Multirate Processing

87203 – Multimedial Signal Processing 1st Module

Politecnico di Milano –
Polo regionale di Como

Agenda

- Ex1: Downsampling
- Ex2: Upsampling
- Ex3: Decimation
- Ex4: Interpolation
- Ex5: Frequency modification with rational factor
- Ex6: Polyphase Decimation
- Ex7: Polyphase Interpolation
- Ex8: Polyphase Filter Banks
- Ex9: Quadrature Mirror Filters

Exercise 1 (1/5)

- Goal: Downsample by factor M the input sequence x

$$x(m) \longrightarrow \boxed{\downarrow M} \longrightarrow y(n)$$

- Keep every Mth sample and discard the rest:

$$y(n) = x(nM) \qquad\qquad Y(z) = \frac{1}{M}\sum_{p=0}^{M-1} X\left(e^{-j\frac{2\pi p}{M}} z^{\frac{1}{M}}\right)$$

Exercise 1 (2/5)

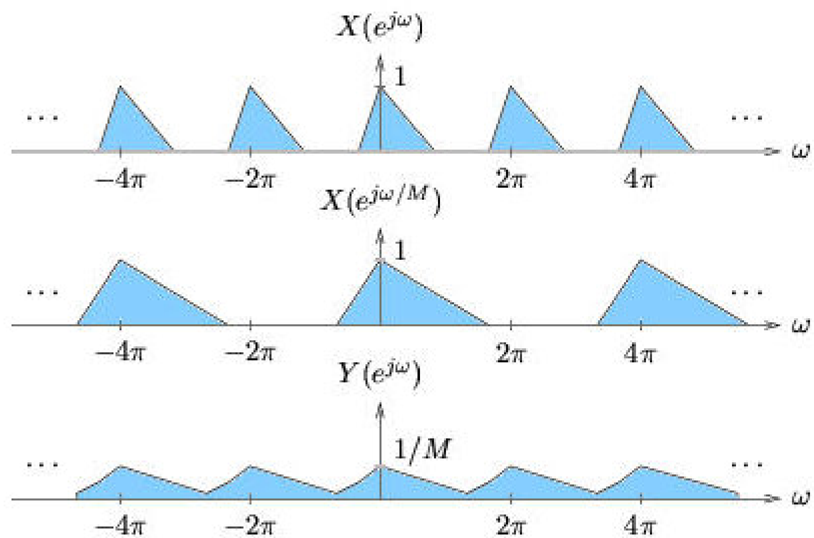
- pseudocode
  - built the input signal
    - x=bartlett(N); % triangular signal
  - downsample: keep every M^{th} sample and discard the rest
    - x_down=x(1:M:end);   %length(x_down)=length(x)/M
  - compare the output and input spectra: see the effect of downsampling
    - Nfft=1024;
    - X=fft(x,Nfft);
    - X_down=fft(x_down,Nfft);

Exercise 1 (3/5)

- HINT: Downsampling expands each 2π-periodic repetition of X(ω) by a factor of M along the ω axis, and reduces the gain by a factor of M.

(see figure 3 and 4 of ex1)

- Tdown = T*M
- Fs_down=Fs/M



Exercise 1 (4/5)

- HINT: If x(m) is not bandlimited to π /M , aliasing may result from spectral overlap.
- pseudocode
  - build the signal
    - w1=pi/16; w2=pi/8; w3=pi/2;
    - n=[0:200];
    - x = cos(w1*n)+0.5*cos(w2*n)+2*cos(w3*n);
  - downsample (M=4)
    - x_down=x(1:M:end); %length(x_down)=length(x)/M
  - compare the output and input spectra: see the alias

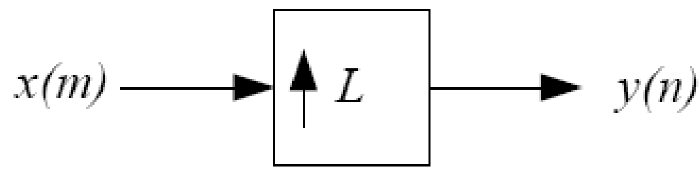
Exercise 1 (5/5)

- N.B.:
  - w1=pi/16 = 0.1963
  - w2=pi/8 = 0.3927
  - w3=pi/2 = 1.5708

- downsample (M=4)
  - w1-->pi/4 = 0.7854
  - w2-->pi/2 = 1.5708
  - w3-->2*pi = 6.2832   ALIAS !!!

Exercise 2 (1/3)

- Goal: Upsample by factor L the input sequence x



- Insert L – 1 zeros between every sample of the input signal:

$$Y(z) = X(z^L)$$

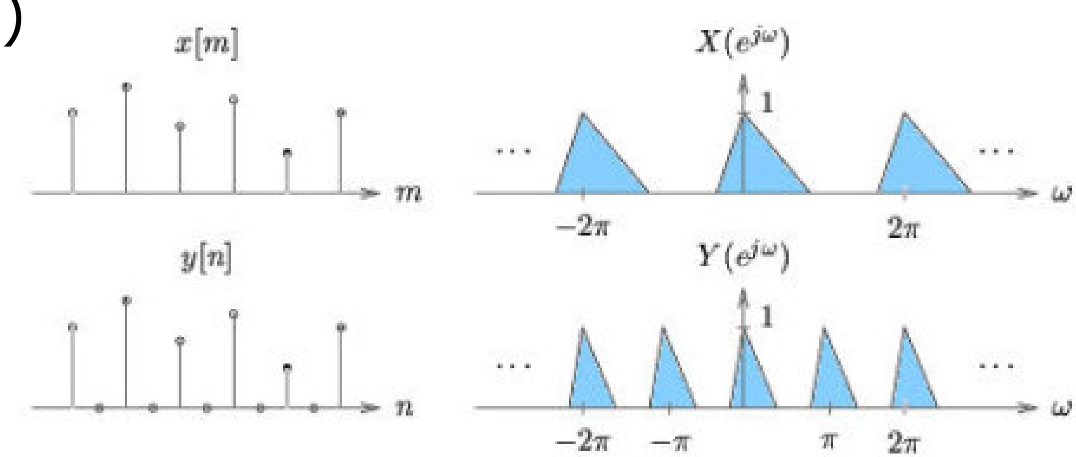
Exercise 2 (2/3)

- pseudocode
  - built the input signal
    - x=bartlett(N); % triangular signal
  - upsample: insert L−1 zeros between every sample of the input signal
    - x_up = zeros(2*N, 1);
    - x_up(1:2:end) = x; %length(x_up)=L*length(x)
  - compare the output and input spectra: see the effect of downsampling
    - Nfft=1024;
    - X=fft(x,Nfft);
    - X_up=fft(x_up,Nfft);

Exercise 2 (3/3)

- HINT: Upsampling compresses the DFT by a factor L along the ω axis.

(see figure 4 of ex2)



  - Tup = T/L
  - Fs_up=L*Fs

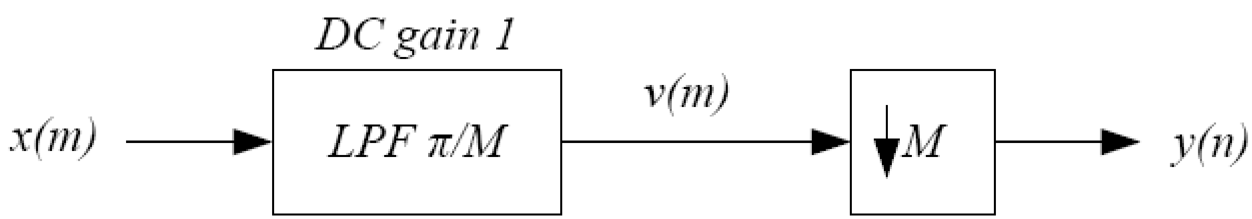
- HINT: Cut the frequencies f such that |f|>π /L , if you don’t want to see the repetition of the original signal.

