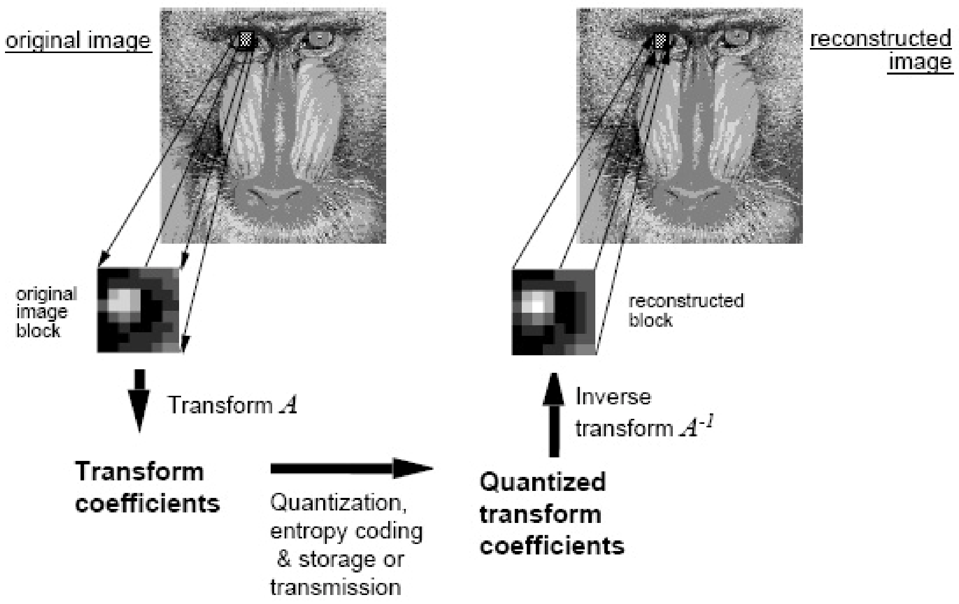
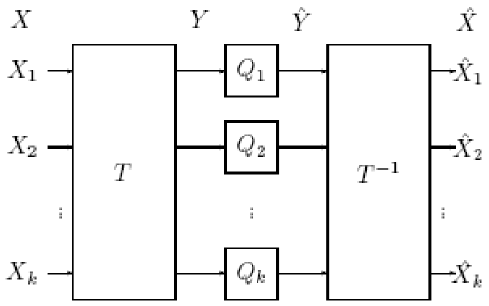
6. TRANSFORM CODING TECHNIQUES


original image
original image block
Transform A
Transform coefficients
Quantization, entropy coding & storage or transmission
Quantized transform coefficients
Inverse transform A^{-1}
reconstructed block
reconstructed image

General idea: Coding, compression, and other key processing operations are performed after a transformation where things are more favorable for the task. This is achieved by collecting samples into vector of dimension *k* that is multiplied by a transformation matrix *T* and the resulting coefficients are processed either as a vector or individually as scalars. Processing could be filtering, feature extraction, and compression. Dominant transformations are Fast Fourier transform (FFT) for filtering, Discrete Cosine Transform (DCT) and Wavelet Transforms (WT) for image compression, Subband Coding (SBC), Linear Predictive Coding (LPC) are used for speech compression.



Represent *k* samples (or pixels) of input as a column vector: $X = (X_1, X_2, \cdots, X_k)^t$, where *t* is the usual transpose operation. Let *T* be a *k*x*k* invertible matrix. Then the communication problem can be formulated as;

$$Y = T(X), \qquad \hat{Y} = Q(Y), \qquad \hat{X} = T^{-1}(\hat{Y}) \tag{6.1}$$

- It is important to note that MSE same in both domains (before and after the transformation), which ensures that inverse transform will not amplify quantization error.
- Determinant of the autocorrelation matrix of the input vector is unchanged:

$$R_Y = E[Y.Y^t] = T.E[X.X^t].T^* = T.R_X.T^*; \quad |\det(T)| = 1; \quad \det(R_X) = \det(Y_X) \tag{6.2}$$

In most applications, we use a linear transform to represent a signal in another domain. In this case, the formulation is:

$$y_n = \sum_{i=0}^{N-1} a_{n-i}.x_i \qquad (6.3a)$$

and its inverse

$$x_n = \sum_{i=0}^{N-1} b_{n-i}.y_i \qquad (6.3b)$$

But it is more common to use the matrix form:

$$\mathbf{Y=A.X} \qquad (6.4a)$$

$$\mathbf{X=B.Y} \qquad (6.4b)$$

Where **A** and **B** are NxN matrices.

In the case of two-dimensional linear transforms for a block size of NxN we have:

$$y_{k,l} = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} a_{i,j,k,l}.x_{i,j} \qquad (6.5a)$$

It should be noted that all 2-D transformations in applications in use today are "***separable***" transforms; i.e., we take the transform of a 2-D block by first taking the transform along one dimension, then repeating the operation along the other direction. In matrix terminology, we take the 1-D transform along rows, and later taking the column-by-column transform of the result in the first step. The inverse also is guaranteed to exist for linear transforms:

$$x_{i,j} = \sum_{k=0}^{N-1}\sum_{l=0}^{N-1} b_{i,j,k,l}.y_{k,l} \qquad (6.5b)$$

In matrix notation, we have:

$$\mathbf{Y=AXA}^T \qquad (6.6a)$$

$$\mathbf{X=BYB}^T \qquad (6.6b)$$

Furthermore, all transforms in signal/image processing are "**orthonormal transforms**" with the property:

$$\mathbf{B=A^{-1}=A^T} \qquad (6.7)$$

For an orthonormal transform, the inverse operation is significantly simplified to:

$$\mathbf{X= A^T\ YA} \qquad (6.8)$$

It is important to note that orthonormal transforms are "**energy preserving**:"

$$\sum_{n=0}^{N-1} x_n^2 = \sum_{i=0}^{N-1} y_i^2 \qquad (6.9)$$

Typical transforms commonly used:

