Chapter 11

Applications

11.1 Digitized Fingerprints and the FBI

The FBI has 30 million sets of fingerprints (300 million fingers) from felony arrests. I always assumed that they have my fingerprints too, because I once applied for a security clearance. (I believe it was granted. So far I have not been arrested. This is the first author, still at large.) The 40,000 sets of prints that arrive each day are divided into three groups:

5000	new prints to be saved
15000	repeat prints (recidivists) to be compared
20000	security clearances to check and return to other agencies.

Up to now, the prints are on cards in Washington. The file cabinets fill a whole floor in the FBI building. The final identification is always done by trained examiners. The "minutiae" that they watch for are ridge endings and bifurcations, which form permanently in childhood. In a British court, 12 of these indicators provide a legal match (fewer than 12 in most US courts — there are about 150 minutiae per finger). The FBI stores the cards in a special order, which begins with an automated classification using arches, loops, and whorls. Then come finer details until manual search is possible. Multiresolution is natural for fingerprints.

This data has to be digitized, and transmitted more quickly. The prints will be captured by electronic "live scans" instead of ink. At present most cards are simply mailed to the FBI. The criminal has gone from the booking station long before the fingerprints reach Washington. The new turnaround time, for digital information sent to the West Virginia office, is to be two hours (if requested).

At present, the rules do not allow the FBI to keep fingerprint files for juveniles. That group is responsible for a large fraction of breaking and entering felonies. They leave fingerprints all over the place! Some states do maintain files on juveniles, and it seems likely that changes in technology will bring changes in the law. The digital revolution is reaching criminology.

Our focus is on the specifications for the compression of gray scale images (256 levels of gray). The FBI chose a *wavelet/scalar quantization* (WSQ) algorithm. It was initially expected that the JPEG standard would win, but at compression of 15:1 and 20:1 the blocking from the DCT was severe. Ridges that are separated in the true image were found to merge during compression. This is unacceptable. It did not happen for wavelets. The linear phase 9/7 filter bank appeared just in time to be compared with the Daubechies 8/8 bank, and 9/7 was better for two main reasons:

1. Symmetry (and symmetric extension at the boundaries)
2. The image contents do not shift between the subbands.

A ridge has the same position within each subband, when filters have zero phase. The FBI confirmed that symmetric extension is better than circular convolution.

At 500 pixels per inch, ridges typically repeat every 10 to 16 pixels. The dominant frequency band is $\omega = \pi/8$ to $\omega = \pi/4$. The subband tree (Figure 11.1) goes down four levels in this range. All WSQ encoders will use the same tree. Symmetric filters up to 31 or 32 taps are permitted — and the 9/7 pair is recommended.

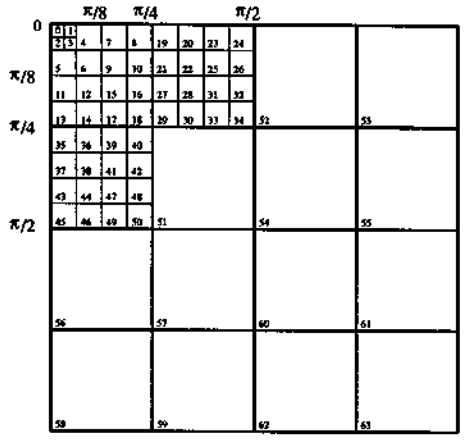


Figure 11.1: The WSQ subbands.

Each of the 64 subbands has its own uniform scalar quantization. The coefficient is set to $p = 0$ if it falls between $-Z/2$ and $Z/2$. A high percentage of the compression comes from this zeroth bin. The compressed image has strings of zeros, and the entropy coding step (Huffman coding) transmits a long string by giving its length. The other bin widths Q are smaller (about $.8Z$). Then the coefficient $a > 0$ falls into the pth bin in Figure 11.2 if $\frac{1}{2}Z + (p-1)Q \le a < \frac{1}{2}Z + pQ$.

Thus a is coded by the small integer p. The coding rule is similar for $a < 0$. The *decoder* (inverse direction) assigns to p the number a at the midpoint of the bin. The decoder also has an option to shift away from the bin center.

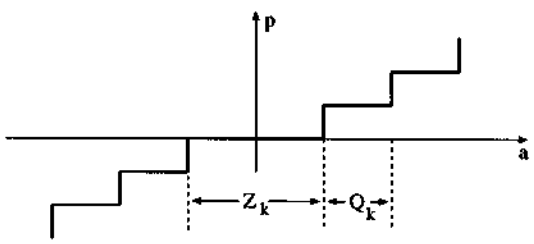


Figure 11.2: The quantization map from real numbers a to integers p.

The key question is the choice of Z and Q for each subband. This is the problem of *bit allo-*

cation in Section 11.2. The FBI uses an empirical relationship $Q_k = qC_k/\log\sigma_k$ controlled by the variance σ_k of the kth subband. Flat signals, with small variance, contain little information. The bin widths are correspondingly larger. The constant C_k can give extra emphasis to subbands that happen to be at the scale of ridges and pores. Then the constant q controls the overall compression. In the FBI system, q is partly determined by the size of the blank background that surrounds the print. The ten megabytes of data per card are compressed by 15 :1.

Theoretically, all bands should be at the same stage on their rate-distortion curves. The difficulty is to measure the distortion that our eyes actually perceive. It is not the L^2 norm! Energy is a convenient measure of error, but it does not agree well with human perception.

Experiments indicate that the improvement over the JPEG algorithm (the DCT in 8×8 blocks) ***increases with the compression ratio***. At high compression, the blocking effects in the JPEG output overwhelm the signal. It becomes impossible to follow the ridge lines, and the FBI examiners frankly said *no*. One effort of Tom Hopper, who is leading the move to digital, is to convert all the groups who deal with fingerprints — including state and local police and the registry of motor vehicles. They have to believe in the investment, and the quality of the compressed signal is what they go by. Then they build the systems.

Entropy Coding

A set of 254 symbols captures bin values p from -73 to 74 and the lengths of zero runs up to 100. A few escape symbols account for all remaining possibilities. In a highpass channel, the symbols $p = \pm 1$ might occur with frequency up to 25% each, and $p = \pm 2$ up to 5% each. Zero runs might account for 12% (length 1) and 6% (length 2) and 3% (length 3). The lowpass channels, with smaller bin widths, have more information in higher values of p. A block of subbands will use the same Huffman table to encode the quantized values. This yields the compressed image data in the figure below. The decoder reverses the entropy coding, then the quantization, and finally the wavelet transform.

Original Image

Reconstructed Image at 0.25 bpp

11.2 Image and Video Compression

Image Compression

"A picture is worth a thousand words." This English aphorism reminds us of the importance of images. It is especially true in the age of information highway and multimedia. Computers, fax machines, videophones, teleconferencing systems and storage devices impact our workplace. Text, data, sound, image and video clip are grouped together to send over data networks or to store. The amount of data is astronomical. Compression increases the throughput of the network and the capacity of the storage device. For satellite transmission, compression greatly reduces cost.

A 24-bit color picture with 256 by 256 pixels needs more than 0.2 MByte of storage. A high density diskette with capacity of 1.4 MB can store about 7 pictures. If the picture can be compressed by 50 to 1 without any perceptual distortion, the capacity of the diskette increases to 350 pictures. This is significant. The key point here is the notion of *perceptual* lossless compression. A good coding algorithm should study and incorporate the human visual system to exploit redundancy in the image.

There are many techniques for image coding. Subband coding is today the most successful. Pyramid coding was and is effective for high bit-rate compression. Transform coding based on the Discrete Cosine Transform became popular in the 1980s because of low complexity and effective bit allocation. This became the JPEG standard in image coding. For gray-scale images, it performs well for compression ratios up to 16 to 1. At 24 to 1 compression, JPEG's synthesized image suffers from blocking, which manifests the short basis functions used in reconstruction.

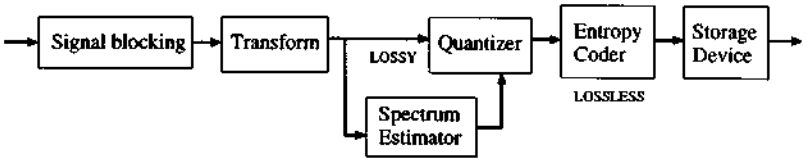


Figure 11.3: The steps of a transform-based image coder.

Subband coding using wavelets (this means tree-structured filter banks) avoids blocking at medium bit-rate, because its basis functions have variable length. Long basis functions represent flat background (low frequency). Short basis functions represent regions with texture. At a reasonable distance, one can not detect the errors easily. At low bit-rate, wavelet coding suffers from *ringing* when high frequencies (textures) are deleted. The ringing artifacts are significant around edges with high intensity. They have the shape of the basis functions that are emphasized in synthesis.

Part of the "Barbara" image and its Discrete Cosine Transform are in Figure 11.4. The image is blocked (8 by 8) and then transformed. The transformed subimage also has size 8 by 8. The intensities of the first few coefficients (the upper left corner of the transform) are the largest. The DCT preserves energy and the essential information is in those few coefficients. Quantization assigns more bits to these pixels. The objective of bit allocation is to minimize the distortion. The quantized subbands are then scanned and coded using lossless compression. This *entropy coder* watches for runs of zeros and transmits their length (roughly 3 : 1 compression for free).

Similar procedures are used in the wavelet-based transform coder. Figure 11.5 shows the

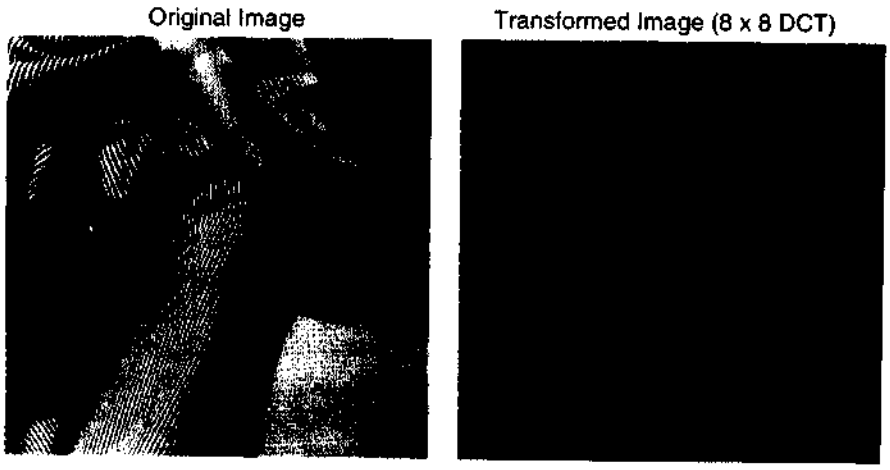


Figure 11.4: The discrete cosine transform shows most energy in the DC coefficients (bright points).

transform using a two-channel filter bank. The upper left subimage is obtained by lowpass filtering in both the horizontal and vertical directions, indicated by *LL*. The other three subimages (*much lower intensity*) have details involving high frequencies. The bit allocation algorithm will assign many bits to *LL* and few bits to *HH*. The normal number of iterations on the *LL* subimages is 4 or 5, as shown on the right side of Figure 11.5.

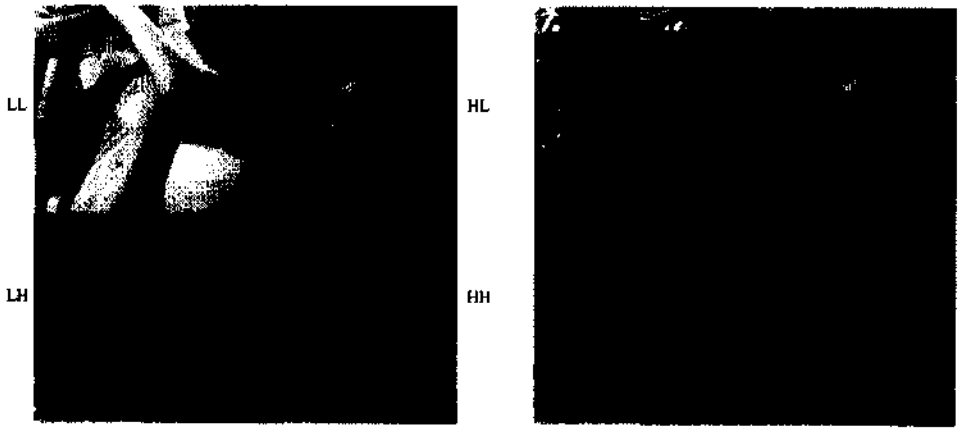


Figure 11.5: The discrete wavelet transform of "Barbara" (one level and four levels).

The subsections below discuss these building blocks in detail. The organization is:

- Choice of transformations and their effects on image coding.
- Bit allocation algorithm and quantization.
- Entropy coding algorithms.

- Error measures.
- Comparison of block transforms and wavelets.

Choice of Transformations and Their Effects on Image Coding

Consider the block transform coder in Figure 11.6. The input blocks are

$$v(n) = \begin{bmatrix} x(nM) & x(nM-1) & \cdots & x(nM-N+1) \end{bmatrix}^T.$$

Each block is transformed to length M by $y(n) = P^T v(n)$. The figure shows a uniform filter bank with M channels. The kth row of P^T contains the filter coefficients $h_k(n)$. Examples are the DCT ($N = M$), the Lapped Orthogonal Transform ($N = 2M$), and the Generalized LOT ($N = LM$).

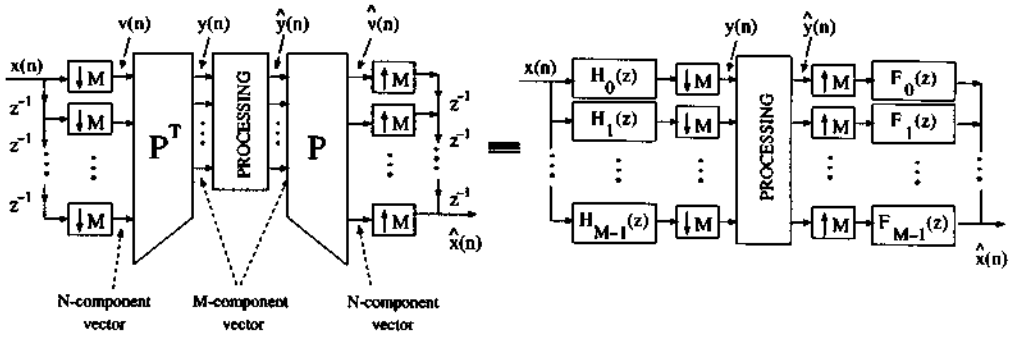


Figure 11.6: Block transform and the equivalent filter bank. The kth row of $\mathbf{P}^T$ contains $\mathbf{h}_k(n)$.

