CMPT 365 Multimedia Systems

Lossy Compression

Spring 2017

Edited from slides by Dr. Jiangchuan Liu

Lossless vs Lossy Compression

- If the compression and decompression processes induce no information loss, then the compression scheme is **lossless**; otherwise, it is **lossy**.
- Why is lossy compression possible ?

Original

Compression Ratio: 7.7

Compression Ratio: 12.3

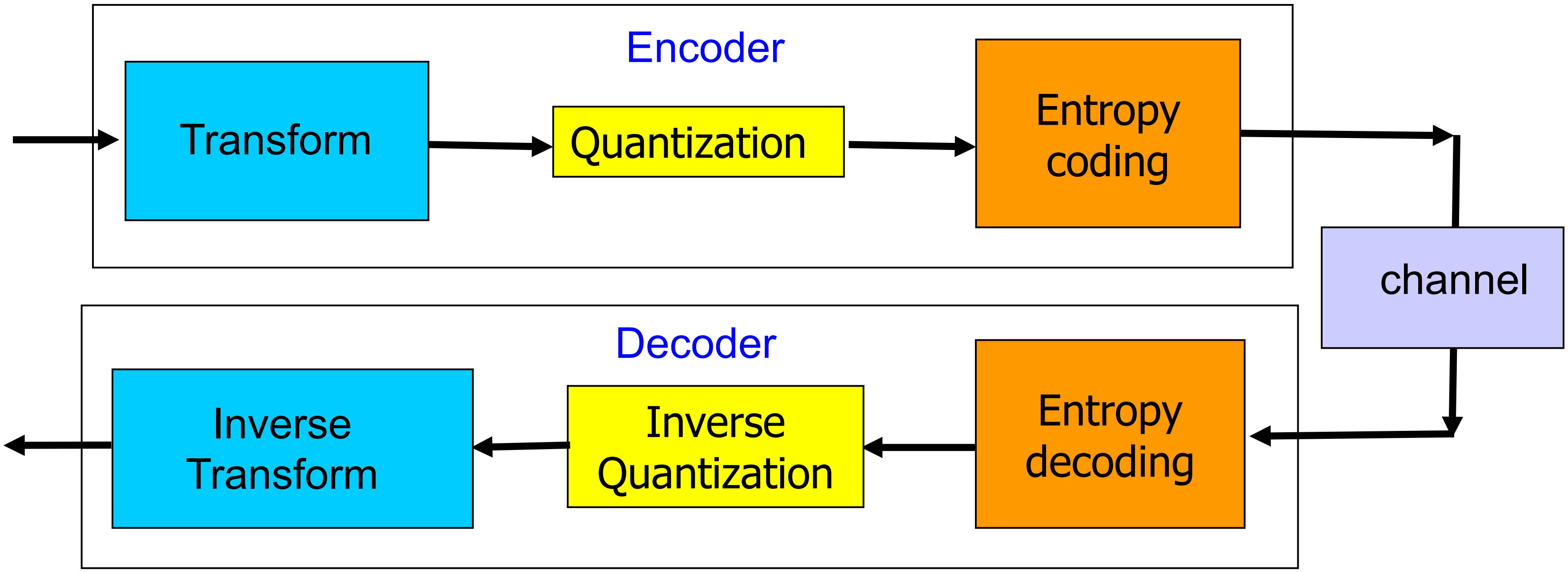
Compression Ratio: 33.9

Outline

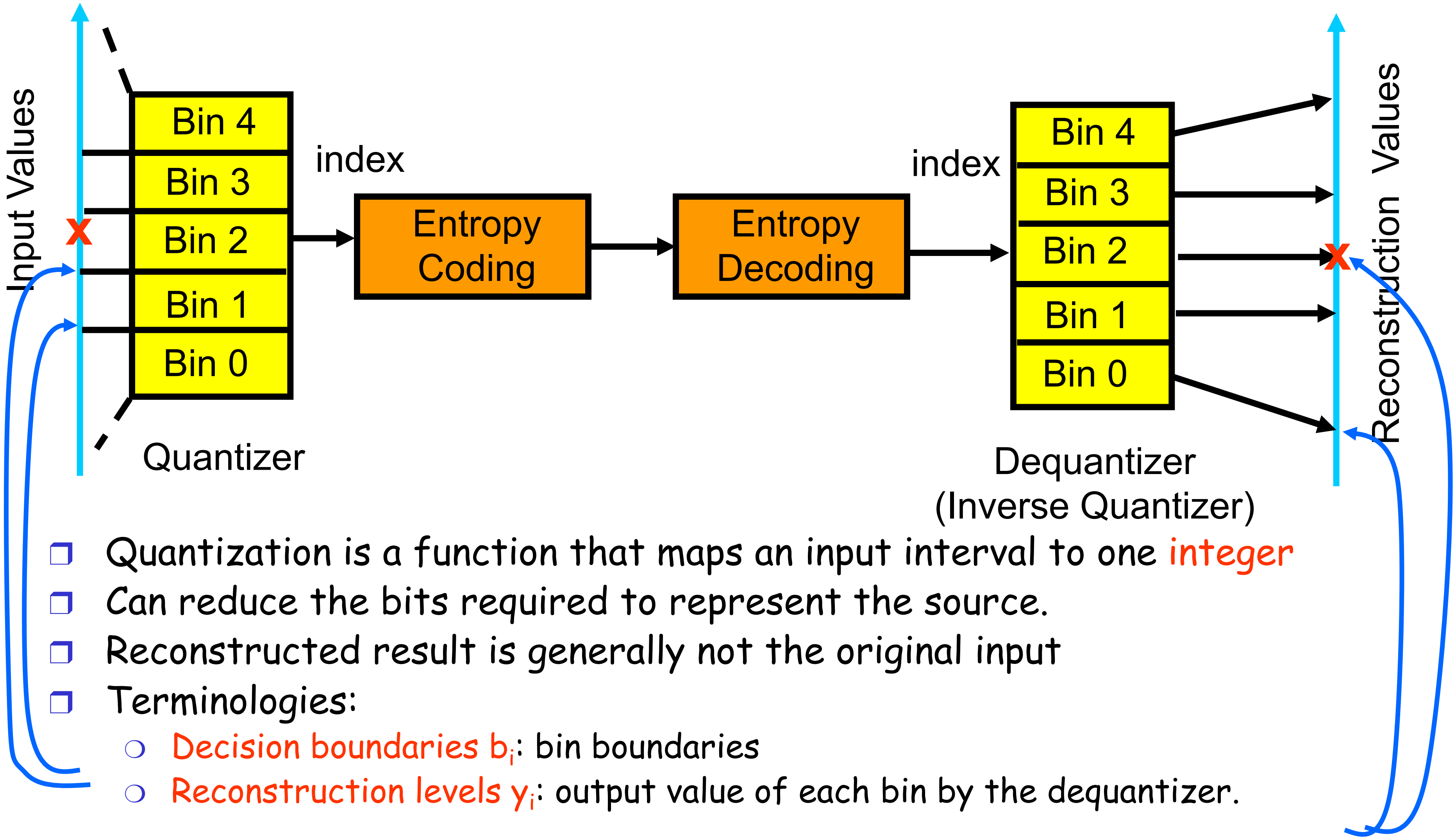
- Quantization
  - Uniform
  - Non-uniform
- Transform coding
  - DCT

Quantization

- The process of representing a large (possibly infinite) set of values with a much smaller set.
  - Example: A/D conversion
- An efficient tool for lossy compression
- Review …



Review: Basic Idea

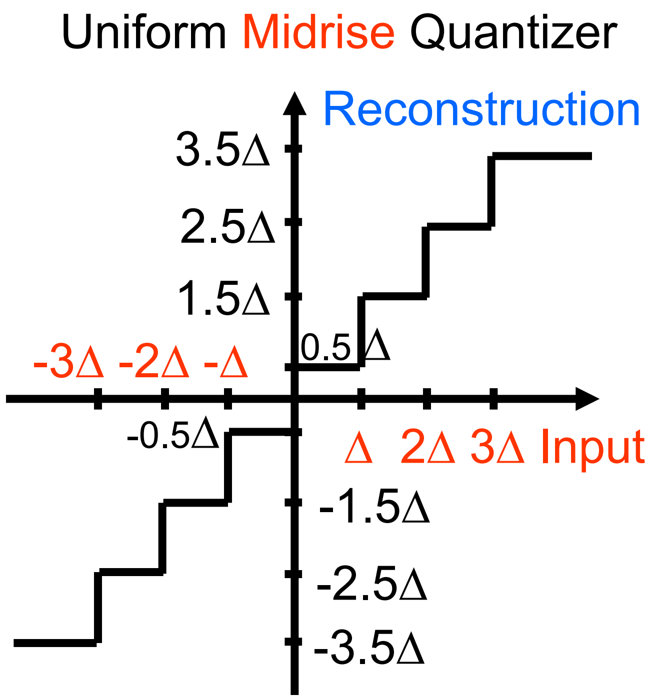


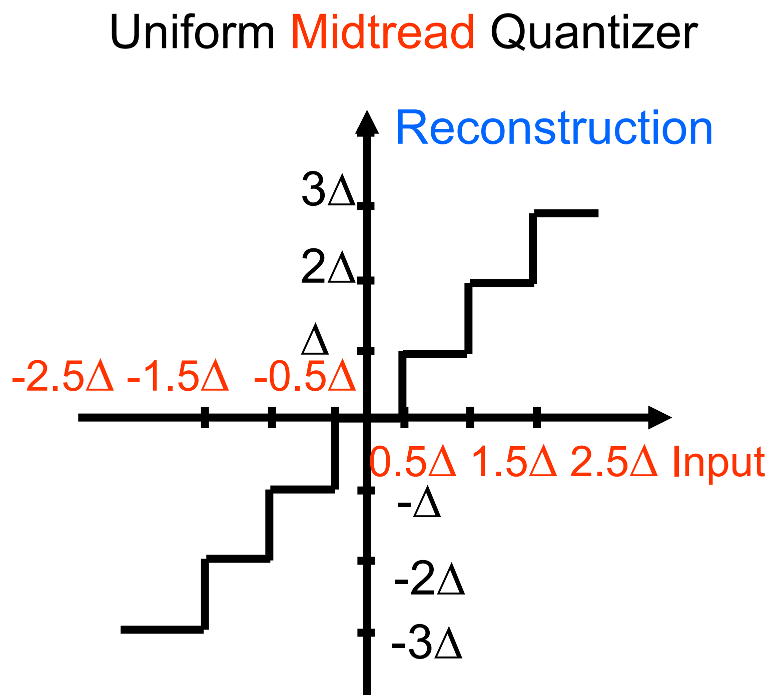
- Quantization is a function that maps an input interval to one integer
- Can reduce the bits required to represent the source.
- Reconstructed result is generally not the original input
- Terminologies:
  - Decision boundaries b_i: bin boundaries
  - Reconstruction levels y_i: output value of each bin by the dequantizer.

