CS 6351 DATA COMPRESSION

SUBBAND CODING PART I

Instructor: Abdou Youssef

OBJECTIVES OF THIS LECTURE

By the end of this lecture, you will be able to:

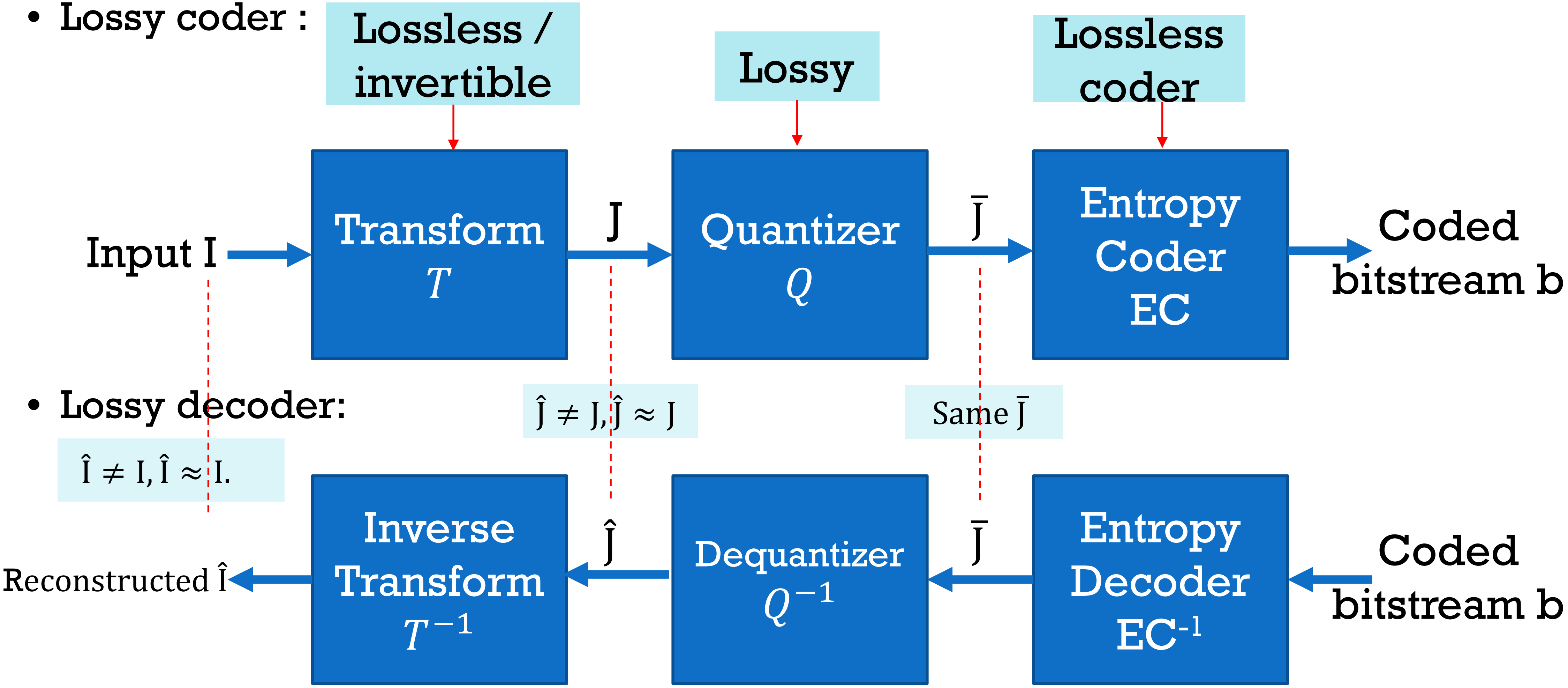
- Describe linear filters and how they work
- Relate filtering to Fourier transforms, frequencies, polynomial multiplication, and spectrum shaping
- Specify low-pass filters and high-pass filters, explain their effect on signals, and preliminarily use filters for certain standard applications
- Explain subband coding using filter banks, down-sampling and up-sampling
- Describe and visualize the effects of subband coding on 1D signals and 2d signals
- Apply subband coding to lossy compression in a fundamental way

OUTLINE

- Reminder of why an alternative to DCT is needed
- Linear filters
- Filters as weighted averaging
- Convolution theorem: relating filtering to the Fourier transform
- Filters as spectrum-shaping devices, affecting (bands of) frequencies
- The z-transform, and the convolution theorem revisited
- Low-pass and high-pass filters
- Frequency response of filters
- Effect filters on 1D and 2D signals
- Subband coding scheme, and its effect on signals, and its application to compression
- A peak into next lecture

GENERAL SCHEME OF LOSSY COMPRESSION

- Lossy coder :
- Lossy decoder:



Can we have different transforms than the ones seen so far?

BACKGROUND AND MOTIVATION
-- PROBLEMS WITH DCT-BASED COMPRESSION --

- Blocking artifacts, especially at low bitrate

BACKGROUND AND MOTIVATION
-- INADEQUACY OF REMEDIES TO BLOCK-DCT --

- The methods for reducing blocking artifacts, such as *overlapped transforms*, are costly and complicated

- Applying DCT on the whole image, rather than on small blocks
  - Ignores the significant differences in frequency contents in various local regions of the image – **non-localization problem**
  - Which leads to less quality-bitrate performance

BACKGROUND AND MOTIVATION
-- ADVANTAGES OF WAVELETS/SUBBAND CODING --

- Wavelets/Subband-coding operate on the whole image as one single block
  - Thus avoiding blocking artifacts
  - While dynamically adjusting the spatial/frequency resolution to the appropriate level in various regions of the image
- In practice, wavelets/subband coding perform as well as DCT and sometimes better, especially at low bitrate, without blocking artifacts

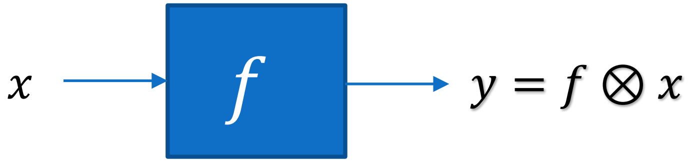
LINEAR FILTERS

- Definition of a **linear filter**
  - A linear filter f is characterized by a sequence $(f_k)_k$ of real numbers
    - the f_k's are called the ***filter* taps**, or **filter coefficients**, and we write $f = (f_k)_k$
  - Filtering an input signal $x = (x_n)_n$ through filter f gives an output signal $y = (y_n)_n$:

$$\boldsymbol{y_n = \sum_k f_k x_{n-k} = \sum_k f_{n-k} x_k} \text{ for all } n$$

The range of k is important, and is sometimes left implicit

- Mathematical notation: $\boldsymbol{y = f \otimes x}$
  - That is called the ***convolution*** of f and x



- Notes about indexing notation:
  - Indices k can range from anywhere to anywhere
  - Any term where its index is "out of range" is by default = 0

If $x = [x_0, x_1, x_2, \dots, x_{100}]$,
Then $x_{101} = 0, x_{-1} = 0, \dots$

EXAMPLES OF FILTERS (1)

- Take filter $f = [f_0, f_1] = [1, -1]$
  - This means that $f_k = 0$ for any $k \neq 0, 1$
- Take input signal $x = [x_0, x_1, x_2, \dots, x_{100}]$
- Then, the output y of the filtering is:
  - $\boldsymbol{y_n = \sum_k f_k x_{n-k} = f_0 x_n + f_1 x_{n-1} = x_n - x_{n-1}}$ for all n
  - Thus, $\boldsymbol{y_0 = x_0 - x_{-1} = x_0}$, $\boldsymbol{y_1 = x_1 - x_0}$, $\boldsymbol{y_2 = x_2 - x_1}$, $\boldsymbol{y_3 = x_3 - x_2}, \dots,$
    $\boldsymbol{y_{100} = x_{100} - x_{99}, y_{101} = x_{101} - x_{100} = -x_{100}}$, $\boldsymbol{y_{102} = 0, y_{103} = 0}, \dots$
- Concretely, if $x = [x_0, x_1, x_2, \dots, x_{100}] = [1, 2, 3, \dots, 100]$
  - Then $y = [y_0, y_1, y_2, \dots, y_{100}, y_{101}] = [1,1,1, \dots, 1, -100]$
  - You could stipulate where the indexing of y ends, like at 100.

EXAMPLES OF FILTERS (2)

- Take filter $f = [f_{-1}, f_0, f_1] = [-\frac{1}{2}, 1, -\frac{1}{2}]$
  - This means that $f_k = 0$ for any $k \neq -1, 0, 1$
- Take input signal $x = [x_0, x_1, x_2, \dots, x_{100}]$
- Then, the output y of the filtering is:
  - $\boldsymbol{y_n = \sum_k f_k x_{n-k} = f_{-1} x_{n+1} + f_0 x_n + f_1 x_{n-1} = -\frac{1}{2} x_{n+1} + x_n - \frac{1}{2} x_{n-1} = x_n - \frac{x_{n-1} + x_{n+1}}{2}}$

    Thus, $\boldsymbol{y_{-1} = -\frac{1}{2} x_0}$, $\boldsymbol{y_0 = x_0 - \frac{1}{2} x_1}$, $\boldsymbol{y_1 = x_1 - \frac{x_0 + x_2}{2}}$, $\boldsymbol{y_2 = x_2 - \frac{x_1 + x_3}{2}}$, …
- Concretely, if $x = [x_0, x_1, x_2, \dots, x_{100}] = [1, 2, 3, \dots, 100]$
  - Then $y = [y_{-1}, y_0, y_1, y_2, \dots, y_{100}, y_{101}] = [-\frac{1}{2}, 0,0, 0, \dots, 0, 50.5, -50]$

EXAMPLES OF FILTERS (3)

- Take filter $f = [f_0, f_1, f_2] = [-\frac{1}{2}, 1, -\frac{1}{2}]$
  - This means that $f_k = 0$ for any $k \neq 0, 1, 2$
- Take input signal $x = [x_0, x_1, x_2, \dots, x_{100}]$
- Then, the output y of the filtering is:
  - $\boldsymbol{y_n = \sum_k f_k x_{n-k} = f_0 x_n + f_1 x_{n-1} + f_2 x_{n-2} = -\frac{1}{2} x_n + x_{n-1} - \frac{1}{2} x_{n-2}}$
- Concretely, if $x = [x_0, x_1, x_2, \dots, x_{100}] = [1, 2, 3, \dots, 100]$
  - Then $y = [y_0, y_1, y_2, \dots, y_{100}, y_{101}, y_{102}] = [-\frac{1}{2}, 0,0, 0, \dots, 0, 50.5, -50]$
  - Compare that with the output of the previous filter:

$$y = [y_{-1}, y_0, y_1, y_2, \dots, y_{100}, y_{101}] = [-\frac{1}{2}, 0,0, 0, \dots, 0, 50.5, -50]$$

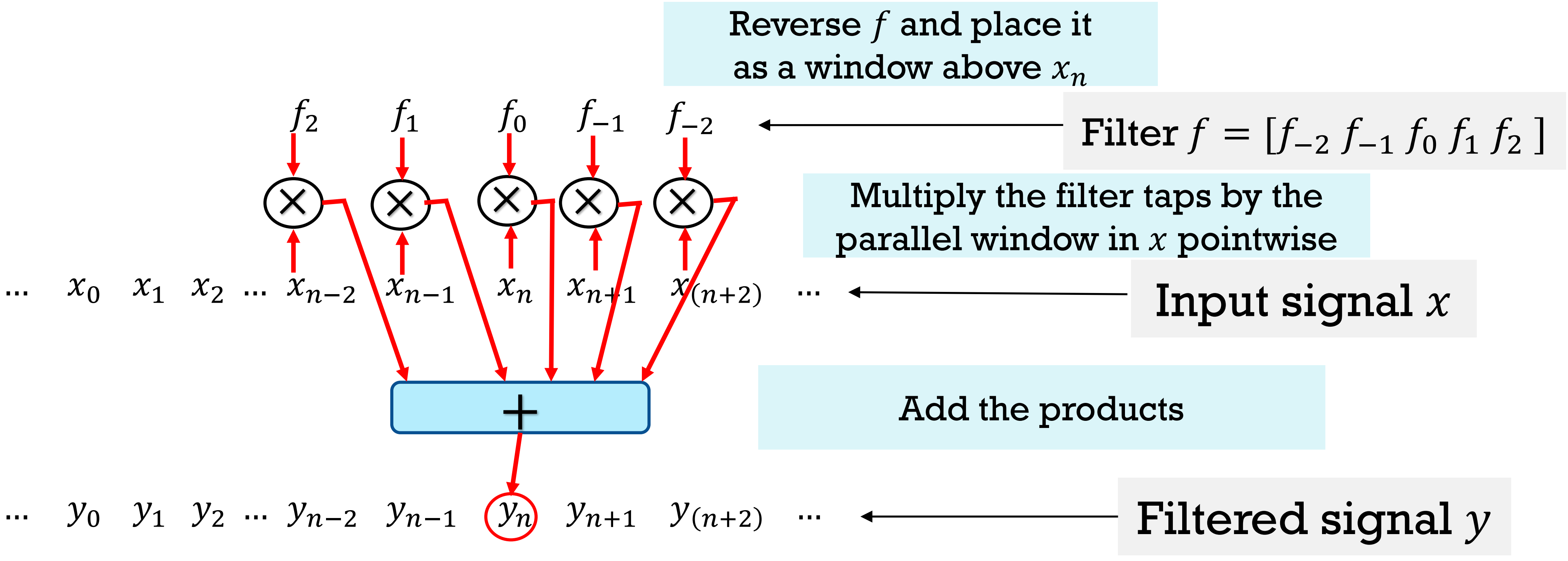
- Almost same filter as the last one, $f = [f_{-1}, f_0, f_1] = [-\frac{1}{2}, 1, -\frac{1}{2}]$
- But different in **indexing range**

EXAMPLES OF FILTERS (4)