- Fourier
- DCT
- Wavelet
- Karhunen-Loeve
- Hotelling, Hartley, and Hadamard
- Principal Value

DFT and its Inverse

DFT: It is a transformation that maps an *N*-point Discrete-time (DT) signal *x*[*n*] into a function of the N complex discrete harmonics. That is, given $x[n]; \quad n = 0,1,2,\cdots,N-1$, an *N*-point Discrete-time signal *x*[*n*] then DFT is given by **(analysis equation):**

$$X(k) = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}nk} \qquad for \quad k = 0,1,2,\cdots,N-1 \tag{6.10a}$$

and the inverse **DFT (IDFT)** is given by **(synthesis equation):**

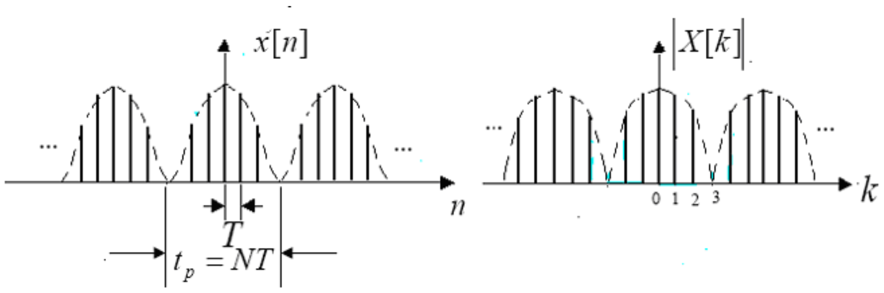
$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X(k) e^{+j\frac{2\pi}{N}nk} \qquad for \quad n = 0,1,2,\cdots,N-1 \tag{6.10b}$$

1. These two equations form DFT pair.
2. They have N-point resolution both in the discrete-time domain and discrete-frequency domain.
3. Always the scaling factor $1/N$ is associated with the synthesis equation (inverse DFT).
4. $X(k)$ is periodic in N or equivalently in $\Omega_k = 2\pi/N$; that is,

$$X(k) = X(\Omega_k) = X(\Omega_k + 2\pi) = X(\frac{2\pi}{N}(k+N)) = X(k+N) \tag{6.11a}$$

5. $x[n]$ determined from (7.10) is also periodic in N;

$$x[n] = x[n+N] \tag{6.11b}$$



Matrix Representation of DFT

Write the variables involved in matrix form: $x = [x[0], x[1], \cdots, x[N-1]]^T$ and $W_N = e^{-j2\pi/N}$:

$$X(k) = \sum_{n=0}^{N-1} x[n].W_N^{kn} \qquad for \;\; k = 0,1,2,\cdots,N-1 \tag{6.12}$$

Then the weight matrix is simply:

$$W = \begin{bmatrix} W_N^0 & W_N^0 & W_N^0 & \cdots & W_N^0 \\ W_N^0 & W_N^1 & W_N^2 & \cdots & W_N^{N-1} \\ \vdots & \vdots & & \cdots & \vdots \\ W_N^0 & W_N^{N-1} & W_N^{2(N-1)} & \cdots & W_N^{(N-1)(N-1)} \end{bmatrix} \tag{6.13}$$

which is normally referred as DFT matrix and the resulting transform vector becomes:

$$X = [X(0), X(1), \cdots, X(N-1)]^T = W.x \qquad (6.14a)$$

- $[W]_{nk} = [W]_{kn}$ (6.14b)
- $W = W^T$ (6.14c)
- $W_N^{-1} = W_N^*$; where * stands for the complex conjugate. (6.14d)

With these we can write the inverse DFT (IDFT) as follows:

$$x[n] = \frac{1}{N}\sum_{n=0}^{N-1} X(k).W_N^{-kn} \quad for \;\; n = 0,1,2,\cdots,N-1 \qquad (6.15a)$$

$$x = \frac{1}{N}W^*.X = W^{-1}.X \qquad (6.15b)$$

$$W^{-1} = \frac{1}{N}.W^* \qquad W^*.W = N.I_N \qquad \text{with } I_N : NxN\ identity\ matrix \qquad (6.15c)$$

Example 6.1: Consider the discrete-time representation of a cosine signal:

$$x[n] = A.Cos(\frac{2\pi}{N}n) = 0.5.A.(e^{j\frac{2\pi}{N}n} + e^{-j\frac{2\pi}{N}n})$$

$$X(k) = A.\sum_{n=0}^{N-1}(\frac{e^{j\frac{2\pi}{N}n} + e^{-j\frac{2\pi}{N}n}}{2}).e^{-j\frac{2\pi}{N}nk} = \frac{A}{2}\sum_{n=0}^{N-1} e^{j\frac{2\pi}{N}n(1-k)} + \frac{A}{2}\sum_{n=0}^{N-1} e^{-j\frac{2\pi}{N}n(1+k)}$$

$$X(k) = \frac{A}{2}\sum_{n=0}^{N-1}(e^{j\frac{2\pi}{N}(1-k)})^n + \frac{A}{2}\sum_{n=0}^{N-1}(e^{-j\frac{2\pi}{N}(1-k)})^n$$

$$= \frac{A}{2}\frac{1-e^{j2\pi(1-k)}}{1-e^{j\frac{2\pi}{N}(1-k)}} + \frac{A}{2}\frac{1-e^{j2\pi(1+k)}}{1-e^{j\frac{2\pi}{N}(1+k)}} \qquad for\ k = 0,1,2,\cdots,N-1$$