Uniform Quantizer

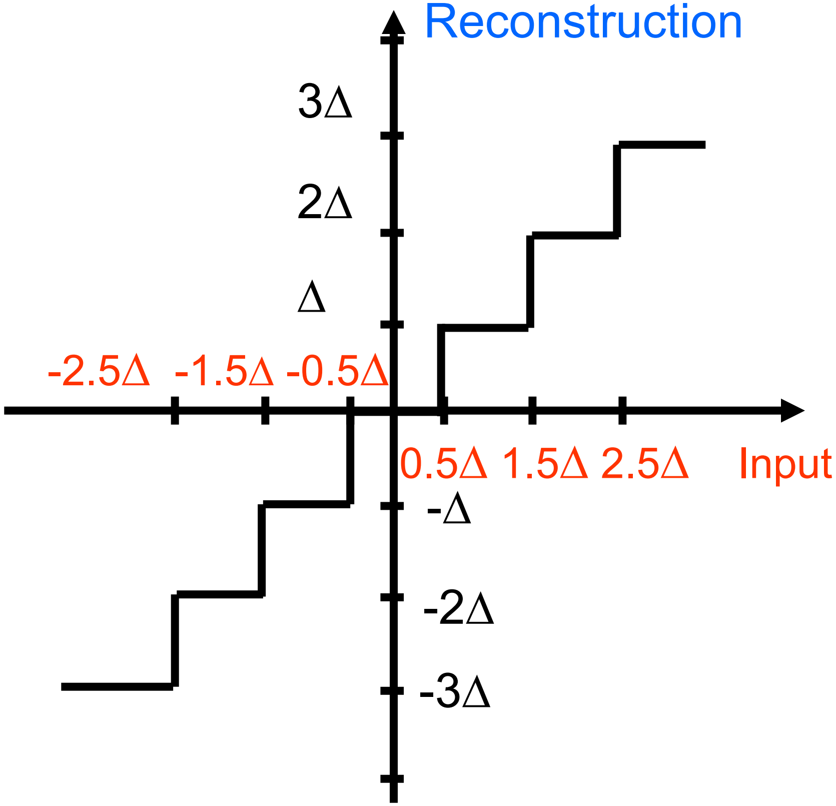
- All bins have the same size except possibly for the two outer intervals:
  - bi and yi are spaced evenly
  - The spacing of bi and yi are both **Δ (step size)**

$$y_i = \frac{1}{2}\left(b_{i-1} + b_i\right)$$ for inner intervals.





Midtread Quantizer



- Quantization mapping:
  Output is an index

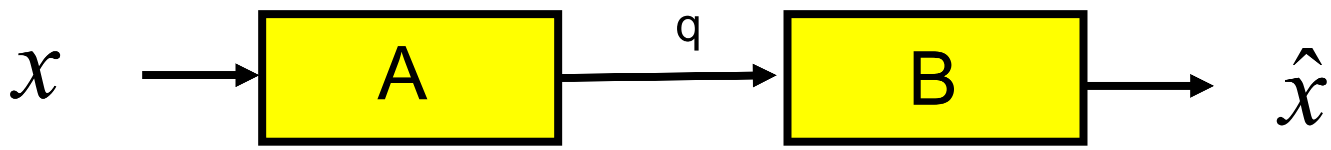
$$q = A(x) = sign(x)\left\lfloor \frac{|x|}{\Delta} + 0.5 \right\rfloor$$

- Example:
  $x = -1.8\Delta$, $q = -2$.

- De-quantization mapping:

$$\hat{x} = B(q) = q\Delta$$

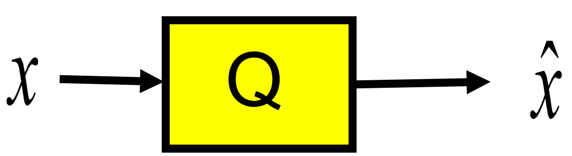
Model of Quantization



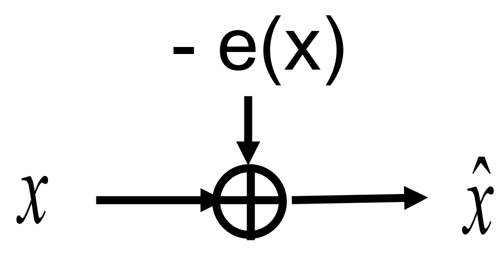
- Quantization: $q = A(x)$
- Inverse Quantization: $\hat{x} = B(q) = B(A(x)) = Q(x)$

B(x) is not exactly the inverse function of A(x), because $\hat{x} \neq x$

- Quantization error: $e(x) = x - \hat{x}$
- Combining quantizer and de-quantizer:

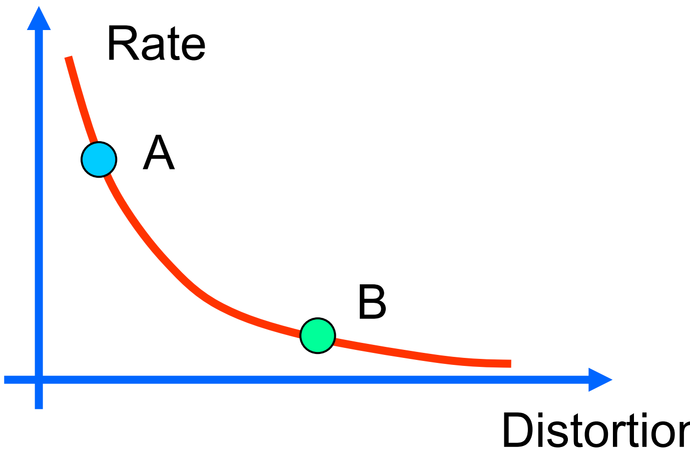


or



Rate-Distortion Tradeoff

- Things to be determined:
  - Number of bins
  - Bin boundaries
  - Reconstruction levels



- A tradeoff between rate and distortion:
  - To reduce the size of the encoded bits, we need to reduce the number of bins
  - Less bins ➔ More reconstruction errors

Measure of Distortion

- Quantization error: $e(x) = x - \hat{x}$
- Mean Squared Error (MSE) for Quantization
  - Average quantization error of all input values
  - Need to know the probability distribution of the input

- Number of bins: M
- Decision boundaries: b_i, i = 0, …, M
- Reconstruction Levels: y_i, i = 1, …, M
- Reconstruction:

$$\hat{x} = y_i \quad \text{iff } b_{i-1} < x \le b_i$$

- MSE:

$$MSE_q = \int_{-\infty}^{\infty} (x - \hat{x})^2 f(x)dx = \sum_{i=1}^{M} \int_{b_{i-1}}^{b_i} (x - y_i)^2 f(x)dx$$

  - Same as the variance of e(x) if μ = E{e(x)} = 0 (zero mean).
  - Definition of Variance:

$$\sigma_e^2 = \int_{-\infty}^{\infty} (e - \mu_e)^2 f(e)de$$

Rate-Distortion Optimization

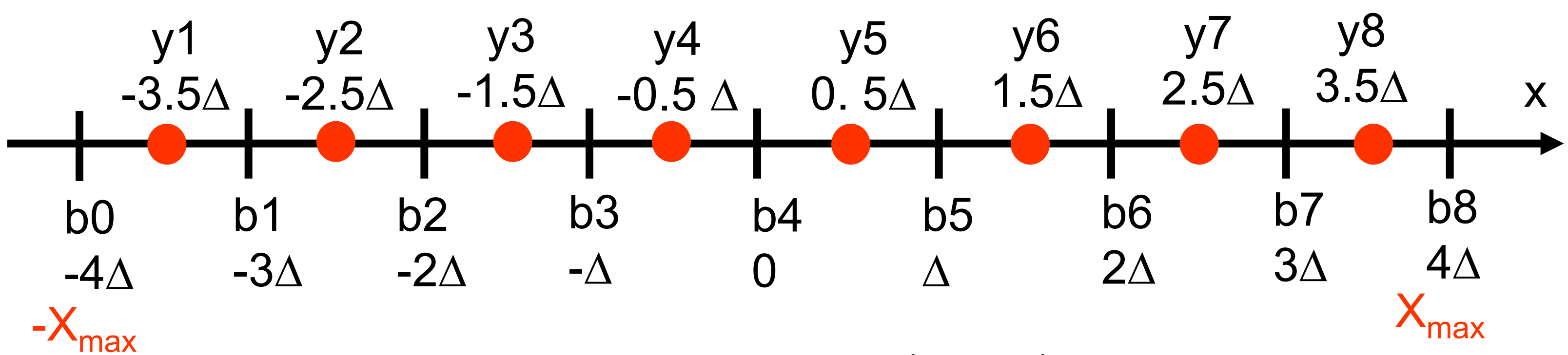
- Two Scenarios:
  - Given M, find b_i and y_i that minimize the MSE.
  - Given a distortion constraint D, find M, b_i and y_i such that the MSE $\leq$ D.

Outline

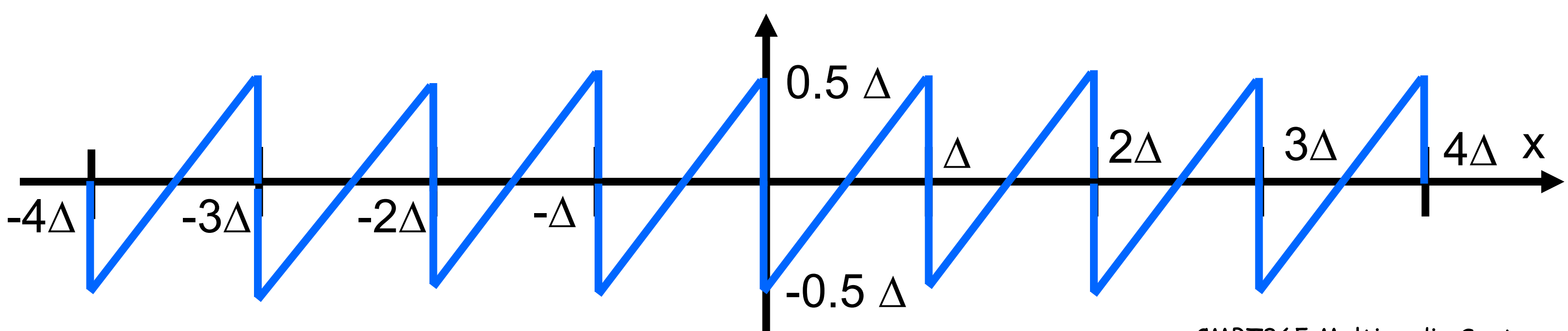
- Quantization
  - Uniform
  - Non-uniform
  - Vector quantization
- Transform coding
  - DCT

Uniform Quantization of a Uniformly Distributed Source

- Input X: uniformly distributed in $[-X_{max}, X_{max}]$: $f(x)= 1 / (2X_{max})$
- Number of bins: M (even for midrise quantizer)
- Step size is easy to get: $\Delta = 2X_{max} / M$.
- $b_i = (i - M/2)\,\Delta$



- ➔ e(x) is uniformly distributed in $[-\Delta/2, \Delta/2]$.