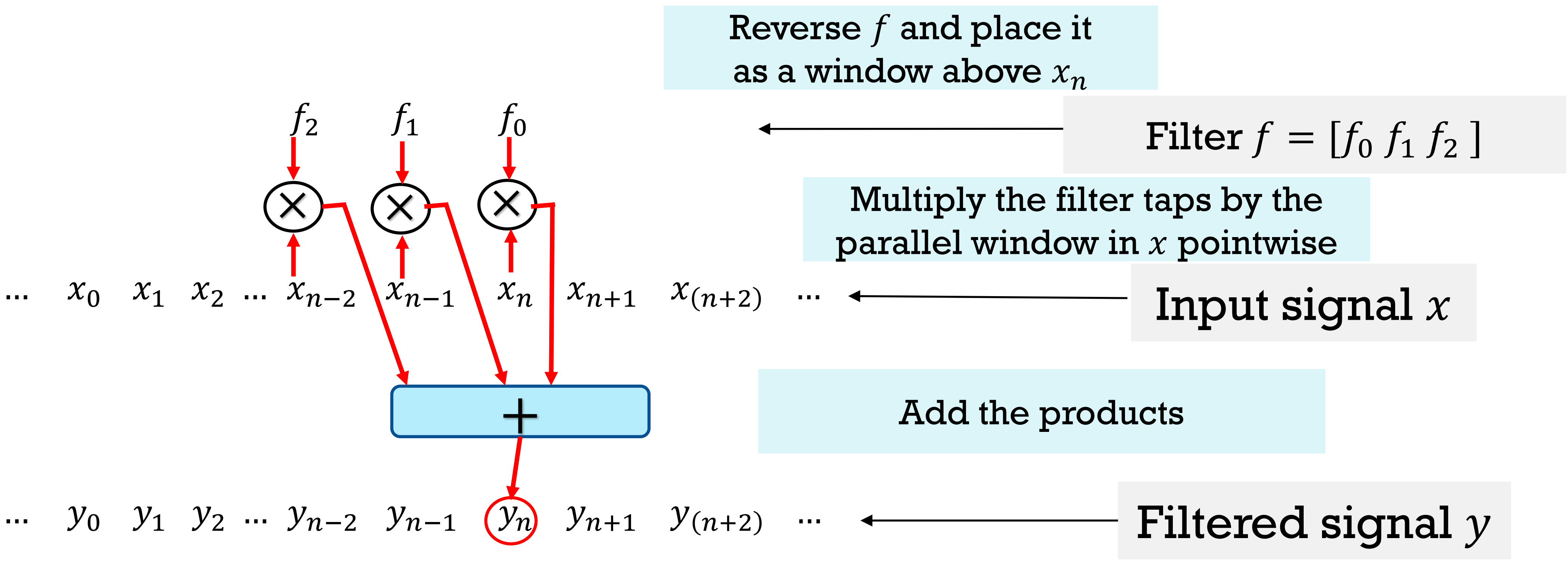
- What should the filter f be so that $y_n = x_n - \frac{x_{n-1}+x_{n-2}}{2}$?
- Answer:
  - $y_n = x_n - \frac{x_{n-1}+x_{n-2}}{2} = 1.\, x_n + \left(-\frac{1}{2}\right) x_{n-1} + \left(-\frac{1}{2}\right) x_{n-2}$
  - $y_n \qquad\qquad\qquad = f_0 x_{n-0} + f_1 x_{n-1} \quad + \quad f_2\, x_{n-2}$
  - Therefore, the filter $f = [f_0, f_1, f_2] = [1, -\frac{1}{2}, -\frac{1}{2}]$

FILTERING AS A WEIGHTED "AVERAGE"
-- THE FILTER TAPS ARE THE WEIGHTS (1) --

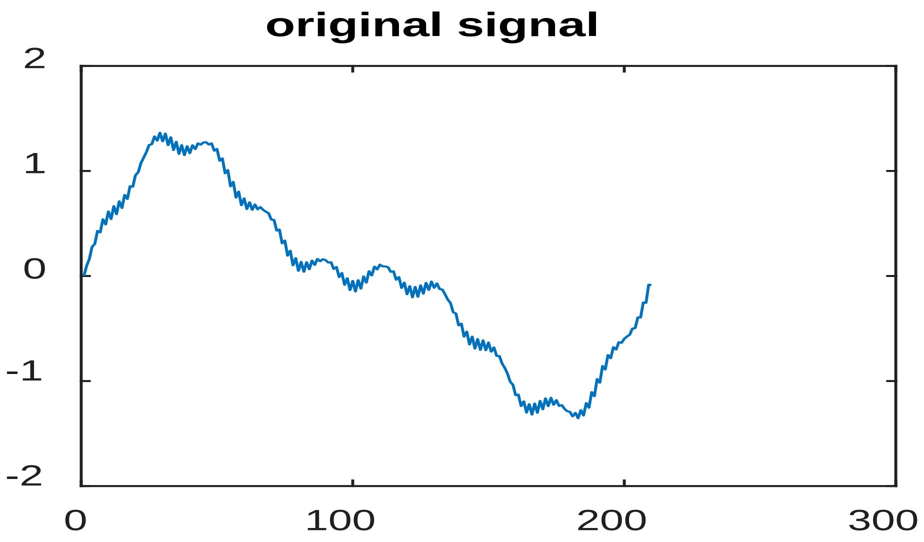
- Take filter $f = [f_{-2}\ f_{-1}\ f_0\ f_1\ f_2\ ]$, and a signal x
- $y_n = \sum_k f_k x_{n-k} = f_{-2}x_{n+2} + f_{-1}\, x_{n+1} + f_0 x_n + f_1 x_{n-1} + f_2 x_{n-2}$

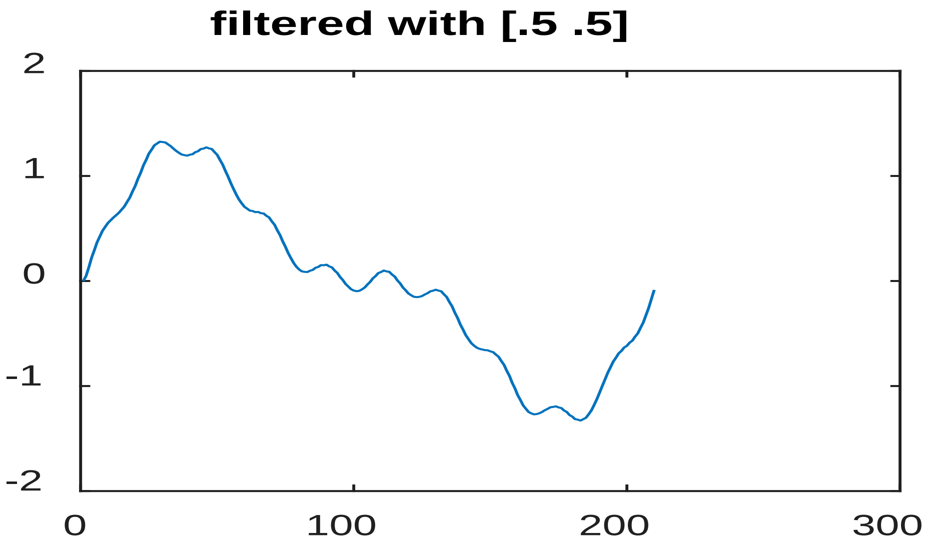


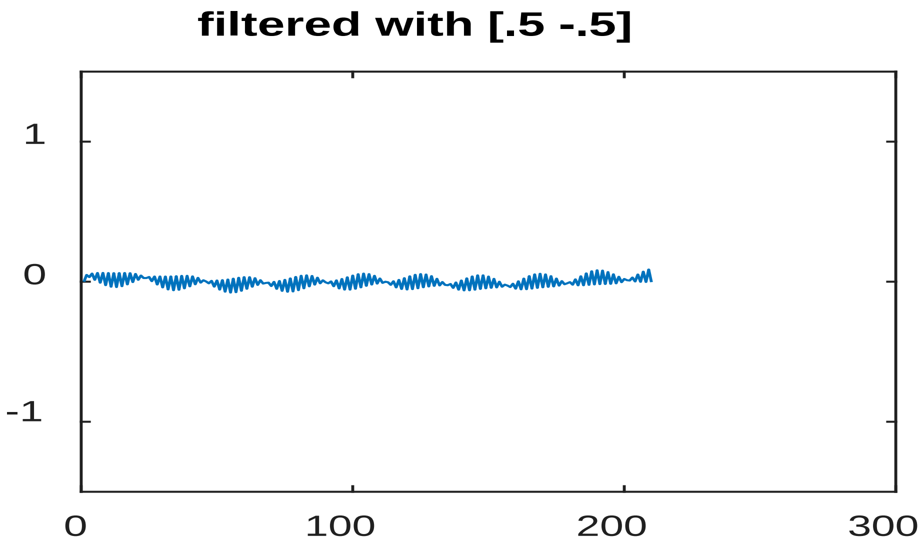
FILTERING AS A WEIGHTED "AVERAGE"
-- THE FILTER TAPS ARE THE WEIGHTS (2) --



EFFECT OF FILTERS ON SIGNALS





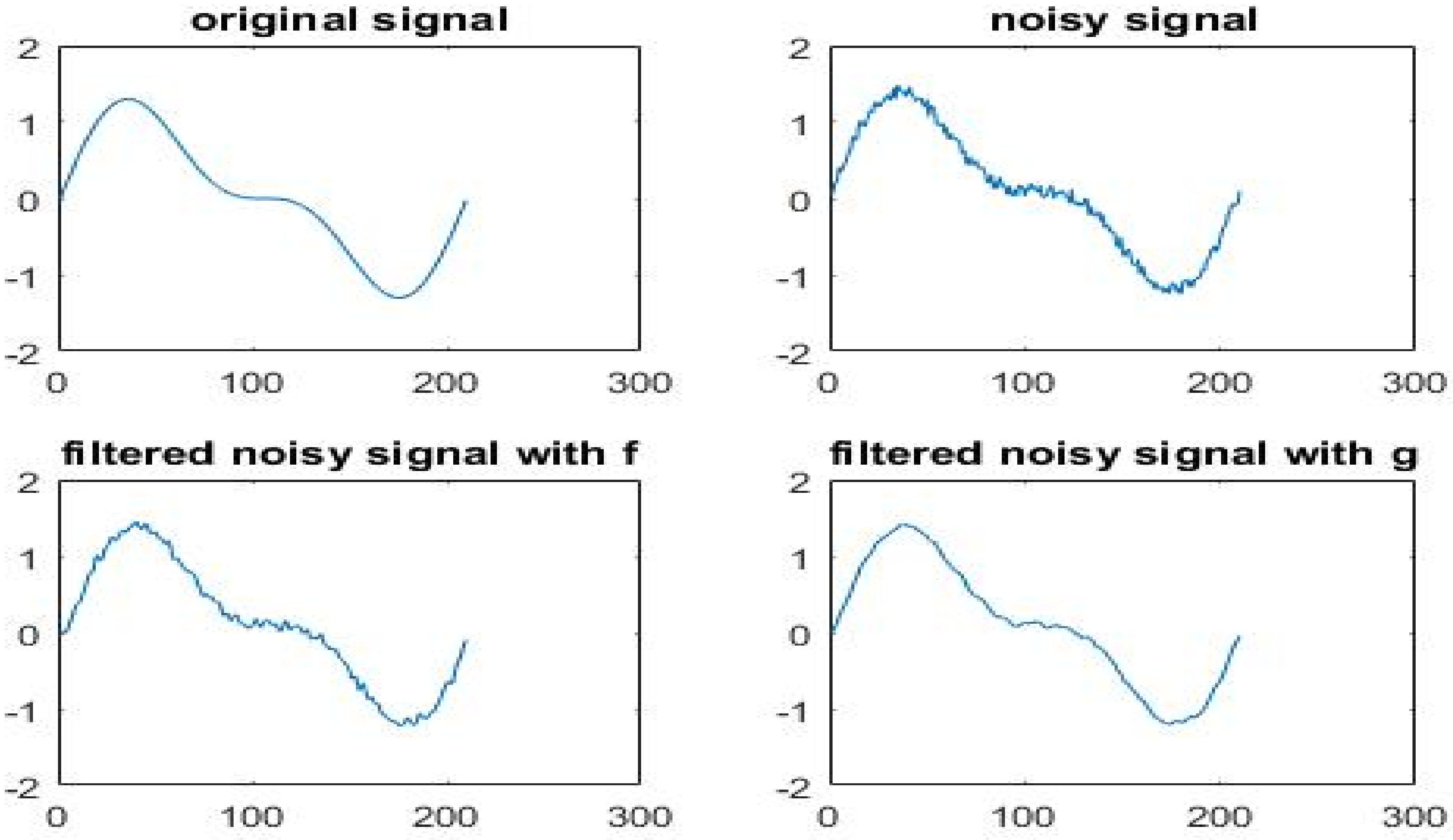


Code:
```
t=0:.03:2*pi;
S=sin(t)+0.1*sin(2*t)+0.01*sin(10*t);
subplot(2,2,1); plot(S);title('original signal')
FS=filter([1/2 1/2],[1],S);
subplot(2,2,2); plot(FS);title('filtered with [.5 .5]')
HS=filter([1/2 -1/2],[1],S);
subplot(2,2,3); plot(HS);  ylim([-1.5,1.5]);
title('filtered with [.5 -.5]')
```

APPLICATIONS OF FILTERING
-- NOISE REDUCTION (1/3) --

- Consider an original signal S, which got corrupted with noise r into NS
  - t=0:.03:2*pi; S=sin(t)+0.5*sin(2*t);
  - r=rand(1,length(S))/5; % random noise
  - NS=S+r; % signal plus noise
- Consider two filters f and g which will be used to reduce the noise
  - f=[0.0267 -0.0169 -0.0782 0.2669 0.6029 0.2669 -0.0782 -0.0169 0.0267];
  - g=[1 1 1 1 1]/5;
- Let
  - FNSf be the signal NS after denoising with filter f
  - FNSg be the signal NS after denoising with filter g

APPLICATIONS OF FILTERING
-- NOISE REDUCTION 2/3 --



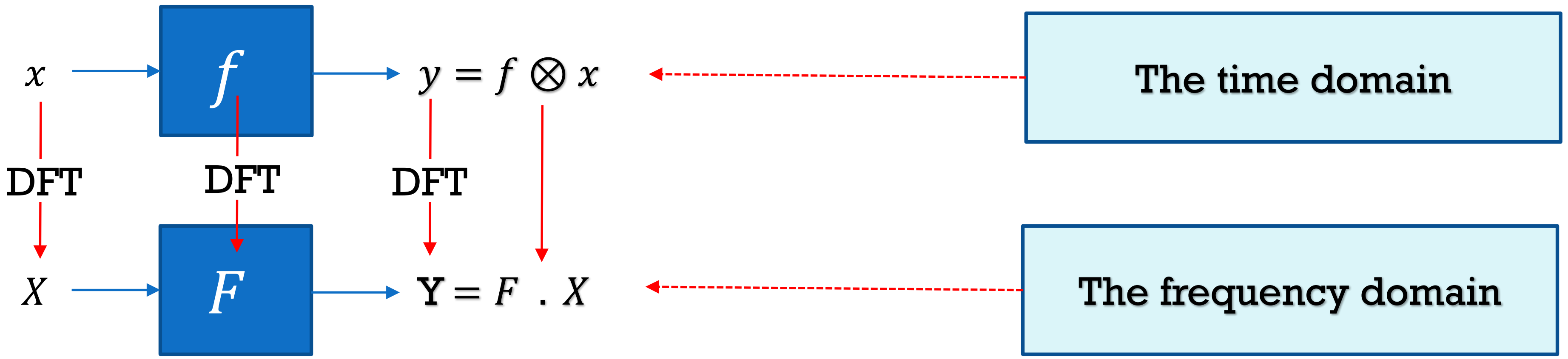
APPLICATIONS OF FILTERING
-- NOISE REDUCTION 3/3 --