- First term is 0 when $k \neq 1$ and it is $NA/2\ for\ k = 1$.
- Second term is 0 when $k \neq N-1$ and it is $NA/2\ for\ k = N-1$.
- Final result becomes: $X(k) = \frac{NA}{2}[\delta(k-1) + \delta(k-(N-1))]$

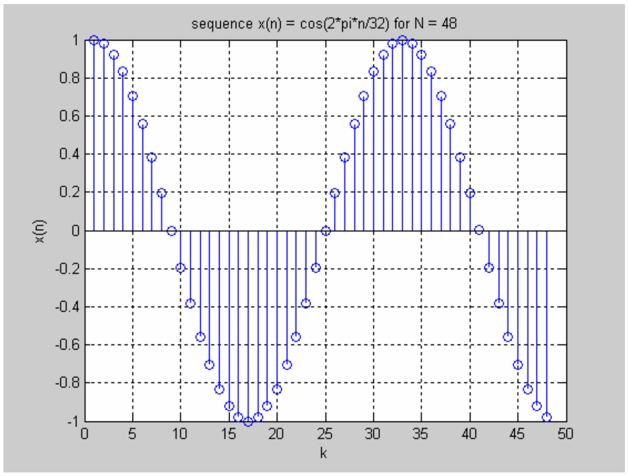
where the first term represents the positive frequency term and the other one is the mirror image as expected.

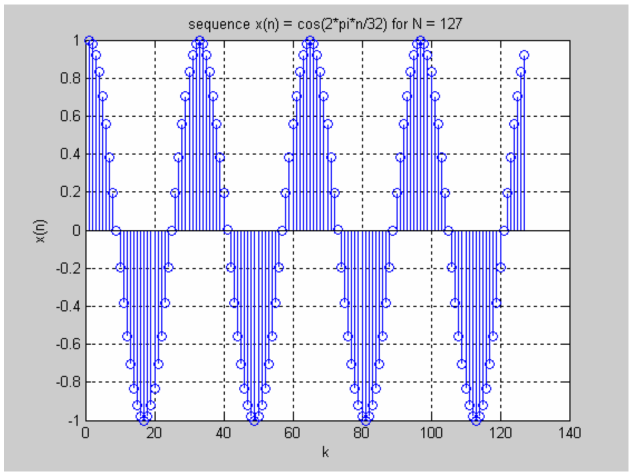
Now let us try to implement that using DFT with 48-points and 127-point of sampled versions:

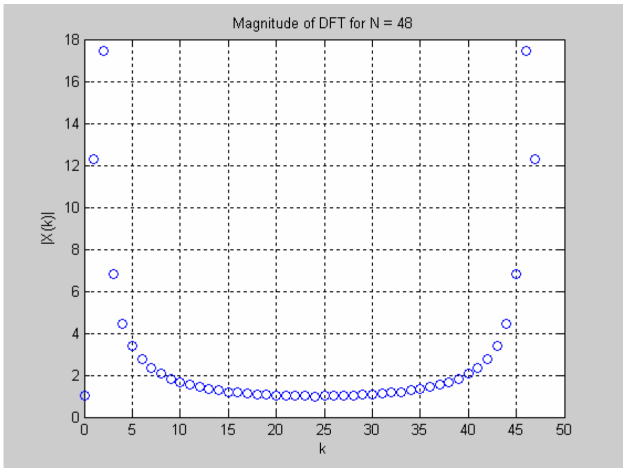
% E1xample 6.1 Sampled Cosine with N=48	% Now try it with
N = 48;	N = 127;
n =[0:1:N-1];	n =[0:1:N-1];
k = n; M = 32;	k = n; M = 32;
xn = cos(2*pi*n/M);	xn = cos(2*pi*n/M);
Xk = fft(xn);	Xk = fft(xn);
magXk = abs(Xk);	magXk = abs(Xk);
PhaseXk = angle(Xk);	PhaseXk = angle(Xk);
% Plots	% Plots
axis([1 2 3 4]); axis;	axis([1 2 3 4]); axis;
stem(xn); xlabel('k');ylabel('x(n)');	stem(xn); xlabel('k');ylabel('x(n)');
title('sequence x(n) = cos(2*pi*n/32) for N = 48');	title('sequence x(n) = cos(2*pi*n/32) for N = 127');
grid; figure; axis([0,47,0,18]);	grid; figure; axis([0,47,0,18]);

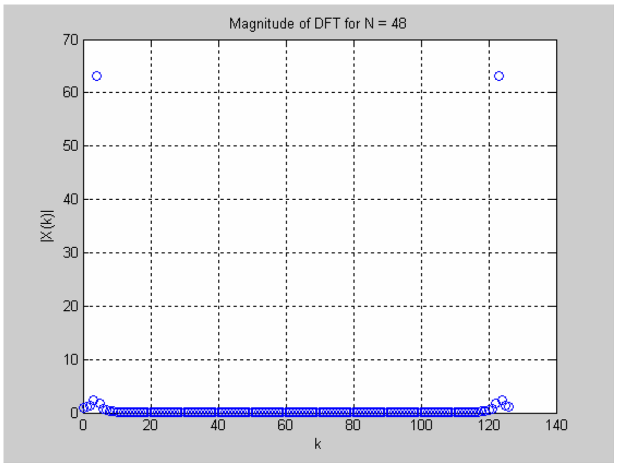
```
plot(n,magXk,'o');
xlabel('k');ylabel('|X(k)|');
title('Magnitude of DFT for N = 48'); grid;
figure; axis([0,47,-2,2]);
plot(n,PhaseXk,'*');
xlabel('k');ylabel('|X(k)|');
title('Phase of DFT for N = 48');
grid;axis;
```