Uniform Quantization of a Uniformly Distributed Source

- MSE

$$MSE_q = \int_{-\infty}^{\infty} (x-\hat{x})^2 f(x)dx = \sum_{i=1}^{M} \int_{b_{i-1}}^{b_i} (x-y_i)^2 f(x)dx$$

$$= M \frac{1}{2X_{\max}} \int_0^{\Delta} \left(x - \frac{\Delta}{2}\right)^2 dx = \frac{M}{2X_{\max}} \frac{1}{12} \Delta^3 = \frac{1}{12} \Delta^2$$

- M increases, Δ decreases, MSE decreases
- Variance of a random variable uniformly distributed in [- Δ/2, Δ/2]:

$$\sigma^2{}_q = \int_{-\Delta/2}^{\Delta/2} (x-0)^2 \frac{1}{\Delta} dx = \frac{1}{12} \Delta^2$$

- Optimization: Find M such that MSE ≤ D

$$\frac{1}{12}\Delta^2 \le D \Rightarrow \frac{1}{12}\left(\frac{2X_{\max}}{M}\right)^2 \le D \Rightarrow M \ge X_{\max}\sqrt{\frac{1}{3D}}$$

Signal to Noise Ratio (SNR)

- Variance is a measure of signal energy
- Let $M = 2^n$
- Each bin index is represented by n bits

$$SNR(dB) = 10\log_{10}\frac{Signal\ \mathrm{Energy}}{Noise\ Energy} = 10\log_{10}\frac{1/12(2X_{\max})^2}{1/12\Delta^2}$$

$$= 10\log_{10}\frac{(2X_{\max})^2}{(2X_{\max}/M)^2} = 10\log_{10}M^2 = 10\log_{10}2^{2n} = (20\log_{10}2)n$$

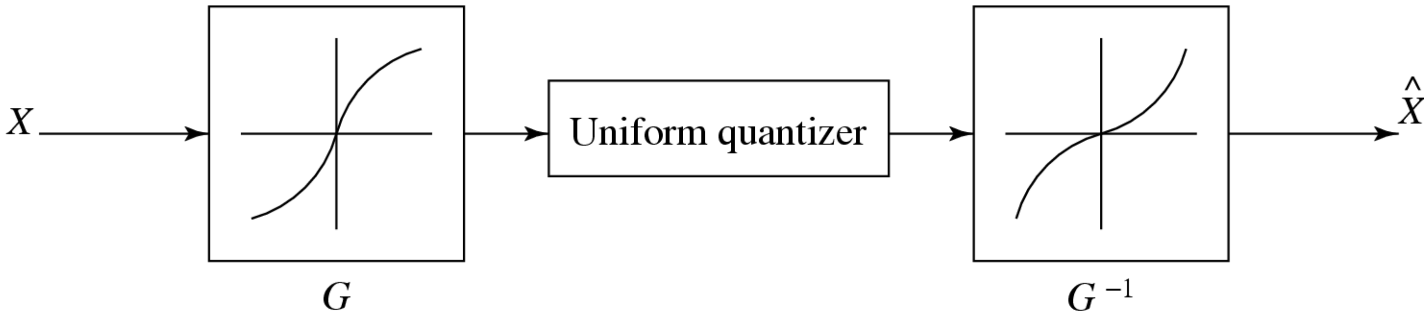
$$\approx 6.02n\ \ dB$$

- If n→n+1, Δ is halved, noise variance reduces to 1/4, and SNR increases by 6 dB.

Outline

- Quantization
  - Uniform
  - Non-uniform
- Transform coding
  - DCT

Non-uniform Quantization

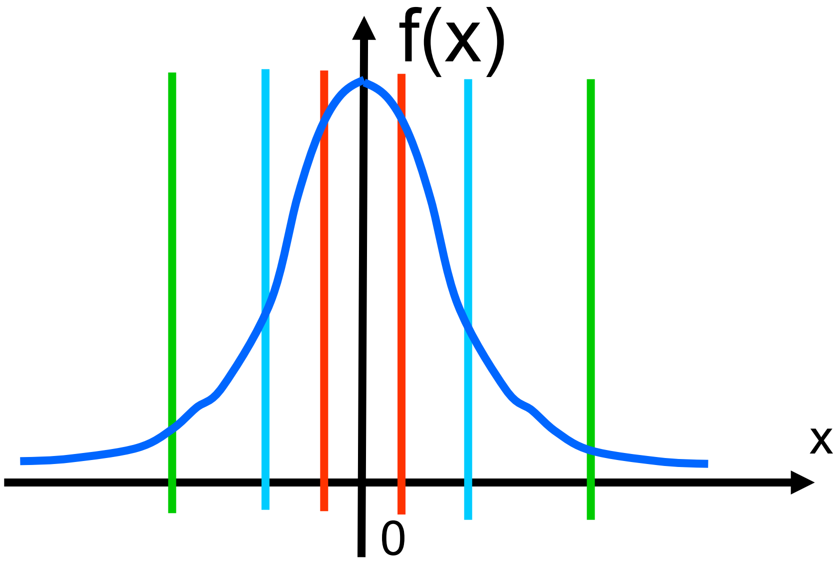


- *Companded quantization* is **nonlinear**.
- As shown above, a *compander* consists of a *compressor function* G, a uniform quantizer, and an *expander function* G^{-1}.
- The two commonly used companders are the *μ*-law and A-law companders.

Non-uniform Quantization

- Uniform quantizer is not optimal if source is not uniformly distributed
- For given M, to reduce MSE, we want narrow bin when f(x) is high and wide bin when f(x) is low

$$\sigma_q^2 = \int_{-\infty}^{\infty} (x - \hat{x})^2 f(x)dx = \sum_{k=1}^{M} \int_{b_{k-1}}^{b_k} (x - y_k)^2 f(x)dx$$



Lloyd-Max Quantizer

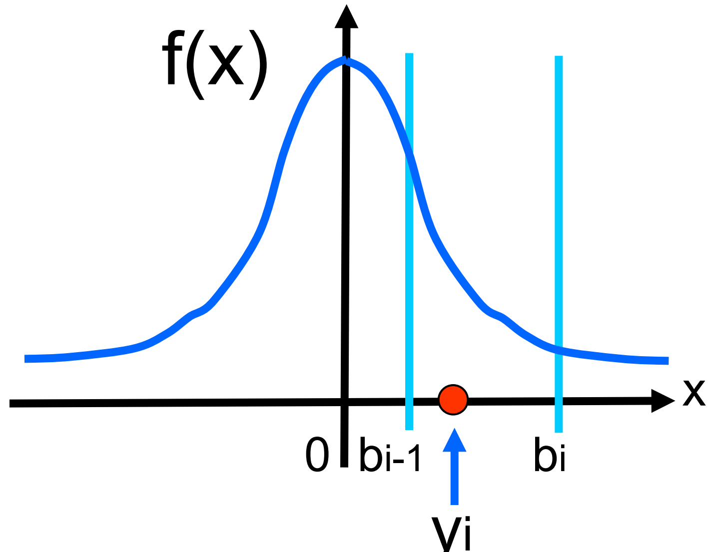
- Also known as pdf-optimized quantizer

$$\sigma_q^2 = \int_{-\infty}^{\infty} (x-\hat{x})^2 f(x)dx = \sum_{k=1}^{M} \int_{b_{k-1}}^{b_k} (x-y_k)^2 f(x)dx$$

- Given M, the optimal b_i and y_i that minimize MSE, satisfying

$$\text{Lagrangian condition}: \frac{\partial \sigma_q^2}{\partial y_i} = 0, \quad \frac{\partial \sigma_q^2}{\partial b_i} = 0.$$

$$\frac{\partial \sigma_q^2}{\partial y_i} = 0 \Rightarrow y_i = \frac{\int_{b_{i-1}}^{b_i} x\, f(x)dx}{\int_{b_{i-1}}^{b_i} f(x)dx}$$



y_i is the centroid of interval $[b_{i-1}, b_i]$.

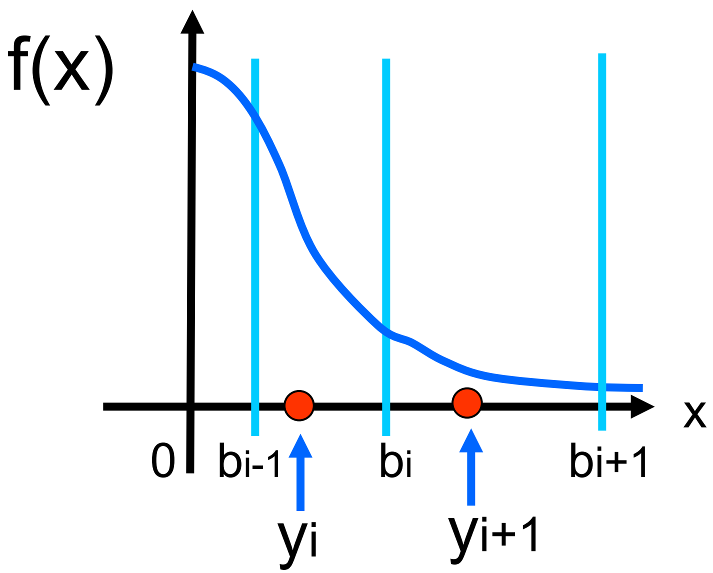
Lloyd-Max Quantizer

- If f(x) = c (uniformly distributed source):

$$y_i = \frac{\int_{b_{i-1}}^{b_i} x\, f(x)dx}{\int_{b_{i-1}}^{b_i} f(x)dx} = \frac{c\int_{b_{i-1}}^{b_i} x\, dx}{c(b_i - b_{i-1})} = \frac{\frac{1}{2}(b_i^2 - b_{i-1}^2)}{b_i - b_{i-1}} = \frac{1}{2}(b_i + b_{i-1})$$

$$\frac{\partial \sigma_q^2}{\partial b_i} = 0 \Rightarrow b_i = \frac{y_i + y_{i+1}}{2}$$

➔ b_i is the midpoint of y_i and y_{i+1}



Lloyd-Max Quantizer

- Summary of conditions for optimal quantizer:

$$y_i = \frac{\int_{b_{i-1}}^{b_i} x\, f(x)dx}{\int_{b_{i-1}}^{b_i} f(x)dx} \qquad b_i = \frac{y_i + y_{i+1}}{2}$$

- Given b_i, can find the corresponding optimal y_i
- Given y_i, can find the corresponding optimal b_i

- How to find optimal bi and yi simultaneously?
  - A deadlock:
    - Reconstruction levels depend on decision levels
    - Decision levels depend on reconstruction levels
  - Solution: iterative method !

Lloyd Algorithm (Sayood pp. 267)

1. Start from an initial set of reconstruction values y_i.
2. Find all decision levels $$b_i = \frac{y_i + y_{i+1}}{2}$$
3. Computer MSE: $$\sigma_q^2 = \sum_{k=1}^{M} \int_{b_{k-1}}^{b_k} (x - y_k)^2 f(x)dx$$
4. Stop if MSE changes little from last time.
5. Otherwise, update y_i, go to step 2.

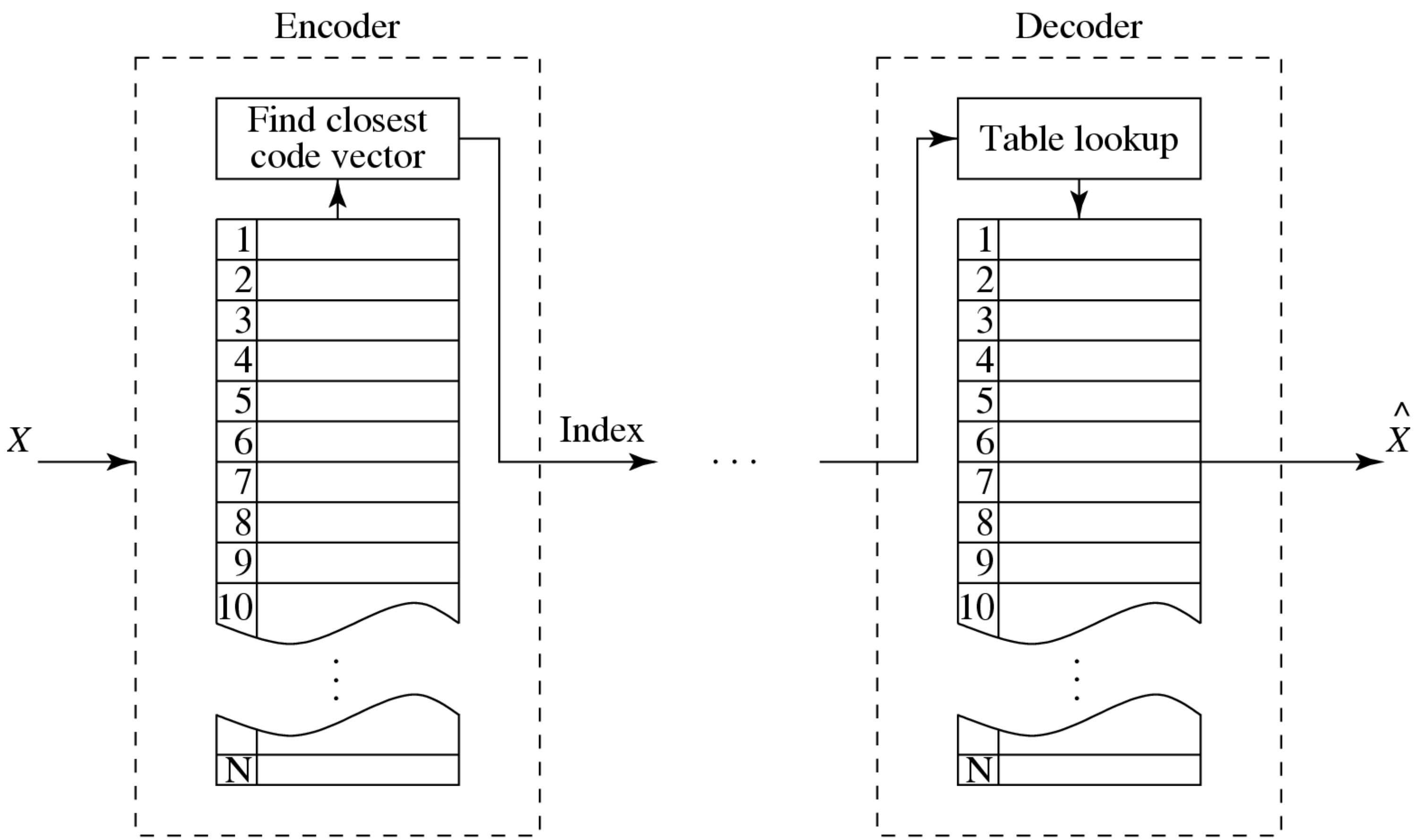
$$y_i = \frac{\int_{b_{i-1}}^{b_i} x\, f(x)dx}{\int_{b_{i-1}}^{b_i} f(x)dx}$$