The subband signal $y(n)$ is quantized, entropy coded, and stored. In synthesis, the stored signal is decoded (to produce $\hat{y}(n)$) and then transformed by the matrix P. In the absence of quantization, $P = \begin{bmatrix} P_0 P_1 \cdots P_{L-1} \end{bmatrix}$ gives an orthogonal transform:

$$\sum_{m=0}^{L-1-\ell} P_m P_{m+\ell}^T = \delta(\ell) I; \quad 0 \le \ell \le L-1. \tag{11.1}$$

The synthesis filters are time-reversed versions of the analysis filters. The reconstructed signal $\hat{x}(n)$ is a linear combination of the basis vectors. What properties of those vectors (columns of P) will produce a good perceptual image coder?

- *They should be smooth and symmetric (or antisymmetric).* Smoothness controls the noise in a region with constant background. Symmetry allows the use of symmetric extension to process the image's borders.

- *They should decay to zero smoothly at both ends.* Figure 11.7(a) shows the basis functions of the DCT. Note that they do not decay to zero. This creates discontinuity between blocks (subimages) when the image is compressed. This *blocking artifact* can be seen from the reconstructed image in Figure 11.7(b). The compression rate is 32 to 1 (0.25 bpp).

- *The bandpass and highpass filters should have no DC leakage.* Higher frequency bands will be quantized severely. It is desirable for the lowpass band to contain all of the DC information. Otherwise, if the bandpass and highpass responses to $\omega = 0$ are not zero, we see the *checkerboard artifact* in Figure 11.8(b).

  An equivalent form of the DC leakage condition can be derived for the lowpass filter: $H_0(2k\pi/M) = \delta(k)$ for $k < M$. These ω_k are the mirror frequencies.

- *The basis functions should be chosen to maximize coding gain.*

- *Their lengths should be reasonably short to avoid excessive ringing and reasonably long to avoid blocking.* Figure 11.9 shows transforms of length 8 (DCT) and 48 (GenLOT) used in 32 :1 compression. The first transform has blocking and the second transform has ringing. Intuitively, one would like to represent texture by short functions and background by long functions. This is offered by wavelets.

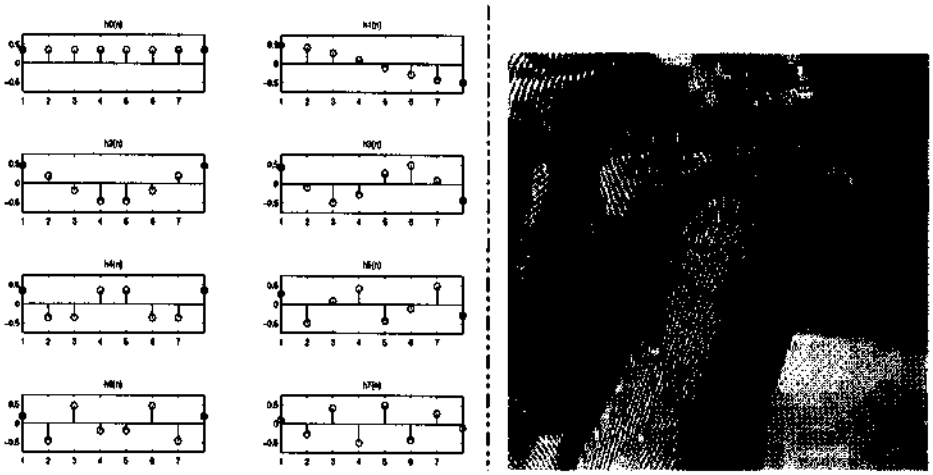


Figure 11.7: *Blocking* at 32 :1 comes from discontinuities of the 8 DCT basis functions at edges.

- *In the frequency range* $|\omega| \leq \frac{\pi}{M}$, *the bandpass and highpass responses should be small.* This minimizes the quantization effect on bandpass and highpass.

Wavelet-based transform One attractive property of wavelets is their ability to adjust the lengths of basis functions. A four-level wavelet decomposition and its equivalent nonuniform filter bank are in Figure 11.10. The low frequency basis function is a cascade of interpolated versions of the lowpass filter $\boldsymbol{H_0}$. Its effective length is large. Higher frequencies are iterated less; the basis functions become shorter. The signal is approximated by a few basis functions. After four levels in Figure 11.5, most of the energy is in the lowpass subband. This upper left subimage is a coarse approximation of the original. The other bands add details. *Bit allocation becomes crucial.* Clearly, subimages with low energy levels should have fewer bits.

We comment further on desirable properties of a two-channel filter bank, now emphasizing what becomes important with *iteration.*

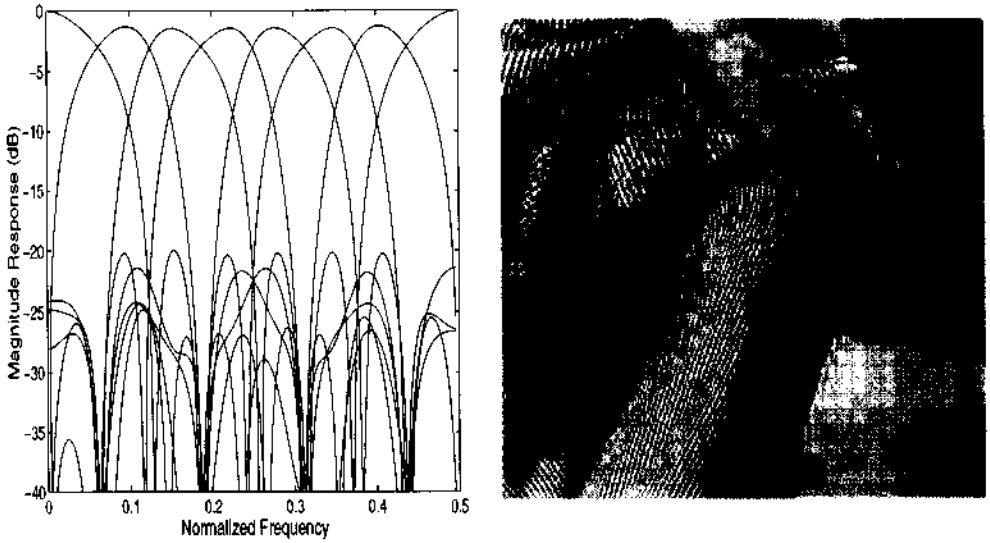


Figure 11.8: Frequency responses of GenLOT with $M = 8$ and $N = 24$. The bandpass responses at $\omega = 0$ (DC) are not zero. The DC component leaks over to produce *checkerboarding*.

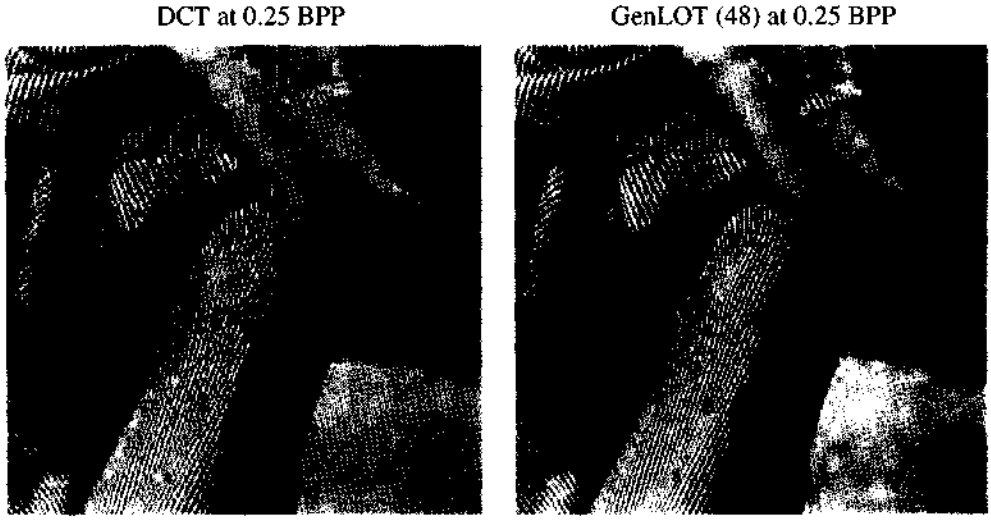


Figure 11.9: Short DCT basis vectors produce blocking. Long GenLOT vectors produce ringing.

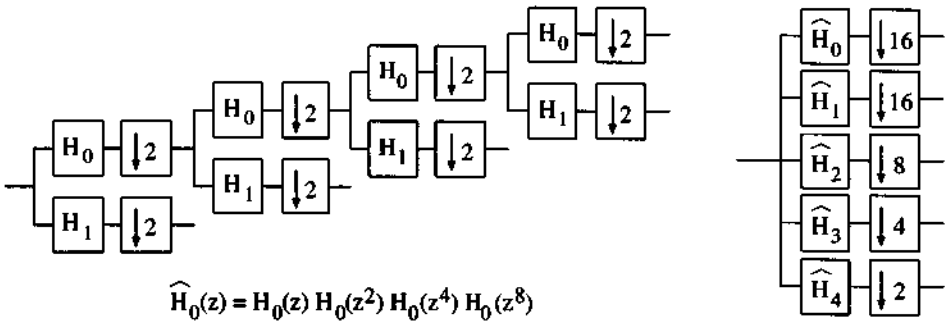


Figure 11.10: A four-level discrete wavelet transform and its equivalent nonuniform filter bank.

- *The synthesis scaling function should be smooth and symmetric.* Figure 11.11 shows several sawtooth subsections, plus two large jumps. Those jumps in the synthesis scaling function produce "blocking-like artifacts" in the reconstructed image.
- *The highpass analysis filter should have no DC leakage.* When output from this filter is quantized, we do not want to affect the DC component of the signal. It is desirable that the lowpass subband contains all the DC energy. Figure 11.12 shows the filter banks, their scaling functions, and the reconstructed image (at 1 bpp). Note the checkerboard artifact and the nonzero responses at $\omega = \pi$. Wavelet theory requires the lowpass filters to have at least one zero. This is also beneficial for image coding.
- *The analysis filters should be chosen to maximize coding gain.*
- $F_0(z)$ *should be long and* $F_1(z)$ *should be short.* A long $F_0(z)$ will help the coder to represent flat regions. A short $F_1(z)$ will minimize ringing due to quantization. Figure 11.13(a) shows blocking at 0.25 bpp from the short Haar filter. Figure 11.13(b) shows ringing for a longer 9-tap $F_1(z)$.
- $H_1(z)$ *should have good stopband attenuation to minimize the leakage of quantization noise into low frequencies.*
- How does one factor a given halfband $P(z)$ into $H_0(z)\ F_0(z)$? One should choose $H_0(z)$ short (and therefore $F_1(z)$ short) and $F_0(z)$ long. $H_0(z)$ should have at least one zero at $\omega = \pi$. $F_0(z)$ should have many zeros at π to obtain a smooth scaling function. An example is the filter bank with length (2, 14) (factors of the maxflat filter). The total of eight zeros at π is the same as the (9, 7) FBI filter bank. Figure 11.14 shows the reconstucted images at 0.5 bpp using the (2, 14) and FBI systems. The PSNR is 24.68 dB and 25.30 dB for (2, 14) and (9, 7). The image qualities are comparable, although (2, 14) has smaller coding gain (9.45 dB) compared to 9.787 dB of the FBI system.

Bit Allocation and Quantization

After transformation the subband signals are quantized, entropy coded and stored (or transmitted). Figure 11.15 shows typical distributions for the subband signals. For bandpass and highpass subbands, zero-mean Gaussian is a good approximation. The lowpass subband is image-dependent and its distribution is roughly uniform. *Given M subimages and a fixed bit rate R,*

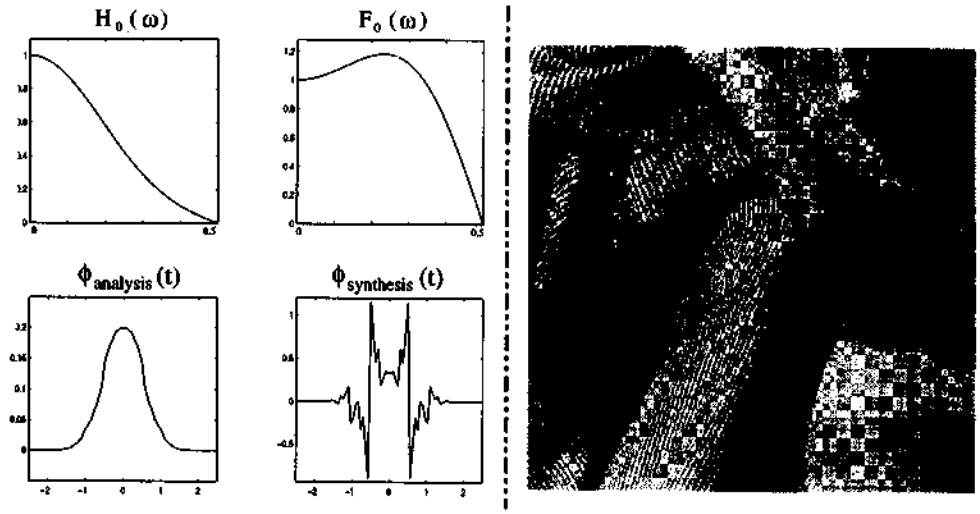


Figure 11.11: Frequency responses and scaling functions. The synthesis scaling function is rough and has large jumps. This produces blocking-like artifacts.

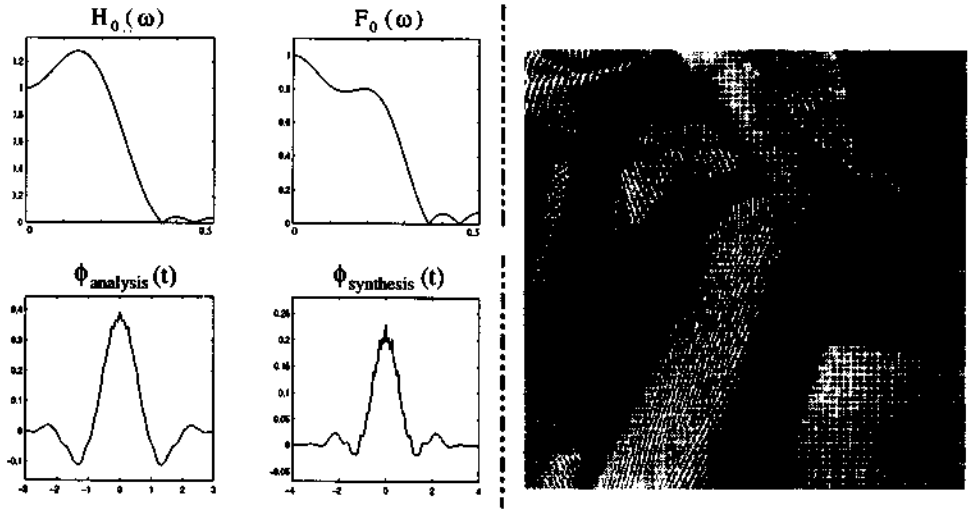


Figure 11.12: Frequency responses and scaling functions of a filter bank. The lowpass responses are not zero at $\omega = \pi$. This DC leakage yields a checkerboard in the reconstructed image.