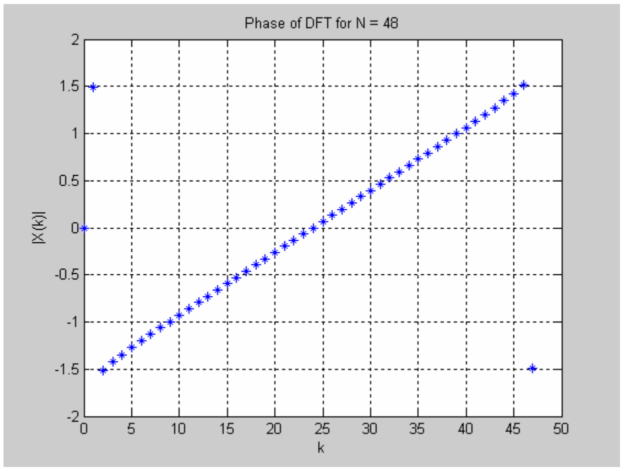
```
plot(n,magXk,'o');
xlabel('k');ylabel('|X(k)|');
title('Magnitude of DFT for N = 48');grid;
figure; axis([0,47,-2,2]);
plot(n,PhaseXk,'*');
xlabel('k');ylabel('|X(k)|');
title('Phase of DFT for N = 127');
grid;axis;
```

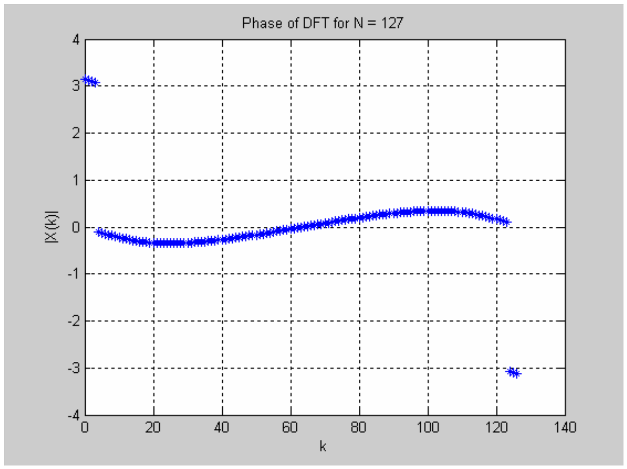












Even though we were expecting two harmonics we have ended up plots needs some explanation:

- Sampled signals are becoming more like a continuous signal as N increases.
- Magnitude plots are not located at two harmonics but expand over the frequency range, which is called "Leakage" in the business.
- Symmetry is respect to the center of the plots rather than 2-sided spectral halves, which is due to the periodic behavior of the DFT.

Fast Fourier Transform (FFT): FFT since its introduction by Cooley-Tukey almost a half century ago has been playing historically sustained significant role in the development of DSP since it is the most widely used "fast algorithm" in solving many engineering challenges, designing filters, performing spectral analysis, estimation, noise cancellation and benchmark testing devices and systems, etc. Also, it is also very readily useable for computing the inverse transforms. Consider the definition of DFT in (6.10)

$$X(k)=\sum_{n=0}^{N-1}x[n]e^{-j\frac{2\pi}{N}nk}=\sum_{n=0}^{N-1}x[n]\{Cos(\frac{2\pi}{N}nk)-jSin(\frac{2\pi}{N}nk)\}\quad for\quad k=0,1,2,\cdots,N-1 \qquad (6.16)$$

To appreciate its computational efficiency let us define:

$$\begin{aligned}1\,OP &= 1\,Complex\ Multiply + 1\,Complex\ Add\\ &= 1\,Cos\ Multiply + 1\,Sin\ Multiply + 1\,Complex\ Add + 1\,Complex\ Accumulate\end{aligned} \qquad (6.17)$$

- To compute N-Point DFT we need $N^2\ OPS$.
- Cooley-Tukey basic FFT algorithm requires $N.\log_2 N\ OPS$.

m	$N=2^m$	DFT OPs	Cooley-Tukey FFT OPs	FFT Savings
1	2	4	2	50%
2	4	16	8	50%
3	8	64	24	62.5%
4	16	256	64	75%
5	32	1024	160	84.4%
6	64	4096	384	90.6%
7	128	16384	896	94.5%
8	256	65536	2048	96.9%
9	512	262144	4608	98.2%
10	1024	1048576	10240	99.0%

- 99% savings in computational complexity is unheard of in any other fast algorithm in science. Even for reasonable and frequently observed FTT sizes of 128 or 256-points FFT we are in the 90% savings range.

DCT and its Inverse

Let us introduce the discrete cosine transform and its inverse based on 8x8 image pixel blocks:

$$F(u,v)=\frac{1}{4}.C(u).C(v)[\sum_{x=0}^{7}\sum_{y=0}^{7}f(x,y).Cos[\frac{(2x+1)u\pi}{16}].Cos[\frac{(2y+1)v\pi}{16}]] \qquad (6.18a)$$

$$f(x,y)=\frac{1}{4}.C(u).C(v)[\sum_{u=0}^{7}\sum_{v=0}^{7}F(u,v).Cos[\frac{(2x+1)u\pi}{16}].Cos[\frac{(2y+1)v\pi}{16}]] \qquad (6.18b)$$

$$C(u),\,C(v)=\begin{cases}1/\sqrt{2} & for\ u,v=0\\ 1 & Otherwise\end{cases} \qquad (6.18c)$$

Why DCT so important?