Outline

- Quantization
  - Uniform
  - Non-uniform
  - Vector quantization
- Transform coding
  - DCT

Vector Quantization (VQ)

- According to Shannon's original work on information theory, any compression system performs better if it operates on vectors or groups of samples rather than individual symbols or samples.
- Form vectors of input samples by simply concatenating a number of consecutive samples into a single vector.
- Instead of single reconstruction values as in scalar quantization, in VQ code *vectors* with n components are used. A collection of these code vectors form the *codebook*.



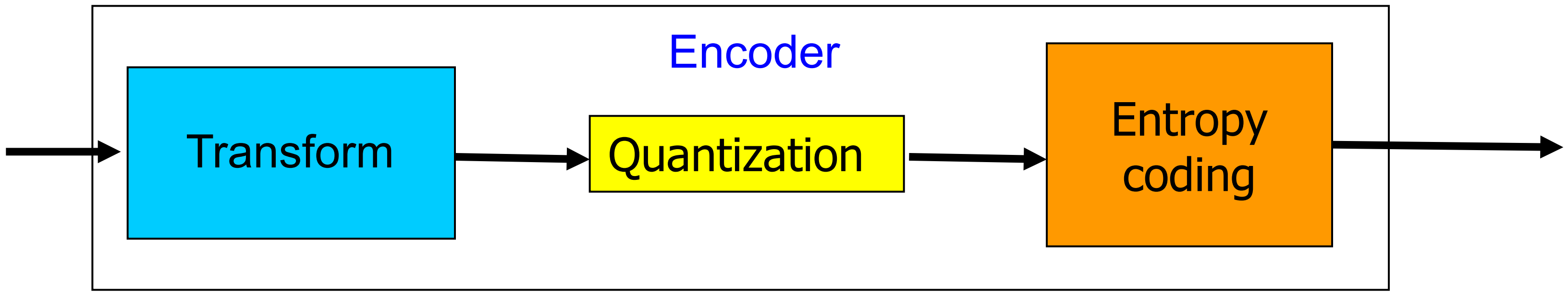
- **Fig. 8.5**: Basic vector quantization procedure.

Outline

- Quantization
  - Uniform quantization
  - Non-uniform quantization
- Transform coding
  - Discrete Cosine Transform (DCT)

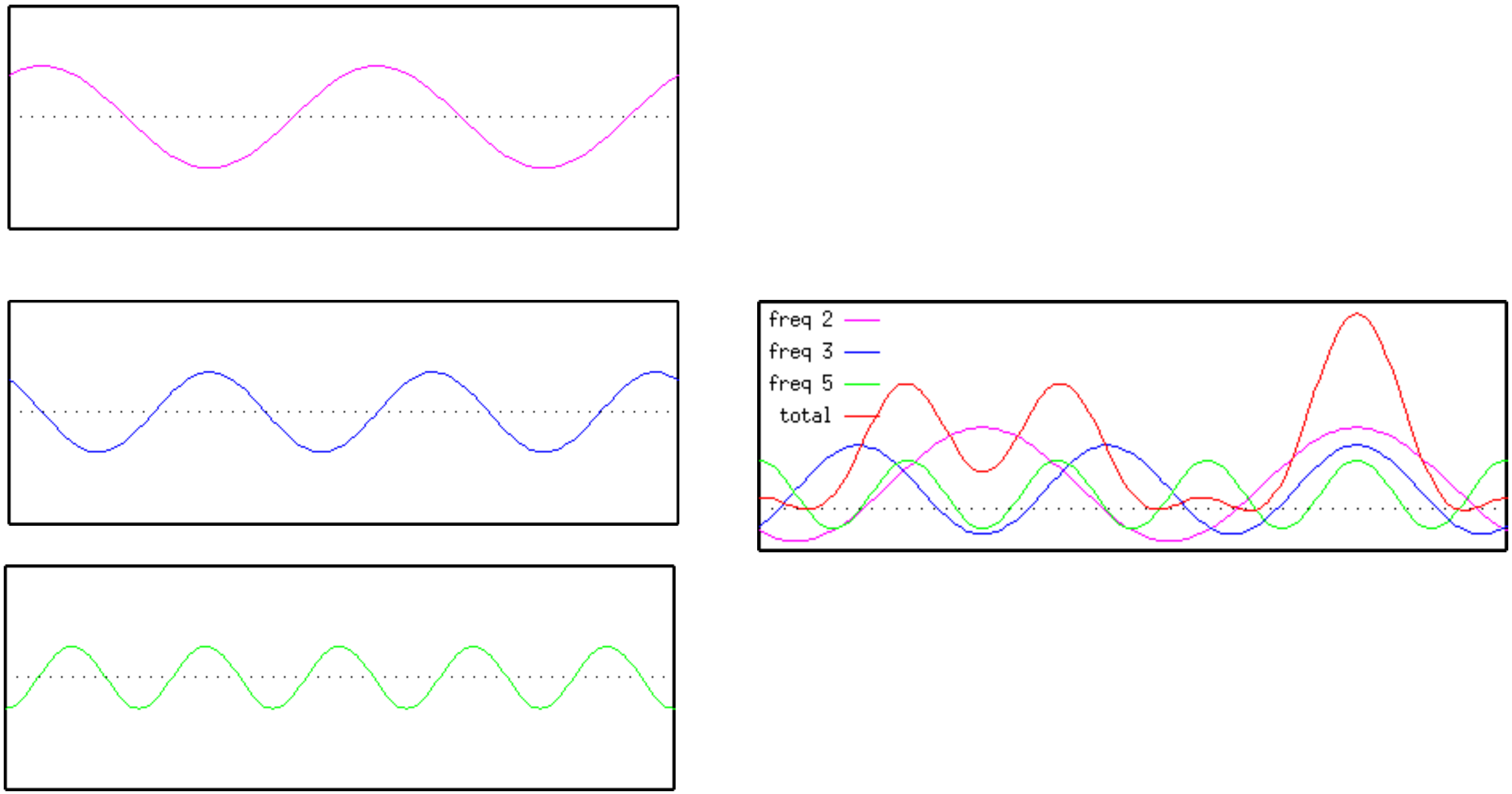
Why Transform Coding ?

- Transform
  - From one domain/space to another space
  - Time -> Frequency
  - Spatial/Pixel -> Frequency
- Purpose of transform
  - Remove correlation between input samples
  - Transform most energy of an input block into a few coefficients
  - Small coefficients can be discarded by quantization without too much impact to reconstruction quality



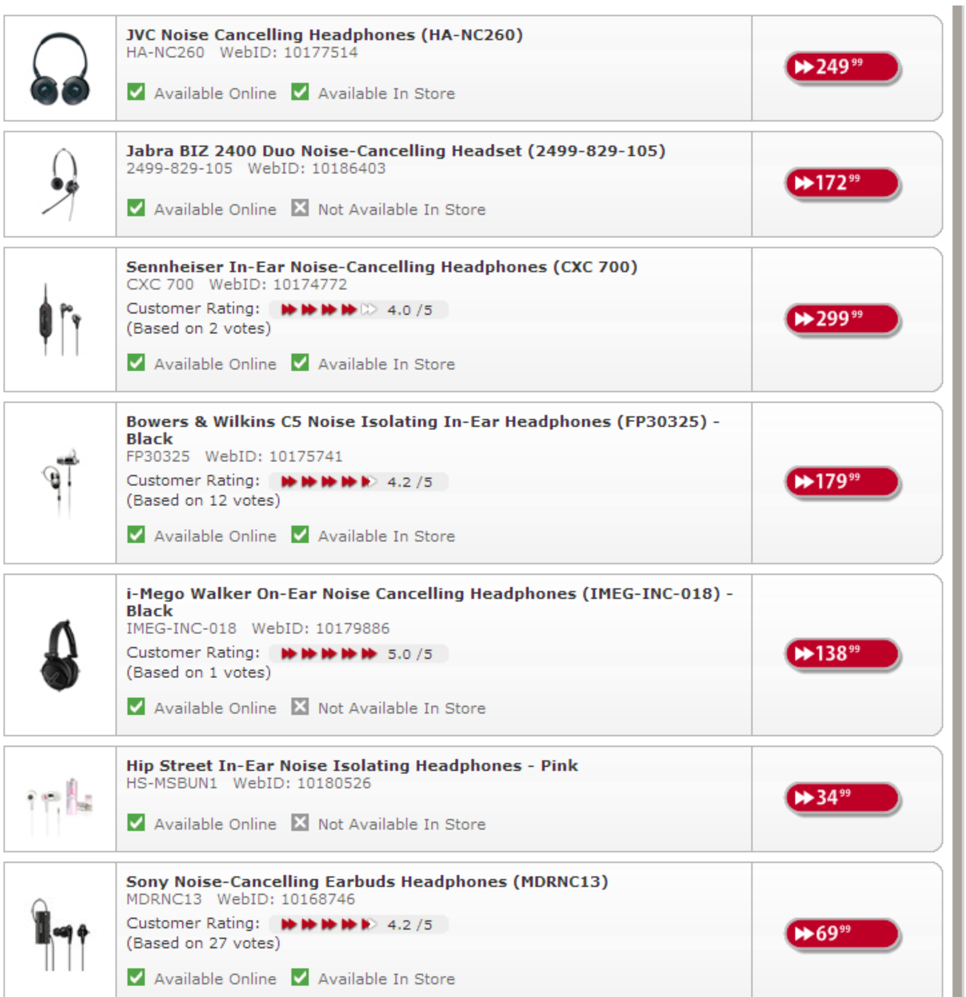
1-D Example

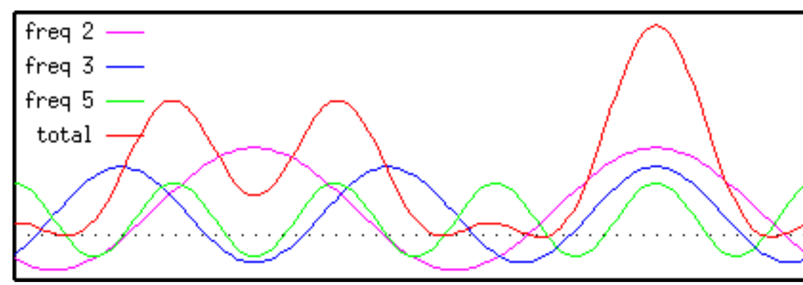
- Fourier Transform



1-D Example

- Application (besides compression)
  - Boost bass/audio equalizer
  - Noise cancellation