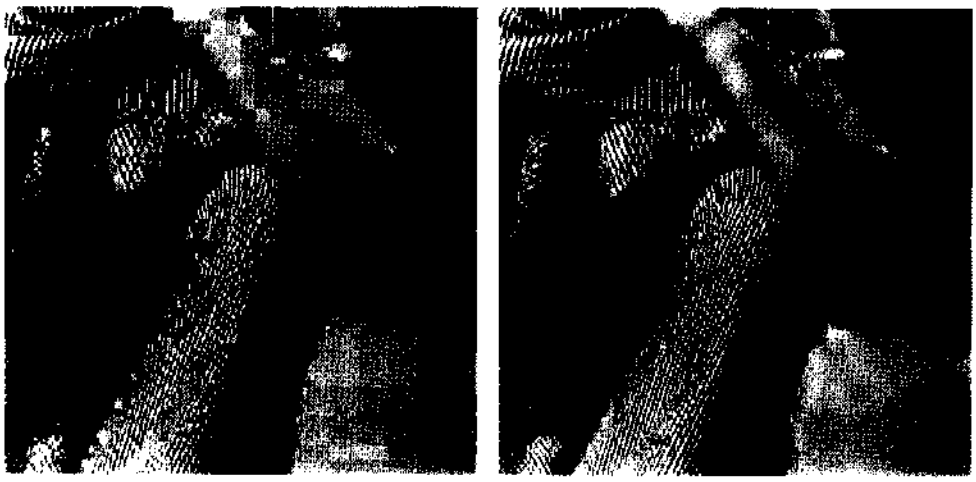


Figure 11.13: Blocking from a short synthesis lowpass filter. Ringing from a long synthesis highpass filter.

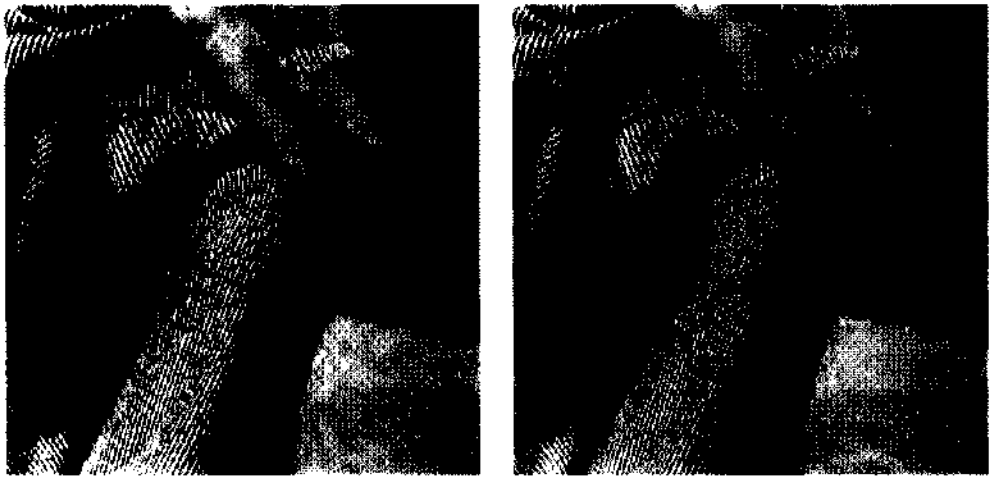


Figure 11.14: Image compression at 0.5 bpp using (2, 14) and (9, 7) filter lengths.

how does one assign bits to the subimages? The bit allocation algorithm needs a cost function D. Examples of D are distortion, energy and entropy. We discuss the minimization of distortion for a uniform quantizer with variable bit length next.

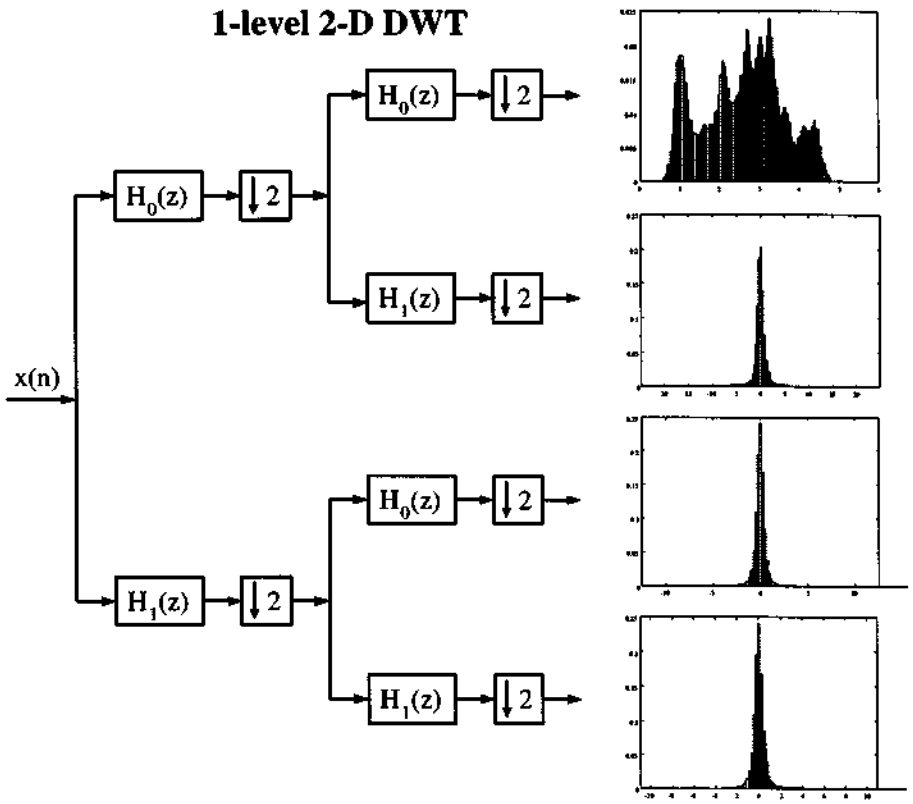


Figure 11.15: Typical distributions of subband coefficients.

Consider the 4-level wavelet decomposition in Figure 11.5. The original image size is 256 by 256 with 8 bits per pixel, for a total of 524,288 bits. How does one assign 16,384 bits to the subbands for a 32 to 1 compression? The numbers of pixels in the subbands go from 16 by 16 up to 128 by 128. Assigning one extra bit to the coefficients in the small subband does not increase the total number of bits as much as assigning the extra bit to the large subbands. Let

- N be the total number of bits in the original image
- M be the number of subbands ($M = 13$ for four levels)
- $\alpha_k = N_k/N$ be the relative subband size
- $\boldsymbol{b} = (b_1, \ldots, b_M)$ contain the bit lengths allocated to the subbands
- ϵ_k be the quantizer performance index
- σ_k^2 be the subband variance
- w_k be the weighting factor (for perceptual coding).

Define the quantization error and the total bit rate as:

$$D(\boldsymbol{b}) = \sum_{k=1}^{M} \alpha_k w_k \epsilon_k^2 \, 2^{-2b_k} \, \sigma_k^2 \quad \text{and} \quad R(\boldsymbol{b}) = \sum_{k=1}^{M} \alpha_k \, b_k.$$

The bit allocation problem is to minimize $D(\boldsymbol{b})$ subject to $R(\boldsymbol{b}) = R_c$ = fixed bit rate. This can be solved using a Lagrange multiplier λ:

$$\min \; \{D(\boldsymbol{b}) + \lambda R(\boldsymbol{b})\} = \min \sum \alpha_k \left(w_k \epsilon_k^2 \, 2^{-2b_k} \, \sigma_k^2 + \lambda \, b_k \right). \tag{11.2}$$

Assume that all $\epsilon_k = \epsilon$, and set $\widetilde{\lambda} = \lambda/\epsilon^2$. Then (11.2) reduces to

$$\min \; \left\{ w_k \, 2^{-2b_k} \, \sigma_k^2 + \widetilde{\lambda} \, b_k \right\} \quad \forall k. \tag{11.3}$$

Differentiate (11.3) with respect to b_k to obtain a closed form solution:

$$b_k = \tfrac{1}{2} \log_2 \left(\frac{(2 \log_e 2) w_k \sigma_k^2}{\widetilde{\lambda}} \right). \tag{11.4}$$

Then the bit-rate constraint $R(\boldsymbol{b}) = R_c$ becomes

$$\sum_{k=1}^{M} \alpha_k b_k = \tfrac{1}{2} \sum_{k=1}^{M} \alpha_k \log_2 \left(\frac{(2 \log_e 2) w_k \, \sigma_k^2}{\widetilde{\lambda}} \right) = R_c.$$

This yields the multiplier $\widetilde{\lambda} = \lambda/\epsilon^2$:

$$\widetilde{\lambda} = 2^{\left\{ \sum_{k=1}^{M} \alpha_k \log_2 ((2 \log_e 2) w_k \, \sigma_k^2) - 2R_c \right\}} \tag{11.5}$$

In summary, the average subband variances σ_k^2 lead to $\widetilde{\lambda}$, subject to the constraint. Then the allocated bit length b_k is computed from (11.4). When R_c is too small, the resulting bit length can be negative for subbands with very small signal variances. Since one can not assign negative bits, the bit allocation algorithm is restarted with these subbands removed (0 bits). The iterative algorithm is:

1. Find bit lengths b_k and set all negative b_k to zero.
2. Reduce the number of subbands appropriately, $M_{new} = M_{old} - M_{zero}$.
3. Adjust $R_c = R_{c,new}$ and repeat step 1. Stop if all $b_k \geq 0$.

For a linear-phase filter bank without orthogonality, the signal energy is not preserved. Quantization noise introduced in the subbands is amplified by the synthesis bank. Assuming that the quantization errors are uncorrelated and defining $B_k = \alpha_k \sum_n f_k^2(n)$, the reconstruction error variance is $\sigma^2 = \sum_{k=0}^{M-1} B_k \, \sigma_k^2$. B_k normally ranges from $0.9\alpha_k$ to $1.1\alpha_k$ for practical systems, and it is reasonable to assume that $B_k = 1$ (as for an orthogonal filter bank).

From b_k and σ_k, we compute the step size Δ_k for the quantizer. The maximum allowable quantization error for the kth subband is $T_k = c_k \, \sigma_k / 2^{b_k}$, where $c_k = 8$ is a reasonable choice. The step size Δ_k is normally chosen to be

$$\Delta_k = \max \, (2T_k, \, \Delta_{k,min}) \text{ for } b_k > 0 \quad \text{and} \quad \Delta_k = 2X_{k,max} + \epsilon \text{ for } b_k = 0. \tag{11.6}$$

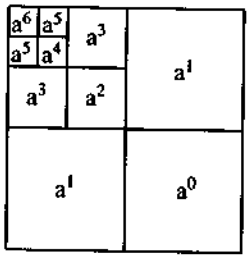
Here $X_{k,max}$ is the largest nonzero value, $b_{k,max}$ is the maximum number of bits for coding a nonzero normalized coefficient, and $\Delta_{k,min}$ is the minimum step size to guarantee that $b_{k,max}$ is not overridden for coding $X_{k,max}$. The Huffman table of the sequential baseline coder has a limited size and should be kept small. The table is also quantized and transmitted. Therefore, Δ_k must be greater than or equal to the smallest δ_k that is not quantized to zero. Summarizing,

$$\Delta_{k,min} = \max \left\{ \frac{X_{k,max}}{2^{b_{k,max}} - 1}, \quad \delta_k \right\}; \quad \epsilon \geq \delta_k.$$

The exception for the case where $b_k = 0$ in (11.6) assures that all coefficients are quantized to zero if no bit is assigned. $\epsilon \geq \delta_k$ is necessary to keep all normalized coefficients $\tilde{y}_k = \text{round}\ (y_k/\Delta_k)$ smaller than 0.5.

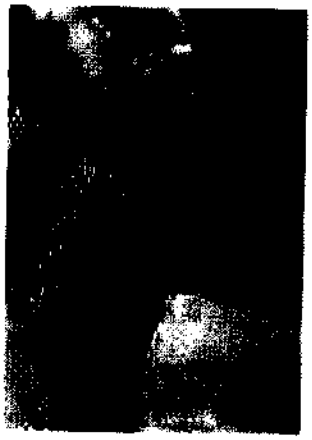
The lowpass channel variance strongly depends on the signal. Its distribution is approximately uniform. Δ_1 computed from (11.6) would be too large since the uniform quantizer is more effective for uniform distribution (rather than Gaussian). This error is equalized appropriately by scaling the lowpass subband variance by $\tilde{\sigma}_1^2 = c_1\, \sigma_1^2$ before bit allocation. c_1 is determined experimentally and is different for every signal. It is about 1.5 for the image Lenna.

Perceptual weighting for wavelet-based image coding It is well known that minimizing the squared error does not guarantee optimal results in the perceptual sense. At medium and low bit-rates, human eyes are less sensitive to loss of high frequencies. One needs to weight the quantization noise in the subband using the sensitivity of the human eye. But perceptual quality is complicated. We summarize here a simple and effective weighting scheme for the estimated quantization noise before bit allocation. More bits are allocated to low and medium bands, and high frequency noise is increased. The figure below shows the weight factors for three levels with $a > 1$ and the improvement for $a = 2$.



(9,7) system at 0.125 bpp, a=1

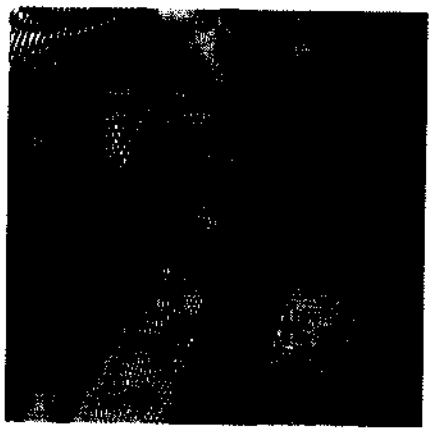
(9,7) system at 0.125 bpp, a=2



The choice of weight for an M-channel GenLOT is $w = \begin{bmatrix} a^{M-1} & \cdots & a & 1 \end{bmatrix}^T$. Extending to 2-D yields $W = w\, w^T$. Here are the reconstructed images using GenLOT of length 48, with $a = 1$ and $a = 2$. Note again the improvement with $a = 2$.