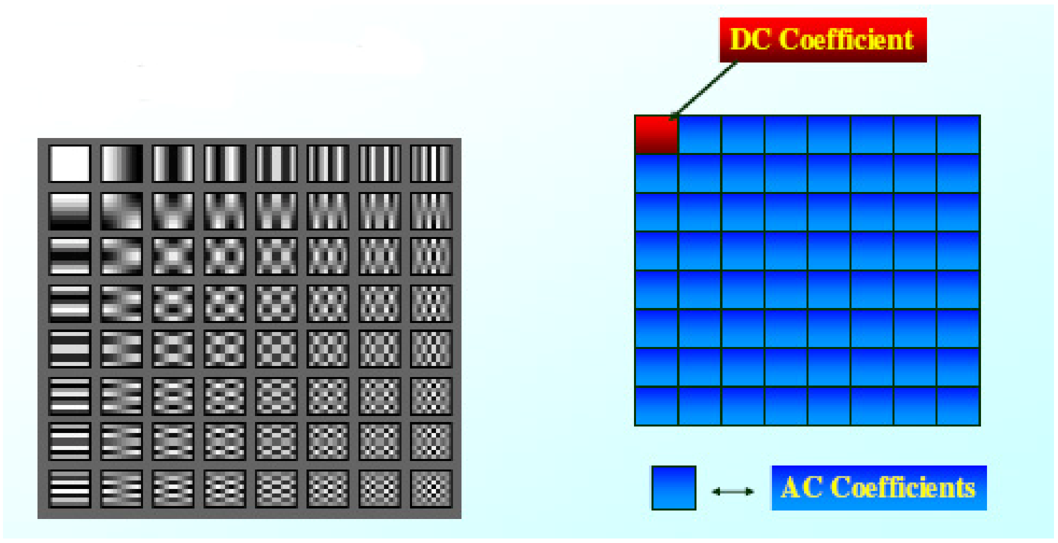
- It is real, and can be computed by an FFT. It has excellent energy compaction for highly correlated data.
- It won in the open competition for early international image compression standards (1992-93 JPEG standard).
- Once we have the transform, the question is how to quantize & code them?

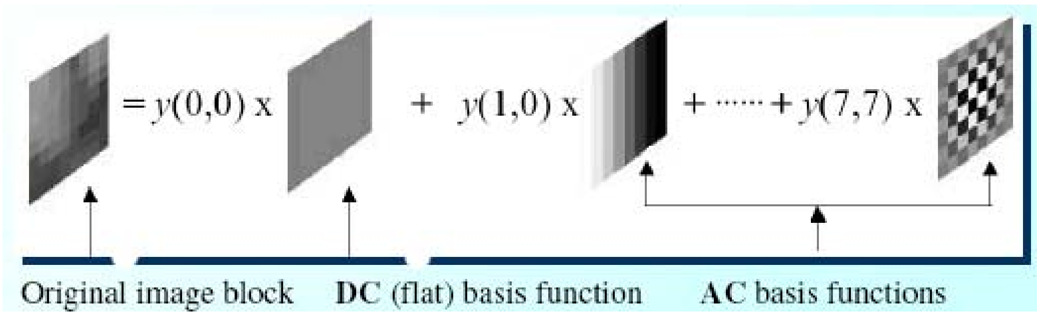
Several options with two broad classes:

1. Scalar quantization: Simple and pretty good if the encoding rate is high and we normally use subsequent entropy coding. Only need to pick the number of quantizer size for each quantizer.

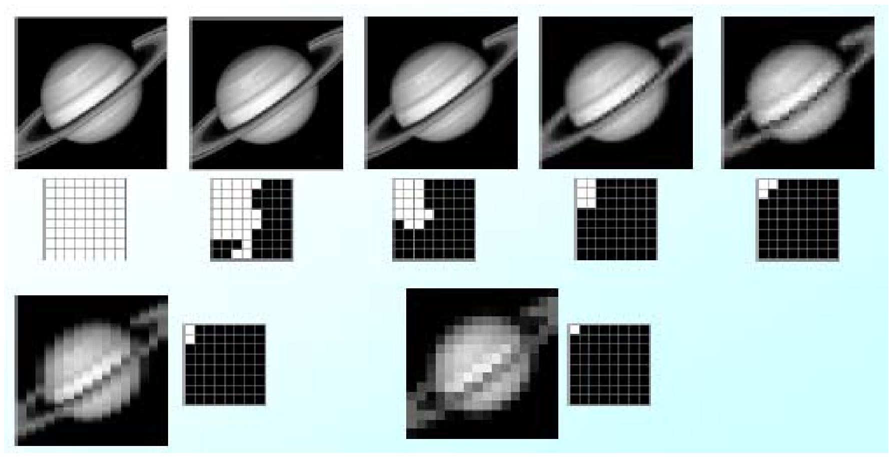
2. Vector quantization: Quantize groups (subbands, wavelets) as vectors (complicated); followed by a runlength coding scheme and an entropy coder.

Example 6.2: 64-point (8x8) DCT basis functions: DCT coefficients can be viewed as weighting functions that, when applied to the 64 cosine basis functions of various spatial frequencies (8 x 8 templates), will reconstruct the original block.





Example 6.3: Let us explore the above concept for encoding the image of Saturn (Matlab demo) keeping and encoding only a group of basis functions (white squares).