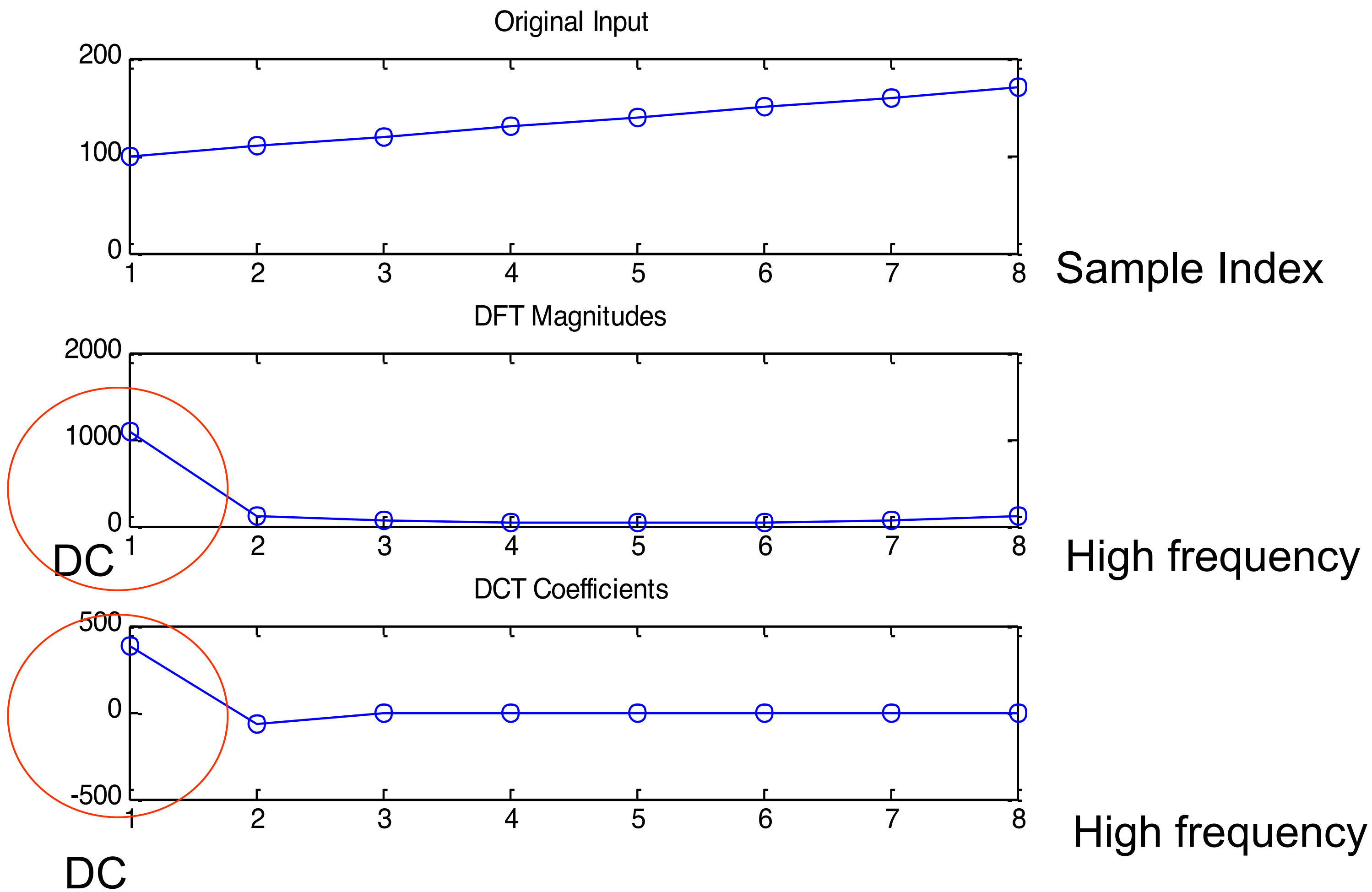
1-D Example

- http://www.mathdemos.org/mathdemos/trigsounddemo/trigsounddemo.html
  - Sine wave/sound/piano

- www.sagebrush.com/mousing.htm
  - An electronic instrument that allows direct control of pitch and amplitude

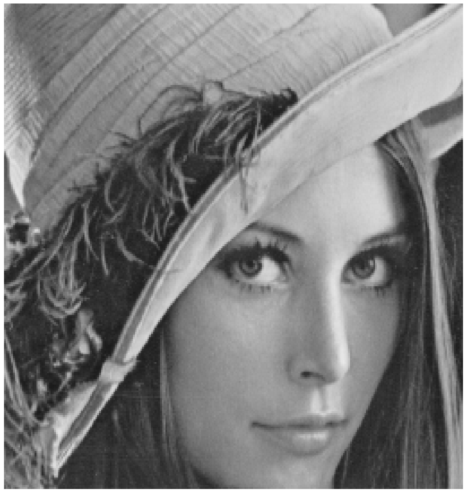
1-D Example

- Smooth signals have strong DC (direct current, or zero frequency) and low frequency components, and weak high frequency components

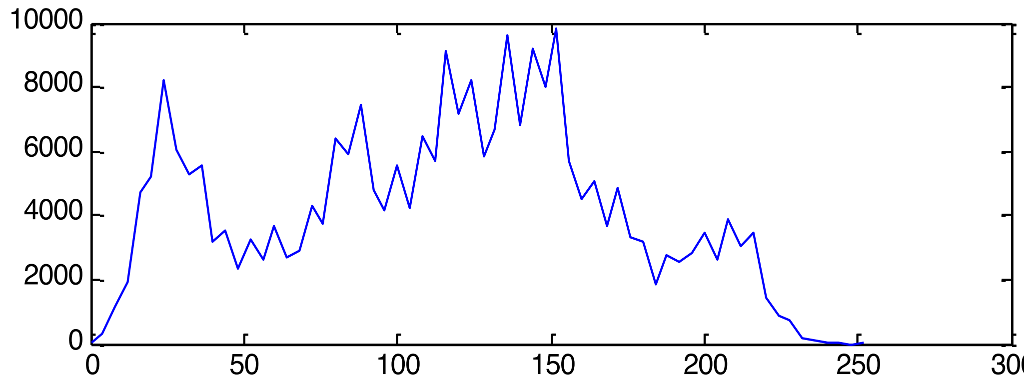


2-D Example

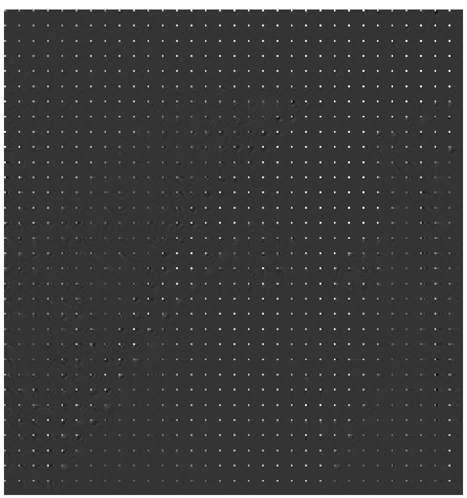
Original Image

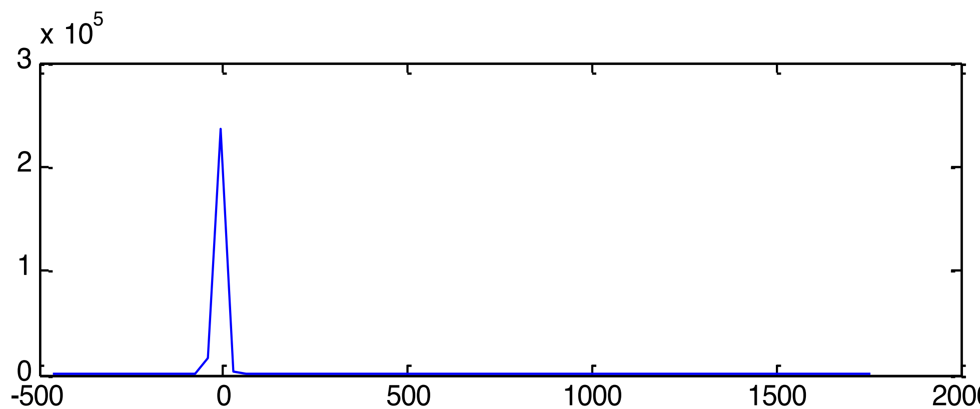


- Apply transform to each 8x8 block
- Histograms of source and DCT coefficients



2-D DCT Coefficients. Min= -465.37, max= 1789.00





- Most transform coefficients are around 0.
- Desired for compression

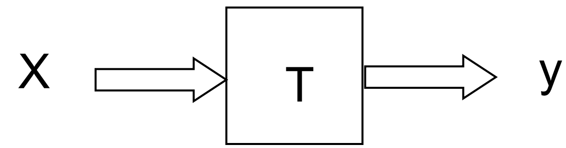
Rationale behind Transform

- If **Y** is the result of a linear transform **T** of the input vector **X** in such a way that the components of **Y** are much less correlated, then **Y** can be coded more efficiently than **X**.
- If most information is accurately described by the first few components of a transformed vector, then the remaining components can be coarsely quantized, or even set to zero, with little signal distortion.

Matrix Representation of Transform

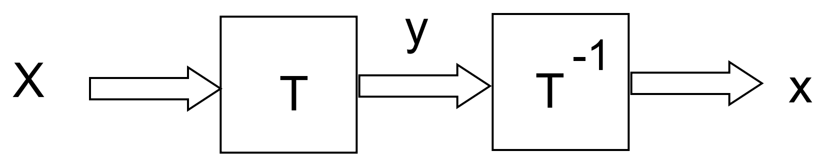
- Linear transform is an N x N matrix:

$$\mathbf{y}_{N\times 1} = \mathbf{T}_{N\times N}\mathbf{x}_{N\times 1}$$



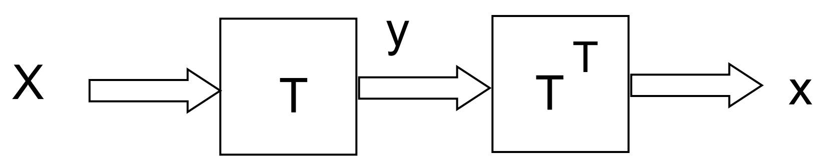
- Inverse Transform:

$$\mathbf{x} = \mathbf{T}^{-1}\mathbf{y}$$



- Unitary Transform (aka orthonormal):

$$\mathbf{T}^{-1} = \mathbf{T}^{T}$$



- For unitary transform: rows/cols have unit norm and are orthogonal to each others

$$\mathbf{T}\mathbf{T}^{T} = \mathbf{I} \;\Rightarrow\; \mathbf{t}_i\mathbf{t}_j^{T} = \delta_{ij} = \begin{cases} 1, & \mathrm{i} = \mathrm{j} \\ 0, & \mathrm{i} \neq \mathrm{j} \end{cases}$$

Discrete Cosine Transform (DCT)

- DCT – close to optimal (known as KL Transform) but much simpler and faster

- Given an input function $f(i, j)$ over two integer variables i and j (a piece of an image), the 2D DCT transforms it into a new function $F(u, v)$, with integer u and v running over the same range as i and j. The general definition of the transform is:

$$F(u,v)=\frac{2\,C(u)\,C(v)}{\sqrt{MN}}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\cos\frac{(2i+1)\cdot u\pi}{2M}\cdot\cos\frac{(2j+1)\cdot v\pi}{2N}\cdot f(i,j) \qquad (8.15)$$