GenLOT(48) at 0.125 bpp, a=1

GenLOT (48) at 0.125 bpp, a=2



Since PSNR is maximal for $a = 1$, this weighting is a tradeoff between perceptual quality and error measure of the reconstructed image. At lower rates, perceptual quality is more important. The weighting scheme is simple, efficient, and image independent. It does not guarantee perceptually optimal results! A reliable measure for perceptual quality (and the best weight) is an ongoing research problem.

Entropy Coding

After bit allocation and quantization, we have subimages with discrete levels represented by integers. How do we store or transmit these subimages? Many highpass coefficients are zero after quantization. These coefficients should be grouped so that the entropy coder can take full advantage of long strings of zeros. This is accomplished by *scanning*.

Run-length coding or Huffman coding or a combination should be used to reduce the redundancy of the images [PM]. We will discuss the baseline entropy coder which is a combination. JPEG also uses the baseline coding method.

Scanning of the discrete wavelet coefficients To demonstrate scanning, consider the three-level wavelet transform in Figure 11.16. Subbands 2, 5 and 8 are highly correlated since 2 is the coarse approximation of 5 and 5 is the coarse approximation of 8. Suppose the pixel at the upper left corner of subband 2 is zero. *Then it is very likely that the pixels in a* 2×2 *shaded square of subband* 5 *are zero.* Similarly, the pixels in a 4×4 shaded square of subband 8 are probably zero. One can group these pixels into an "AC sequence" of length 21 ($= 1+4+16$) by vertically scanning the shaded squares. Figure 11.16 also shows the scanning patterns for subbands 3, 6 and 9 (horizontal) and for subbands 4, 7 and 10 (diagonal). When the original image has size 32, the 16 pixels each in subbands 2, 3, 4 give 48 AC sequences. The low frequency band is scanned horizontally and grouped into the DC sequence of length 16. This scanning method is similar to the *zero-tree coder* proposed by [Shapiro].

Scanning of the block transform Consider an image of size 32×32 transforms to 16 blocks of size 8×8, using an 8-channel GenLOT. The quantized coefficients are scanned and entropy-coded. The ℓ coefficients in block k are correlated with the ℓ coefficients in blocks $k \pm m$. There-

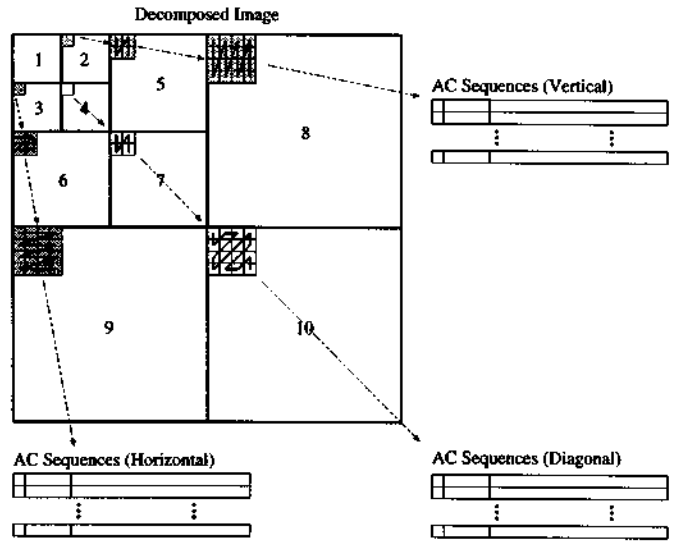


Figure 11.16: Scanning method used in the Discrete Wavelet Decomposition.

fore one should scan them in a zig-zag pattern, as shown in the figure. These scanned coefficients are grouped to sequences of length 16. There are 64 such sequences.

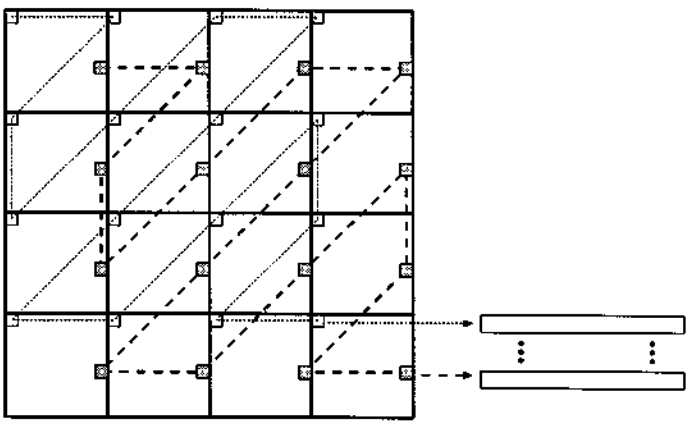


Figure 11.17: Scanning method used in the block transform.

Sequential Baseline Coding After scanning the quantized subimages, we have a set of DC and AC sequences to be stored. The baseline coder takes advantage of the correlation in the AC sequences. This algorithm combines Sequential Baseline Coding and Huffman entropy coding. The basic principle is similar to the JPEG coder, but does not restrict to the DCT.

Coding of the AC sequence: The sequence $9\ 0\ 0\ 2\ 0\ 1\ 0\ 0\ 0\ -3\ 0\ 0\ 3\ 0\ -1\ -1$ has strings of zeros interlacing with nonzeros. An efficient representation remembers the number of zeros before each nonzero. ***Symbol-1*** is the pair (*Runlength, Size*) where *Runlength* specifies the

number of preceding zeros. *Size* determines the number of bits to encode the current nonzero. ***Symbol-2*** gives the (*Amplitude*) of the nonzero: *Size* = n corresponds to *Amplitude* less than 2^n (but not less than 2^{n-1}).

This representation of the example gives a string of ***symbol-1*** and ***symbol-2***:

$$(0,1)4\ \ (2,2)2\ \ (1,1)1\ \ (3,2)-3\ \ (2,2)3\ \ (1,1)-1\ \ (0,1)-1\ \ (0,0).$$

Note the terminal ***symbol-1*** (0, 0) at the end. Also (1, 1) and (0, 1) and (2, 2) occur twice in the string. Huffman entropy coding (see Figure 11.18) can exploit this redundancy in ***symbol-1***. A simple way to code ***symbol-2*** "a" in binary is:

$$b = \begin{cases} a - 2^{Size-1} & \text{if } a > 0 \\ |a| & \text{otherwise.} \end{cases}$$

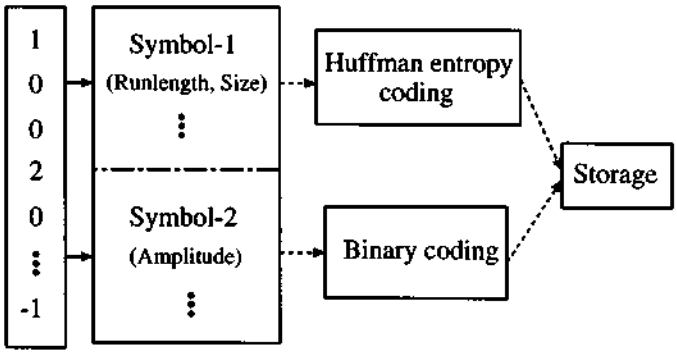


Figure 11.18: Steps in a Sequential Baseline Coder.

Coding of the DC sequence: DC coefficients measure the average energy of the input signal. There is usually a strong correlation between neighboring coefficients. For efficiency we use ***differential coding***: save the first coefficient and then the ***differences between successive coefficients***. These are coded as for AC coefficients. Since one would not expect long zero strings, *Runlength* is not used. ***Symbol-1*** only gives *Size*. An excellent source for Sequential Baseline Coder is [PM].

Three error measures are often used to compare coders and perceptual quality:

$$\begin{cases} \text{Mean Square Error} & MSE = \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}|x(m,n) - \hat{x}(m,n)|^2 \\ \text{Peak Signal Noise Ratio} & PSNR = 10\ \log_{10}\left(\frac{255^2}{MSE}\right) \\ \text{Maximum Error} & MaxError = \text{Max}\ |x(m,n) - \hat{x}(m,n)| \end{cases} \tag{11.7}$$

The image is $M \times N$. The MSE and PSNR are directly related, and one normally uses PSNR to measure the coder's objective performance. At high rate, images with PSNR above 32 dB are considered to be perceptually lossless. At medium and low rates, the PSNR does not agree with the quality of the image.

Comparison of image coder based on block transforms and wavelets

Block Transform Image Coder: GenLOT with M *channels* Figure 11.19(a) shows the coding gain as a function of the overlapping factor N, for various values of M. Notice the large improvement from 4 channels to 8 channels. The added improvement for $M = 10$ and 16 channels is less. For fixed M, one observes a steady increase in coding gain as N increases. GenLOT with 8 channels is a good compromise between the implementation complexity and performance.

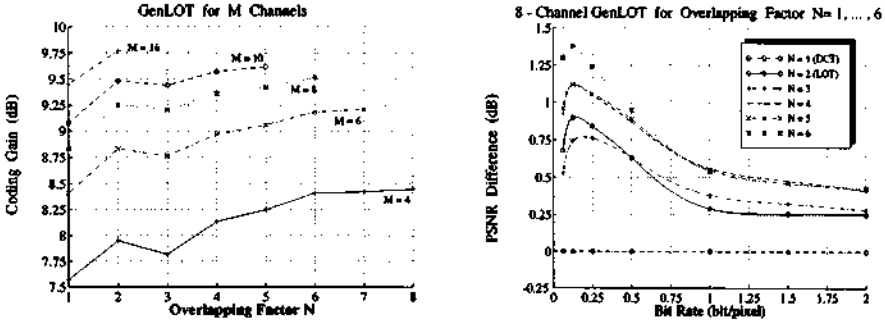


Figure 11.19: (a) Plots of coding gains as functions of overlapping factor. (b) Plots of PSNR differences for the "Lenna" image using 8-channel systems for various bit rates.

Figure 11.19(b) shows the PSNR difference between GenLOT and DCT, for several overlapping factors N. At high rate ($\geq$ 1 bpp), the improvement is not significant. All block image coders would work well at high rate. At medium rate the improvement is larger, since DCT algorithms suffer from blocking. At low rate, blurring and ringing artifacts control the performance of the image coder.

One should also compare the *subjective* performance. Figure 11.20 shows the reconstructed images for DCT and GenLOT (length 48) at 1 bpp and 0.125 bpp. Although the PSNR improvement at 0.125 bpp is only 0.52 dB, one observes a large difference in subjective quality.

Wavelet-based Image Coder with L levels and p zeros at π All filters used in this study are factors of a halfband maximally-flat $P(z)$. One way to form symmetric $F_0(z)$ and $H_0(z)$ from $P(z)$ of length 15 is to assign four zeros at π and the four complex zeros to $F_0(z)$. Then $H_0(z)$ will have the remaining four zeros at π and the two real zeros. This yields the (9, 7) system used by the FBI. To obtain filter banks with even length (Type A), move one zero at π from $H_0(z)$ to $F_0(z)$. This yields the (10, 6) system.

Figure 11.21(a) shows the unified coding gain as function of the number of levels L and the p zeros at π in $P(z)$. For a given $P(z)$, we compute the unified coding gain for all linear-phase systems and plot the best value. See also [Maj1]. This plot assumes that the image is an AR(1) model with $\rho = 0.95$. The unified coding gain saturates around 4 or 5 levels of decomposition. ***More than 5 levels of decomposition help very little.*** Type A and Type B systems show a similar coding gain.

The best filter bank for a given $P(z)$ is used in a wavelet-based study with four levels of decomposition for the "Lenna" image. The PSNRs of the reconstructed images are computed for various rates and plotted in Figure 11.22 for Type A systems. Notice how Haar wavelets are worst. Odd-length B systems show a similar PSNR performance.

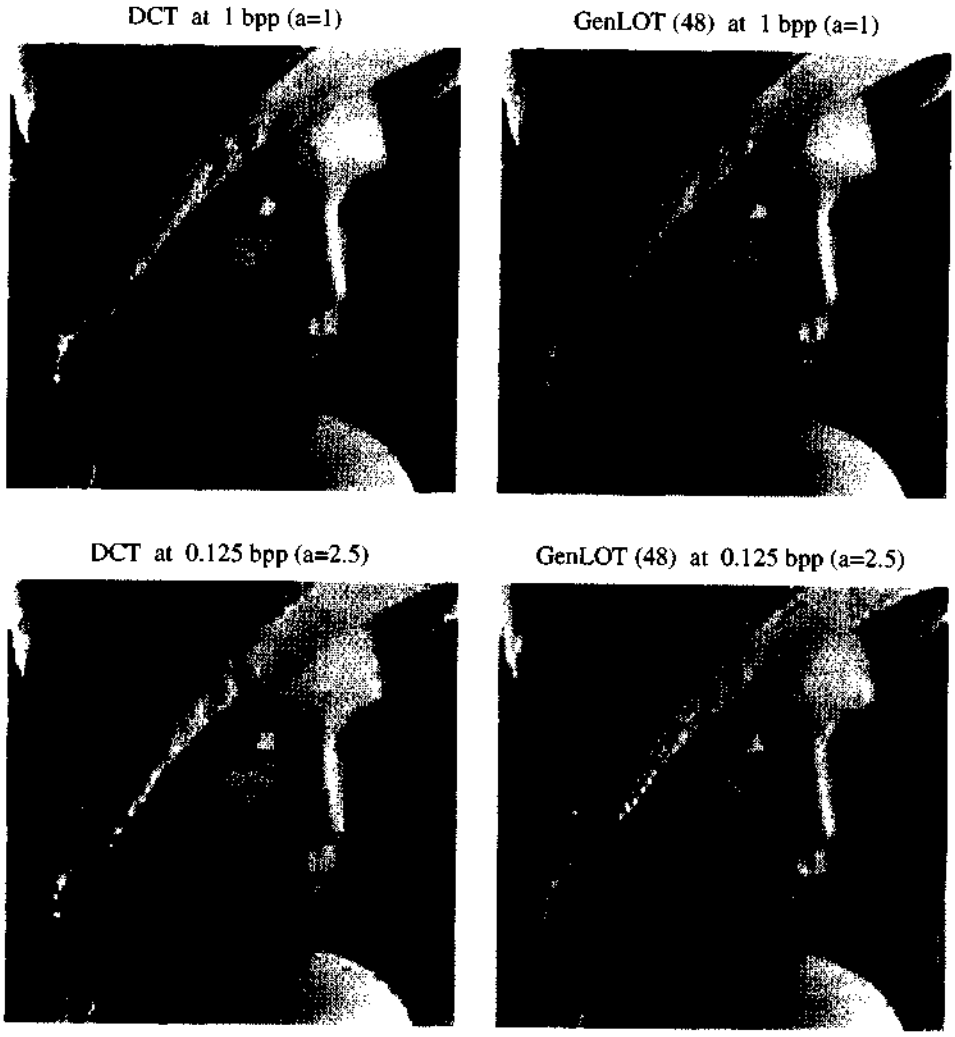


Figure 11.20: Reconstructed images using DCT and GenLOT (length 48) for 1 bpp and 0.125 bpp.

