency response magnitude of the filters using a 1024-point DFT. Assume the time sampling rate is 10000 samples/s so that the 512th frequency bin corresponds to 5000 Hz. Having the impulse responses hk[n] of your filter bank, write a MATLAB function to filter the unit sample impulse in *ex7M2.mat* with your filter bank, and then combine all impulse responses to create the composite (summed) filter bank impulse response. Explain the observed composite impulse response using the FBS constraint. Finally, plot the impulse response to the second and the fifteenth bandpass filter and explain the difference in time structure of the two signals [recall that the filter bank is missing the STFT demodulation term $e^{-j\frac{2\pi}{N}kn}$ (Figure 7.5 in book Discrete Time Speech Signal Processing by Thomas F. Quatieri)]

4. Repeat part (3) with analysis filter *filter2*. Using the FBS constraint, explain the reason for the deviation in the composite response from an impulse.

5. Repeat part (3) with *filter3* (using a plot factor of 2). Superimpose the composite impulse response on *filter3* (plotting the first 400 samples) and again explain your observation using the FBS constraint.

6. Repeat part (3) with *filter4*, which is *filter3* shifted by fifty samples to the right. Superimpose the composite impulse response on filter4 (plotting the first 450 samples) and again explain your observation using the FBS constraint. Also comment on the difficulty in creating an impulsive composite response by simply shifting the analysis filter.

Solution.
There is no function called 'uniform_bank' in MATLAB or any of its plugins. ■



MATLAB Code

```
file = load('./Data_assignment_4/ex7M2.mat')
f1 = file.filter1;
f2 = file.filter2;
f3 = file.filter3;
f4 = file.filter4;
impulse = file.impulse;

subplot(2, 2, 1);
plot(f1);
title('Filter 1')
subplot(2, 2, 2);
plot(f2);
title('Filter 2')
subplot(2, 2, 3);
plot(f3);
title('Filter 3')
subplot(2, 2, 4);
plot(f4);
title('Filter 4')
```