- where $i, u = 0, 1, \ldots, M-1$; $j, v = 0, 1, \ldots, N-1$; and the constants $C(u)$ and $C(v)$ are determined by

$$C(\xi)=\begin{cases}\frac{\sqrt{2}}{2} & if \quad \xi = 0,\\ 1 & otherwise.\end{cases}$$

1D Discrete Cosine Transform (1D DCT):

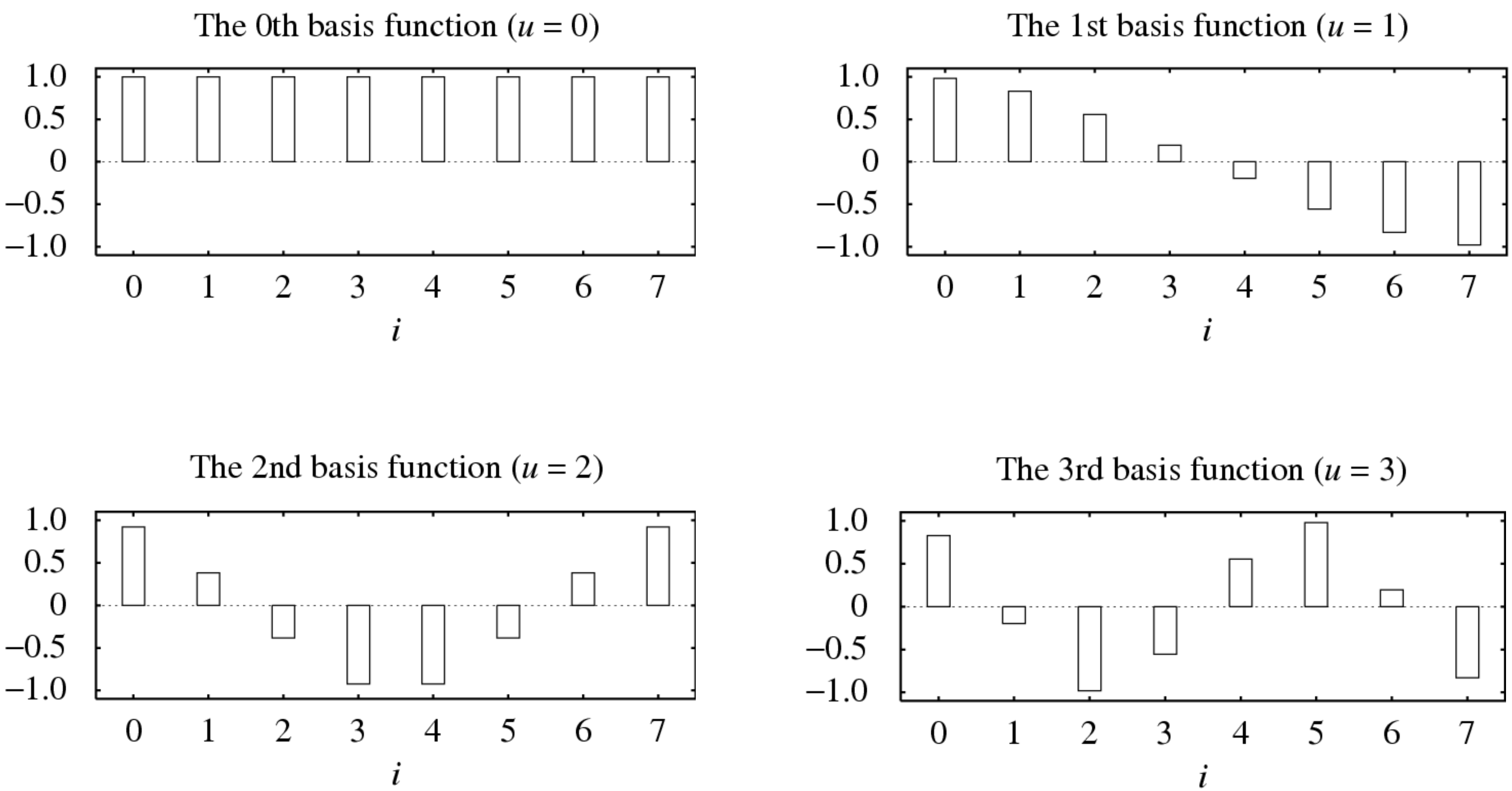
$$F(u) = \frac{C(u)}{2} \sum_{i=0}^{7} \cos \frac{(2i+1)u\pi}{16} f(i) \qquad (8.19)$$

- where i = 0, 1, . . . , 7, u = 0, 1, . . . , 7.

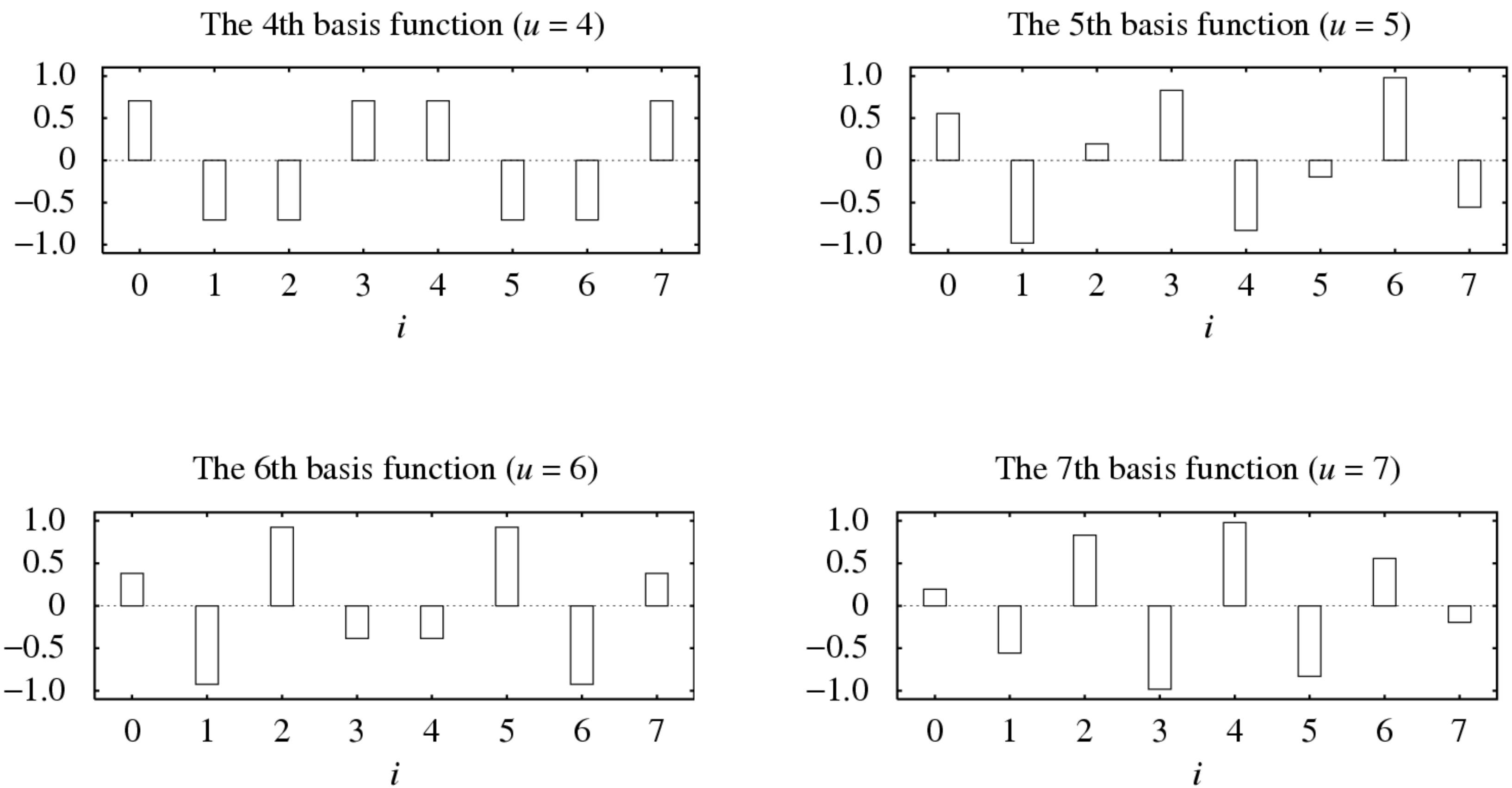
- **1D Inverse Discrete Cosine Transform (1D IDCT):**

$$\tilde{f}(i) = \sum_{u=0}^{7} \frac{C(u)}{2} \cos \frac{(2i+1)u\pi}{16} F(u) \qquad (8.20)$$

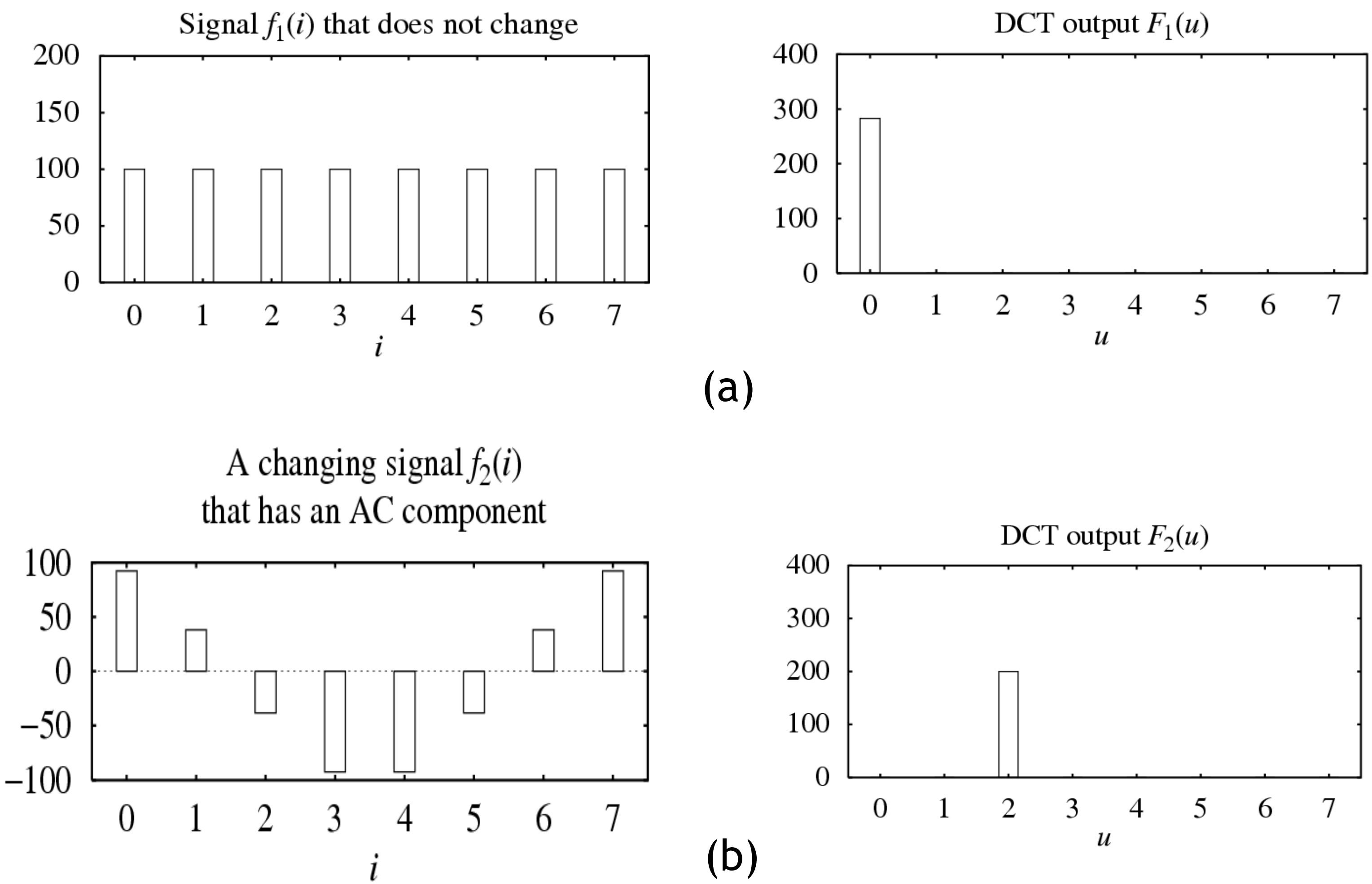
- where i = 0, 1, . . . , 7, u = 0, 1, . . . , 7.