Agenda

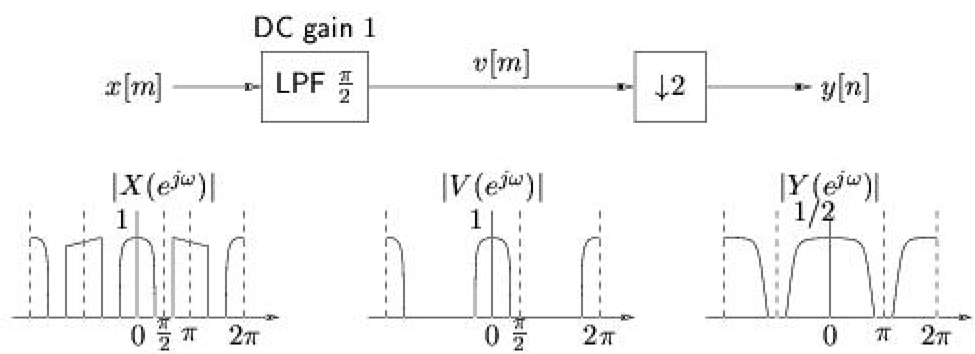
- Ex1: Downsampling
- Ex2: Upsampling
- Ex3: Decimation
- Ex4: Interpolation
- Ex5: Frequency modification with rational factor
- Ex6: Polyphase Decimation
- Ex7: Polyphase Interpolation
- Ex8: Polyphase Filter Banks
- Ex9: Quadrature Mirror Filters

Exercise 3 (1/14)

- Goal: Decimate by a factor M the input sequence x



- Decimation=
  - filtering(to prevent aliasing)
  - downsampling



Exercise 3 (2/14)

- Goal: decimate a speech signal by a factor M=2. Perform filtering:
  - In the time domain
  - In the frequency domain
  - In the frequency domain using OLA
- Compare the output and input spectra

- Data:
  - Input signal: ‘Toms_diner.wav’
  - Original sampling frequency: Fs = 16 kHz
  - As low pass filter you can load the file h_filter.mat

Exercise 3 (3/14)

- Perform filtering:
  - In the time domain:
    - N_x = length(x);
    - N_h = length(h);

    - y_td = filter(h, 1, x);
      % length(y_td)=N_x
    - xfilt = conv(h,x);
      % length(xfilt)=Nx+Nh-1

Exercise 3 (4/14)

- Perform filtering:
  - In the frequency domain:
    - N_x = length(x);
    - N_h = length(h);
    - N = 2^ceil(log2(N_x + N_h - 1));

    % It establishes the closer power of two for FFT transformation
    - X = fft(x, N);
    - H = fft(h, N);
    - y_fd = ifft(X.*H, N);

Exercise 3 (5/14)

- Perform filtering:
  - In the frequency domain using OLA:
    - M=1000;   % Window size
    - x=x(1:floor(length(x)/M)*M);

    % Reduce the data size to a multiple of the window size
    - K=length(x)/M; % Number of frames (no overlap)
    - N = 2^ceil(log2(N_x + N_h - 1)); > M+Nh-1

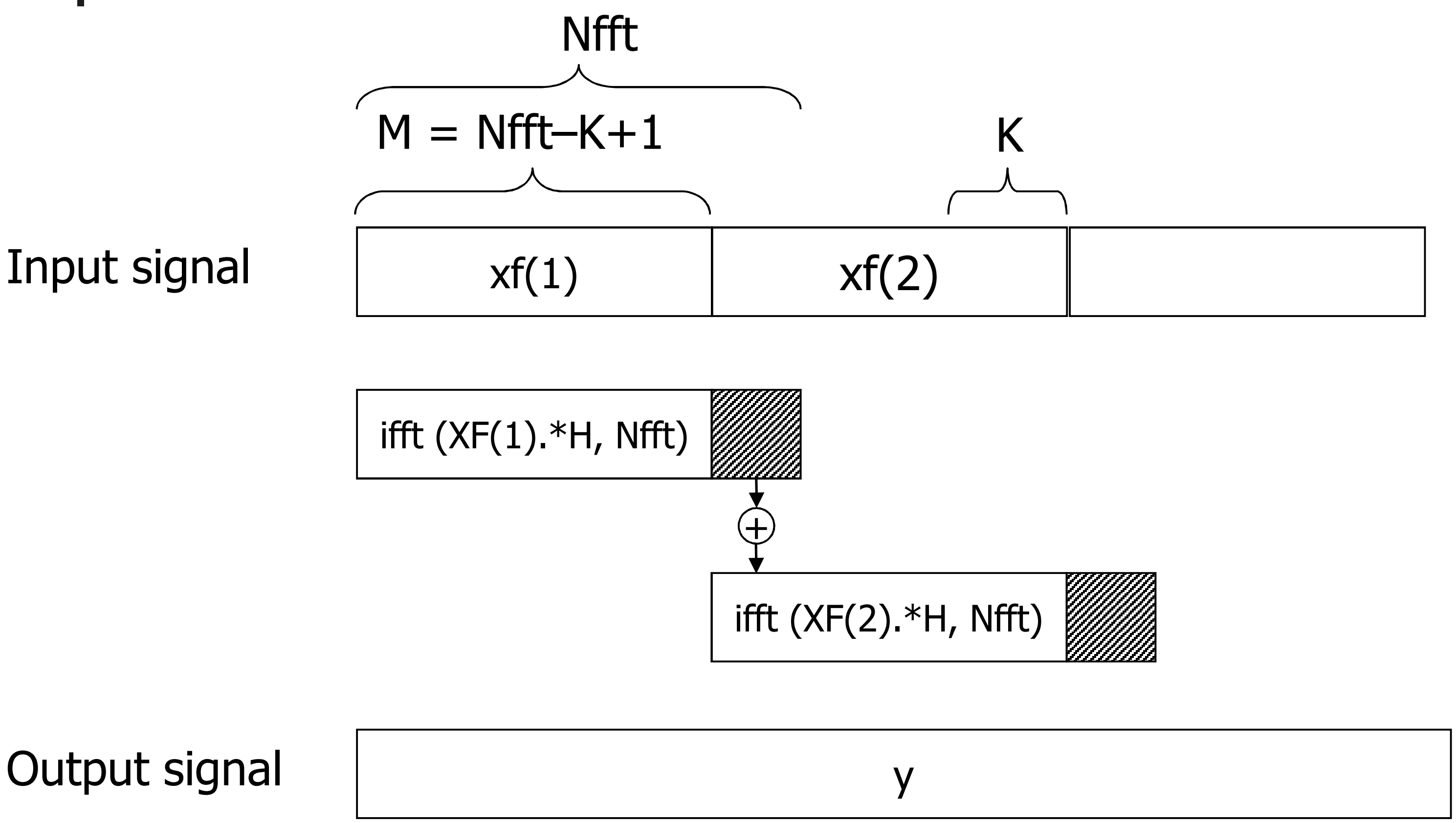
    % It establishes the closer power of two for FFT transformation

    % N=length of the output of each block
    - y_ola = zeros(length(x)+N-1, 1);

Exercise 3 (6/14)

- Perform filtering:
  - In the frequency domain using OLA (continued):
    - Hm = fft(h, N);
    - for k=0:K-1
    - xm = x(k*M+1:k*M+M);
      %windowing (rect, no overlap)
    - Xm = fft(xm, N);
    - ym = ifft(Xm.*Hm, N);
    - y_ola(k*M +1:k*M+N)=y_ola(k*M+1:k*M+N)+ym;
    - end;

Exercise 3 (7/14) Scheme of OLA for R=M:



Exercise 3 (8/14)

- Goal: decimate a signal by a factor M=4.
- Compare the output and input signal in time domain and in frequency domain
- Compare the decimated and the downsampled signals spectra
- See also “d=decimate(x,M);”
- Try to swap the lowpass filter and the downsample command. Can you notice any difference? Why?