This is the code that was used for the previous slide:

```
t=0:.03:2*pi;
S=sin(t)+0.5*sin(2*t);
r=rand(1,length(S))/5;          % random noise
NS=S+r;                         % signal plus noise
f=[0.0267 -0.0169 -0.0782 0.2669 0.6029 0.2669 -0.0782 -0.0169 0.0267];
g=[1 1 1 1 1]/5;                % another filter
FNSf=filter(f,[1],NS);  % filter NS with filter f
FNSg=filter(g,[1],NS);          % filter NS with filter g
subplot(2,2,1); plot(S);   title('original signal')
subplot(2,2,2); plot(NS);  title('noisy signal')
subplot(2,2,3); plot(FNSf); title('filtered noisy signal with f')
subplot(2,2,4); plot(FNSg); title('filtered noisy signal with g')
```

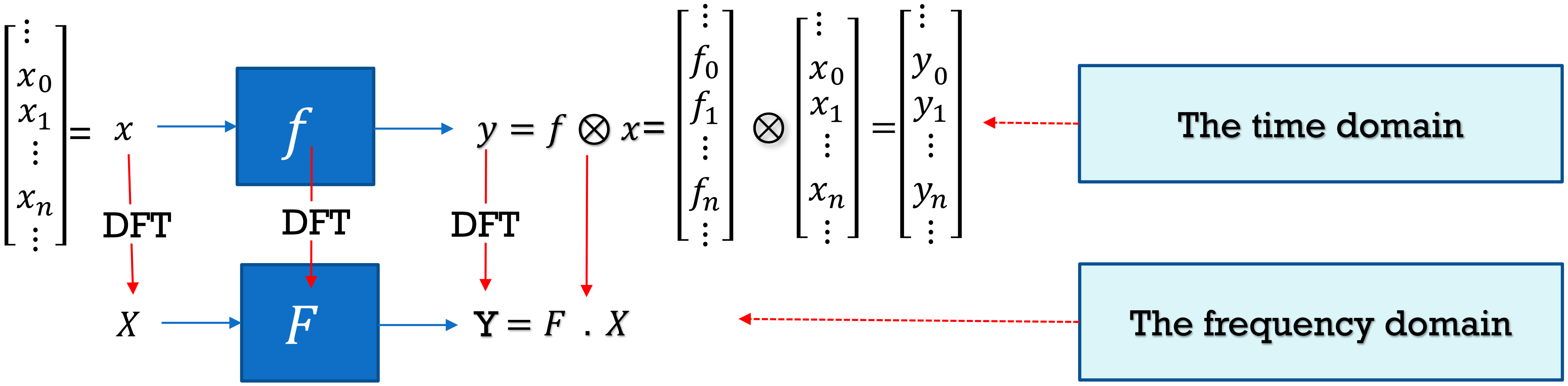
THE CONVOLUTION THEOREM

- **The convolution theorem:**
  - Let $x = (x_n)_n$ be a digital signal and $f = (f_k)_k$ be a filter, and let $y = (y_n)_n \stackrel{\text{def}}{=} f \otimes x$ be the output of filtering x with f.
  - Let X, Y and F denote the Fourier Transforms of x, y and f, respectively.
  - Then, $Y = F.X$ (pointwise multiplication).



THE CONVOLUTION THEOREM

- **The convolution theorem:**



$$\begin{bmatrix} \vdots \\ Y_0 \\ Y_1 \\ \vdots \\ Y_n \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ F_0 \\ F_1 \\ \vdots \\ F_n \\ \vdots \end{bmatrix} . \begin{bmatrix} \vdots \\ X_0 \\ X_1 \\ \vdots \\ X_n \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ F_0 . X_0 \\ F_1 . X_1 \\ \vdots \\ F_n . X_n \\ \vdots \end{bmatrix}$$

THE Z-TRANSFORM

- Let $a = (a_k)_k$ be a sequence (like a discrete signal or a filter)
- The z-transform transforms a sequence $a = (a_k)_k$ into a complex function $A(z)$:

$$\boldsymbol{A(z) = \sum_k a_k z^k} \quad \text{// a polynomial in z}$$

- We use the notation that the input sequence is denoted with a lower case letter, and its z-transform is denoted by the upper-case of the same letter:
  - $a = (a_k)_k \to A(z) = \sum_k a_k z^k$
  - $x = (x_k)_k \to X(z) = \sum_k x_k z^k$
  - $y = (y_k)_k \to Y(z) = \sum_k y_k z^k$
  - $f = (f_k)_k \to F(z) = \sum_k f_k z^k$

MULTIPLICATION OF POLYNOMIALS

- Take $X(z) = \sum_k x_k z^k$ and $F(z) = \sum_k f_k z^k$
- Multiply $F(z).X(z) = \left(\sum_k f_k z^k\right)\left(\sum_k x_k z^k\right) = \sum_n a_n z^n$

The a_n are to be determined

- Let's do an example first:
  - $(f_0 z^0 + f_1 z^1 + f_2 z^2 + f_3 z^3 + f_4 z^4).(x_0 z^0 + x_1 z^1 + x_2 z^2 + x_3 z^3 + x_4 z^4) =$
  - $f_0 x_0 z^0 + (f_0 x_1 + f_1 x_0) z^1 + (f_0 x_2 + f_1 x_1 + f_2 x_0) z^2 + (f_0 x_3 + f_1 x_2 + f_2 x_1 + f_3 x_0) z^3 + \cdots$
  - How did we get each coefficient of z^n? For example, the coefficient of z^3?
    - Multiply $f_k z^k$ by $x_j z^j$ for all k and j where $k + j = 3$, and adding those products
    - Each product is $f_k x_j z^{k+j} = f_k x_{3-k} z^3$,
    - Their sum is $\left(\sum_k f_k x_{3-k}\right) z^3 = (f_0 x_3 + f_1 x_2 + f_2 x_1 + f_3 x_0) z^3$
- In general for z^n, it is $\left(\sum_k f_k x_{n-k}\right) z^n$
- Thus, $a_n = \sum_k f_k x_{n-k}$, and so $y_n = a_n$, where $y = f \otimes x$

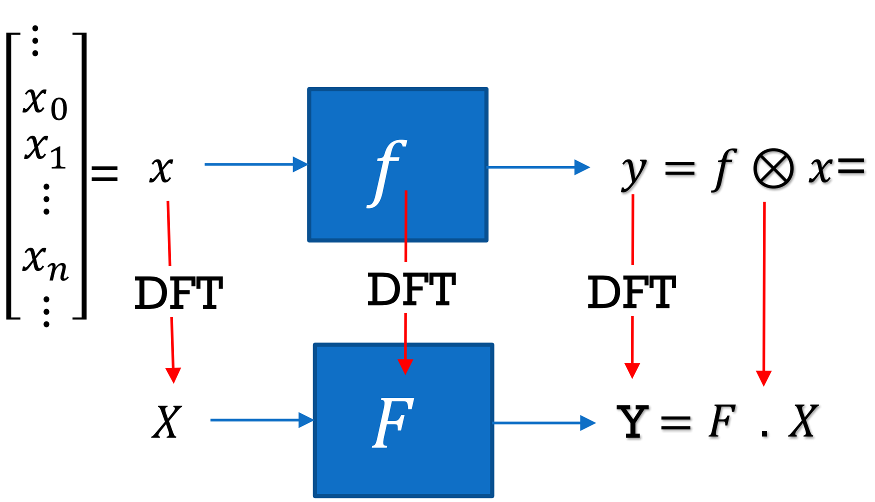
$\Rightarrow F(z).X(z) = \sum_n a_n z^n = \sum_n y_n z^n$
$\Rightarrow F(z).X(z) = Y(z)$

Z-TRANSFORM AND FILTERING

- **Convolution Theorem in terms of the z-transform**:
  - Let $x = (x_n)_n$ be a digital signal and $f = (f_k)_k$ be a filter, and let $y = (y_n)_n \stackrel{\text{def}}{=} f \otimes x$ be the output of filtering x with f.
  - Let $X(z), Y(z)$ and $F(z)$ denote the z-transforms of x, y and f, respectively.
  - Then, $Y(z) = F(z).X(z)$ (polynomial multiplication)
- **Proof**: the derivation we did in the previous slide.
- Exercise: find a connection between DFT and z-transform

THE CONVOLUTION THEOREM
-- IMPLICATIONS (1/3)--

- **Recall:**



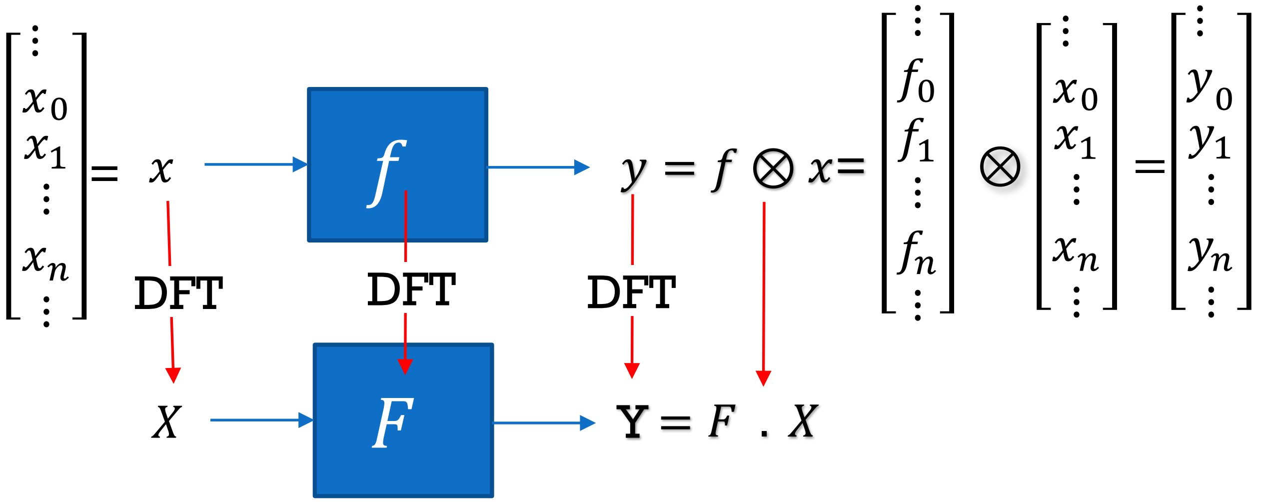
$$\begin{bmatrix} \vdots \\ f_0 \\ f_1 \\ \vdots \\ f_n \\ \vdots \end{bmatrix} \otimes \begin{bmatrix} \vdots \\ x_0 \\ x_1 \\ \vdots \\ x_n \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ y_0 \\ y_1 \\ \vdots \\ y_n \\ \vdots \end{bmatrix}$$

$$\begin{bmatrix} \vdots \\ Y_0 \\ Y_1 \\ \vdots \\ Y_n \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ F_0 \\ F_1 \\ \vdots \\ F_n \\ \vdots \end{bmatrix} . \begin{bmatrix} \vdots \\ X_0 \\ X_1 \\ \vdots \\ X_n \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ F_0 . X_0 \\ F_1 . X_1 \\ \vdots \\ F_n . X_n \\ \vdots \end{bmatrix}$$

- That means if you want to keep certain frequencies of input x, and throw out certain other frequencies, do:
  - Create a filter f whose DFT F
    - has $F_k = 1$ for the frequencies to be kept
    - has $F_k = 0$ for the frequencies to be thrown away

THE CONVOLUTION THEOREM
-- IMPLICATIONS (2/3)--

- **Recall:**



$$\begin{bmatrix} \vdots \\ Y_0 \\ Y_1 \\ \vdots \\ Y_n \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ F_0 \\ F_1 \\ \vdots \\ F_n \\ \vdots \end{bmatrix} . \begin{bmatrix} \vdots \\ X_0 \\ X_1 \\ \vdots \\ X_n \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ F_0 . X_0 \\ F_1 . X_1 \\ \vdots \\ F_n . X_n \\ \vdots \end{bmatrix}$$

- Also, if you want to enhance certain frequencies of input x, and reduce certain other frequencies, do:
  - Create a filter f whose DFT F
    - has $|F_k| > 1$ for the frequencies to be enhanced
    - has $|F_k| < 1$ for the frequencies to be reduced

THE CONVOLUTION THEOREM
-- IMPLICATIONS (3/3)--

- Therefore, a **filter is a <u>spectrum-shaping</u> device**
- That is, to change the frequencies of an input signal x, simply design the right filter, and filter x with it

The **<u>spectrum</u>** of x is the whole range of the frequencies of x, i.e., the DFT(x), namely, X_0, X_1, X_2 ...

- The design of filters that meet certain spectrum-shaping requirements is a well-established field
- In the case of subband coding, we will be designing quartets of filters (**filter banks**) that must meet certain conditions in a coordinated way

SPECIAL KINDS OF FILTERS

- We just saw that filters are spectrum-shaping devices
- We will define next two broad kinds of filters:
  - **Low-pass filters**
  - **High-pass filters**
- More generally, one can define what is called band-pass filters
  - But for our compression purposes, we only need low-pass and high-pass filters
- But to understand such filters, we need the notion of ***frequency response*** (next)

FREQUENCY RESPONSE OF A FILTER (1)

- Since a filter is a spectrum-shaping tool
  - To understand the behavior/effect of a filter, better look at the filter in the frequency domain
- That is, look at filter f by looking at its Fourier transform
- Or, equivalently, look at its z-transform $F(z)$ for $z = e^{-i\omega}$, where
  - $F(z) = \sum_k f_k z^k$
- That is, $F\left(e^{-i\omega}\right) = \sum_k f_k . \left(e^{-i\omega}\right)^k = \sum_k f_k e^{-ik\omega}$
- **Definition**: the function $F(\omega) = \sum_k f_k e^{-ik\omega}$ is called the ***frequency response*** of the filter f
- It is a complex function, periodic of period 2π