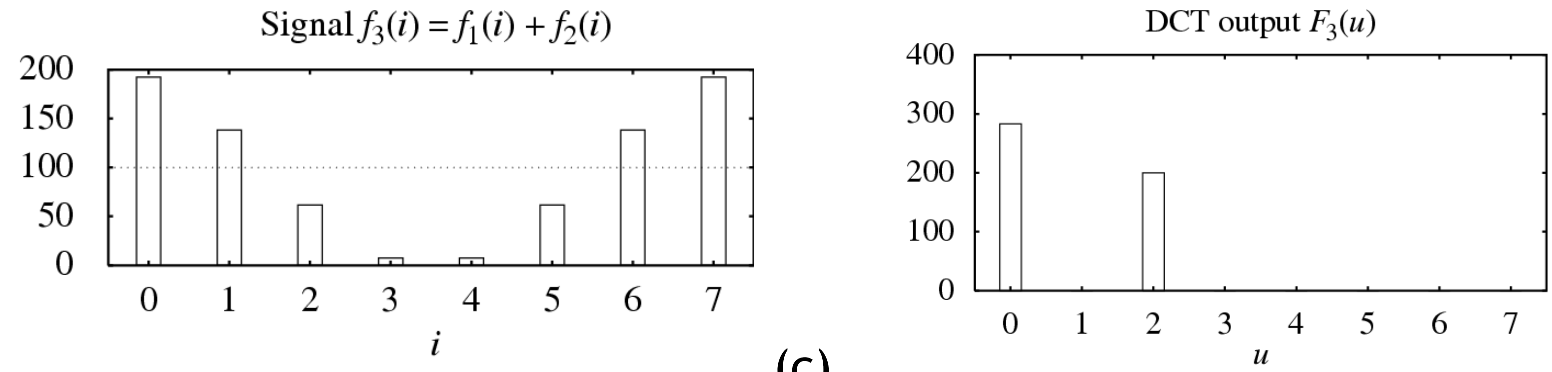
- **Fig. 8.6**: The 1D DCT basis functions.



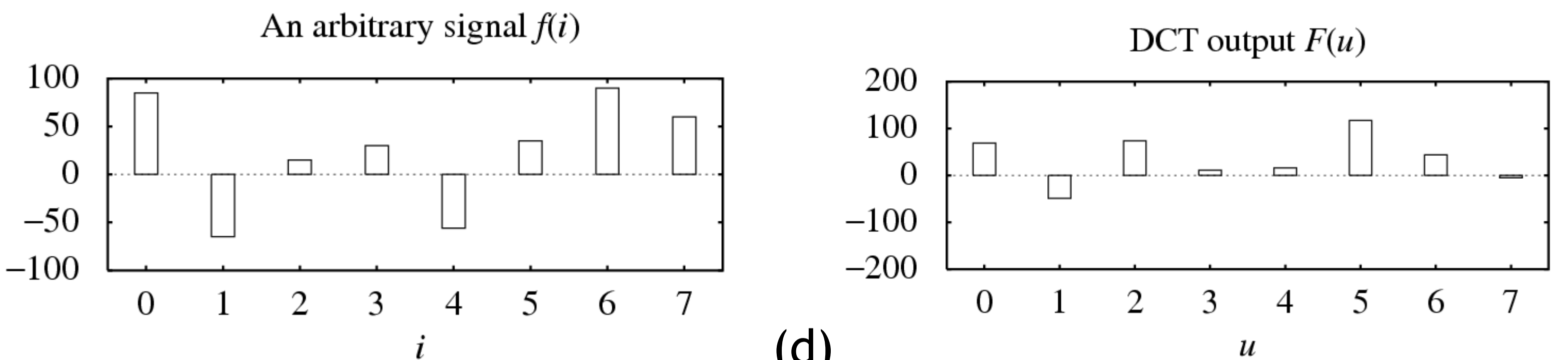
- **Fig. 8.6 (Cont'd):** The 1D DCT basis functions.



- **Fig. 8.7:** Examples of 1D Discrete Cosine Transform: **(a)** A DC signal $f_1(i)$, **(b)** An AC signal $f_2(i)$.

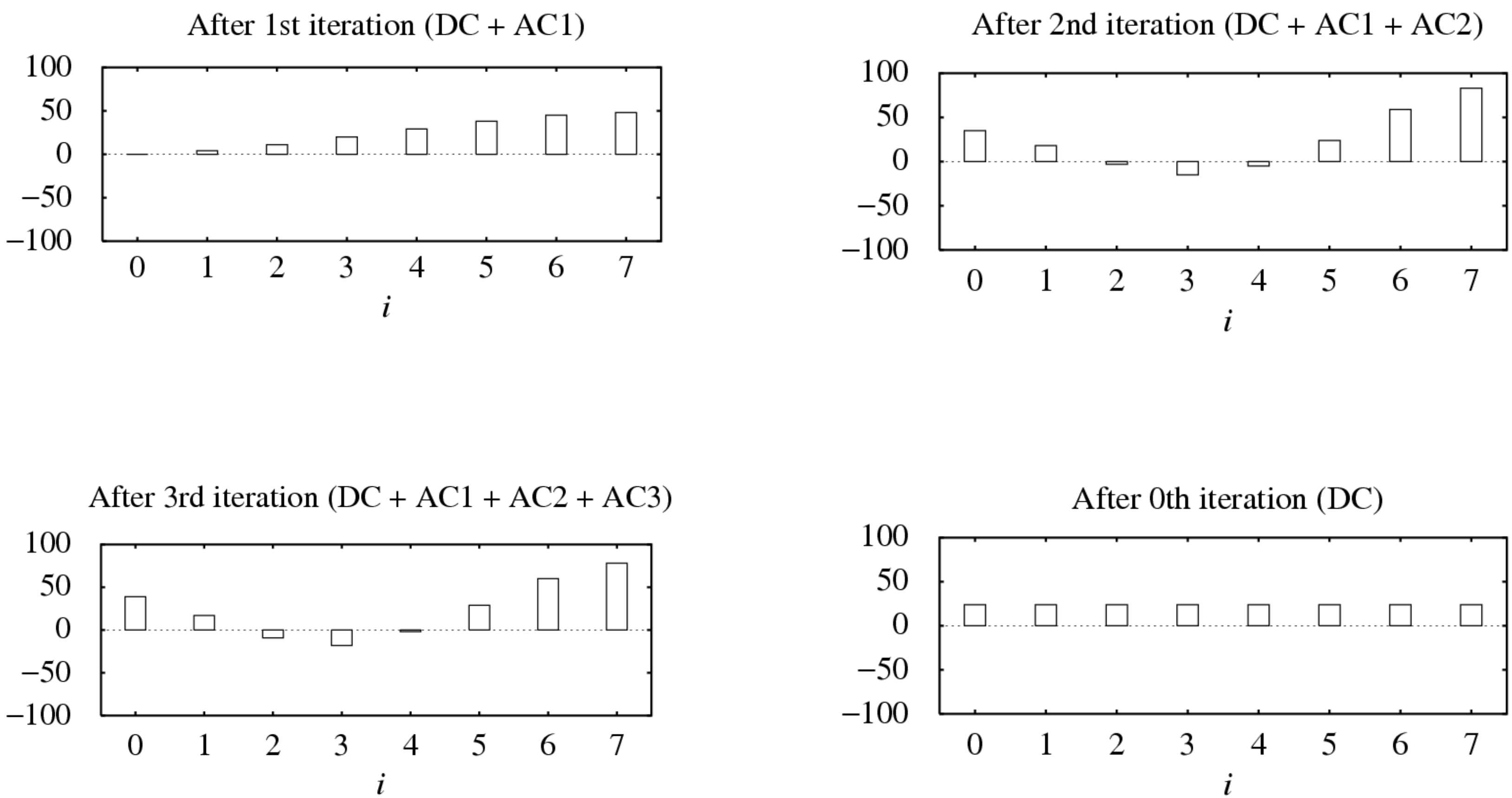


(c)

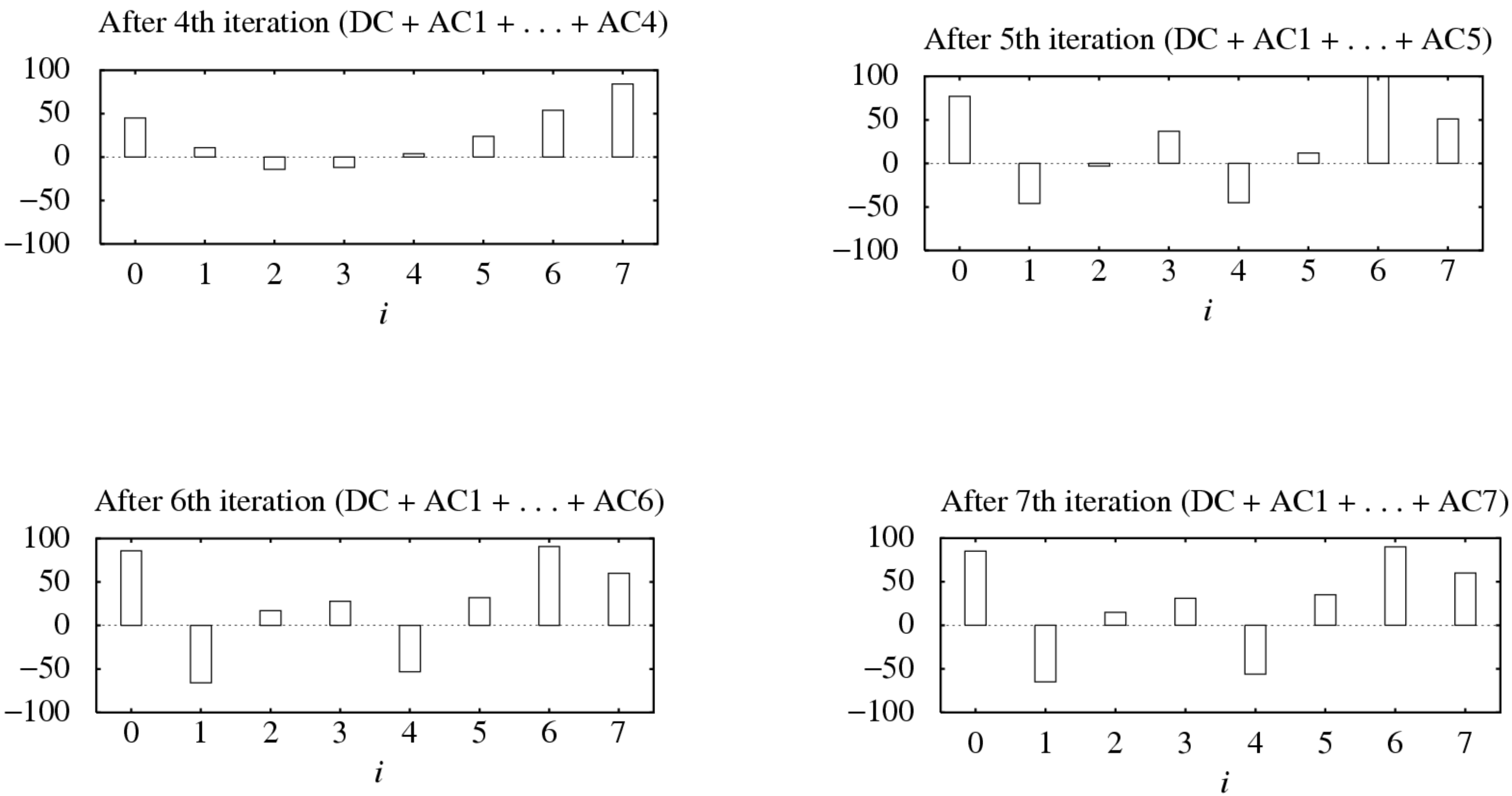


(d)

- **Fig. 8.7 (Cont'd):** Examples of 1D Discrete Cosine Transform: **(c)** $f_3(i) = f_1(i)+f_2(i)$, and **(d)** an arbitrary signal $f(i)$.