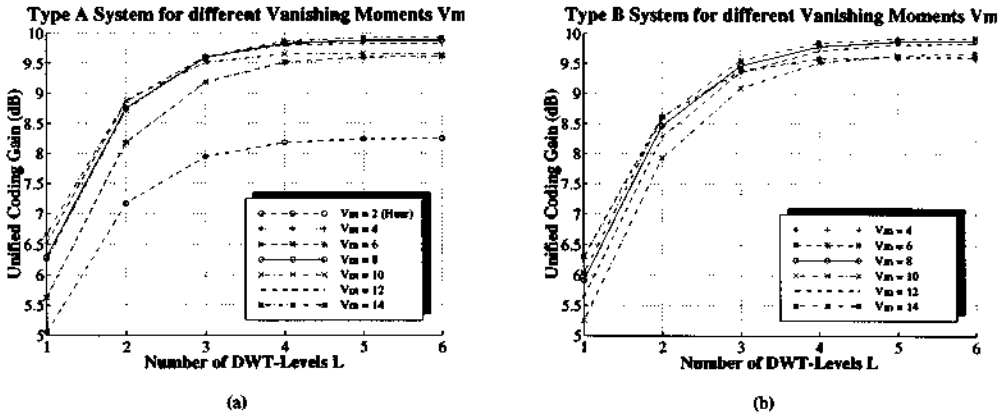


Figure 11.21: Unified coding gain for the best even-length Type-A systems and odd-length Type-B systems. Vm is the number of vanishing moments of the halfband filter $P(z)$.

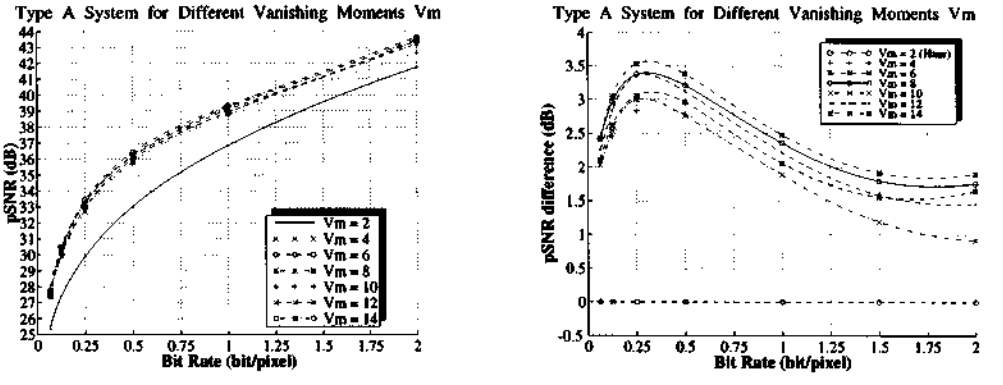
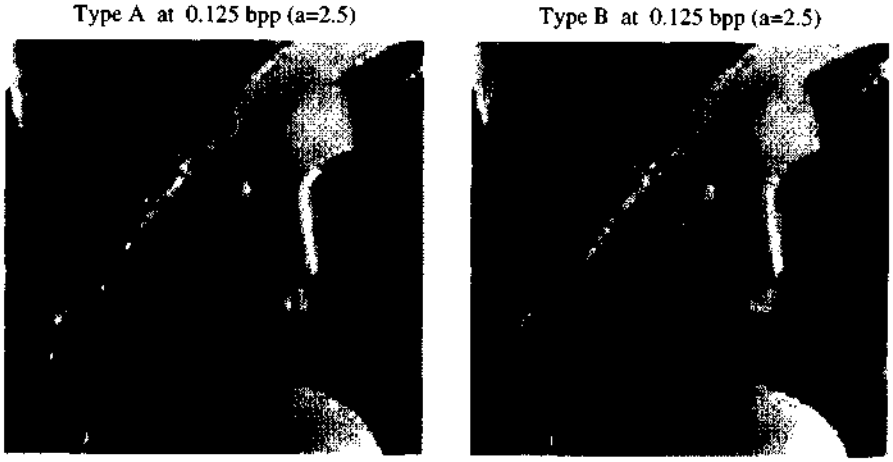


Figure 11.22: PSNR for even length filter. Difference from the PSNR of a DCT transform.

Discussion Given an image and a compression ratio, should one use a block transform (GenLOT) or a wavelet-based transform? At high rate (1 bpp), the perceptual quality is very similar. At medium rate, wavelet-based transforms perform a little better (GenLOT has ringing). At low rate, all transforms have their drawbacks and it is hard to pick one over the others. We can only display the outputs and tabulate the error measures as shown in Figure 11.23.



	(6,10)	(7,9)	GenLOT
MSE	189.1	181.7	169.3
PSNR (dB)	25.36	25.54	25.84
Max Error	88.42	93.72	90.16

Figure 11.23: Reconstructed images using Type A (6, 10) and Type B (9, 7). Error measures include GenLOT (length 48) from Figure 11.21.

Software for image compression is available at

http://saigon.ece.wisc.edu/~waveweb/QMF.html

Video Compression

Commercial systems such as video-on-demand, videophone, video conferencing and multimedia applications are being planned and manufactured for home use. The key to their success is video compression. Video on the Internet will be at a low bit-rate!

Video signals are sequences of 2D images updating at about 30 frames per second. *The new dimension is time.* One can extend separable processing from 2D to 3D. Then a video compression system would use a 3D separable filter bank in the front end. The transformed sequences are quantized and entropy coded. A bit allocation algorithm based on rate distortion theory can be used to find the optimal bit assignment.

Another approach to video compression is based on *motion estimation.* At 30 frames/second, the information in frames m and $m \pm 1$ is highly correlated. Suppose that one can estimate the

motion vectors for all pixels, to indicate where each part of the image moves in the following frame. Then it is sufficient to send the first frame (compressed) and the *motion vectors*. At the synthesis bank, the first frame is reconstructed and subsequent frames are formed by using the motion vectors (plus small corrections to the image). The quality of the reconstructed image depends on the accuracy of the estimated motion vectors.

Consider the image coder based on 8×8 blocks transformed by the DCT. The first frame is quantized, entropy coded and transmitted. The second frame is transformed into blocks. For a specific block (K, L), the search algorithm considers the neighbor blocks $(K \pm 1, L \pm 1)$ to estimate the motion vectors. These are also coded and transmitted. However, an imperfect estimate reduces the quality of the reconstructed second frame. Consequently, one also needs to transmit the expected residual error. The MPEG video standard [MPEG2] employs both forward and backward predictions for motion vector estimation.

Similar algorithms based on the wavelet transform are being developed. Where MPEG deals with subblocks, the wavelet algorithm has blocks with different sizes at different resolutions. Motion estimation is more complicated since there are several scales to search. ***We estimate motion first on a coarse scale, then on finer scales.*** The support regions also depend on the filter lengths. The memory requirement for MPEG is lower, but multiresolution is more powerful. An early comparison is in [Zafar].

Problem Set 11.1

1. If A, B, C, D occur with probabilities 0.4, 0.3, 0.2, 0.1, a Huffman entropy coder will create a binary tree:

   A, B, C, D have codes 0, 10, 110, 111

   What are the Huffman codes for X, Y, Z with probabilities 0.2, 0.5, 0.3?

11.3 Speech, Audio, and ECG Compression

Psychoacoustic model of the ear To design effective algorithms for speech and audio compression, one needs to know how hearing works. The more one knows, the better the algorithms. Psychoacoustics is the study of hearing, from which quantitative models are built. The models are based on years of extensive tests on humans, which led to these conclusions:

- *Hearing is associated with **critical bands***. These nonuniform frequency bands can be approximated by tree-structured filter banks. In speech compression, the filter bank is a four-level dyadic tree. Audio compression is based on either M-band uniform or M-band tree-structure cosine-modulated filter banks (Figure 11.24). There is always a tradeoff between complexity and accuracy in the approximation of critical bands. The filter bank in Figure 11.24(c) is more complicated than the other two. However, it approximates the critical bands more accurately.
- *Around any frequency f_m there is **masking***. An adjacent frequency f with magnitude below $T(f_m, f)$ is masked by f_m and is not audible:

$$T(f_m, f) = \begin{cases} M(f_m)\left(\frac{f}{f_m}\right)^{28}; & f \le f_m \\ M(f_m)\left(\frac{f}{f_m}\right)^{-10}; & f > f_m. \end{cases} \tag{11.8}$$

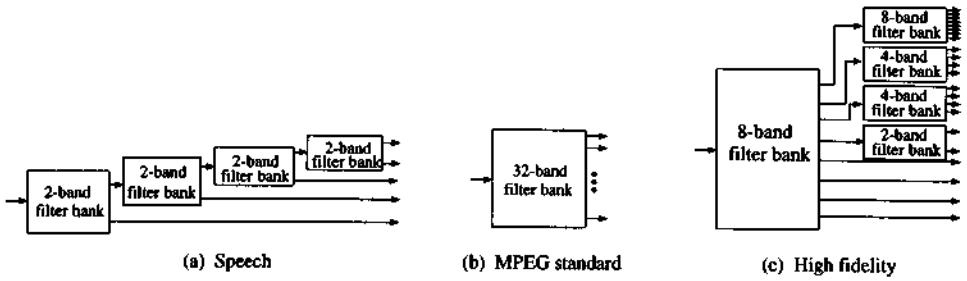
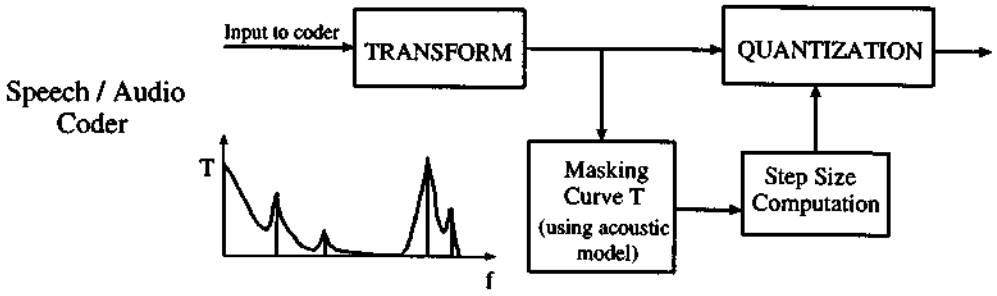


Figure 11.24: Tree-structured filter banks used to approximate the critical bands.

The quantity $M(f_m)$ is the masking threshold at frequency f_m, independent of the signal. The graph of T is linear on a log-log scale. Quantization noise can be freely added to the signal as long as it is below $T(f_m, f)$.



In a speech/audio compression system, the signal is transformed by a tree-structured filter bank. The frequency allocation approximates the critical bands of the human ear. The frequencies f_m with significant power are detected, and their masking envelopes $T(f_m, f)$ are computed. The combined masking envelope forms the *masking curve* of the signal. The quantization noise level is kept below the masking curve. Out-of-band masking noise is negligible as long as the frequency responses of the filter banks have high attenuation.

The allocation of B bits to the signal, assigning b_k bits to subband k, tries to keep the quantization noise inside the masking curve:

$$\text{Minimize} \quad \sum_{k=1}^{M} \frac{1}{12}\left(\frac{2\,p_k}{2^{b_k}-1}\right)^2 \frac{1}{\sigma_{m,k}^2} \quad \text{such that} \quad \sum_{k=1}^{M} b_k = B.$$

Here $\sigma_{m,k}^2$ is the masking power of the kth subband and p_k is its peak value.

Speech Compression

Speech compression is important in mobile communications, to reduce transmission time. Digital answering machines also depend on compression. The bit-rates are low, typically 2.4 kbits per second to 9.6 kbits/second. The best algorithms use either linear predictive models or sinusoidal models.

Speech is classified into *voiced* and *unvoiced* sounds. Voiced sounds are mainly low frequency. In CELP (code excitation linear predictor) the voiced sound is modeled as the output of an all-pole IIR filter with white noise as input. The filter coefficients are found by linear prediction. This filter represents the transfer function of the vocal tract. In a sinusoidal transform, the voiced sounds use a sinusoidal basis. Unvoiced sounds (like sss) have components in all frequency bands and resemble white noise. Model-based techniques achieve reasonable performance at low rates.

At more than 16 kbits/second, subband coding is effective and compatible with the models. Psychoacoustics has associated human hearing to nonuniform critical bands. These bands can be realized roughly as a four-level dyadic tree (Figure 11.24a). For sampling at 8 kHz, the frequency bands of the dyadic tree are: 0–250 Hz, 250–500 Hz, 500–1000 Hz, 1000–2000 Hz and 2000–4000 Hz. These bands can be quantized and coded depending on subband energy; the average signal to noise ratio is maximized. And the *noise masking property* is used.

High-Fidelity Audio Compression

Music signals have no models. The frequency spectrum of a harp is not similar to that of a piano. Subband coding, which makes no assumption on the signal model, is a natural choice. Consider an audio signal of CD quality, sampled at 44.1 kHz with 16 bits of resolution. The total bit-rate is 705.6 kbits/second. For multimedia applications, one would like to compress it to a range from 64 to 192 kbits/second (11:1 to 4 :1). High-fidelity audio compression implies that there is no perceptual loss in the reconstructed signal. This is crucial for digital audio broadcast and satellite TV, where sound quality is the most important feature. Applications of audio compression systems are:

- Digital audio broadcasting
- Satellite TV, High-Definition TV
- Contribution and Distribution links
- Production (tapeless studio, editing systems)
- Storage devices (studio & consumer market)
- Multimedia applications

The noise floor (stopband attenuation) of the filter banks should be below −96 dB (16 bits at 6 dB per bit), although lower attenuation is always preferred. To minimize noise from adjacent subbands, the stopband attenuation is required to have steep roll-off rate. Interested readers should consult [Dehery, Stautner, Brandenburg].

Electrocardiogram Compression

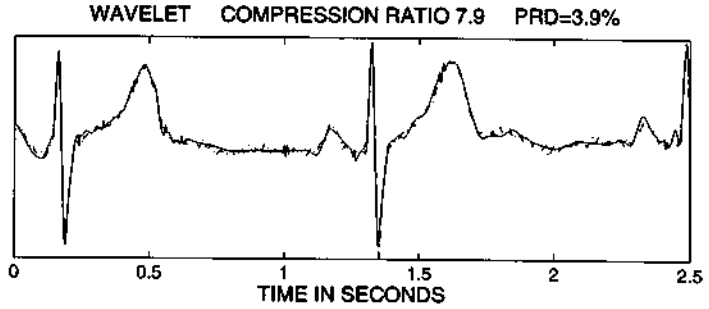
ECG waveforms are signals measured from the heart. They provide essential information to the cardiologist. Normally, a 24-hour recording is desirable to detect heart abnormalities or disorders. Storage requirements can range from 26 Mbytes, with one lead to the heart and 12 bit resolution sampled at 200 Hz, to 138 Mbytes for a system with two leads and 16 bit resolution sampled at 400 Hz. Compression is needed at a low bit-rate for both storage and telemedicine.