Exercise 3 (9/14)

- Decimate a signal by a factor M=4.

- Pseudocode:
  - Build the signal
    - w1=pi/16; %=0.1963 <pi/4
    - w2=pi/8;   % =0.3927<pi/4
    - w3=pi/2;   %=0.5708 >pi/4
    - n=[0:200];
    - x = cos(w1*n)+0.5*cos(w2*n)+2*cos(w3*n);

      $= x_l + x_h$

Exercise 3 (10/14)

- Decimate a signal by a factor M=4.
- Pseudocode:
  - Build the filter
    - N=41;
    - fc=1/(2*M) % =1/8 --> wc = pi/M = 0.7854
    - h = fir1(N,2*fc)
  - Perform filtering
    - xfilt = conv(h,x);
  - Downsampling
    - xdec = xfilt(1:M:end);

Exercise 3 (11/14)

- Matlab provides the function:

  "h_fir = fir1(N, Fc)" that designs an N-th order FIR filter that has cut-off frequency Fc.
  - The cut-off frequency "Fc" must be between 0<Fc< 1.0, with 1.0 corresponding to half the sample rate.
  - The filter is real and has linear phase.
  - The normalized gain of the filter at Fc is -6 dB.

Exercise 3 (12/14)

- Compare the output and input signals in time domain and in frequency domain
  - length(x_dec) = (length(x)+length(h))/M
  - w1-->pi/4 = 0.7854
  - w2-->pi/2 = 1.5708
  - w3-->2*pi = 6.2832 filtered by h
- Compare the decimated and the downsampled signals in time domain and in frequency domain
  - length(x_dec)> length(x_down) = length(x)/M
  - With the downsampled signal spectra we can see the alias

Exercise 3 (13/14)

- See also “d=decimate(x,M);”
  - Matlab provides the function “d = decimate(x,M)” that resamples the sequence in vector x at 1/M times the original sample rate. The resulting resampled vector d is M times shorter (length(d)=length(x)/M).
  - Before resampling, “decimate” filters the data with an eighth order Chebyshev Type I lowpass filter with cutoff frequency:

$$fc = 0.8\frac{Fs}{2M}$$

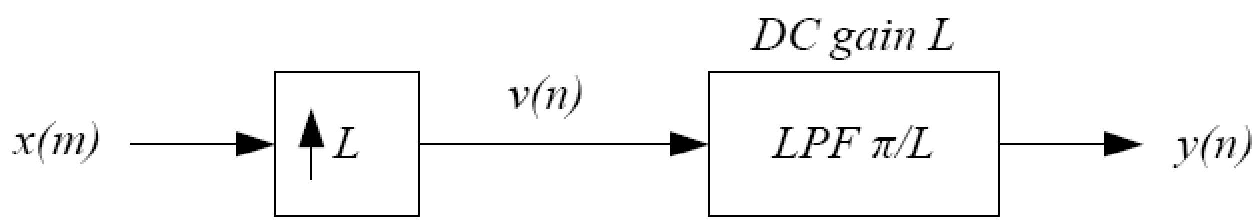
  - N.B. The filter used by “decimate” is better than ours.

Exercise 3 (14/14)

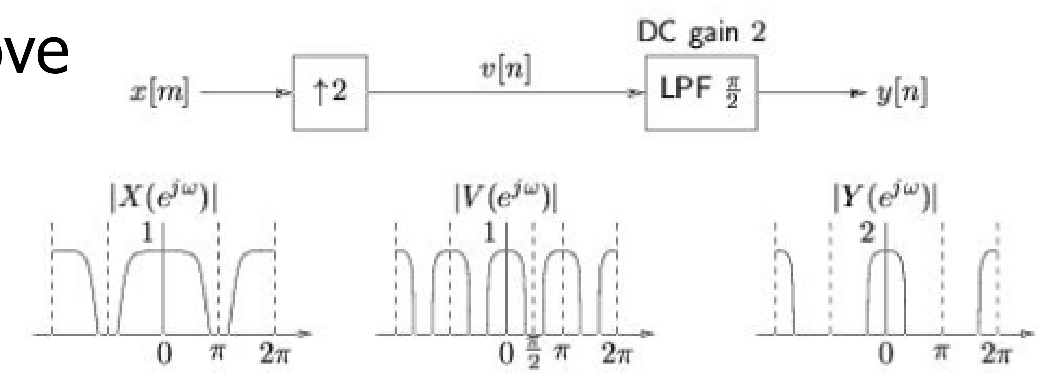
- Try to swap the lowpass filter and the downsample command. Can you notice any difference? Why?
  - CASE 1:
    - `xdown=x(1:M:end);`
    - `hdown=h(1:M:end);`
    - `xdf=conv(xdown,hdown);`
  - CASE2:
    - `xdown=x(1:M:end);`
    - `xdf=conv(xdown,h);`
  - SOLUTION: Polyphase filters

Exercise 4 (1/11)

- Goal: Interpolate by a factor L the input sequence x



- Decimation=
  - upsampling
  - filtering (to remove repetitions of original spectra)



Exercise 4 (2/11)

- Goal: interpolate a signal by a factor M=4.
- Compare the output and input signal in time domain and in frequency domain
- Compare the decimated and the upsampled signals spectra
- See also “d=interp(x,M);”
- Try to swap the upsample command and the lowpass filter. Can you notice any difference? Why?

Exercise 4 (3/11)

- Interpolate a signal by a factor M=4.

- Pseudocode:
  - Build the signal
    - w1=pi/16; %=0.1963 <pi/4
    - w2=pi/8;   % =0.3927<pi/4
    - w3=pi/2;   %=0.5708 >pi/4
    - n=[0:200];
    - x = cos(w1*n)+0.5*cos(w2*n)+2*cos(w3*n);

      $= x_l + x_h$

Exercise 4 (4/11)

- Pseudocode (continued):
  - Build the filter
    - N=41;
    - fc=1/(2*M) % =1/8 --> wc = pi/M = 0.7854
    - h = fir1(N,2*fc)
  - Upsampling
    - xup = zeros(M*length(x), 1);
    - xup(1:M:end) = x ;
  - Perform filtering
    - xint = filter(M*h, 1, xup);

Exercise 4 (5/11)

- Compare the output and input signals in time domain and in frequency domain
  - length(x_int) = M*length(x)+length(h)
  - w1-->pi/(4*16) = 0.0491
  - w2-->pi/(4*8) = 0.0982
  - w3-->pi/(4*2) = 0.3927
- Compare the interpolated and the upsampled signals in time domain and in frequency domain
  - In x_up we find the M-1 zeros between each two samples of the original signal
  - In the upsampled signal spectra we can see the repetitions of the original spectra

Exercise 4 (6/11)

- See also "int= interp(x,M);"
  - Matlab provides the function "int = interp(x,M)" that resamples data at a higher rate using lowpass interpolation. "int=interp(x,M)" resamples the sequence in vector x at M times the original sample rate. The resulting resampled vector int is M times longer (length(int) = M*length(x)).
  - A symmetric filter, B, allows the original data to pass through unchanged and interpolates between so that the mean square error between them and their ideal values is minimized.

  - N.B. The spectra are near to be coincident.