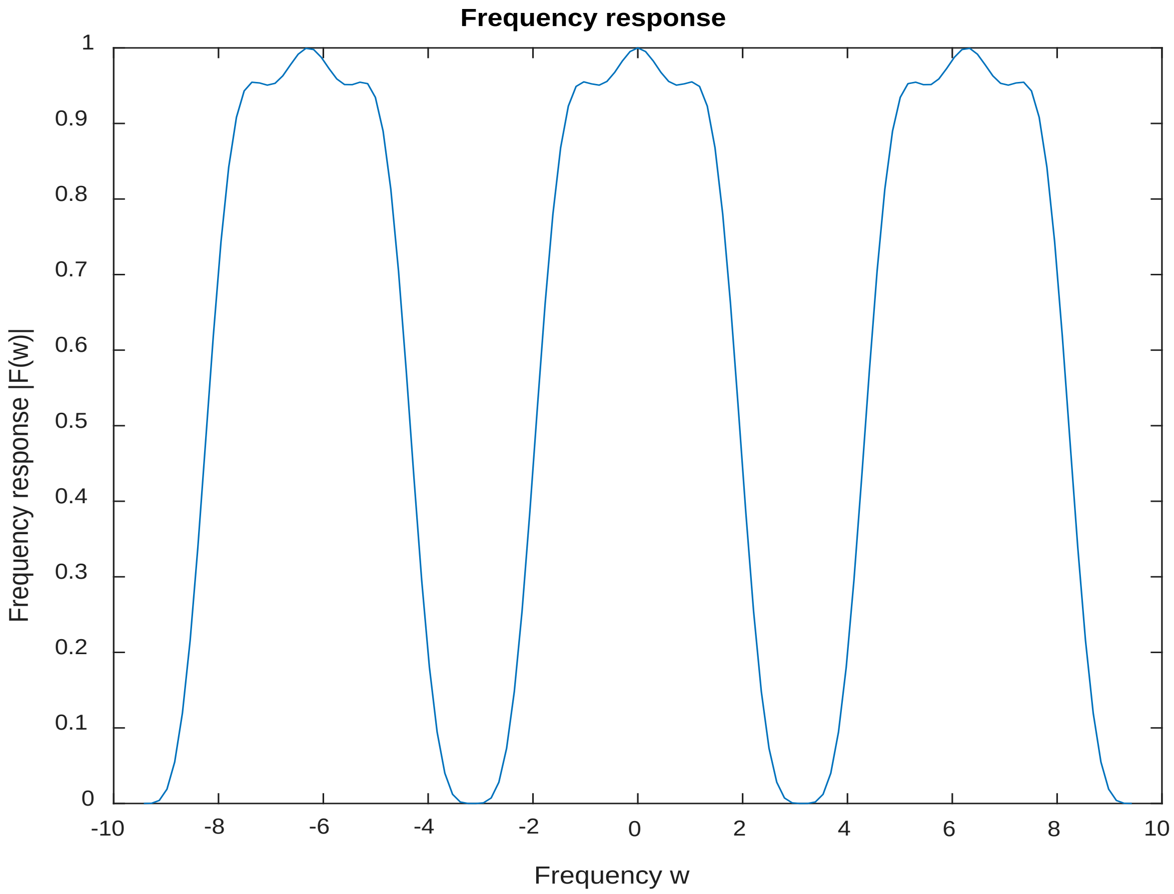
For convenience, people abuse the notation and write $F(\omega) = \sum_k f_k e^{-ik\omega}$

FREQUENCY RESPONSE OF A FILTER (2)

- The frequency response function $F(\omega) = \sum_k f_k e^{-ik\omega}$ is a complex function, periodic of period 2π
- ω is called the ***frequency***, and in the form $F(\omega) = \sum_k f_k e^{-ik\omega}$ above, ω is a <u>*continuous*</u> frequency
- To do discrete frequencies, take $\omega = \frac{2\pi}{N} l$ for $l = 0,1, \dots, N-1$ if the filter f has N taps, that is
  - If $f = [f_0, f_1, \dots, f_{N-1}]$, and its DFT is $[F_0, F_1 \ \dots, F_{N-1}]$, then:
  - $F_l = F\left(\frac{2\pi}{N} l\right) = \sum_k f_k e^{-i\frac{2\pi}{N}kl}$ (Exercise: prove it)
- Graphing $F(\omega)$ is not possible because it is a complex function
- Instead, we plot its magnitude $|F(\omega)|$

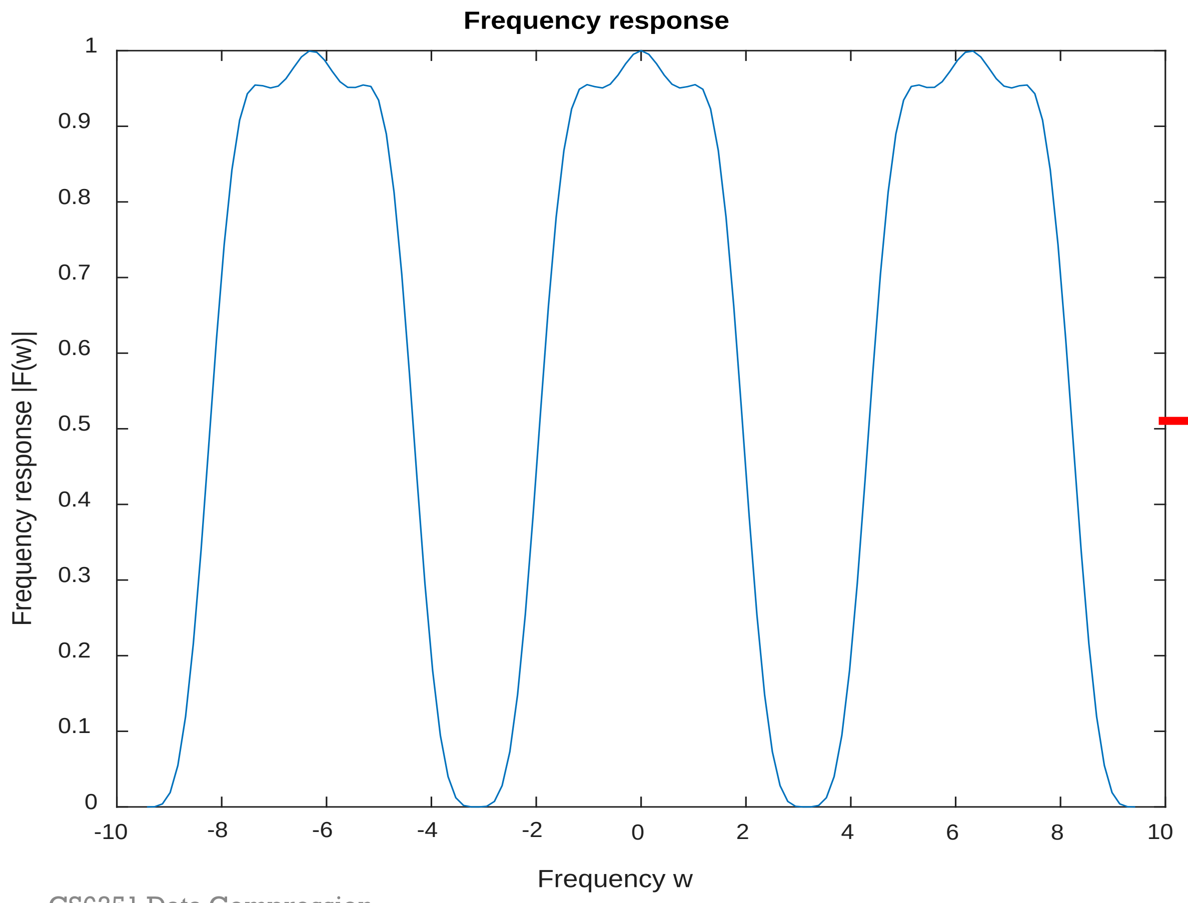
FREQUENCY RESPONSE OF A FILTER (3)

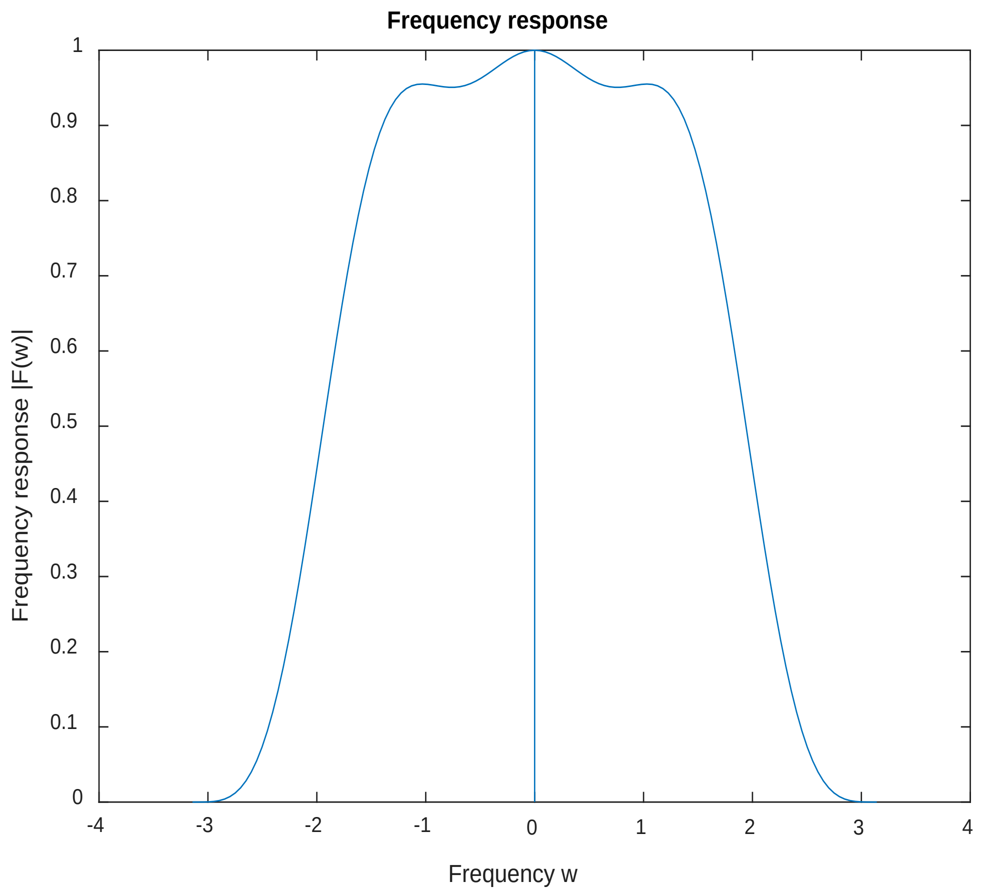
- Instead, we plot its magnitude $|F(\omega)| = |\sum_k f_k e^{-ik\omega}|$
- Ex: f=[0.0267, -0.0169, -0.0782, 0.2669, 0.6029, 0.2669, -0.0782, -0.0169, 0.0267]



FREQUENCY RESPONSE OF A FILTER (4)

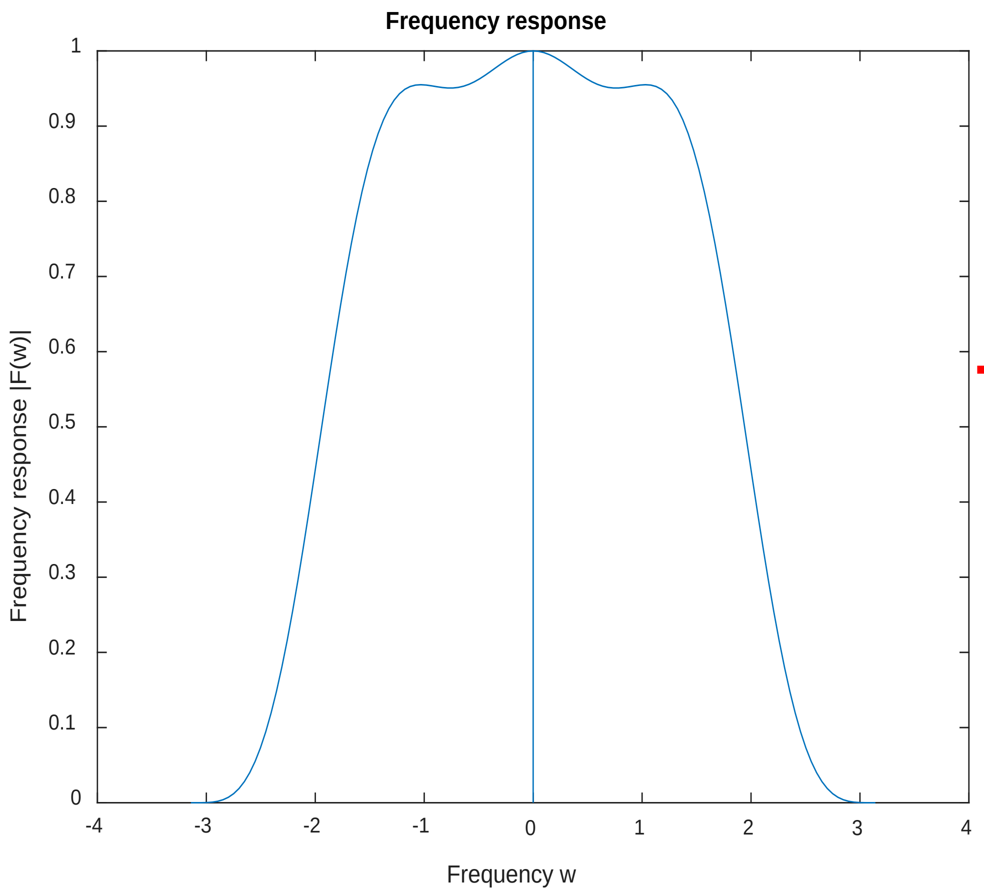
- Because it is <u>periodic</u>, plot $|F(\omega)| = \left|\sum_k f_k e^{-ik\omega}\right|$ in only one period $[-\pi \quad \pi]$
- Ex: f=[0.0267, -0.0169, -0.0782, 0.2669, 0.6029, 0.2669, -0.0782, -0.0169, 0.0267]

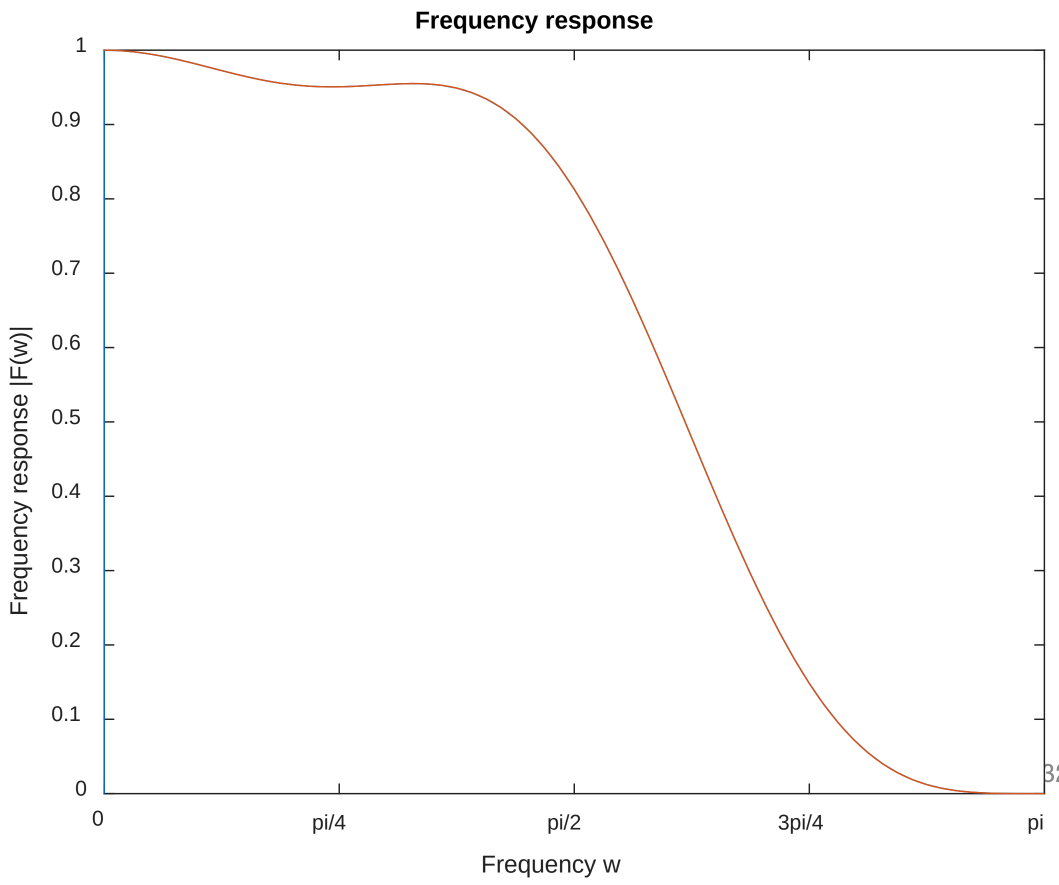




FREQUENCY RESPONSE OF A FILTER (5)