An ECG compression algorithm is judged by its ability to minimize the distortion while retaining all significant features of the signal. An accepted error measure is the percent root-mean-square difference (PRD). Let x_{or} and x_{re} be the original and reconstructed signals of length N:

$$\text{PRD} = \left\{ \sum [x_{or}(n) - x_{re}(n)]^2 / \sum [x_{or}(n)]^2 \right\}^{\frac{1}{2}} \times 100\%$$

Reconstruction with low PRD does not necessarily mean clinical acceptance. The crucial requirement is not to distort the diagnostic information used by the physician. This is a challenging problem since GenLOT and wavelet compression typically produce blocking and ringing. Moreover, ECG is normally used as input to classification. If compression does not change the outcome of the detection and classification algorithms, it is acceptable.

An ECG waveform (dashed line) is shown with its reconstruction (solid line). The compression ratio is 7.9 to 1, using the (13, 11)-tap filter bank. The reconstructed image preserves the significant features of the original and the PRD is 3.9%.



Techniques such as the turning point algorithm, amplitude zone time epoch coding (AZTEC), coordinate reduction time encoding system (CORTES), and the FAN algorithm compress the data by discarding relatively insignificant information [T]. The reconstructed signal is obtained by interpolating the stored samples. These algorithms are simple to implement, but they can produce significant distortion. Figure 11.25 shows the original and reconstructed waveforms using the AZTEC and FAN algorithms. We also show the PRD as function of average bit rate for the AZTEC, FAN and wavelet algorithms. The original signal has 12 bits per sample. In all our tests, wavelet-based compression gave the best objective measures.

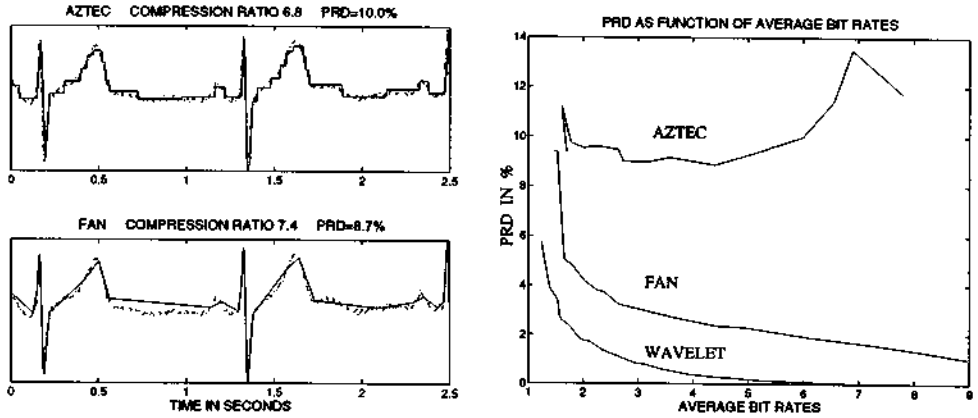


Figure 11.25: ECG Compression using the AZTEC, FAN and wavelet-based algorithms.

The blocks of the wavelet-based ECG algorithm are the same as for image coding. The input is divided into segments of N samples and these vectors are transformed (four levels, linear phase). The spectrum estimator computes a bit allocation to the subbands. They are quantized (scalar uniform) and entropy coded. The segment size N is important in determining the com-

pression ratio and the corresponding PRD. A large N increases the variance of the subband signals and the distortion. By experimenting with many sets of ECG data, we conclude that *a block size of* 2000 *is reasonable.*

Software for ECG compression is available at

http://saigon.ece.wisc.edu/~waveweb/QMF.html

11.4 Shrinkage, Denoising, and Feature Dectection

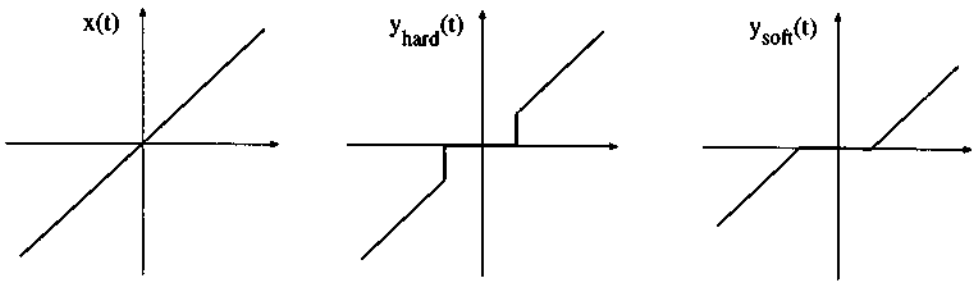
Wavelet Shrinkage

In the L-level wavelet decomposition of a signal, the number of wavelet coefficients with significant energy is small. This is a direct consequence of the approximation property of the wavelets, assuming a sufficient number of vanishing moments (say $p > 2$). The signal can be accurately represented by a small number of coefficients. *Wavelet shrinkage*, developed by Johnstone and Donoho, selects these coefficients based on *thresholding*. The wavelet shrinkage algorithm decomposes the signal into L levels and then:

- For each level, we select a threshold and apply "hard thresholding". This will zero out many small coefficients, which results in efficient representation. Thresholding is a lossy algorithm; the original signal can not be reconstructed exactly. An alternative is soft thresholding at level δ, chosen for compression performance or relative error. The outputs $y_{hard}(t)$ and $y_{soft}(t)$ with threshold δ are

$$y_{hard}(t) = \begin{cases} x(t), & |x(t)| > \delta \\ 0, & |x(t)| \le \delta \end{cases} \quad \textbf{\textit{Hard thresholding}}$$

$$y_{soft}(t) = \begin{cases} sign(x(t))(|x(t)| - \delta), & |x(t)| > \delta \\ 0, & |x(t)| \le \delta \end{cases} \quad \textbf{\textit{Soft thresholding.}}$$



The signal in Figure 11.26 is basically lowpass with noise added. We show the reconstructed signal with threshold levels $\delta = 35$ and 100. After thresholding at these levels, 92.8% and 93.4% of the coefficients are zero. The L_2 norm recoveries are 99.99% and 99.98%, respectively. Reconstruction with $\delta = 35$ is closer to the original since the threshold is lower. But it uses more terms.

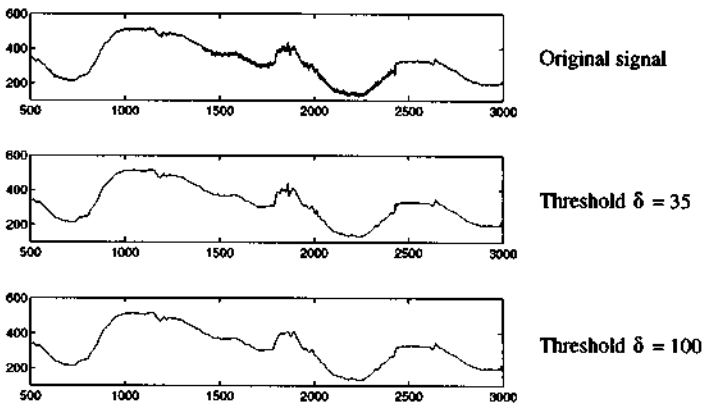


Figure 11.26: Hard thresholding generated using the *Wavelet Toolbox* from *MathWorks*.

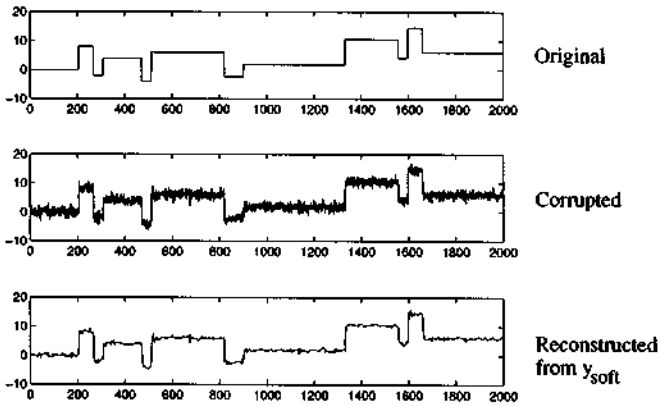
Denoising

Thresholding generally gives a lowpass version of the original signal. By selecting δ, one can suppress the noise $w(n)$ in the signal $x(n) = u(n) + \sigma w(n)$. A simple example of $w(n)$ is Gaussian white noise $N(0, 1)$. ***For denoising we use soft thresholding***.

The noise power σ^2 is assumed to be much smaller than the signal power. Moreover, the signal is assumed to have low frequency components, whereas the noise source is white. Thresholding the detailed coefficients will also remove some of signal's power. It is generally impossible to filter out all the noise without affecting the signal. Algorithms selecting the threshold levels include Stein's Unbiased Risk Estimate, fixed threshold, Minimax criteria or a combination [Donbf1].

The piecewise constant signal below is corrupted by Gaussian white noise. The corrupted signal is decomposed using the Daubechies wavelet D_6. The coefficients at level 4 are thresholded using Stein's Unbiased Risk Estimate. Notice that the reconstruction consists of the original signal and *some* of the noise.

In both wavelet shrinkage and denoising, the output is a cleaned-up version of the input. This works only when one knows the signal characteristics in advance. The algorithm will distort the desired signals when thresholding is applied.



Discontinuity Detection

A discontinuity in the signal can be detected by Haar wavelets. A discontinuity in the first derivative can be detected by D_4. The signal in Figure 11.27(a) is linear, except for an interval of zero slope. The D_4 wavelet can represent straight lines exactly ($p = 2$). The two discontinuities are being captured in the wavelet coefficients. Although wavelets with longer supports could be used, D_4 is the best choice since it has good time localization.

Figure 11.27(b) shows a signal with discontinuity in the second derivative. The wavelet used is D_8, which can represent parabolas exactly (so could D_6). Even though the discontinuity can not be seen from the signal, it is clearly detected by the coefficients in the first level of the decomposition.

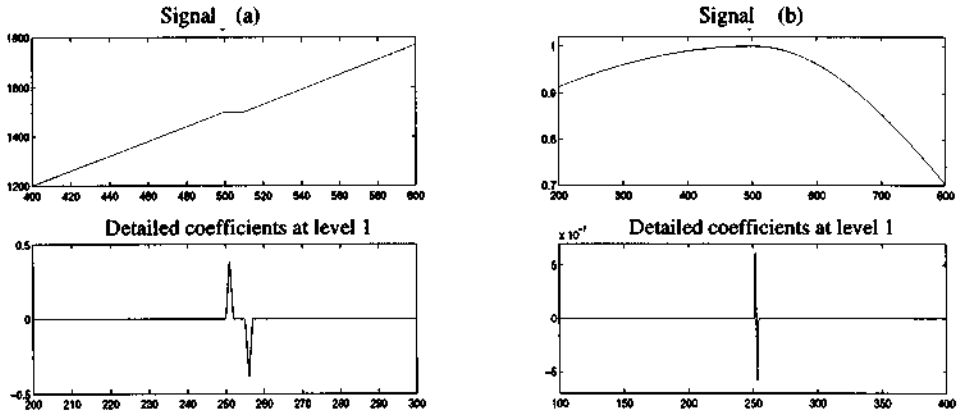


Figure 11.27: Detecting discontinuity in the first derivative and second derivative. (*These plots are generated by the Wavelet Toolbox.*)

Detecting discontinuities using a wavelet transform works well for signals with no noise. When noise dominates, one will obtain many false alarms (wrong detections). For reliable detection, one needs to follow the coefficients for several scales. Mallat pioneered this important application in continuous time for the case with no noise. The section below elaborates on the multiscale feature detection and points out its advantage comparing to the conventional feature detection.

Multiscale Feature Detection

The step edge in Figure 11.28 can be detected from the detailed coefficients using Haar wavelets. An alternative approach uses the Canny edge detector in image processing [Canny]. Both approaches work well for a signal with no noise. The output is clean and the edges can be detected. For the same input with added noise, on the right side, there are many false alarms (wrong detections).

Figure 11.29 shows the same noisy signal with its multiscale representation. Note the occurence of false alarms at each scale. ***The consistency between the peaks at all scales allows the edge locations to be detected.***

There are several issues in a multiscale feature detector. Our example uses *peak* as an indicator for the *edge* feature. However, one could also use *zero crossing* as an indicator. Given a

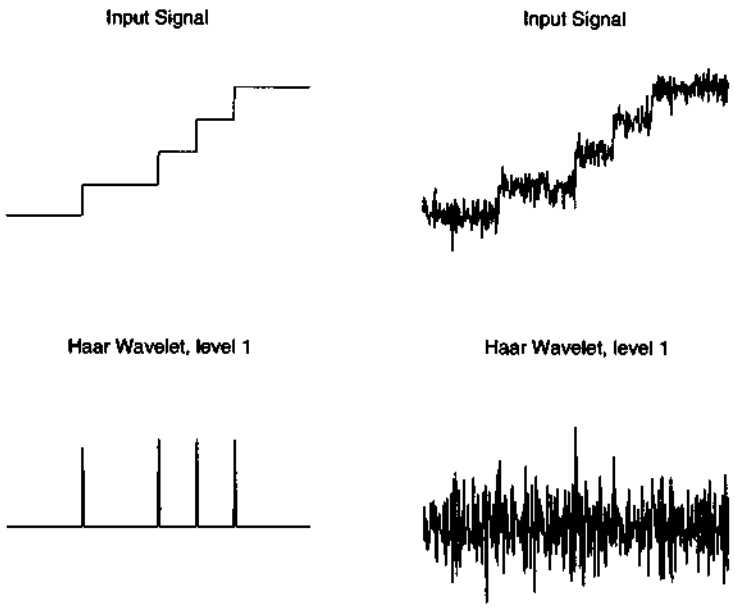


Figure 11.28: Step edge detection using Haar wavelets. The Canny edge detector is very similar. The clean step signal (left) is easy. The signal with noise (right) is difficult at one level.