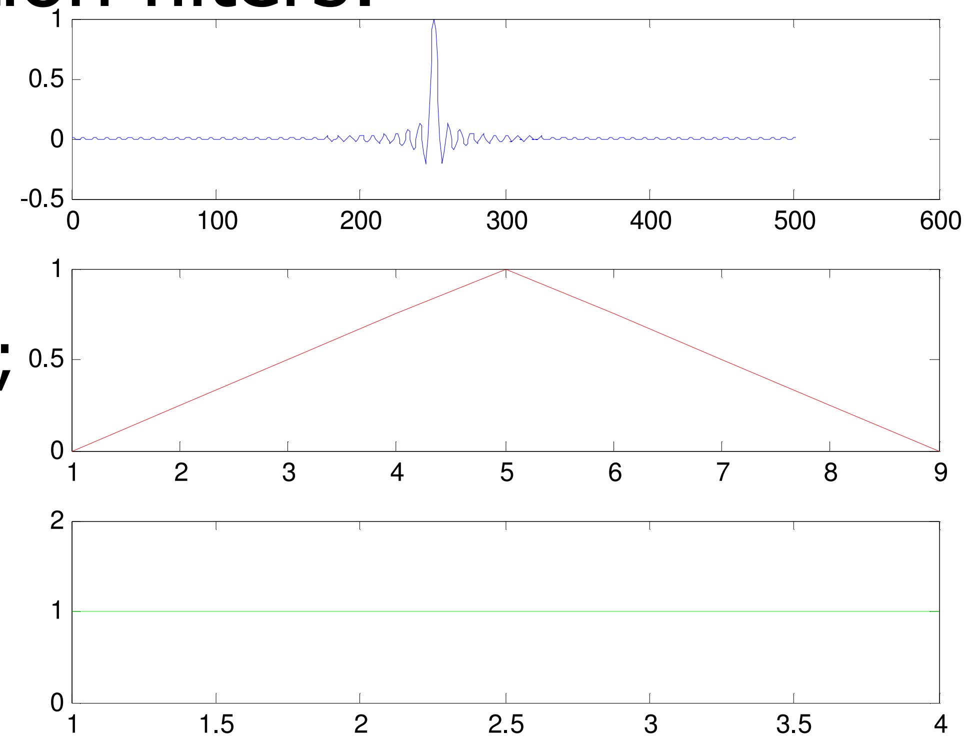
Exercise 4 (7/11)

- Try to swap the upsample command and the lowpass filter. Can you notice any difference? Why?
  - CASE 1:
    - hd=h(1:M:end); % it become an all pass filter
    - xf = filter(M*hd, 1, x);
    - xfup = zeros(M*length(xf), 1);
    - xfup(1:M:end) = xf ;   % it is x_up
  - CASE2:
    - xf = filter(M*h, 1, x);
    - xfup = zeros(M*length(xf), 1);
    - xfup(1:M:end) = xf ;   % it is x_l_up

  - SOLUTION: Polyphase filters

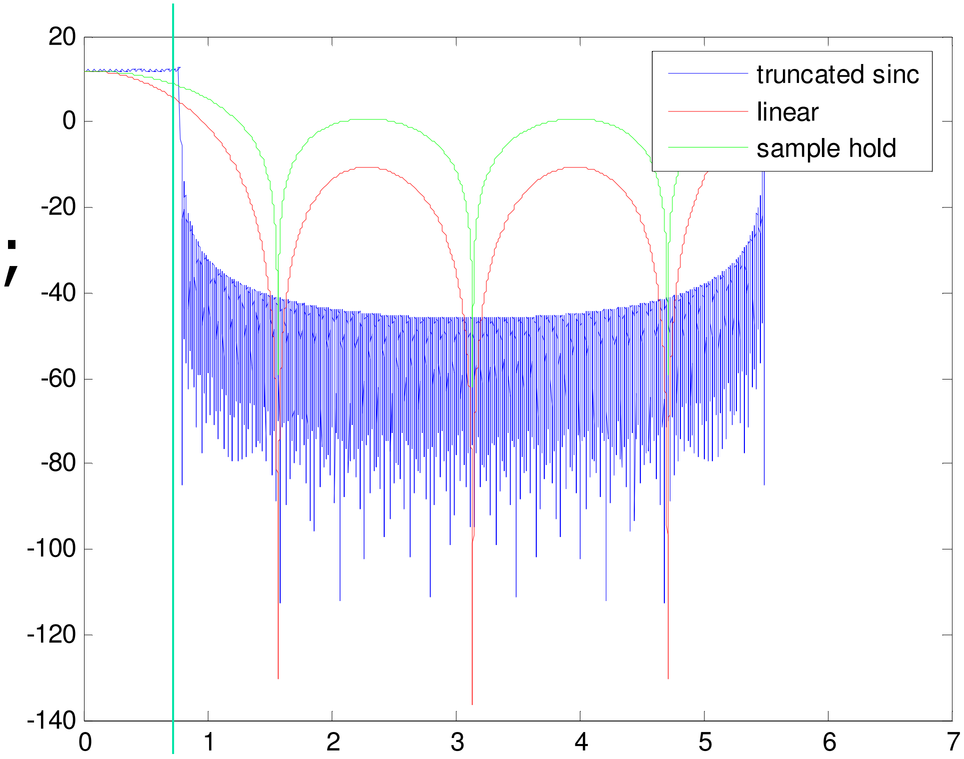
Exercise 4 (8/11)

- Hint: Let's see the effects of three different interpolation filters:
  - h1 = sinc(n/M)';
  - h2 = bartlett(2*M+1);
  - h3 = ones(M,1);



Exercise 4 (9/11)

- Hint: Let's see the effects of three different interpolation filters:
  - h1 = sinc(n/M)';
  - h2 = bartlett(2*M+1);
  - h3 = ones(M,1);
  - wc=pi/4=0.7854;

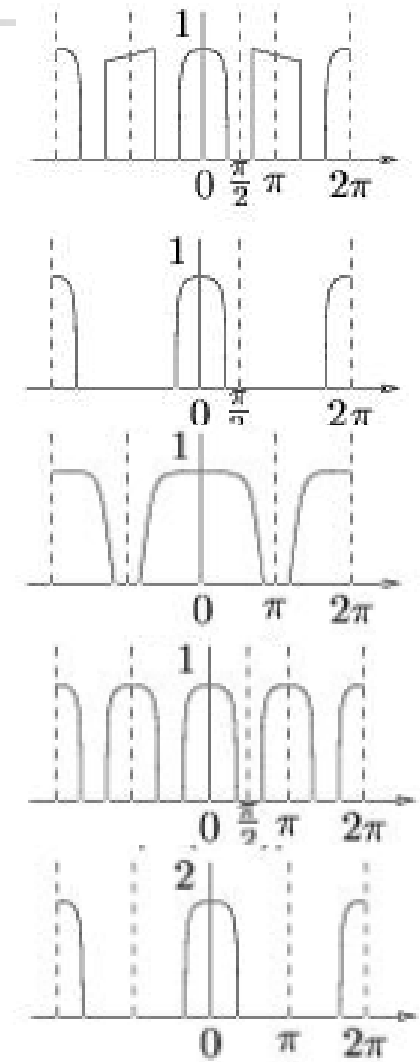


Exercise 4 (10/11)

- Hint: See what happens when we perform a decimation and interpolation by the same factor M:
  - Pseudocode:
    - Build the input signal
    - Build the filter (wc=pi/M)
    - Apply the filter
    - downsample
    - upsample
    - Apply the filter

Exercise 4 (11/11)

- Build the input signal
- Apply the filter (wc=pi/M)
- downsample (gain=1/2)
- upsample
- Apply the filter (gain=2)