- Because it is periodic and symmetric around the y-axis,
  - plot the magnitude $|F(\omega)| = |\sum_k f_k e^{-ik\omega}|$ in only the range $\omega \in [0\ \pi]$
- Ex: f=[0.0267, -0.0169, -0.0782, 0.2669, 0.6029, 0.2669, -0.0782, -0.0169, 0.0267]





MATLAB CODE FOR THE FREQUENCY RESPONSE OF FILTERS

```
function y=fourierplotExt(x, leftLim, rightLim)
% y(t)=sum_k(x_k exp(-it*k)), t=[leftLim, rightLim]
%  plots abs(y(t)).

h=(rightLim-leftLim)/2^7;
R=leftLim:h:rightLim;
X=1:length(R);
T=0:length(x)-1;
for n=X
   y(n)=sum(x.*exp(-j*R(n)*T));
end

y = abs(y);
plot(R,y);  label('Frequency w')
ylabel('Frequency response |F(w)|')
title ('Frequency response')
```

Code that calls y=fourierplotExt(…) to create several plots:

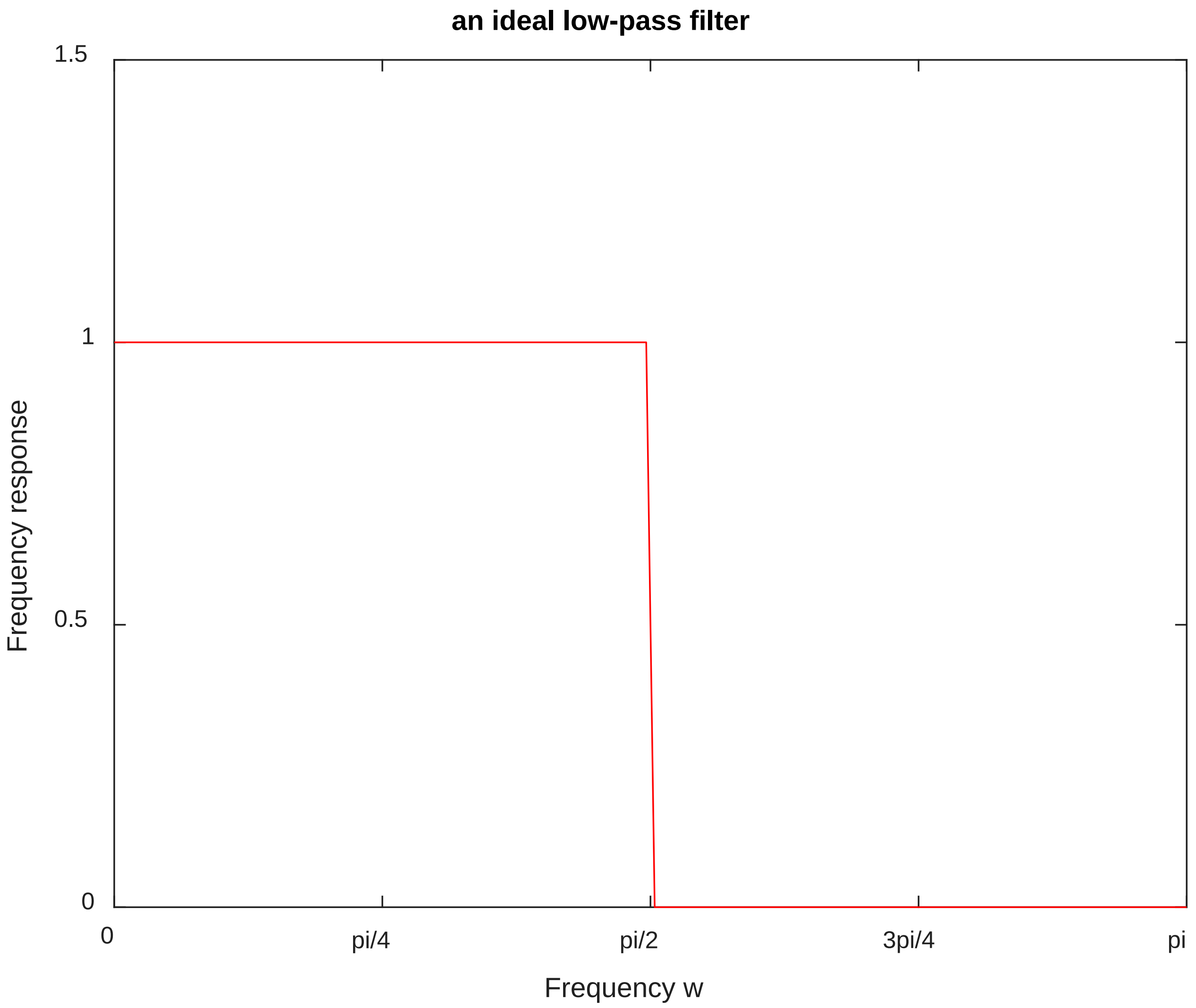
```
>> f=[0.0267, -0.0169, -0.0782, 0.2669, 0.6029, 0.2669, -0.0782, -0.0169, 0.0267];
>> figure; fourierplotExt(f, -3*pi, 3*pi)
>> figure; fourierplotExt(f, -pi, pi)
>> figure; fourierplotExt(f, 0, pi)
>> set(gca,'xtick',[0 pi/4 pi/2 3*pi/4 pi],'xlim',[0 pi],'xticklabels', ['  0  ';'pi/4 ';'pi/2 ';'3pi/4';' pi  '])
```

LOW-PASS FILTERS (LPF)
-- DEFINITION --

- An LPF filter is any filter that
  - eliminates (i.e., blocks) the high-frequency contents of any input signal, and
  - preserves (i.e., passes through) the low-frequency contents
- An ideal LPF f must then have its Fourier Transform F (or its frequency response $F(\omega) = \sum_k f_k e^{-ik\omega}$) as
  - a nonzero constant in a frequency range $[0, a)$, and
  - zero in the remaining range $[a, \pi]$

Applications:
- noise reduction
- Smoothing
- Etc.

LOW-PASS FILTERS (LPF)
-- FREQUENCY RESPONSE --

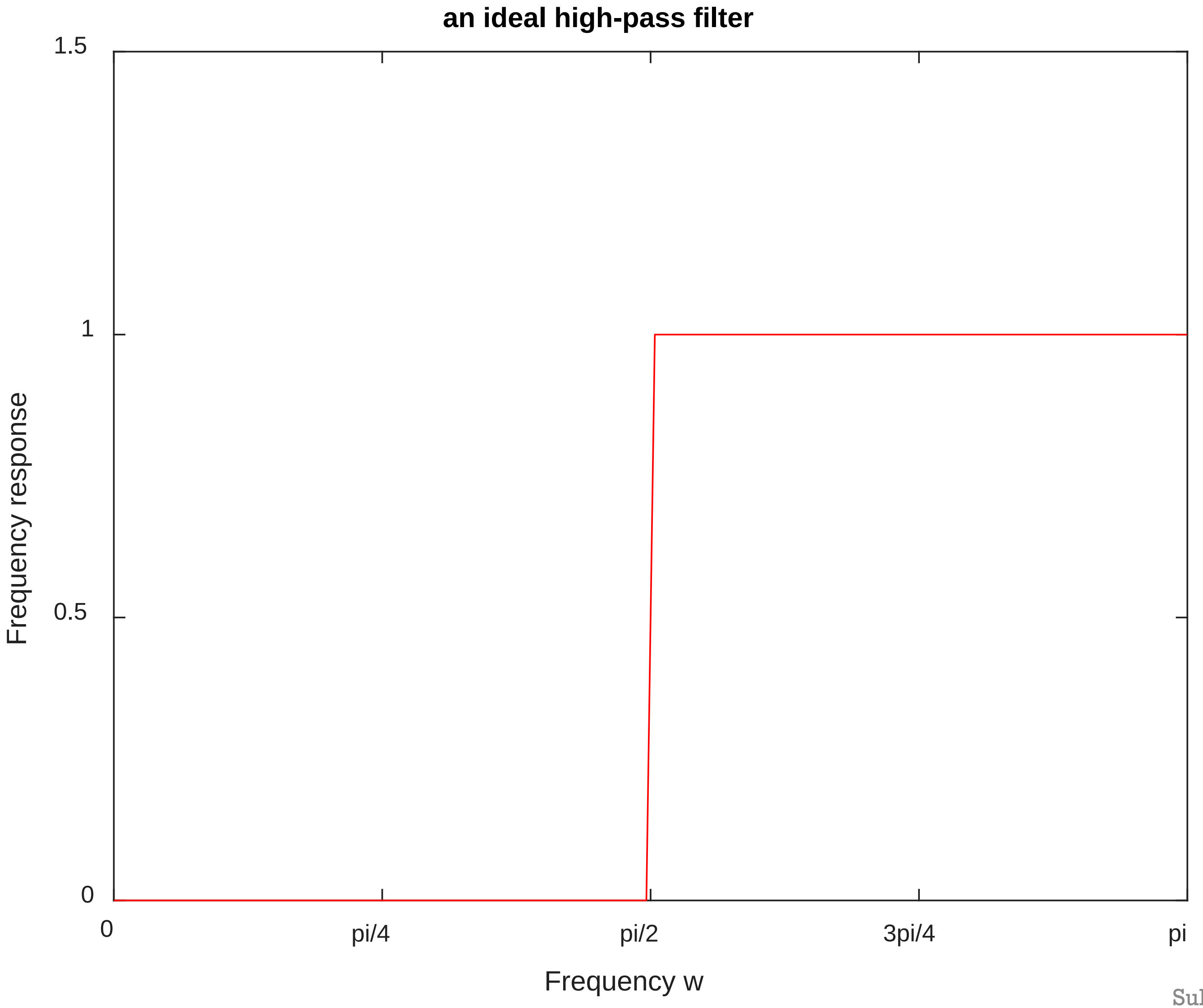


HIGH-PASS FILTERS (HPF)
-- DEFINITION --

- A HPF filter is any filter that
  - eliminates (i.e., blocks) the low-frequency contents of any input signal, and
  - preserves (i.e., passes through) the high-frequency contents
- An ideal HPF f must then have its Fourier Transform F (or its frequency response $F(\omega) = \sum_k f_k e^{-ik\omega}$) as
  - zero in the range $[0, a)$, and
  - a nonzero constant in the remaining frequency range $[a, \pi]$

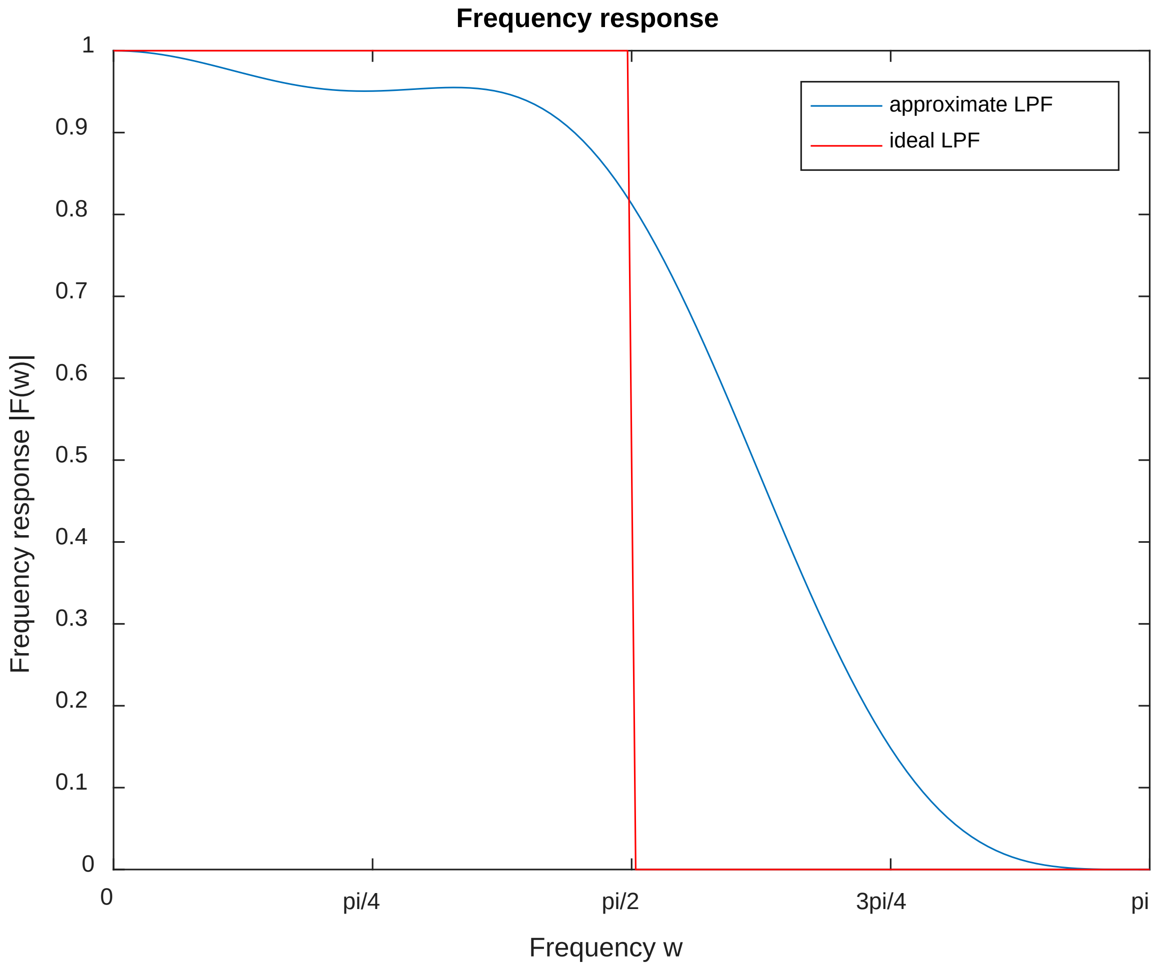
Applications:
- Sharpening
- Edge detection
- Etc.