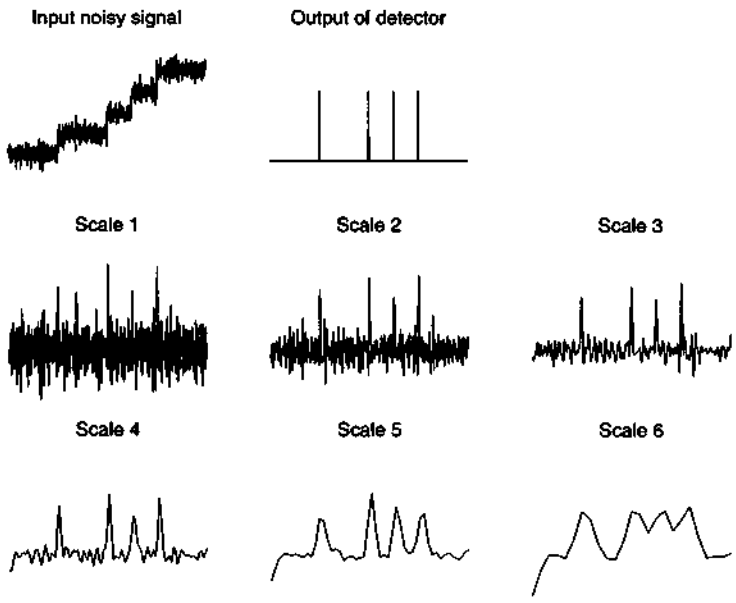


Figure 11.29: Step edge detection using the outputs at different scales.

feature, which indicator is the best one to use? The second issue is the choice of filter bank. How can one design filters that give a multiresolution representation of the indicators? The third issue is the design of an optimal detector for various noise models. Finally, many algorithms are derived in continuous time and can not be used for discrete-time implementation [Mt]. [HaNgCh] presents a general approach to discrete-time multiscale feature detection.

Problem Set 11.4

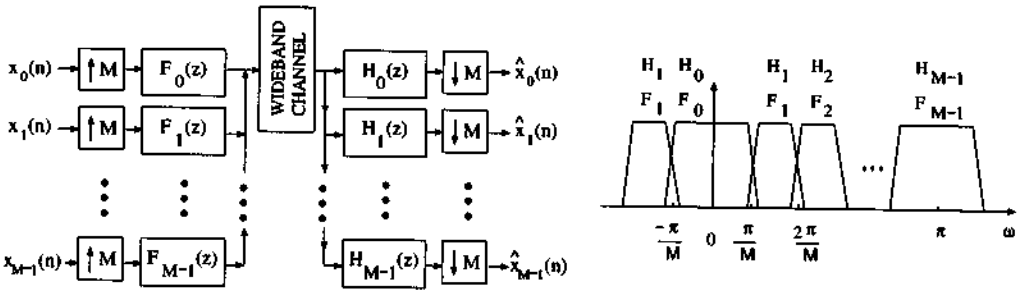
1. For the signal $x(t) = \sin t$, draw the graphs of $y_{hard}(t)$ and $y_{soft}(t)$ after thresholding with $\delta = \frac{1}{2}$.

11.5 Communication Applications and Adaptive Systems

Transmultiplexers

A transmultiplexer is a filter bank with synthesis first. This reverse order is depicted in the figure below. M signals come in, and they are combined into *one* transmitted signal. Then the receiver has to reconstruct the M separate signals.

Low-bandwidth input signals $x_k(n)$ are upsampled and filtered. The combined signal goes through a high-bandwidth channel. The received signal is filtered and downsampled to the original rates. The outputs $\widehat{x}_k(n)$ may suffer from ***distortion*** and ***crosstalk*** because of the decimation, interpolation, and non-ideal filtering.



The output $\widehat{x}_k(n)$ depends on all M inputs by $\widehat{X}_k(z) = \sum S_{k\ell}(z)X_\ell(z)$. The transfer functions $S_{kk}(z)$ give the distortion. The remaining $S_{k\ell}(z)$, $\ell \neq k$, are the crosstalk functions. The objective is to find $H_k(z)$ and $F_k(z)$ such that S(z) is a diagonal matrix, $S_{k\ell}(z) = \delta_{k\ell} z^{-n_k}$. This would be a *perfect reconstruction transmultiplexer.* The design solutions for the PR transmultiplexer are closely related to those in the PR filter bank!

Let $H_k(z)$ and $F_k(z)$ be the analysis and synthesis filters of a perfect reconstruction filter bank. [Vet3, KoNgVa] show how the filters $H_k(z)$ and $z^{-1}F_k(z)$ yield a perfect reconstruction transmultiplexer (for an ideal communication channel).

Discrete Multitone Modulation Transceivers

Transmultiplexers can be used in conjunction with discrete multitone modulation transceivers. Discrete multitone (DMT) or multicarrier modulation is a class of orthogonal frequency modulations. The transmitted data is split into several bit streams and used to modulate several carriers. The spectrum is divided into parallel orthogonal and narrowband subchannels. Each carrier occupies one subchannel. This concept was proposed thirty years ago, but did not receive much attention because of the difficulty in implementation using analog technology.

The advance of digital signal processors and related hardware has made the multicarrier modulation a preferred structure over the single-carrier QAM system. There is important interest in multicarrier modulation for high-speed data transmission over the twisted pair channel of a digital subscriber line. As a result, discrete multitone has been proposed as a standard for high-speed digital subscriber line (HDSL) and asymmetric digital subscriber line (ADSL) communication.

Consider a typical channel where the attenuation variation could be as large as 40 dB. A linear equalizer for a single-carrier QAM system would increase the noise, whereas a decision-feedback equalizer would be too complex. For many channels, multicarrier modulation approximates a constant transfer function in each subchannel. Equalization becomes very simple. Other advantages of multicarrier modulation are the reduced effect of impulse noise as a result of large symbol duration; the flexibility of not transmitting in the corrupted subchannels in case of narrowband interference; and the flexibility of transmitting important data in subchannels with high SNRs.

One disadvantage is the large peak-to-rms ratio of the transmitted signals, which might lead to nonlinear distortion. And the implementation complexity must be controlled. When the frequency separation Δf is $\frac{1}{T}$, where T is the symbol duration, the multicarrier system can use the DFT. That avoids implementing several carriers using analog technology. This is one of the main attractive features of DFT-based multicarrier modulation.

Given an ideal channel, orthogonality is ensured by the $\frac{1}{T}$ spacing between subcarriers, and they can be independently demodulated. In a dispersive channel, the carriers are no longer orthogonal. Interference depends on the spectral overlap between subchannels. DFT-based multicarrier systems, with rectangular pulse shape and -13 dB sidelobes, can have significant interchannel interference. One could employ pulse shaping to reduce spectral overlap, but the resulting transmultiplexer is not necessarily orthogonal (nor perfect reconstruction).

An alternative is to use a ***cosine-modulated*** orthogonal transmultiplexer. Its spectral containment leads to superior robustness against narrowband radio frequency interference (RFI). Interested readers should consult [RiPrNg, SaTz, TzTzPh, TzTzRe] for algorithms and comparisons between DFT and cosine-modulated multicarrier systems.

Adaptive Systems

Adaptive systems are important when the signals or environments are changing with time. Typical applications are inverse filtering, room acoustics modelling, and echo cancellation. We also mention adaptive array processing (beamforming and direction finding) and channel equalization. In these applications, an FIR filter models the signal or the environment. Its coefficients are adapted in time to minimize an error measure. Popular adaptive algorithms are LMS (Least-Mean-Square) and RLS (Recursive-Least-Square). A detailed treatment of adaptive filter theory is in [H, PRCN].

The adaptive system in Figure 11.30 is used in inverse filtering. The objective is to find $f(n)$ such that the error $e(n)$ is as small as possible. Here, the desired signal $d(n)$ is distorted by an unknown system $h(n)$ and is corrupted by noise. The z-transform of $e(n)$ is

$$E(z) = \left[F(z)\, H(z) - z^{-N}\right]\, D(z).$$

Assuming that $H(z)$ is an FIR filter, the ideal choice for $F(z)$ is $z^{-N}/H(z)$. But normally $F(z)$ is constrained to be FIR. Its length affects the convergence rate and the error.

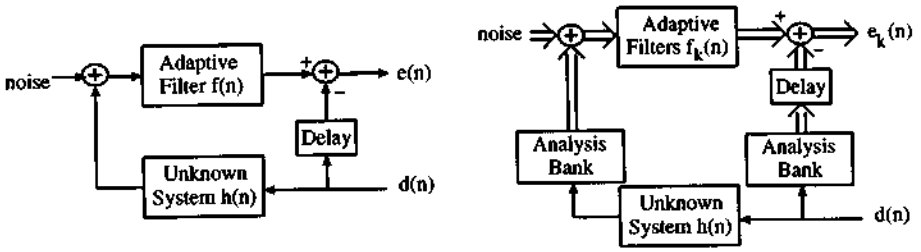


Figure 11.30: Adaptive systems in inverse filtering (left) and using a filter bank (right).

If the spectrum of $H(e^{j\omega})$ is complicated, the inverse spectrum is equally complicated and the resulting $f(n)$ is long. By separating $H(e^{j\omega})$ into many bands $H_k(e^{j\omega})$, the inverse filter in the k-th subband would not be too large. These M subbands in Figure 11.30 are adapted using M FIR adaptive filters $f_k(n)$, much shorter than $f(n)$ since the analyzing spectrum is simplified band by band. The convergence rates are faster. This is a tradeoff between hardware complexity and convergence rate.

Figure 11.31(a) shows a very different adaptive system. From the error $[H(z) - F(z)]\, D(z)$, one observes that $F(z)$ is no longer an inverse but is an approximation to $H(z)$. The length of $f(n)$ is comparable to that of $h(n)$. This system is used to cancel telephone echo coming from an impedance mismatch of the residence line and the switching center. It is most noticeable and annoying for long distance calls. Here, $s(n)$ is the speech signal, $d(n)$ is the reference signal, $y(n)$ is the echo signal received at the handset, and you hear $x(n)$.

The objective of $f(n)$ is to reproduce $s(n)$ while suppressing the echo $y(n)$. For a complicated and nonlinear echo path, an adequate $f(n)$ is too long. ***A bank of shorter filters $f_k(n)$ is better.*** The signals $s(n)$ and $d(n)$ are decomposed into M subbands, on which adaptation is performed.

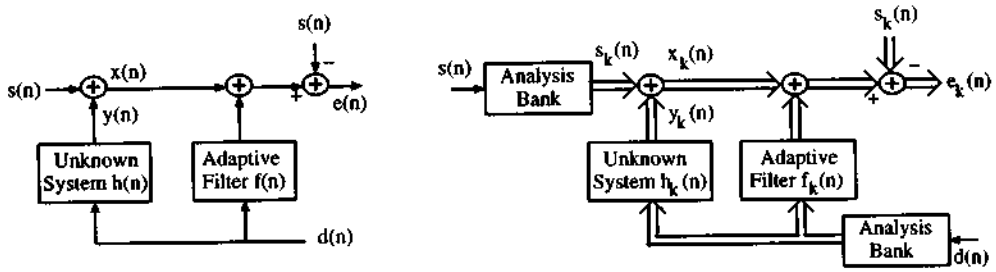


Figure 11.31: Adaptive systems for echo cancellation (left) and using a filter bank (right).

Filter bank adaptive systems give an improved convergence rate and error reduction [GilVet,

JinWoLu]. Subbands with large energy can be adapted by long filters, whereas subbands with insignificant energy can be either discarded or adjusted using short filters. The price is the additional complexity (minimized by using lattice structure or polyphase). Two-channel adaptive banks are designed in [JinWoLu] and compared with spline and Daubechies wavelets.

11.6 Wavelet Integrals for Differential Equations

The possible application of wavelets to differential equations is important. In principle, this application is available. *In practice, it is not yet a real success.* The overall approach is familiar, but the competition with other methods is severe. We do not necessarily predict that wavelets will win.

The unknown function $u(t)$ (or $u(x)$ or $u(x, y)$) is to be approximated by a finite combination $b_1\phi_1(t) + \cdots + b_N\phi_N(t)$. These "trial functions" are chosen in advance. They are polynomials or sinusoids in the spectral method. The $\phi_k(t)$ are piecewise polynomials in the finite element method. Presumably they are scaling functions and wavelets in the "wavelet method."

A key point is the option to use different scales $\Delta t = 2^{-j}$. The grid becomes a *multigrid*. The number of scale levels can vary with the position t or x, to produce an *adaptive mesh*. Flexibility comes from multiresolution. But the wavelet mesh does not achieve the total flexibility of the finite element method.

Up to this point in the book, all signals were assumed known. Now the coefficients $b_1, \ldots, b_N$ are unknowns because $u(t)$ is unknown. There are two outstanding rules for producing N equations for $b_1, \ldots, b_N$:

1. *Collocation*: Apply the differential equation at N points $t_1, \ldots, t_N$.

2. *Galerkin method*: Choose N test functions $g_1(t), \ldots, g_N(t)$. Replace an equation like $u''(t) = f(t)$ by N "weighted residual equations:"

$$\int \left(b_1\phi_1''(t) + \cdots + b_N\phi_N''(t)\right) g_j(t)\, dt = \int f(t)g_j(t)\, dt, \quad 1 \le j \le N. \tag{11.9}$$

When the test functions g_j are delta functions $\delta(t - t_j)$, this is collocation. When the g_j are the same as the ϕ_j, this is the Rayleigh-Ritz method. When the g_j are powers of t, this is the moment method. The trial functions and test functions should satisfy all *essential* boundary conditions in the problem—these are Dirichlet conditions like $u(0) = 0$. They need not satisfy *natural* boundary conditions like $u'(1) = 0$.

Collocation seems doubtful for wavelets, which are not smooth. The Galerkin method may also look doubtful, because of the integrals in (11.9). But these integrals can be computed by a quadrature formula, in which the coefficients are found once and for all. Then an integral of $f(t)\phi(t)$ does not require point values of $\phi(t)$, only of the smoother function $f(t)$:

$$\int_a^b f(t)\phi(t)\, dt \approx \sum_{k=1}^{r} a_k f(t_k). \tag{11.10}$$

This section discusses several quadrature rules. We have written $\phi(t)$ but similar rules are needed also for the wavelet $w(t)$. We will choose equally spaced evaluation points, so that overlapping integrals can share the same t_k (and $f(t)$ may be known only by those samples anyway).

Note that integrals are often computed by parts. Derivatives are moved away from rough functions and onto smooth functions. When the trial and test functions are the same ($\phi_j = g_j$), integration by parts changes $\int \phi''(t)g(t)\,dt$ into $-\int \phi'(t)g'(t)\,dt$. This needs only one derivative, not two.

We expect exact integrals for functions chosen in advance, and quadrature rules for integrals involving inputs $f(t)$. We discuss quadrature rules and then exact integrals.