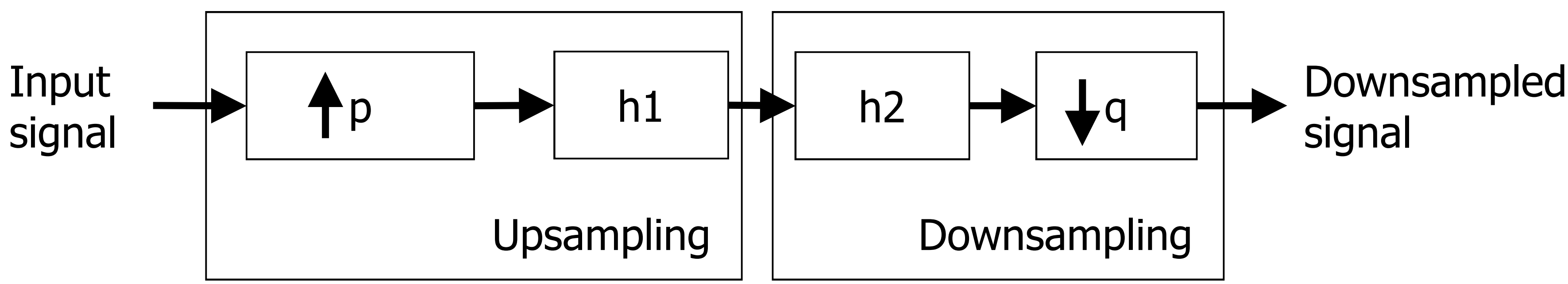


Agenda

- Ex1: Downsampling
- Ex2: Upsampling
- Ex3: Decimation
- Ex4: Interpolation
- **Ex5: Frequency modification with rational factor**
- Ex6: Polyphase Decimation
- Ex7: Polyphase Interpolation
- Ex8: Polyphase Filter Banks
- Ex9: Quadrature Mirror Filters

Exercise 5 (1/5)

- **Goal**: decimate a speech signal from the sample frequency 16 kHz to 10 kHz
  - **Problem**: the input and the target frequencies form a rationale ratio Fout/Fs=r = p/q
  - Therefore, you have upsample the original signal with factor p and then downsample it with a factor q:



Exercise 5 (2/5)

- **Hints**: Pay attention to the cut-off frequencies of the two low-pass filters:
  - f1 = Fs/(2*p);

    % Cut-off frequency of the first low-pass filter
  - f2 = Fs_new/(2*q) = (5*Fs)/(2*q) ;

  % Cut-off frequency of the second low-pass filter. It is now referred to the new sampling frequency (fs*upsample_factor)

  - h1 = fir1(30, 1/p);   (pi/5=0.6283)
  - h2 = fir1(30, 1/q);   (pi/8=0.3927)
  - N.B.: The cut-off frequency must be between 0 and 1, with 1 corresponding to half the sample rate.

Exercise 5 (3/5)

- Pseudocode:
  - Build the input signal
    - w1=pi/16;
    - w2=pi/8;
    - w3=pi/2;
    - n=[0:200];
    - x = cos(w1*n)+0.5*cos(w2*n)+2*cos(w3*n);
  - Build the filters
    - h1 = fir1(30, 1/p);   (pi/5=0.6283)
    - h2 = fir1(30, 1/q);   (pi/8=0.3927)

Exercise 5 (4/5)

- Pseudocode(continued):
  - upsample
    - y_up = zeros(length(x)*upsample_factor, 1);
    - y_up(1:upsample_factor:end) = x;
  - Apply the first filter
    - y_up = filter(upsample_factor*h1, 1, y_up);
  - Apply the second filter
    - y_down = filter(h2, 1, y_up);
  - downsample
    - y_down = y_down(1:downsample_factor:end);

Exercise 5 (5/5)

- Fout = (Fs*5/8):
  - w1=pi/16; --> (pi*8)/(16*5) = 0.3141
  - w2=pi/8 ; --> (pi*8)/(8*5) = 0.6283
  - w3=pi/2 ; --> (pi*8)/(2*5) = 2.5133

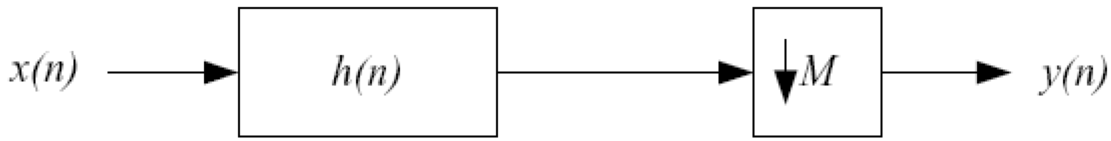
  - without alias or any repetition of the original spectra

Agenda

- Ex1: Downsampling
- Ex2: Upsampling
- Ex3: Decimation
- Ex4: Interpolation
- Ex5: Frequency modification with rational factor
- **Ex6: Polyphase Decimation**
- **Ex7: Polyphase Interpolation**
- Ex8: Polyphase Filter Banks
- Ex9: Quadrature Mirror Filters

Exercise 6 (1/9)

- Goal: Decimate by a factor M the input sequence x



- The direct implementation shown in ex3 is not efficient: the result of several operations is discarded when the output of filtering is decimated.
- Polyphase filters are needed every time we want to achieve computational savings in the decimation phase.

Exercise 6 (2/9)

- Basic ideas:
  - Decompose the filter into its polyphase components
  - Decompose the input signal in subsequences
  - Apply each subfilter to the corrispondent subsequence.
  - Sum the results to obtain the output signal

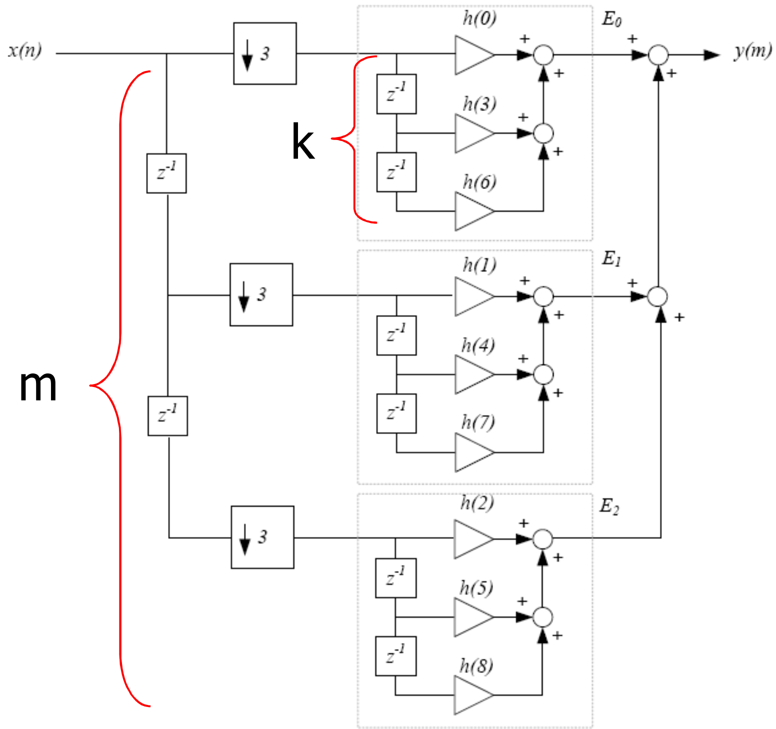
Exercise 6 (3/9)

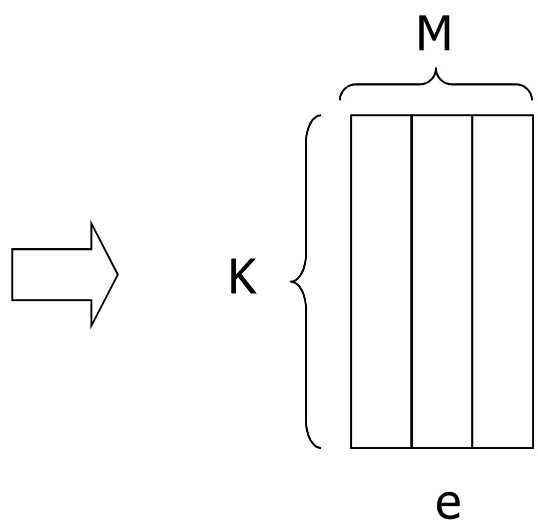
- Notation:
  - M downsampling factor
  - L filter length
  - K = ceil(L/M), number of sample for each subfilter
  - L = K*M
- Decomposition of the lowpass filter is accomplished in the following way:

$$e_m(k) = h(kM + m) \quad \text{for} \begin{cases} k = 0,1,\ldots,K \\ m = 0,1,..,M \end{cases}$$