- Fig. 8.8: An example of 1D IDCT.



- **Fig. 8.8 (Cont'd):** An example of 1D IDCT.

The DCT is a linear transform:

- In general, a transform $\mathcal{T}$ (or function) is linear, iff

$$\mathcal{T}(\alpha p + \beta q) = \alpha \mathcal{T}(p) + \beta \mathcal{T}(q), \quad (8.21)$$

where α and β are constants, p and q are any functions, variables or constants.

- From the definition in Eq. 8.17 or 8.19, this property can readily be proven for the DCT because it uses only simple arithmetic operations.

The Cosine Basis Functions

- Function $B_p(i)$ and $B_q(i)$ are *orthogonal*, if

$$\sum_i [B_p(i) \cdot B_q(i)] = 0 \qquad if \;\; p \neq q$$

- (8.22)

- Function $B_p(i)$ and $B_q(i)$ are *orthonormal*, if they are orthogonal and

$$\sum_i [B_p(i) \cdot B_q(i)] = 1 \qquad if \;\; p = q$$

- (8.23)

- It can be shown that:

$$\sum_{i=0}^{7} \left[\cos\frac{(2i+1)\cdot p\pi}{16} \cdot \cos\frac{(2i+1)\cdot q\pi}{16} \right] = 0 \qquad if \; p \neq q$$

- $$\sum_{i=0}^{7} \left[\frac{C(p)}{2} \cos\frac{(2i+1)\cdot p\pi}{16} \cdot \frac{C(q)}{2} \cos\frac{(2i+1)\cdot q\pi}{16} \right] = 1 \qquad if \;\; p = q$$

2D Discrete Cosine Transform (2D DCT):

$$F(u,v) = \frac{C(u)\,C(v)}{4} \sum_{i=0}^{7} \sum_{j=0}^{7} \cos\frac{(2i+1)u\pi}{16} \cos\frac{(2j+1)v\pi}{16} f(i,j)$$

- where i, j, u, v = 0, 1, . . . , 7, and the constants $C(u)$ and $C(v)$ are determined by Eq. (8.5.16).

2D Inverse Discrete Cosine Transform (2D IDCT):

- The inverse function is almost the same, with the roles of $f(i, j)$ and $F(u, v)$ reversed, except that now $C(u)C(v)$ must stand inside the sums:

$$\tilde{f}(i,j) = \sum_{u=0}^{7} \sum_{v=0}^{7} \frac{C(u)\,C(v)}{4} \cos\frac{(2i+1)u\pi}{16} \cos\frac{(2j+1)v\pi}{16} F(u,v)$$

- where i, j, u, v = 0, 1, . . . , 7.

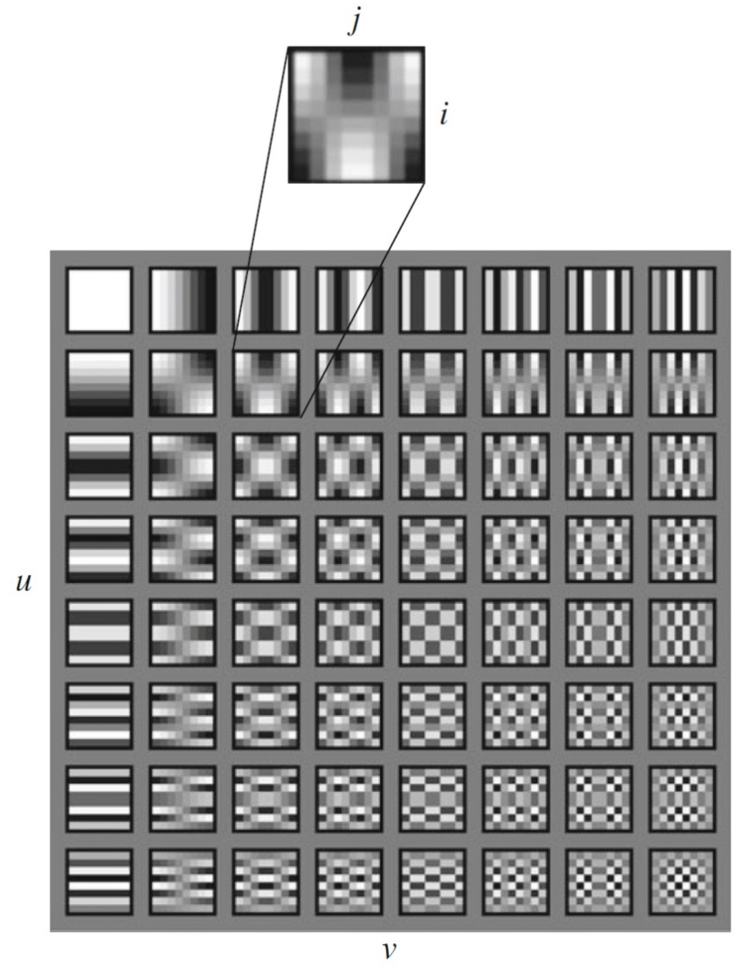
2D Basis Functions

- For a particular pair of *u* and *v*, the respective 2D basis function is:

$$\cos\frac{(2i+1)\cdot u\pi}{16}\cdot\cos\frac{(2j+1)\cdot v\pi}{16},$$

- The enlarged block shown in Fig. 8.9 is for the basis function:

$$\cos\frac{(2i+1)\cdot 1\pi}{16}\cdot\cos\frac{(2j+1)\cdot 2\pi}{16}.$$



- **Fig. 8.9:** Graphical Illustration of 8 × 8 2D DCT basis.

2D Separable Basis

$$F(u,v)=\frac{C(u)C(v)}{4}\sum_{i=0}^{7}\sum_{j=0}^{7}\cos\frac{(2i+1)u\pi}{16}\cos\frac{(2j+1)v\pi}{16}f(i,j)$$

- The 2D DCT can be *separated* into a sequence of two, 1D DCT steps:

$$G(u,j)=\frac{1}{2}C(u)\sum_{i=0}^{7}\cos\frac{(2i+1)u\pi}{16}f(i,j).$$

$$F(u,v)=\frac{1}{2}C(v)\sum_{j=0}^{7}\cos\frac{(2j+1)v\pi}{16}G(u,j).$$

- It is straightforward to see that this simple change saves many arithmetic steps. The number of iterations required is reduced from 8 × 8 to 8+8.

2D DCT Matrix Implementation

- The above factorization of a 2D DCT into two 1D DCTs can be implemented by two consecutive matrix multiplications:

$$F(u, v) = \mathbf{T} \cdot f(i, j) \cdot \mathbf{T}^T. \qquad (8.27)$$

- We will name $\mathbf{T}$ the *DCT-matrix*.

$$\mathbf{T}[i, j] = \begin{cases} \frac{1}{\sqrt{N}}, & \text{if } i = 0 \\ \sqrt{\frac{2}{N}} \cdot \cos\frac{(2j+1)\cdot i\pi}{2N}, & \text{if } i > 0 \end{cases} \qquad (8.28)$$

Where i = 0, … , N-1 and j = 0, … , N-1 are the row and column indices, and the block size is N x N.

- When $N = 8$, we have:

$$\mathbf{T_8}[i, j] = \begin{cases} \frac{1}{2\sqrt{2}}, & \text{if } i = 0 \\ \frac{1}{2} \cdot \cos\frac{(2j+1)\cdot i\pi}{16}, & \text{if } i > 0. \end{cases}$$

- (8.29)

$$\mathbf{T_8} = \begin{bmatrix} \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \cdots & \frac{1}{2\sqrt{2}} \\ \frac{1}{2}\cdot\cos\frac{\pi}{16} & \frac{1}{2}\cdot\cos\frac{3\pi}{16} & \frac{1}{2}\cdot\cos\frac{5\pi}{16} & \cdots & \frac{1}{2}\cdot\cos\frac{15\pi}{16} \\ \frac{1}{2}\cdot\cos\frac{\pi}{8} & \frac{1}{2}\cdot\cos\frac{3\pi}{8} & \frac{1}{2}\cdot\cos\frac{5\pi}{8} & \cdots & \frac{1}{2}\cdot\cos\frac{15\pi}{8} \\ \frac{1}{2}\cdot\cos\frac{3\pi}{16} & \frac{1}{2}\cdot\cos\frac{9\pi}{16} & \frac{1}{2}\cdot\cos\frac{15\pi}{16} & \cdots & \frac{1}{2}\cdot\cos\frac{45\pi}{16} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{2}\cdot\cos\frac{7\pi}{16} & \frac{1}{2}\cdot\cos\frac{21\pi}{16} & \frac{1}{2}\cdot\cos\frac{35\pi}{16} & \cdots & \frac{1}{2}\cdot\cos\frac{105\pi}{16} \end{bmatrix}. \quad (8.30)$$

2D IDCT Matrix Implementation

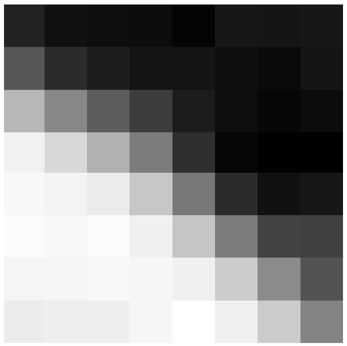
- The 2D IDCT matrix implementation is simply:

$$f(i, j) = \mathbf{T}^T \cdot F(u, v) \cdot \mathbf{T}. \qquad (8.31)$$

- See the textbook for step-by-step derivation of the above equation.
  - The key point is: the DCT-matrix is orthogonal, hence, $\mathbf{T}^T = \mathbf{T}^{-1}.$

2-D 8-point DCT Example

- Original Data:



89	78	76	75	70	82	81	82
122	95	86	80	80	76	74	81
184	153	126	106	85	76	71	75
221	205	180	146	97	71	68	67
225	222	217	194	144	95	78	82
228	225	227	220	193	146	110	108
223	224	225	224	220	197	156	120
217	219	219	224	230	220	197	151

- 2-D DCT Coefficients (after rounding to integers):

Most energy is in the upper-left corner

1155	259	-23	6	11	7	3	0
-377	-50	85	-10	10	4	7	-3
-4	-158	-24	42	-15	1	0	1
-2	3	-34	-19	9	-5	4	-1
1	9	6	-15	-10	6	-5	-1
3	13	3	6	-9	2	0	-3
8	-2	4	-1	3	-1	0	-2
2	0	-3	2	-2	0	0	-1

Further Exploration

- **Textbook 8.1-8.5**
- **Other sources**
  - *Introduction to Data Compression* by Khalid Sayood
  - *Vector Quantization and Signal Compression* by Allen Gersho and Robert M. Gray
  - *Digital Image Processing* by Rafael C. Gonzales and Richard E.Woods
  - *Probability and Random Processes with Applications to Signal Processing* by Henry Stark and John W. Woods
  - *A Wavelet Tour of Signal Processing* by Stephane G. Mallat