Example 6.4: Let us explore the impact of rate, basis functions and the block size on DCT using VcDemo on sample images including Lena. Details of compression using DCT on Lena:

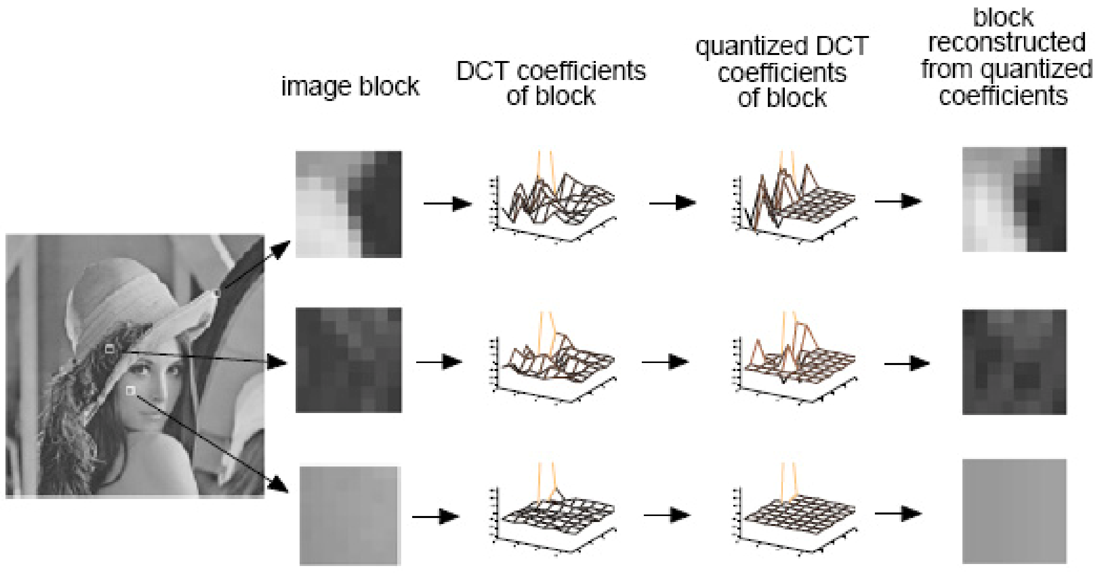
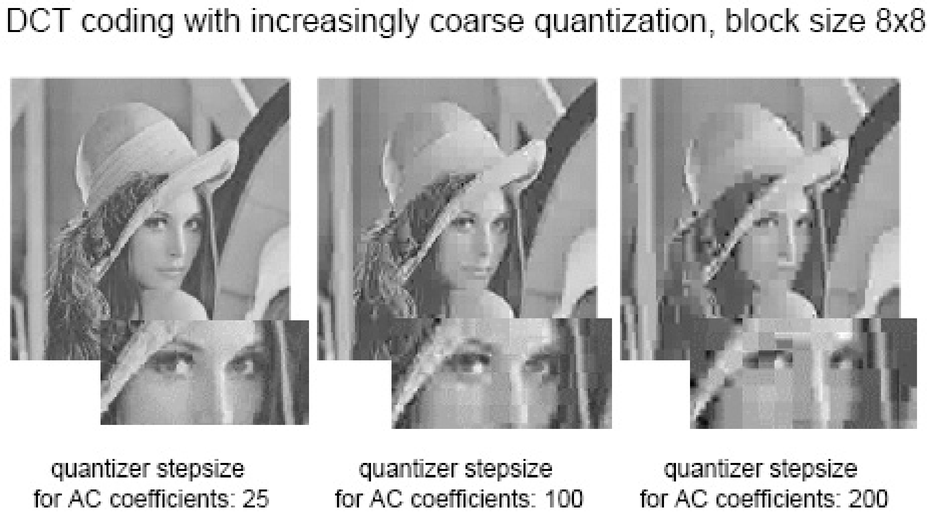


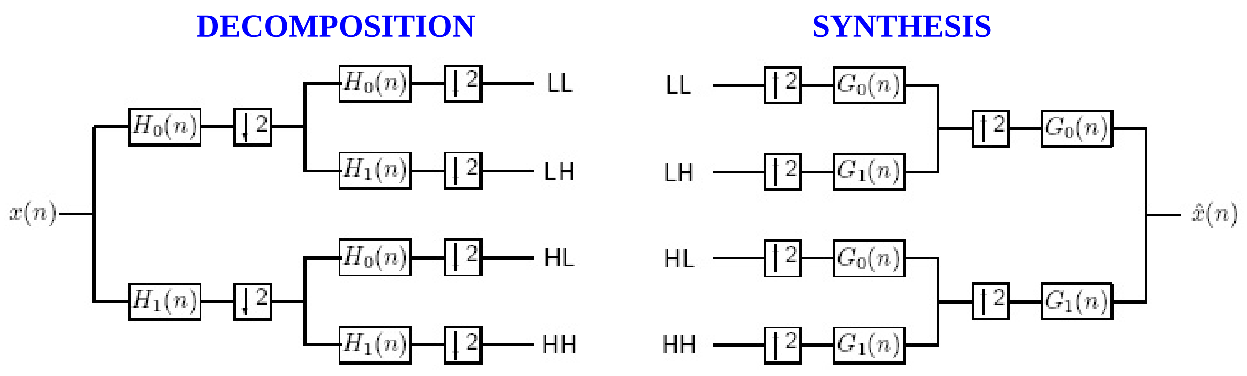
image block
DCT coefficients of block
quantized DCT coefficients of block
block reconstructed from quantized coefficients

Typical artifacts of DCT:


DCT coding with increasingly coarse quantization, block size 8x8
quantizer stepsize for AC coefficients: 25
quantizer stepsize for AC coefficients: 100
quantizer stepsize for AC coefficients: 200

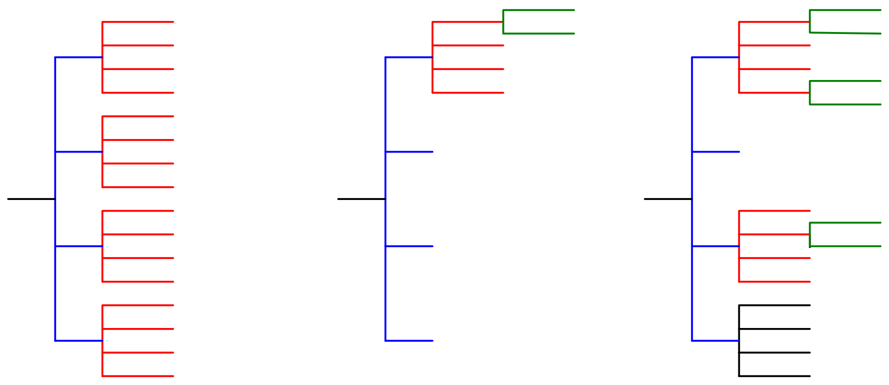
Subband and Wavelet Transforms and Coding

General Idea: Instead of using block transforms, we pass the input image (signal) through a filter bank with approximately non-overlapping pass bands. A form of transform of an entire image as a single block, but implemented by locally sliding-block linear filters. These filters are known as "**quadrature mirror filters** (QMF)" and have exact reconstruction property in the absence of compression (quantization) as depicted below.



Example 6.5: Let us explore the impact of bit rate, number of bands and types of decomposition, and the filter size on subband decomposition using VcDemo on sample images including Lena.

Alternately, we can use pyramid decomposition using wavelet transforms.

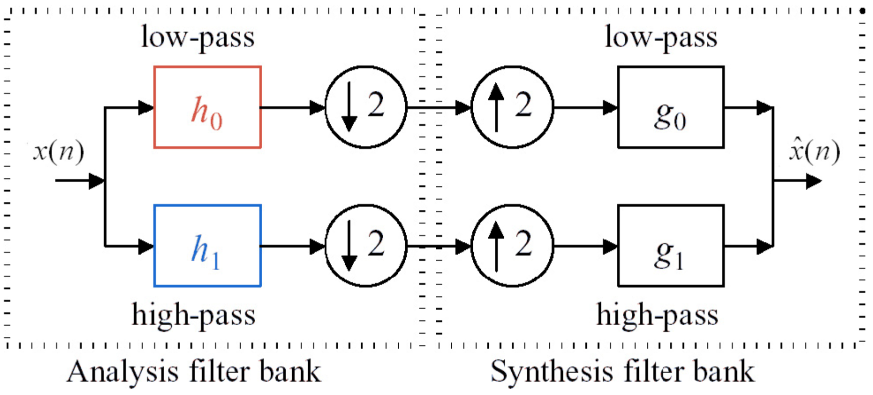


2-stages of uniform decomposition leading to 16 uniform subbands (left); pyramidal decomposition leading to 10 octave-band subbands (incomplete tree) (middle), and a wavelet decomposition (right).

Discrete Wavelet Transform (DWT) and Compression

One-Dimensional DWT decomposes a one-dimensional (1-D) sequence (e.g., line of an image or a segment of speech into two sequences called subbands, each with half the number of samples, according to the following procedure:

- 1-D sequence is separately low-pass and high-pass filtered.
- Filtered signals are down-sampled by a factor of two to form the low-pass and high-pass subbands.
- This process is called the wavelet transform analysis stage.
- In the absence of compression or channel errors, the synthesis can recover an exact copy of the original data.



The decomposition and synthesis process in the z-domain (after z-transform) results in:

$$\hat{X}(z) = \frac{1}{2}[h_0(z).g_0(z) + h_1(z).g_1(z)].X(z) + \underbrace{\frac{1}{2}[h_0(-z).g_0(z) + h_1(-z).g_1(z)].X(-z)}_{\textbf{Aliasing distortion}} \qquad (6.19)$$

Aliasing will be cancelled and the exact recovery will take place if these filters have the following properties:

$$g_0(z) = h_1(-z) \qquad \text{and} \qquad -g_1(z) = h_0(-z) \qquad (6.20)$$

Example 6.6: LeGall filters for two-channel filter bank with perfect reconstruction.

Impulse responses of analysis filters:

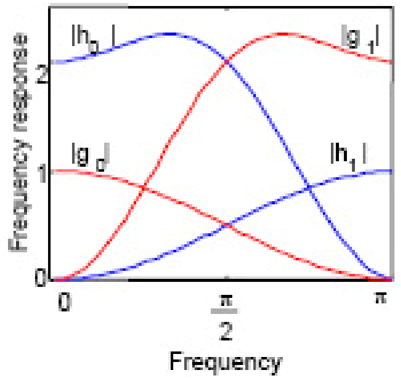
LP filter: $h_0(n) = -0.25\delta(n+2) + 0.5\delta(n+1) + 1.5\delta(n) + 0.5\delta(n-1) - 0.25\delta(n-2)$ (6.21a)

HP filter: $h_1(n) = 0.25\delta(n+1) - 0.5\delta(n) + 0.25\delta(n-1)$ (6.21b)

Impulse responses of synthesis filters:

LP filter: $g_0(n) = 0.25\delta(n+1) + 0.5\delta(n) + 0.25\delta(n-1)$ (6.22a)