Exercise 6 (4/9)

Decomposition schematization for M = 3 and L = 9





Exercise 6 (5/9)

- $$H(z) = \sum_{m=0}^{M-1} z^{-m} E_m(z^M)$$ it works at MT respect to the original filter
- $$E_m(z) = \sum_{k=0}^{K-1} e_m(k)\, z^{-k}$$
- $$e_m(k) = h(kM + m) \quad \text{for} \begin{cases} k = 0,1,\ldots,K \\ m = 0,1,..,M \end{cases}$$

delayed by m-samples

Exercise 6 (6/9)

- Goal:
  - Decimate a signal from 16kHz to 8kHz
  - Use polyphase implementation of decimation
  - As low pass filter use the h_filter file
- Input:
  - sum of two sinusoids at frequencies 1000 Hz and 4500 Hz
    - f1=1000/16000 --> w1=2*pi*f1=0.3972
    - f2=4500/16000 --> w2=2*pi*f2=1.767

Exercise 6 (7/9)

- Decompose the filter into its polyphase components

```
function e = decompose(h, M)

N = length(h);
K = ceil(N/M);    %Number of samples in each subfilter
L = M*K;
h=[h;zeros(L-N,1)];    % zero pad
e = zeros(K,M);

for k = 0:K-1         % For each sample
    for  m=0:M-1          % For each subfilter
        e(k+1,m+1)=h(k*M+m+1);
      %KxM matrix. Each column is a subfilter
    end;
end;
```

Exercise 6 (8/9)

```
for m  = 1:M
```

- Decompose the input signal in subsequences

```
        x_m = x(m:M:end);
        % shift and downsample
```

- Apply each subfilter to the corrispondent subsequence.

```
        xf_m = conv(x_m, e(:,m));
```

- Sum the results to obtain the output signal

```
       y = y + xf_m;   % length(y)=(Nx+L-1)/M
```

```
end
```

Exercise 6 (9/9)

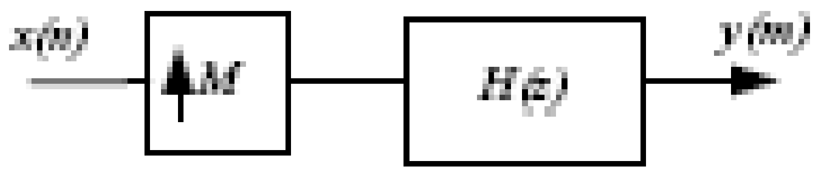
- Compare the output and input signals spectra
  - w1= 0.3972 --> w1*M= 0.7944
  - w2 = 1.767 >pi/2 --> w2*M = 3.534 (alias!!)

    (w2 is stopped by the lowpass filter)

- Compare the output obtained by means polyphase and direct implementations of decimation

Exercise 7 (1/8)

- Goal: Interpolate by a factor M the input sequence x



- The direct implementation shown in ex4 is not efficient.
- Polyphase filters are needed every time we want to achieve computational efficiency in the interpolation phase.

Exercise 7 (2/8)

- Basic ideas:
  - Decompose the filter into its polyphase components
  - Apply each subfilter to the input signal.
  - Upsample each filtered subsequence.
  - Sum the results to obtain the output signal

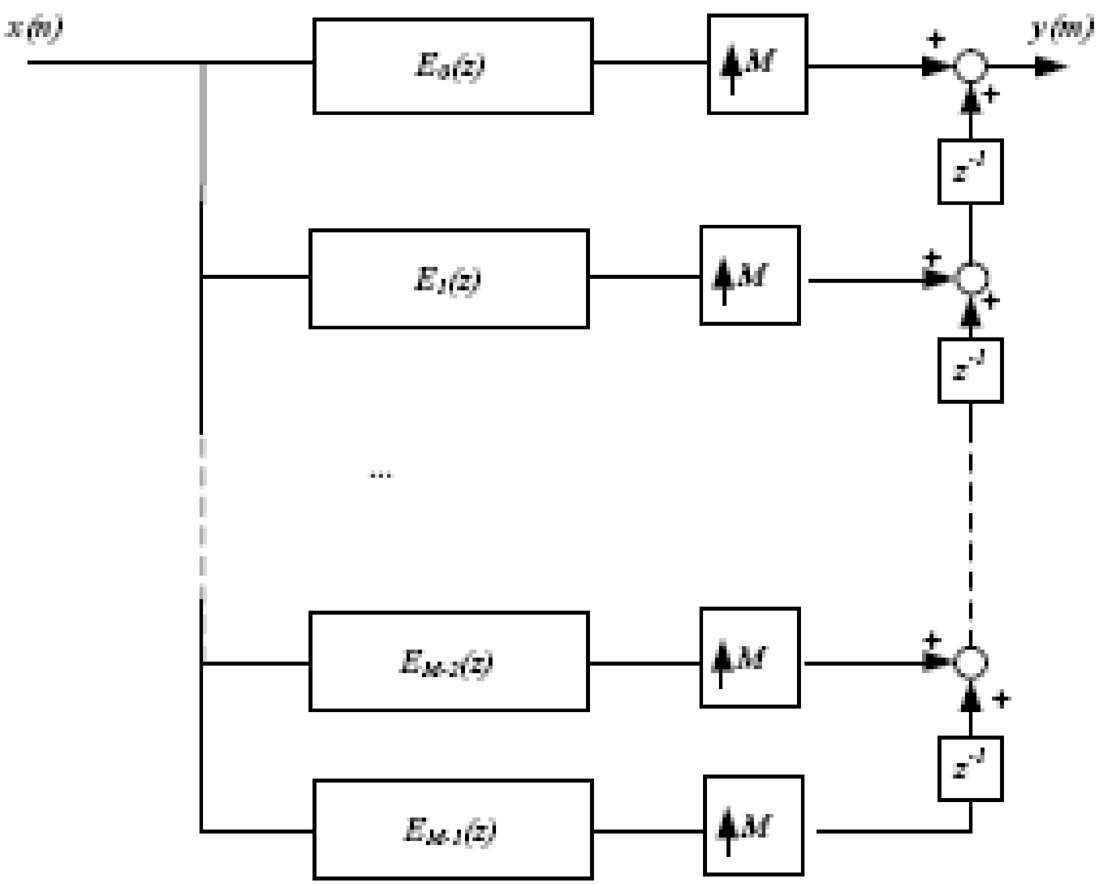
Exercise 7 (3/8)

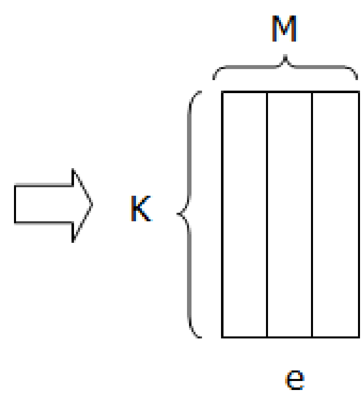
- Notation:
  - M upsampling factor
  - L filter length
  - K = ceil(L/M), number of sample for each subfilter
  - L = K*M
- Decomposition of the lowpass filter is exact the same as for decimation:

$$e_m(k) = h(kM + \underset{\substack{\downarrow \\ \text{delayed by m-samples}}}{m}) \quad \text{for} \quad \begin{cases} k = 0,1,\ldots, K \\ m = 0,1,.., M \end{cases}$$

Exercise 7 (4/8)