HIGH-PASS FILTERS (HPF)
-- FREQUENCY RESPONSE --



OBSERVATIONS ABOUT LPF'S AND HPF'S

- Ideal LPF's and HPF's are not realizable in practice, but
  - many realizable filters are good approximations of ideal filters



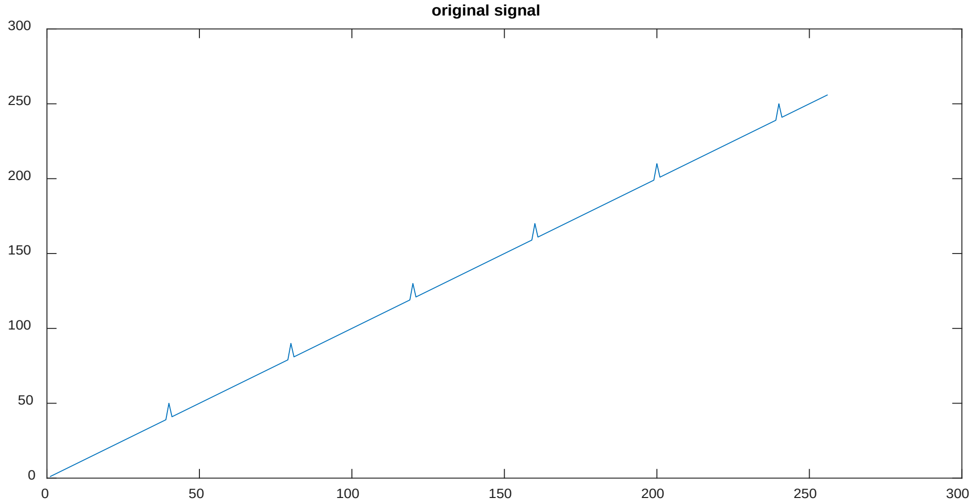
FIR AND IIR FILTERS

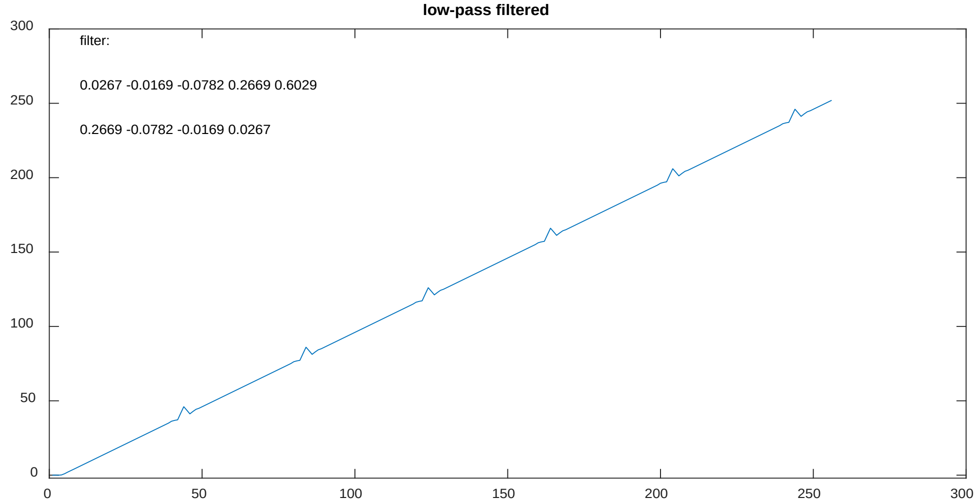
- A filter $f = (f_k)_k$ is called a ***finite-impulse-response (FIR) filter*** if it has a finite number of taps (i.e., coefficients), that is, the range of k is **finite**

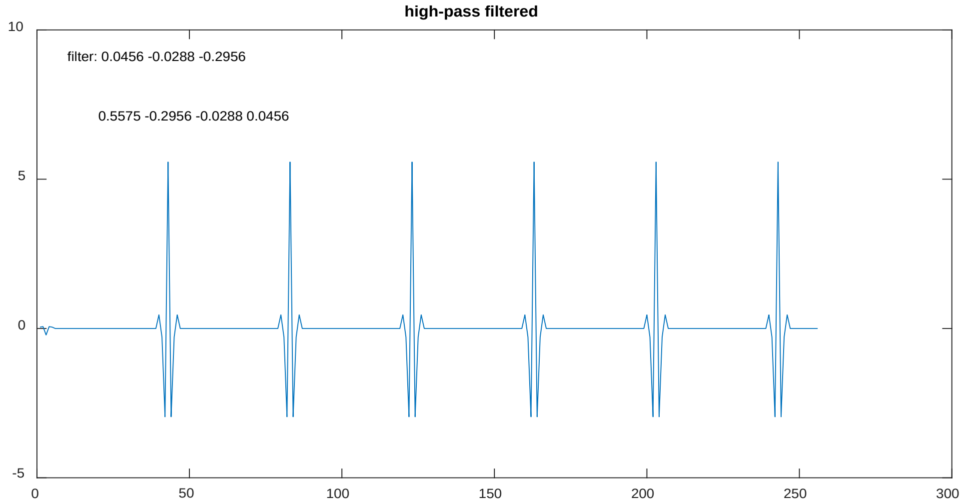
- Otherwise, the filter is called an ***infinite-impulse response (IIR) filter***

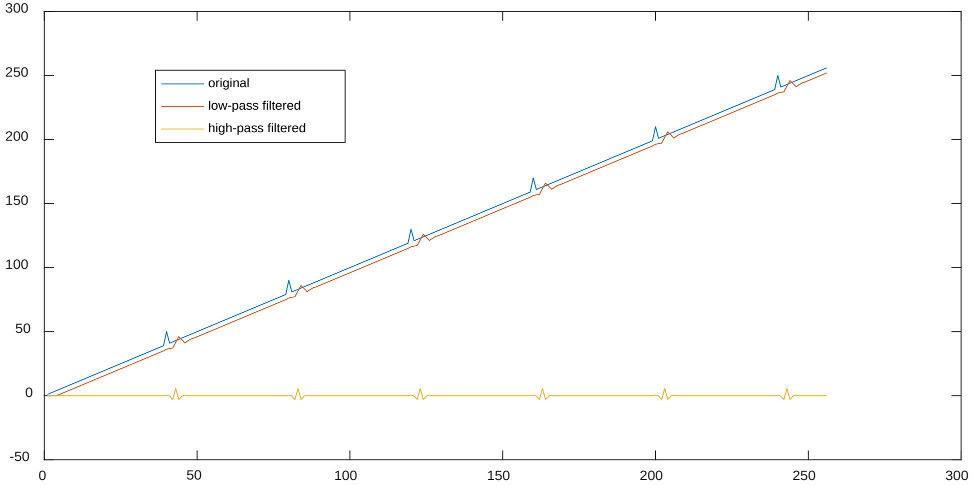
- The filters that we will use in subband coding are FIR filters

EXAMPLES OF LPF'S AND HPF'S AND THEIR EFFECT







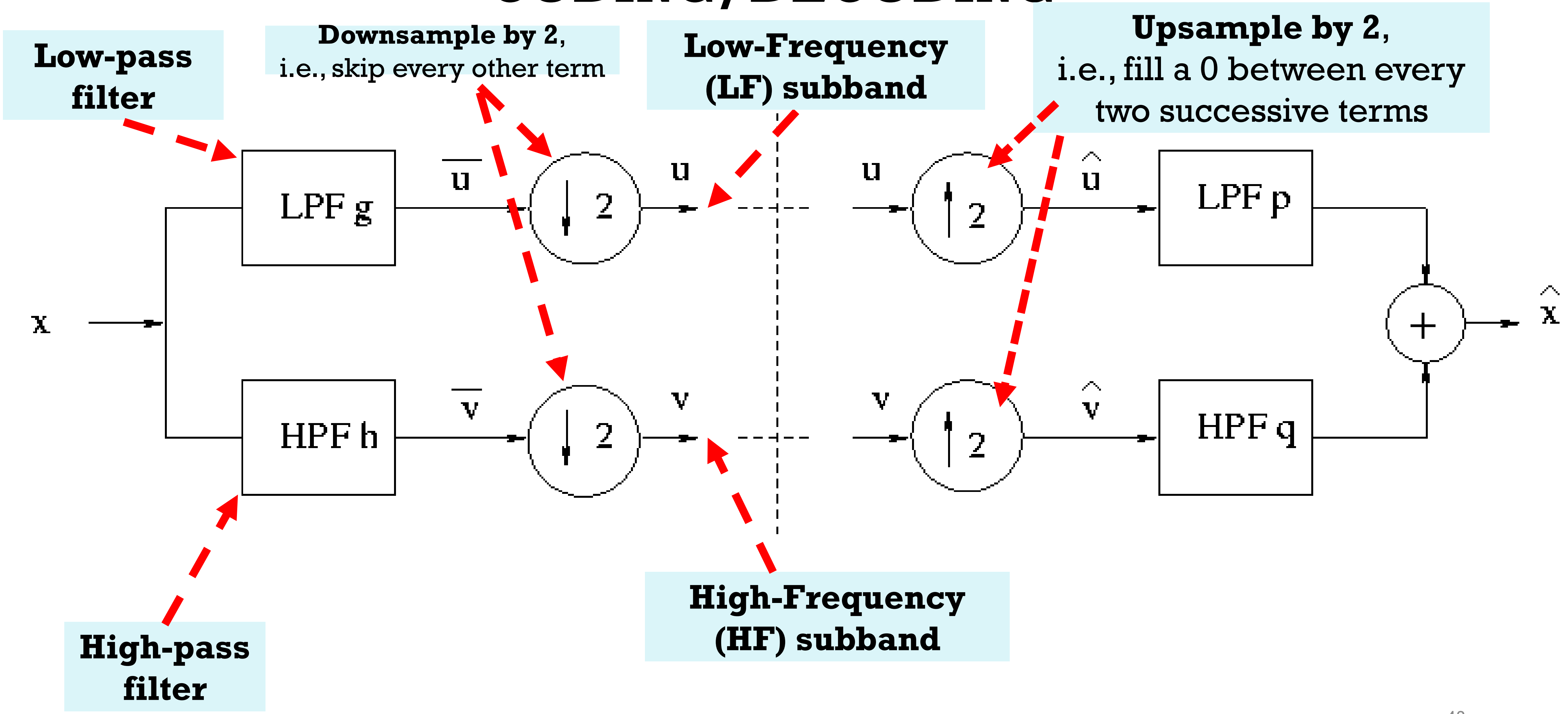


MATLAB CODE FOR THE PREVIOUS FIGURES

Code:

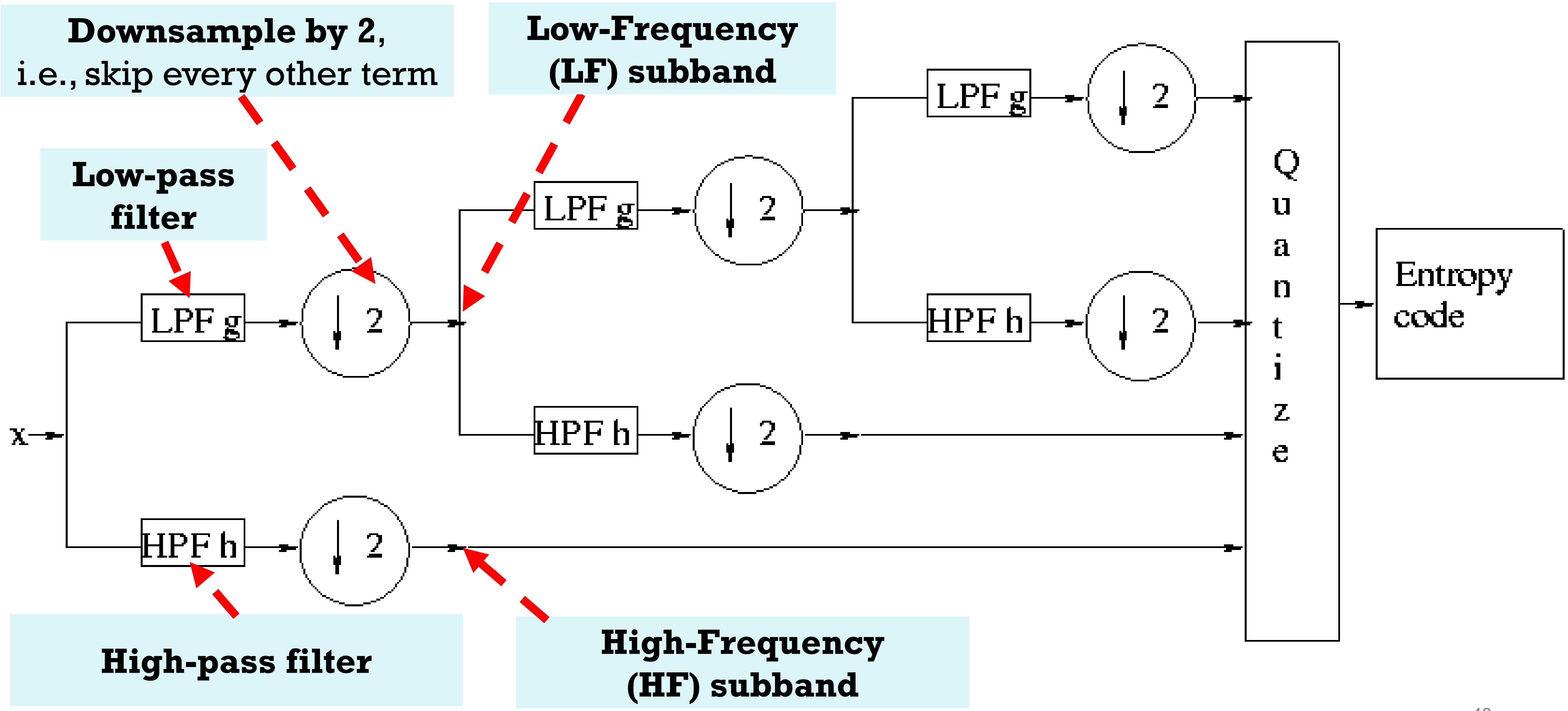
```
x=1:256;  xn=x;
xn(40:40:256)= x(40:40:256)+10;
subplot(2,2,1); plot(xn);title('original signal')
y=filter(g,[1],xn);   subplot(2,2,2);   plot(y);   title('low-pass filtered');ylim([-2,300])
z=filter(h,[1],xn);   subplot(2,2,3);    plot(z);   title('high-pass filtered');
subplot(2,2,4); plot(x,xn,x,y,x,z); legend('original','low-pass filtered', 'high-pass filtered')
text(10,260,'0.0267 -0.0169 -0.0782 0.2669 0.6029')
text(10,230,'0.2669 -0.0782 -0.0169 0.0267')
subplot(2,2,3); ylim([-5,10])
text(10, 9,'filter: 0.0456 -0.0288 -0.2956')
text(10, 7,'        0.5575 -0.2956 -0.0288 0.0456')
```