HP filter: $g_1(n) = 0.25\delta(n+2) + 0.5\delta(n+1) - 1.5\delta(n) + 0.5\delta(n-1) - 0.25\delta(n-2)$ (6.22b)

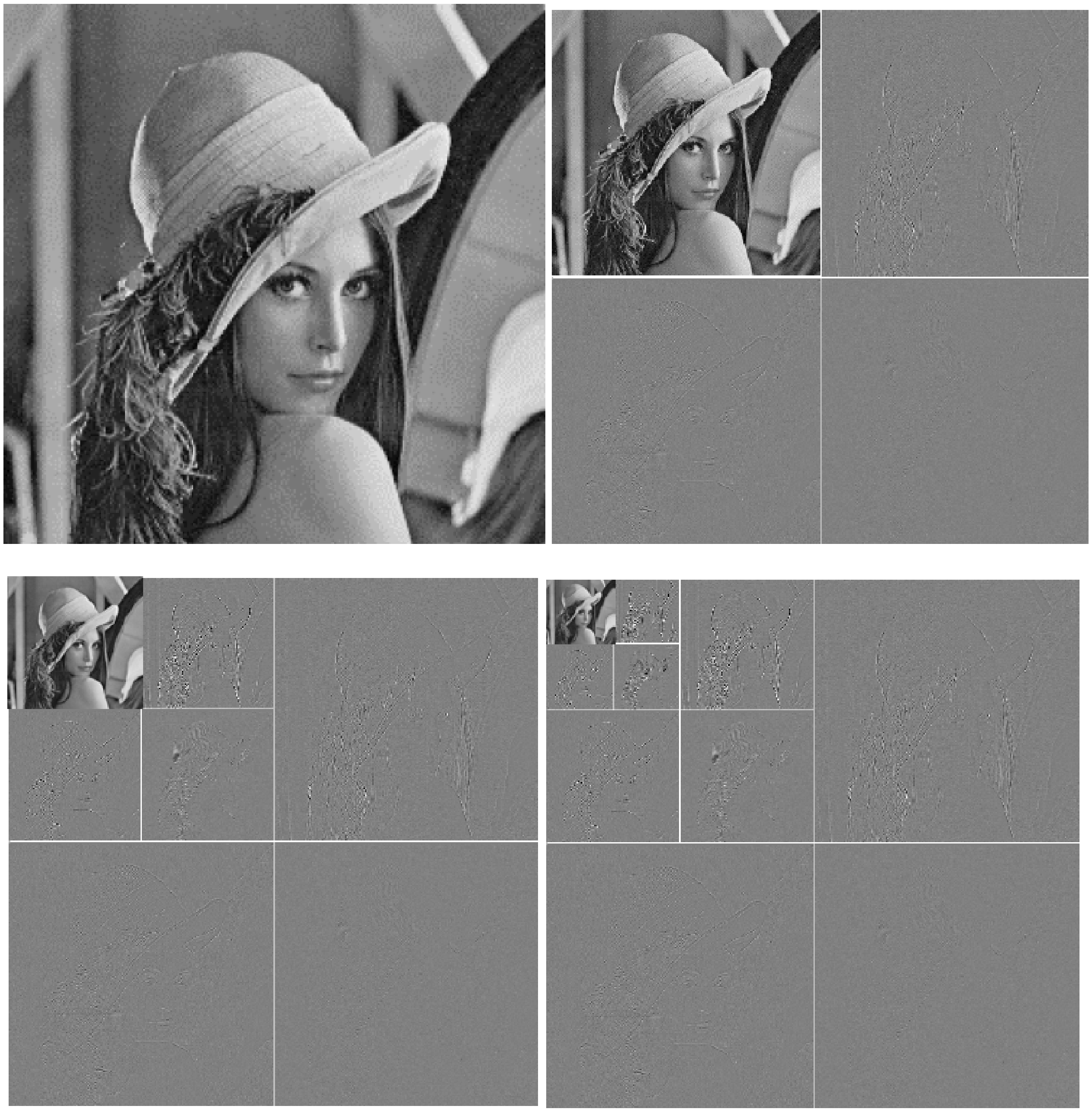


DWT Benefits:

- Multiple resolution representation.
- Lossless representation with integer filters.
- Better de-correlation than DCT, resulting in higher compression efficiency
- Use of visual models: DWT provides a frequency band decomposition of the image where each subband can be quantized according to its visual importance (similar to the quantization table specification in JPEG-DCT).

2-d Discrete Wavelet Transform and Examples

Example 6.7: Consider the Lena image and perform 3-level wavelet decomposition.



Example 6.8: Let us explore using Matlab Slideshow demo of the 2-D wavelet compression.

Example 6.9: Let us the performance of encoders using 2-D Wavelet from VcDemo on Lena for levels 2,4, 6, and bit rate 0.5 bit/pixel

Encoder: #Resolution levels : 2

Target bit rate : 0.50 (bpp)	Decoded bit rate: 0.50 (bpp)
Target file size : 131072 (bits)	Decoded #bits : 131072 (bits)

```
 Mean Square Error       : 353.4
 Signal-to-Noise Ratio   :   8.1 (dB)                     PSNR              :  22.6 (dB)

Encoder: #Resolution levels        :  4
Mean Square Error        : 19.6
 Signal-to-Noise Ratio   :  20.7 (dB)                     PSNR              :  35.2 (dB)

Encoder: #Resolution levels        :  6
Mean Square Error        : 17.7
 Signal-to-Noise Ratio   :  21.1 (dB)                   PSNR              :  35.6 (dB)
```

We can see the impact of bit (compression) rate by varying the rate parameter up and down.