Interpolation schematization:





$$E_m(z) = \sum_{k=0}^{K-1} e_m(k)\ z^{-k}$$

Exercise 7 (5/8)

- Goal:
  - Interpolate a signal from 16kHz to 32kHz
  - Use polyphase implementation of interpolation
  - As low pass filter use the h_filter file
- Input:
  - sum of two sinusoids at frequencies 1000 Hz and 4500 Hz
    - f1=1000/16000 --> w1=2*pi*f1=0.3972
    - f2=4500/16000 --> w2=2*pi*f2=1.767

Exercise 7 (6/8)

- Decompose the filter into its polyphase components

```
function e = decompose(h, M)

N = length(h);
K = ceil(N/M);    %Number of samples in each subfilter
L = M*K;
h=[h;zeros(L-N,1)];    % zero pad
e = zeros(K,M);

for k = 0:K-1        % For each sample
    for  m=0:M-1          % For each subfilter
        e(k+1,m+1)=h(k*M+m+1);
      %KxM matrix. Each column is a subfilter
    end;
end;
```

Exercise 7 (7/8)

```
for m  = 1:M
```

- Apply each subfilter to the input signal

```
    xf(m,:) = conv(x, M*e(:,m));
```

- Upsample and shift each filtered signal

```
    xup(m,:) = zeros((M*length(xf))+M-1, 1);
    xup(m,m:M:end-(M-1)) = xf(m,:);
```

- Sum the results to obtain the output signal

```
     y = y + xup(m,:);   % length(y)=M*(Nx+K-1)
```

```
end
```

Exercise 7 (8/8)

- Compare the output and input signals spectra
  - w1= 0.3972 --> w1/M= 0.1986
  - w2 = 1.767 >pi/2 --> w2/M = 0.8835

  (The original spectra repetitions [pi-0.1986=2.943 and pi-0.8835=2.3066] are cutted by the lowpass filter)

- Compare the output obtained by means polyphase and direct implementations of interpolation

Agenda

- Ex1: Downsampling
- Ex2: Upsampling
- Ex3: Decimation
- Ex4: Interpolation
- Ex5: Frequency modification with rational factor
- Ex6: Polyphase Decimation
- Ex7: Polyphase Interpolation
- **Ex8: Polyphase Filter Banks**
- Ex9: Quadrature Mirror Filters

Exercise 8 (1/10)

- A filter bank is a set of parallel filters that enables to decompose the input signal into a number of subbands.

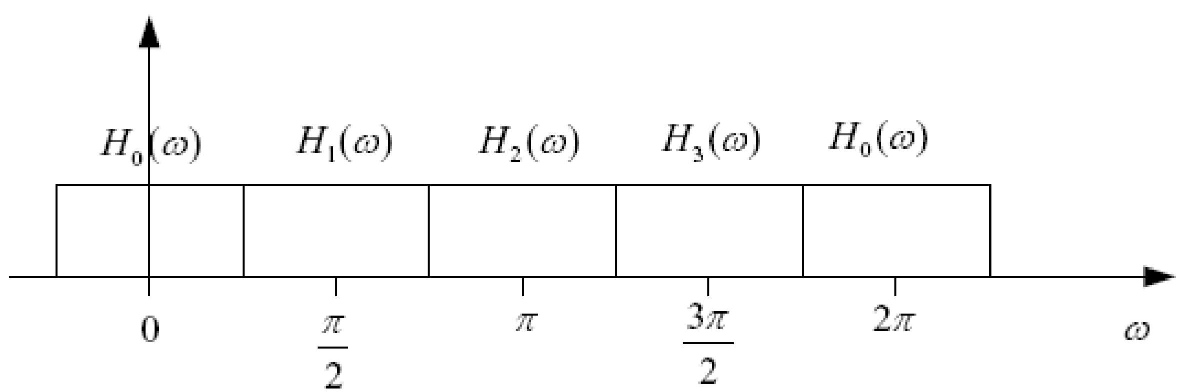


Fig. 5.19. Frequency response of a uniform filter bank with $M = 4$ subbands

Exercise 8 (2/10)

- $H_0(z) = \sum_{k=0}^{M-1} z^{-k} E_k(z^M)$ it works at MT respect to the original filter

- $E_m(z) = \sum_{k=0}^{K-1} e_m(k)\, z^{-k}$

- $e_m(k) = h(kM + m) \quad \text{for} \begin{cases} k = 0,1,\ldots,K \\ m = 0,1,..,M \end{cases}$

$$\underset{\text{delayed by m-samples}}{\downarrow}$$

delayed by m-samples

Exercise 8 (3/10)

- We can demonstrate that:

$$H_m(z) = H_0\left(z\,e^{j\frac{2\pi m}{M}}\right) = \sum_{k=0}^{M-1} z^{-k} W^{mk} E_k(z^M)$$

$$\begin{bmatrix} H_0(z) \\ H_1(z) \\ . \\ H_{M-1}(z) \end{bmatrix} = \begin{bmatrix} W^{0,0} & W^{0,1} & . & W^{0,M-1} \\ W^{1,0} & W^{1,1} & . & W^{1,M-1} \\ . & . & & . \\ W^{M-1,0} & W^{M-1,1} & . & W^{M-1,M-1} \end{bmatrix} \begin{bmatrix} E_o(z^M) \\ z^{-1}E_1(z^M) \\ . \\ z^{-(M-1)}E_{M-1}(z^M) \end{bmatrix}$$

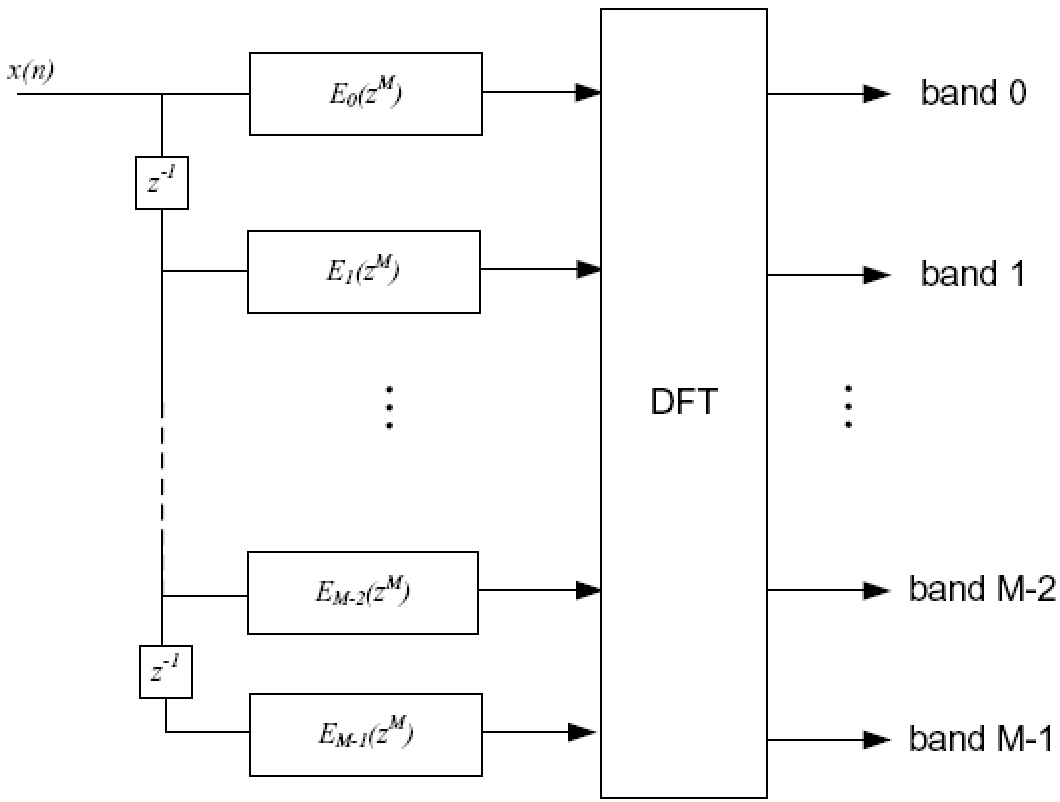
Exercise 8 (4/10)

- N.B.: $W^{m,k}$ is the DFT matrix

$$W^{m,k} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & e^{-j\frac{2\pi}{M}} & e^{-j2\frac{2\pi}{M}} & \dots & e^{-j(M-1)\frac{2\pi}{M}} \\ . & . & . & & . \\ 1 & e^{-j(M-1)\frac{2\pi}{M}} & e^{-j2(M-1)\frac{2\pi}{M}} & \dots & e^{-j(M-1)(M-1)\frac{2\pi}{M}} \end{bmatrix}$$

Exercise 8 (5/10)

Implementation of a polyphase filterbank



Exercise 8 (6/10)

- Goal: Decompose the original signal in M=4 subbands.
- Basic ideas:
  - Compute the filters
  - Apply each subfilter to the input signal
  - Apply the DFT matrix at the filtered signals.