THE MAIN SCHEME OF SUBBAND CODING/DECODING



HOW SUBBAND CODING IS GENERALLY APPLIED

-- A TREE-LIKE STRUCTURE: THE ENCODER --



HOW SUBBAND CODING IS GENERALLY APPLIED
-- A TREE-LIKE STRUCTURE: THE DECODER --

Upsample by 2,
i.e., fill a 0 between every two successive terms

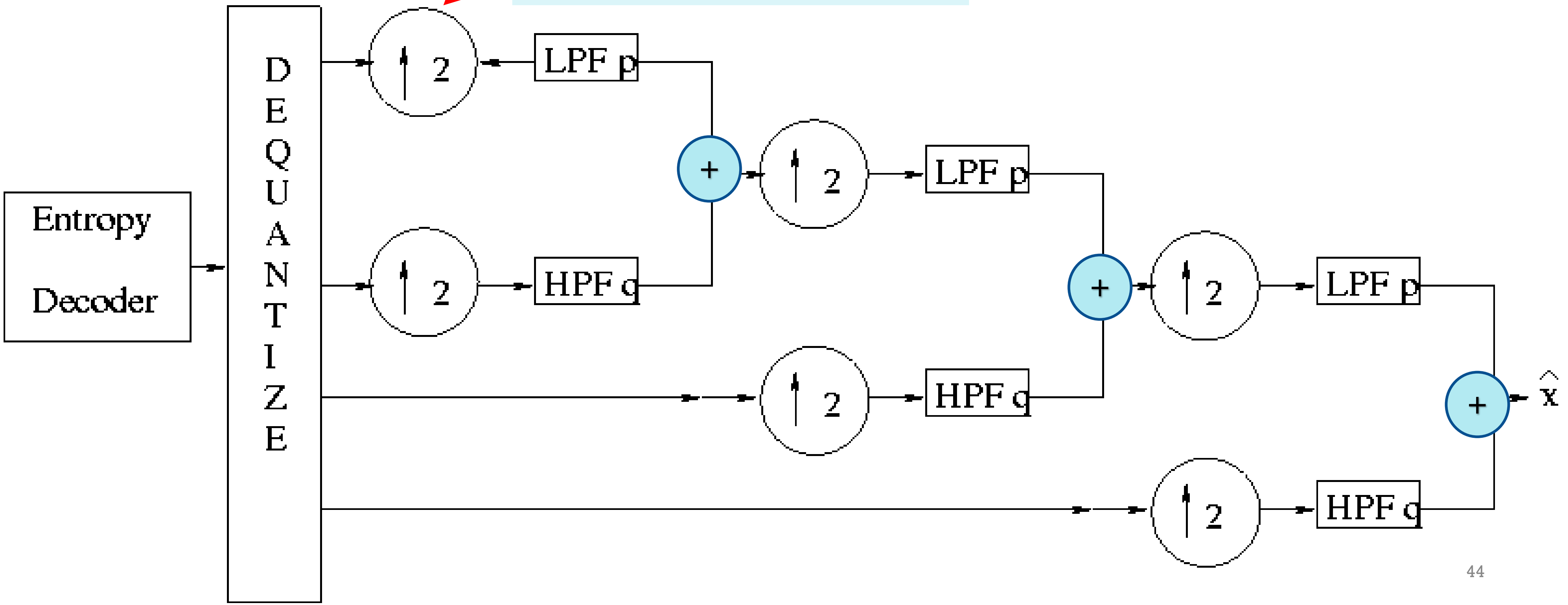
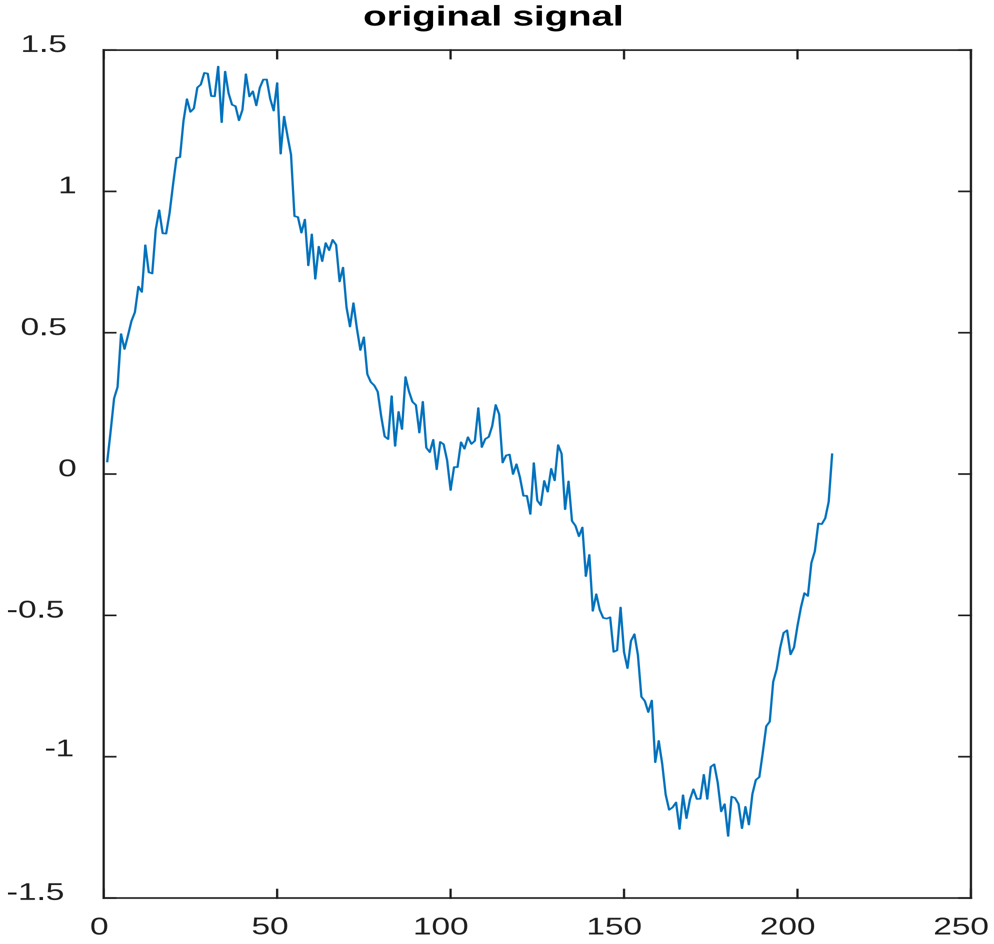


ILLUSTRATION OF SUBBAND CODING ON 1D SIGNALS (1/5)



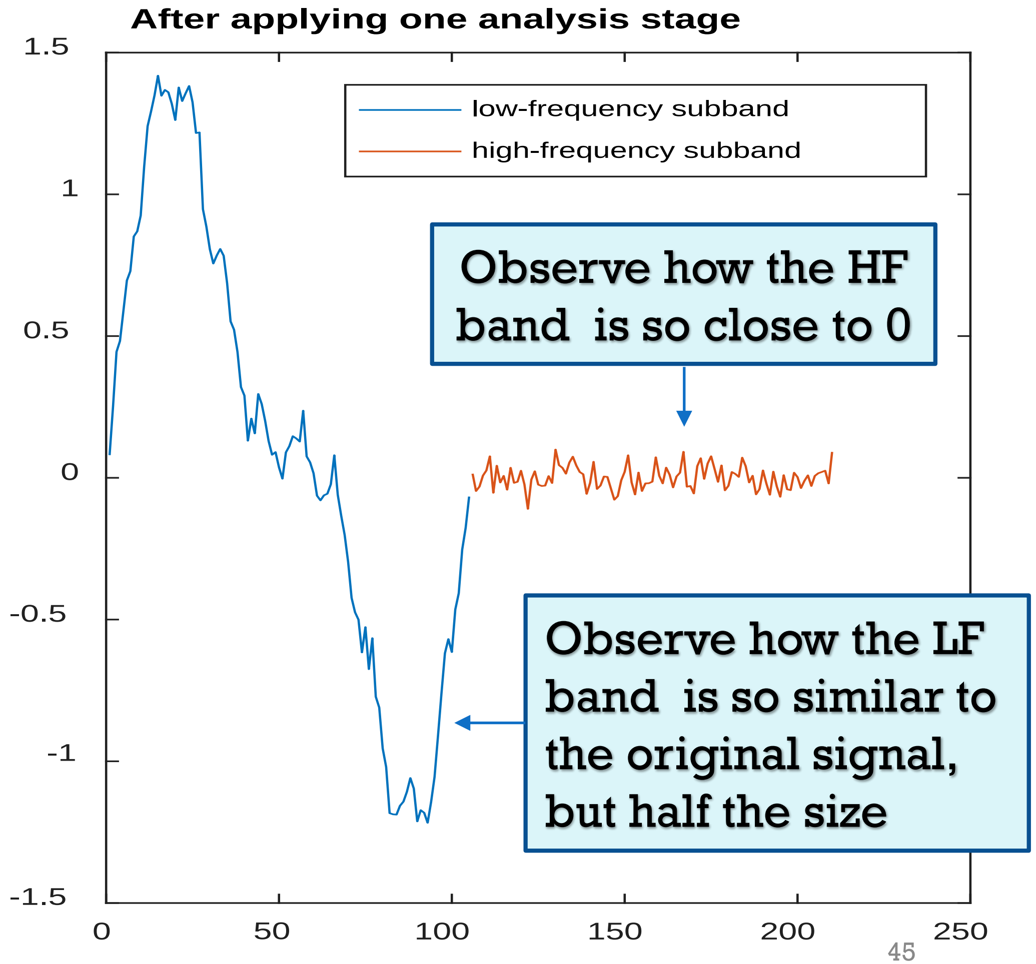
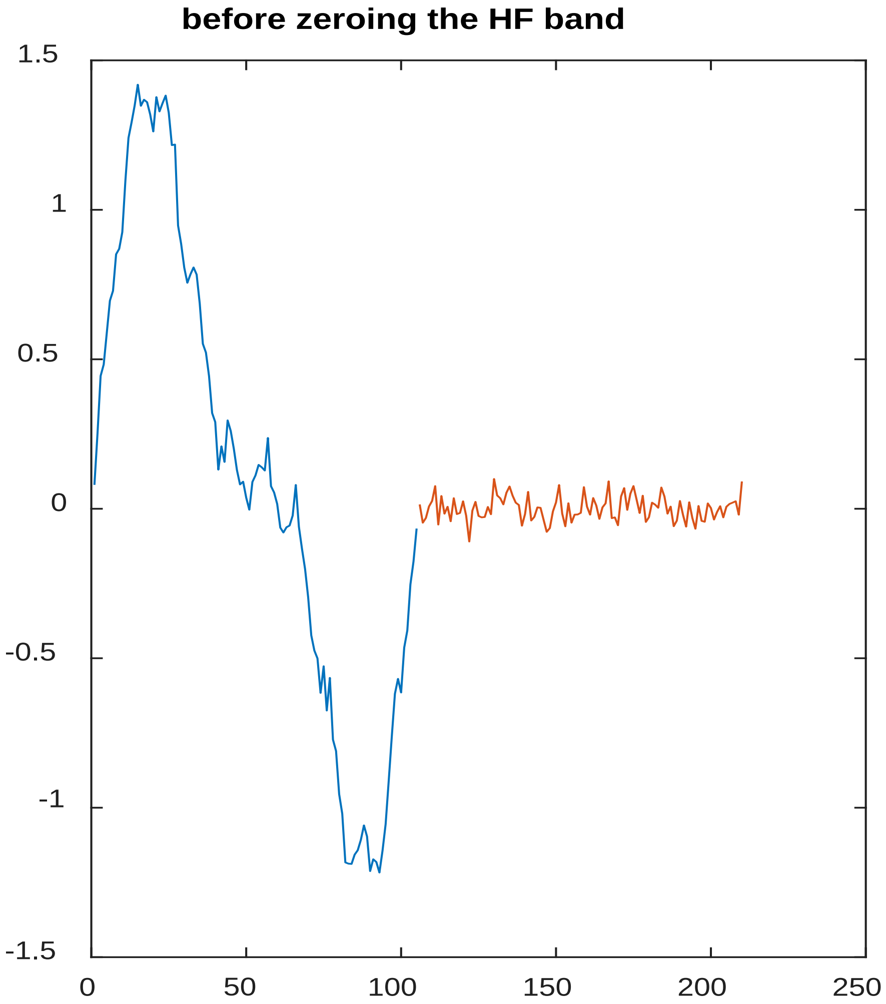


ILLUSTRATION OF SUBBAND CODING ON 1D SIGNALS (2/5)

- Zero out the HF band



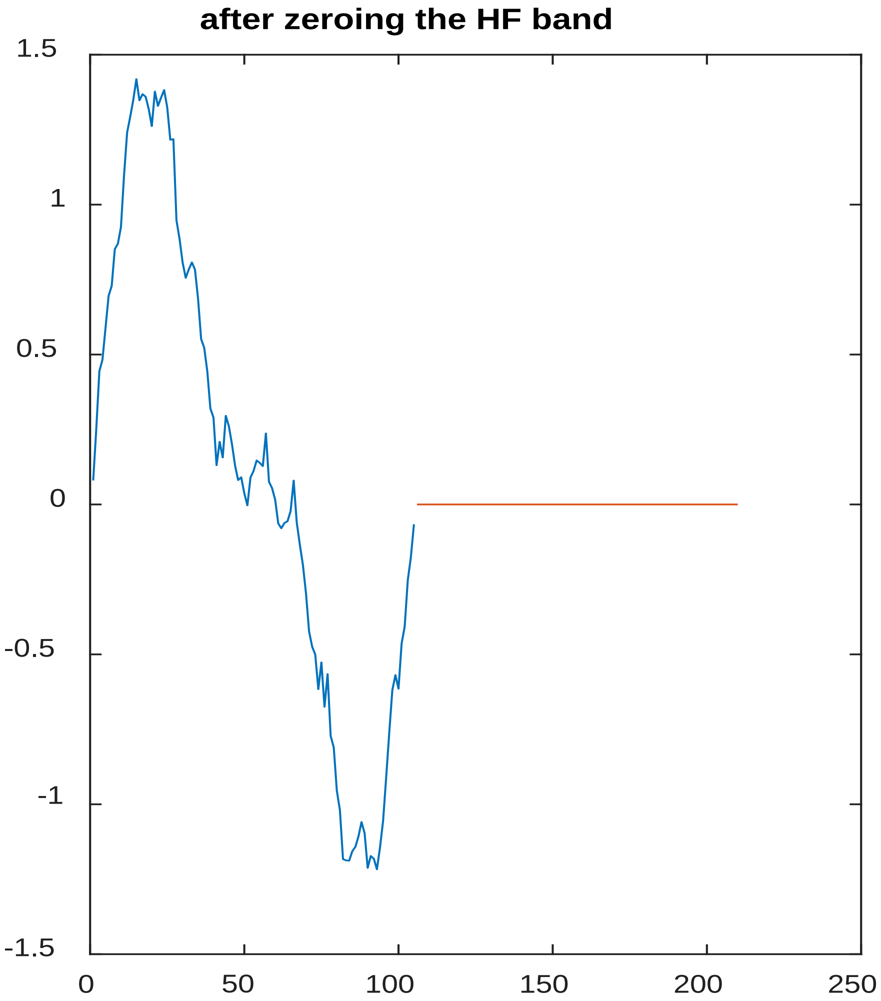
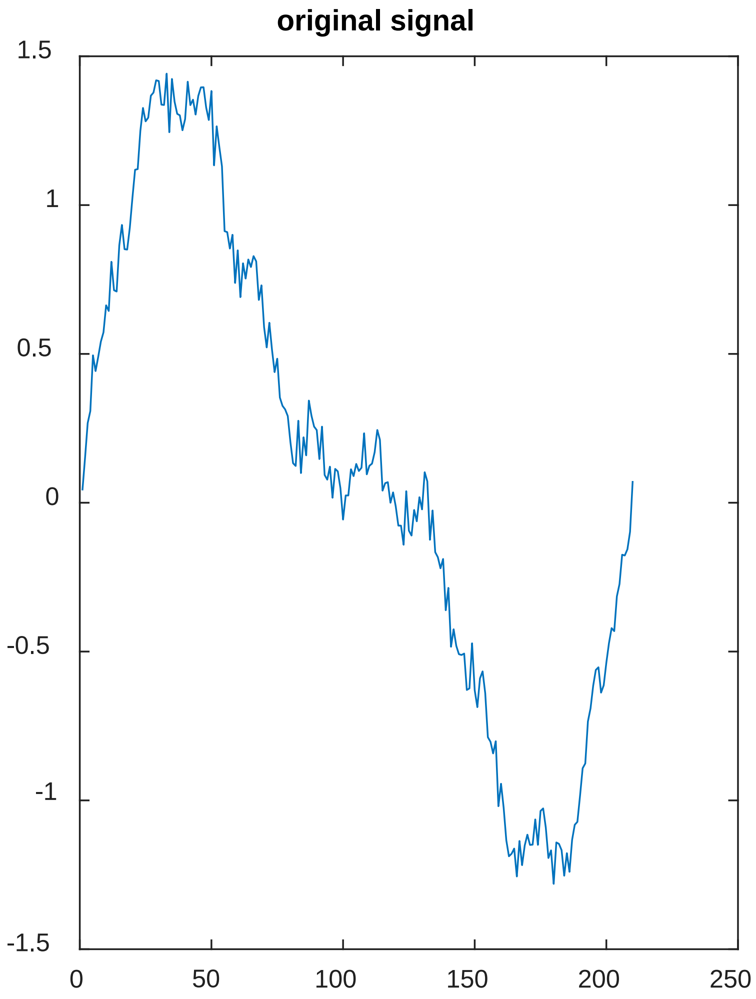