Exercise 8 (7/10)

- Notation:
  - M number of subbands
  - L filter length
  - K = ceil(L/M), number of sample for each subfilter
  - L = K*M

- The low pass filter is:
  - h = fir1(L-1,1/M) ;

Exercise 8 (8/10)

- Compute the polyphase filters
  - em = zeros(M, length(h));
  - for m = 1:M
  - em(m,m:M:end) = h(m:M:end) ;
  - end

- N.B.: “em” is a MxL matrix. Each row is a subband filter (we take the null samples).

Exercise 8 (9/10)

- out = zeros(M, length(x)) ;

  % each row is one specific subband of the input signal (in the time domain) = is the input convolved with a particular subband filter

```
for m  = 1:M
        y = filter(em(m, : ), 1, x) ;
        out(m,1:length(y)) = y ;
end
```

- out=fft(out);

Exercise 8 (10/10)

- Show the output spectrum for each subband:

```
for m = 1 :M
    subplot(M, 1 ,m),
    plot(abs(fft(out(m,:),1024)))
end
```

Agenda

- Ex1: Downsampling
- Ex2: Upsampling
- Ex3: Decimation
- Ex4: Interpolation
- Ex5: Frequency modification with rational factor
- Ex6: Polyphase Decimation
- Ex7: Polyphase Interpolation
- Ex8: Polyphase Filter Banks
- **Ex9: Quadrature Mirror Filters**

Exercise 9 (1/8)

- Goal: Obtain a system that allows the perfect recontruction of a signal.   Ip. M=2

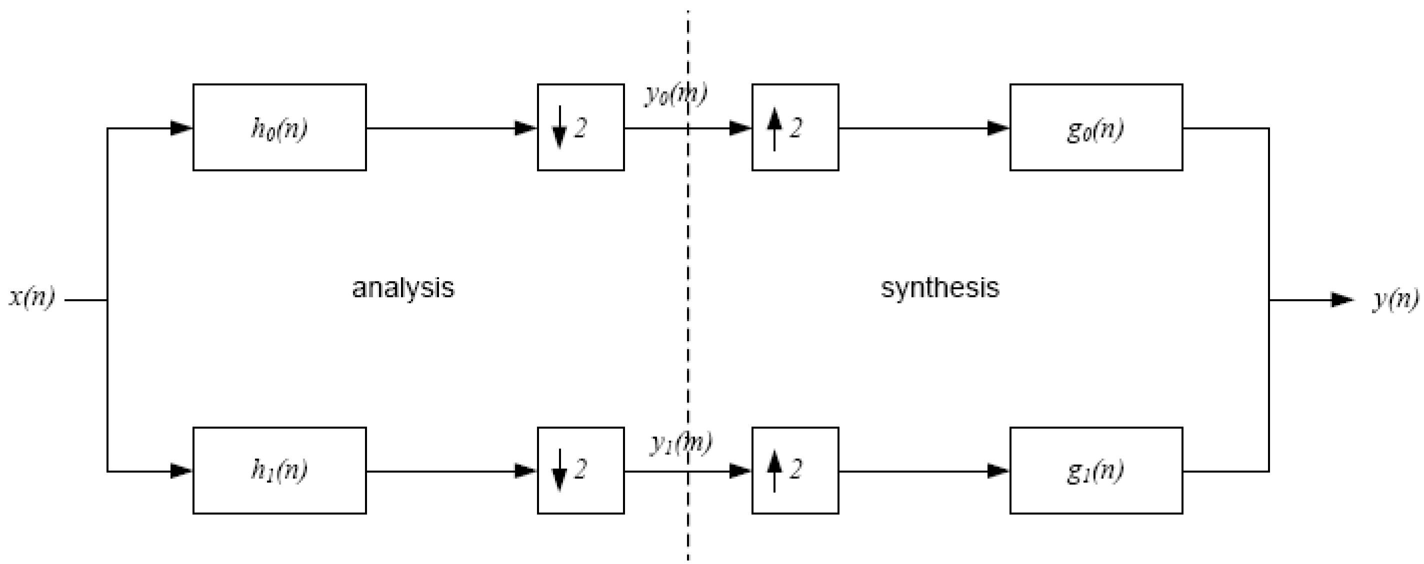


Fig. 5.21. Bank of two filters

$$y(n) = x(n)$$

Exercise 9 (2/8)

- Goal: Generate the analysis and synthesis filters based on QUADRATURE MIRROR FILTERS conditions:

  - linear phase FIR philter (N even)
  - $H_1(\omega) = H_0(\omega - \pi)$
  - $\left|H_0(\omega)\right|^2 + \left|H_0(\omega - \pi)\right|^2 = \cos t$

Exercise 9 (3/8)

- QUADRATURE MIRROR FILTERS.
  - $H_1(z) = H_0(-z)$ $\qquad h_1(n) = (-1)^n h_0(n)$
  - $G_0(z) = H_1(-z)$ $\qquad g_0(n) = (-1)^n h_1(n)$
  - $G_1(z) = -H_0(-z)$ $\qquad g_1(n) = (-1)^{n+1} h_0(n)$

Exercise 9 (4/8)

- Pseudocode:
  - Compute the filters
  - Build the signal
  - Analysis:
    - filtering
    - dowmsamplig
  - Synthesis:
    - upsampling
    - filtering
  - Compute the reconstruction error

Exercise 9 (5/8)

- Compute the filters:
  - h0 = [sqrt(2)/2, sqrt(2)/2] ;
  - h1 = h0.*[1 -1];
  - g0 = h0;
  - g1 = -h1 ;

- Verify that :

T = abs(H0).^2 + abs(H1).^2 ;

is constant for each frequency

Exercise 9 (6/8)

- Load the input signal
  - [x Fs Nbits] = wavread('flute2.wav');
  - x = x(:,1);
- Analysis: filtering - dowmsamplig
  - % low pass subband
  - y0 = conv(x,h0); % filtering
  - y0 = y0(1:2:end); % downsampling
  - % high pass subband
  - y1 = conv(x,h1); % filtering
  - y1 = y1(1:2:end); % downsampling

Exercise 9 (7/8)

- Synthesis: upsampling – filtering
  - % low pass subband
  - y0up = zeros(2*length(y0), 1);
  - y0up(1:2:end) = y0 ;
  - y0up = conv(y0up,g0);

  - % high pass subband
  - y1up = zeros(2*length(y1), 1);
  - y1up(1:2:end) = y1 ;
  - y1up = conv(y1up,g1);

Exercise 9 (8/8)

- Recontruct the signal
  - y = y0up + y1up ;

- Compute the reconstruction error
  - e = x-y ;
  - figure, plot(e)
  - display(['max error:' num2str(max(abs(e)))]);