ILLUSTRATION OF SUBBAND CODING ON 1D SIGNALS (3/5)

- Reconstruct the data (the synthesis stage):



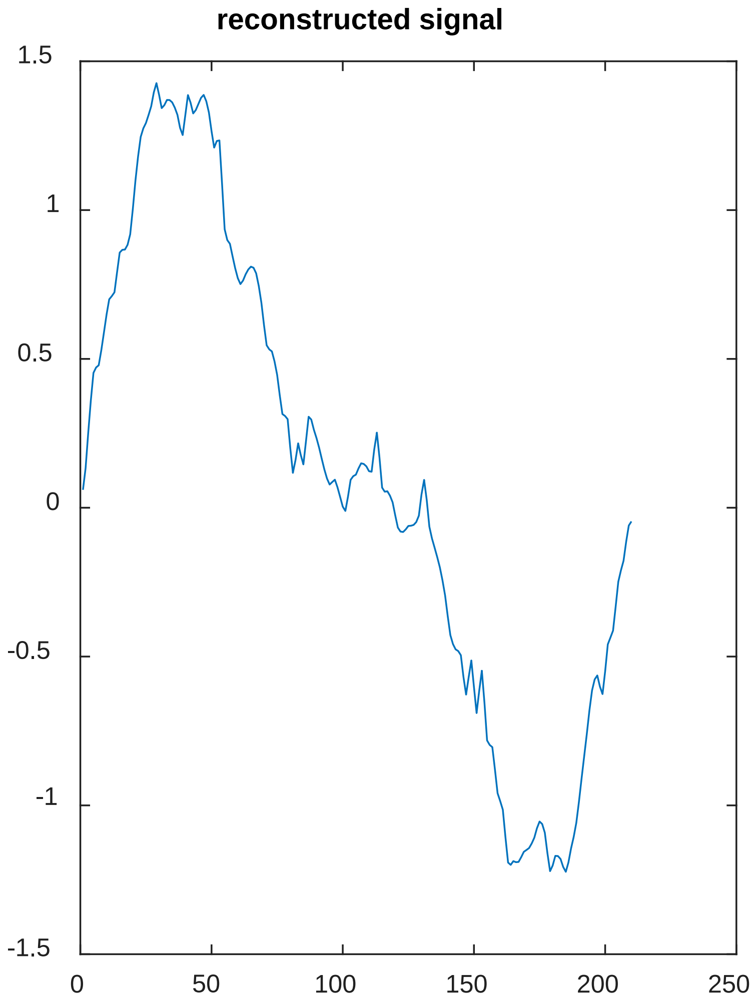


ILLUSTRATION OF SUBBAND CODING ON 1D SIGNALS (4/5)

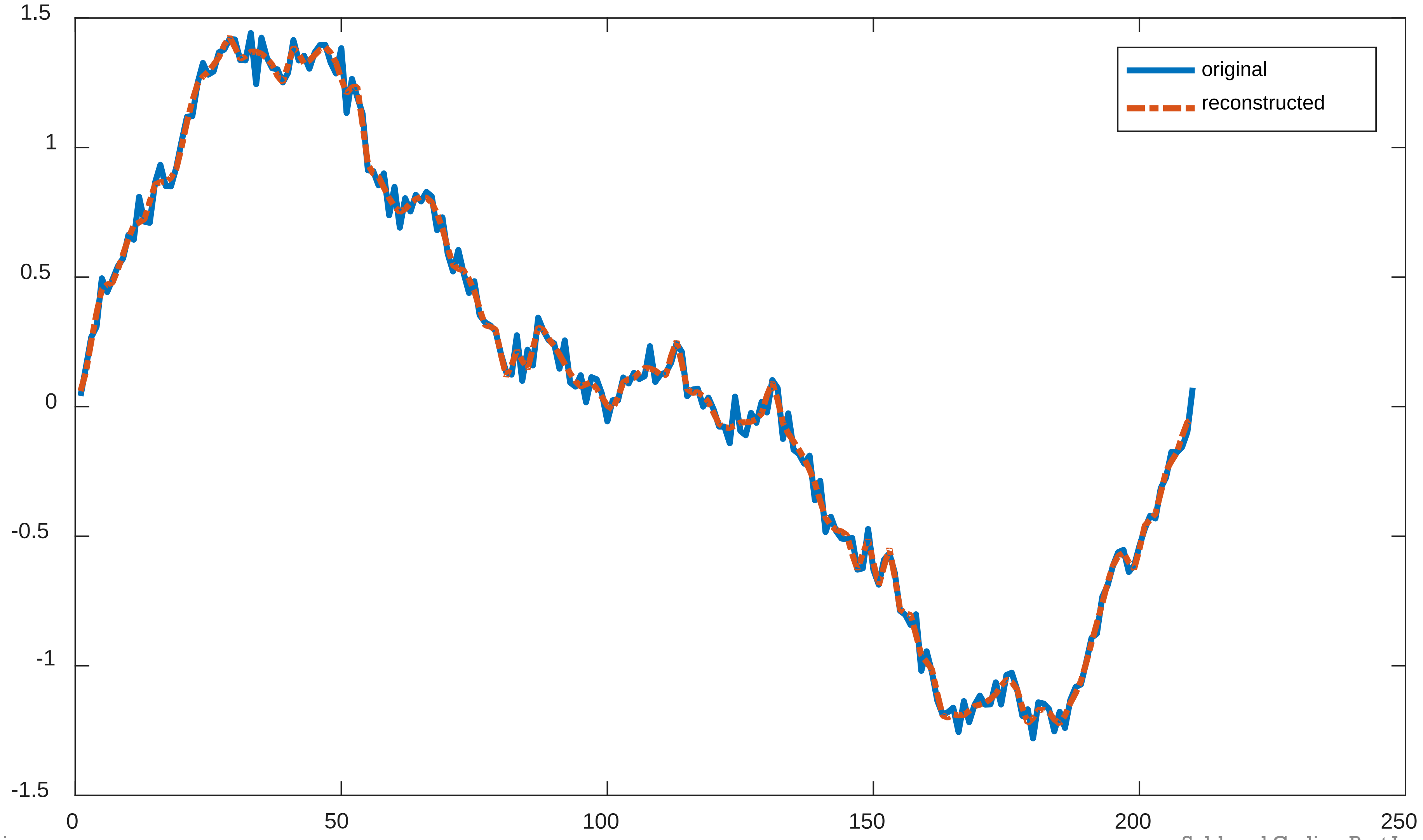


ILLUSTRATION OF SUBBAND CODING ON 1D SIGNALS (5/5)

- Measuring the error between the original and the reconstructed

- MSE: 0.0017

- Pretty good!

CODE FOR THE PREVIOUS ILLUSTRATIONS

Code for getting the LF subband and the HF subband:

```
figure
t=0:.03:2*pi;
x = sin(t) + 0.5*sin(2*t) + 0.1*sin(10*t) +
0.01*sin(100*t)+rand(1,length(t))/5;
subplot(1,2,1);
plot(x); title('original signal')
xlow=analfilter(x,g,0);
xhigh=analfilter(x,h,1);
subplot(1,2,2); plot(1:105,xlow,106:210,xhigh);
title('After applying one analysis stage')
legend('low-frequency subband', 'high-
frequency subband')
```

```
figure
subplot(1,2,1); plot(1:105,xlow,106:210,xhigh);
title('before zeroing the HF band')
subplot(1,2,2);
plot(1:105,xlow,106:210,zeros(1,105));
title('after zeroing the HF band')
```

```
figure
twobands=[xlow, zeros(1,105)];
xhat=synfilter(twobands,g,h);
subplot(1,2,1);
plot(x); title('original signal')
subplot(1,2,2);
plot(xhat); title('reconstructed signal')
```

```
figure
plot(1:210,x,'-', 1:210,xhat,'-.', 'LineWidth',2)
legend('original', 'reconstructed')

mse(x,xhat)
```

ILLUSTRATION OF SUBBAND CODING ON 2D SIGNALS

Original Lena

A one-level transform of Lena (Lena 1)

ILLUSTRATION OF SUBBAND CODING ON 2D SIGNALS

Original Lena

Lena reconstructed from just the low-frequency subband of Lena 1, zeroing out the 3 other bands (CR=4)

ILLUSTRATION OF SUBBAND CODING ON 2D SIGNALS

A one-level transform of Lena (Lena 1)

A two-level transform of Lena (Lena 2)

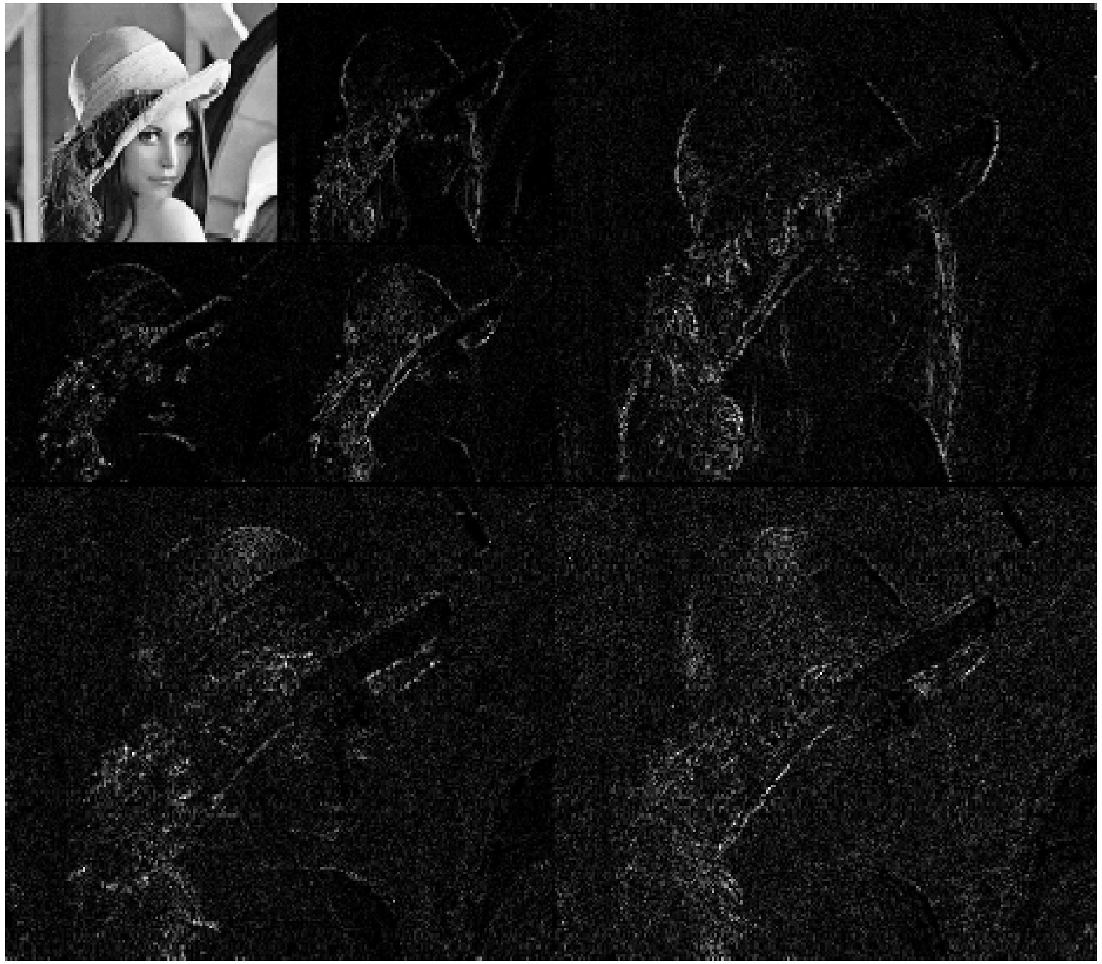


ILLUSTRATION OF SUBBAND CODING ON 2D SIGNALS

Original Lena

Lena reconstructed from just the low-frequency subband of Lena 2, zeroing out the 6 other bands (CR=16)

SUBBAND CODING ISSUES

- Filter design
- Quantization Method
- Shape of the tree
- Same or different filter sets per image or class of images?

FILTER DESIGN

- Classical filter design techniques for LPF's and HPF's
  - Least Mean Square technique
  - Butterworth technique
  - Chebychev technique
- Those techniques are for designing **<u>single</u>** filters, rather than a bank of four filters working together
- The four filters (g, h, p, q) for a subband coding system must have the ***perfect reconstruction*** property (to be seen later)
  - the output signal is identical to the input signal if no quantization takes place

NEXT LECTURE

- Continuation of subband coding
  - Filter design and perfect reconstruction
  - Quantization in the context of subband coding
  - Shape of the tree
  - Same or different filter sets per image or class of images?