Quadrature Rules

The usual test for accuracy is appropriate for smooth functions $f(t)$. The idea is to try $f(t) =$ *polynomial*. The rule (11.10) has accuracy d if it gives the correct integral for all polynomials of degree less than d. The first d terms in the Taylor series of $f(t)$ are integrated correctly, so the error is from the dth derivative:

$$\int f(t)\,\phi(t)\,dt = \sum a_k f(t_k) + O(\|f^{(d)}\|).$$

Changing from t to $2^j t$, this error will be of order $(\Delta t)^d$ when the scale is $\Delta t = 2^{-j}$:

$$\int f(t)2^{j/2}\phi\left(2^j t\right)dt = \sum a_k 2^{-j/2} f\left(2^{-j}t_k\right) + O\left((\Delta t)^d\right). \tag{11.11}$$

We study quadrature formulas at unit scale. Then (11.11) gives the error at other scales.

Example 11.1. One-point quadrature. The rule has only one term:

$$\int f(t)w(t)\,dt \approx 0 \quad \text{and} \quad \int f(t)\phi(t)\,dt \approx f(t_1). \tag{11.12}$$

The first rule is strange but its accuracy is $d = p$, because the wavelet has p vanishing moments. The exact integral is zero up to $f(t) = t^{p-1}$.

The second rule is correct for $f(t) \equiv 1$ provided $\int \phi(t)\,dt = 1$. Thus $d = 1$ at least. The accuracy increases to $d = 2$ when the evaluation point is $t_1 = \int t\phi(t)\,dt$. (The rule is then exact for $f(t) = t$.) This is a low-level example of the Gauss idea, to double the accuracy by choosing intelligent t_k as well as a_k. But higher-order Gauss has unequally spaced t_k and is not ideal for these applications. We follow instead the excellent paper [Sweldens-Piessens], which also gives the coefficients a_k for high order quadrature rules.

How can you find this number $\int t\phi(t)\,dt$? By definition, it is the first moment M_1 of $\phi(t)$. It equals the first moment $m_1 = \sum k\boldsymbol{h}(k)$ of the filter coefficients! There is a simple recursion that gives the moments $M_n = \int t^n\phi(t)\,dt$ exactly from the moments $m_n = \sum k^n\boldsymbol{h}(k)$. As always, the answer is in those coefficients $\boldsymbol{h}(k)$, when we use the dilation equation for $\phi(t)$:

$$\begin{aligned}
2^n \int t^n\phi(t)\,dt &= \sum 2\boldsymbol{h}(k)\int (2t)^n\phi(2t-k)\,dt \\
&= \sum \boldsymbol{h}(k)\int t^n\phi(t-k)\,dt \\
&= \sum \boldsymbol{h}(k)\int (t+k)^n\phi(t)\,dt \\
&= \sum_{j=0}^{n}\binom{n}{j}\sum \boldsymbol{h}(k)k^j\int t^{n-j}\phi(t)\,dt.
\end{aligned}$$

This says that $2^n M_n = \sum_{j=0}^{n} \binom{n}{j} m_j M_{n-j}$. Bring M_n to the left side:

$$\textit{Moment formula} \quad M_n = \frac{1}{2^n - 1} \sum_{j=1}^{n} \binom{n}{j} m_j M_{n-j}. \tag{11.13}$$

In particular $M_1 = m_1$. This is the best evaluation point for wavelet integrals — but it is not generally an integer. The D_4 scaling function has $M_1 = m_1 = (1 + \sqrt{3})/4$.

Three comments. First, orthogonal scaling functions have $M_2 = M_1^2$ (Problem 10). The one-point quadrature $\int f(t)\phi(t)\,dt \cong f(M_1)$ is *third-order* accurate. It is correct for $f(t) = 1, t, t^2$. Second, the Daubechies polynomials are nearly "interpolating" at the points $t = M_1 + k$:

$$\phi(M_1) \approx 1.0002 \quad \text{and} \quad \phi(M_1 + 1) \approx -0.0004 \quad \text{and} \quad \phi(M_1 + 2) \approx 0.0002.$$

See [K] for graphs and discussion. Third, by using $3p$ instead of $2p$ coefficients we can construct "*coiflets*" that have *zero moments* $M_1, M_2, \ldots, M_{p-1}$. Then the one-point rule $\int f(t)\phi(t)dt \approx f(0)$ is pth order accurate.

Exact Wavelet Integrals

We have already computed the inner products $\boldsymbol{a}(k) = \int \phi(t)\phi(t+k)\,dt$ between translates of the scaling function. We will review the key idea, which gives $\boldsymbol{a}(k)$ in terms of the filter coefficients $\boldsymbol{h}(k)$. We never want to do an exact integral any other way. Generally we can't! The applications to differential equations require many more integrals, for example

$$\int w(t)w(t+k)\,dt \qquad \text{and} \qquad \int \phi'(t)\phi'(t+k)\,dt \qquad \text{and} \qquad \int t^m \phi(t)\phi(t+k)\,dt.$$

The last integrand has three factors. Freely available software now allows *four factors* [Kunoth]. These integrals may be needed on an interval I (usually of length 1 or 2^{-n}) as well as on the whole line. Then the integral has a one-zero box function $\chi_I(t)$ as an extra factor.

In all cases the key idea is to use the two-scale equation for each factor. There are two-scale equations for $\phi(t)$ and $\phi'(t)$ and $w(t)$ and the box function $\chi_I(t)$. Recall what happens when two-scale equations are substituted into typical integrals:

$$\boldsymbol{a}(k) = \int \phi(t)\phi(t+k)\,dt = \int \Big(\sum 2\boldsymbol{h}(n)\phi(2t-n)\Big)\Big(\sum 2\boldsymbol{h}(n)\phi(2t+2k-n)\Big)\,dt$$

$$\boldsymbol{a}_w(k) = \int \phi(t)w(t+k)\,dt = \int \Big(\sum 2\boldsymbol{h}(n)\phi(2t-n)\Big)\Big(\sum 2\boldsymbol{h}_1(n)\phi(2t+2k-n)\Big)\,dt$$

The next step is to replace $2t$ by t on the right side. Then dt becomes $\frac{1}{2}dt$. The results are very different in our two examples:

$\boldsymbol{a}(k)$ becomes a combination of $\boldsymbol{a}(k+\ell)$: ***eigenvalue problem*** $\boldsymbol{a} = \boldsymbol{T}\boldsymbol{a}$

$\boldsymbol{a}_w(k)$ also becomes a combination of $\boldsymbol{a}(k+\ell)$: ***explicit equation*** $\boldsymbol{a}_w = \boldsymbol{T}_w\boldsymbol{a}$.

The equation $\boldsymbol{a} = \boldsymbol{T}\boldsymbol{a}$ is homogeneous. Its solution must be suitably normalized. The equation $\boldsymbol{a}_w = \boldsymbol{T}_w\boldsymbol{a}$ gives $\boldsymbol{a}_w$ in terms of the known $\boldsymbol{a}$. The matrices are doubled-shifted Toeplitz:

$$\boldsymbol{T} = (\downarrow 2)2\boldsymbol{H}\boldsymbol{H}^T \quad \text{and} \quad \boldsymbol{T}_w = (\downarrow 2)2\boldsymbol{H}_1\boldsymbol{H}^T. \tag{11.14}$$

This pattern is quite typical. Section 7.2 carried out the steps leading to $\boldsymbol{T}$. We now carry out the same steps for other important integrals (with more factors). Then we deal with integrals involving $w(t)$, which do *not* yield eigenvalue problems. For integrals involving $w(2t)$, the highpass matrix with diagonals $\boldsymbol{h}_1(0), 0, \boldsymbol{h}_1(1), 0, \boldsymbol{h}_1(2), 0, \ldots$ replaces $\boldsymbol{H}_1$.

Integrals of Products of $\phi_i(t)$ and $\phi_i'(t)$

We will describe the exact computation of a much wider class of integrals $\boldsymbol{b}(k)$. All functions to be integrated must be *refinable*. They satisfy two-scale dilation equations (also called refinement equations). The coefficients for $\phi_0(t), \phi_1(t), \phi_2(t)$ are $2\boldsymbol{h}_0(k), 2\boldsymbol{h}_1(k), 2\boldsymbol{h}_2(k)$. The dilation equation is substituted for each $\phi_i(t)$ and for any derivatives. (Each derivative of $2t$ produces an extra 2.) Changing the variable of integration brings $2t$ back to t and yields a two-scale equation for the integral:

$$\boldsymbol{b}(\boldsymbol{n}) = 2^{\alpha} \sum \boldsymbol{g}(\boldsymbol{n})\, \boldsymbol{b}(2\boldsymbol{k} - \boldsymbol{n}). \tag{11.15}$$

That is an eigenvalue problem $\boldsymbol{b} = 2^{\alpha}(\downarrow 2)\boldsymbol{G}\,\boldsymbol{b}$. The eigenvector is $\boldsymbol{b}$, and the operator $(\downarrow 2)$ changes $\boldsymbol{k}$ to $2\boldsymbol{k}$. Our first problem is to find the coefficients in $\boldsymbol{g}$ from the coefficients in $\boldsymbol{h}_0, \boldsymbol{h}_1$, and $\boldsymbol{h}_2$. *The two-scale equation for the integral* $\boldsymbol{b}$ *follows from the two-scale equations for* $\phi_0(t)$, $\phi_1(t)$, $\phi_2(t)$.

Our presentation follows [Kunoth], who has created a valuable and freely available code. She works in d dimensions with $\boldsymbol{x} = (x_1, \ldots, x_d)$ and allows a product of $\ell + 1 = 4$ functions. In principle any product of functions and their derivatives and dilations and translations is workable. We will begin in one dimension with $\ell = 2$ — thus a product of three functions inside the integral. The letters $\boldsymbol{b}$ and $\boldsymbol{g}$ are different from hers, since her conventions with 2's are not the same.

The integrals to be computed are sometimes called ***connection coefficients***:

$$\boldsymbol{b}(\boldsymbol{k}) = \boldsymbol{b}(k_1, k_2) = \int_{-\infty}^{\infty} \phi_0(t)\, D^{\mu_1}\phi_1(t - k_1)\, D^{\mu_2}\phi_2(t - k_2)\, dt.$$

The function $\phi_0(t)$ might be the standard box with dilation coefficients $\frac{1}{2}, \frac{1}{2}$. The functions $\phi_1(t)$ and $\phi_2(t)$ might be scaling functions, when the integral arises from Galerkin's method. If the differential equation is nonlinear, or has variable coefficients, that gives more factors ($\ell > 2$) in the integral. We are not now integrating wavelets! It is the scaling function that has a homogeneous dilation equation and leads to an eigenvalue problem. Integrals involving $w(t)$ are computed afterward. The filters $\boldsymbol{h}_0, \boldsymbol{h}_1, \boldsymbol{h}_2$ are all *lowpass*.

Our standard example is $\boldsymbol{a}(n) = \int \phi(t)\phi(t + k)\, dt = \int \phi(t - k)\phi(t)\, dt$. (From now on we have $t - k$ instead of $t + k$; no problem.) It has $\ell = 1$ with $\mu_1 = 0$; no derivatives. Or it has $\ell = 2$ with $\phi_0(t) \equiv 1$ and $(\mu_1, \mu_2) = (0, 0)$. In this familiar example, the matrix is $\boldsymbol{G} = \boldsymbol{H}\boldsymbol{H}^T$ and the exponent is $\alpha = 1$. The double-shift matrix is $\boldsymbol{T} = (\downarrow 2)2\boldsymbol{H}\boldsymbol{H}^T$ as usual. This example will be a useful check, when the coefficients $\boldsymbol{g}(n)$ come from the autocorrelation $\boldsymbol{g} = \boldsymbol{h} * \boldsymbol{h}^T$:

$$\boldsymbol{g}(n) = \sum \boldsymbol{h}(k)\boldsymbol{h}(k - n). \tag{11.16}$$

We jump directly to the key formula with three factors ϕ_0, ϕ_1, ϕ_2:

$$\boldsymbol{g}(n_1, n_2) = \sum \boldsymbol{h}_0(k)\boldsymbol{h}_1(k - n_1)\boldsymbol{h}_2(k - n_2). \tag{11.17}$$

To establish that formula, and to allow derivatives of the functions $\phi_i(t)$, substitute the dilation equations into the integral $b(k_1, k_2)$. The derivatives bring 2^{μ_1} and 2^{μ_2}:

$$\int_{-\infty}^{\infty} \Big[\sum 2h_0(k)\phi_0(2t-k)\Big]\Big[2^{\mu_1}\sum 2h_1(m)\phi_1(2t-2k_1-m)\Big]\Big[2^{\mu_2}\sum 2h_2(p)\phi_2(2t-2k_2-p)\Big]\,dt.$$

Change $2t-k$ to T and $2\,dt$ to dT. Set $\alpha = 2+\mu_1+\mu_2$. Our integral is

$$2^{\alpha}\sum_{k,m,p} h_0(k)h_1(m)h_2(p)\int_{-\infty}^{\infty}\phi_0(T)\phi_1\left(T-(2k_1+m-k)\right)\phi_2\left(T-(2k_2+p-k)\right)dT. \tag{11.18}$$

That inner integral is $b(2k_1+m-k, 2k_2+p-k)$. Set $n_1 = k-m$ and $n_2 = k-p$:

$$b(k_1,k_2) = 2^{\alpha}\sum_{n_1}\sum_{n_2}\sum_{k} h_0(k)h_1(k-n_1)h_2(k-n_2)b(2k_1-n_1, 2k_2-n_2). \tag{11.19}$$

The coefficients $g(n_1, n_2)$ anticipated in (11.17) have appeared (take the sum on k). The eigenvalue equation for b is

$$b(k_1,k_2) = 2^{\alpha}\textstyle\sum_{n_1}\sum_{n_2} g(n_1,n_2)b(2k_1-n_1, 2k_2-n_2). \tag{11.20}$$

In general $\alpha = \ell+\mu_1+\cdots+\mu_\ell$. The matrix $(\downarrow 2)G$ must have the eigenvalue $2^{-\alpha}$. In case of multiple eigenvectors, there is a correct choice to be made for b. It is given by moment conditions (11.22) discovered in [DaMi]:

Theorem 11.1 *Suppose the filters H_0, H_1, H_2 have at least $1, 1+\mu_1, 1+\mu_2$ zeros at π. There is a unique finite length eigenvector b satisfying* (11.21) *and* (11.22)*:*

$$\sum g(2k-n)b(n) = 2^{-\alpha}b(k) \quad \textit{which is} \quad (\downarrow 2)Gb = 2^{-\alpha}b \tag{11.21}$$

$$\sum k^{\nu_1}k_2^{\nu_2}\,b(k_1,k_2) = \mu_1!\,\mu_2!\,\delta(\nu_1-\mu_1)\,\delta(\nu_2-\mu_2) \textit{ for } \nu_1\le\mu_1,\ \nu_2\le\mu_2. \tag{11.22}$$

Example 11.2. Compute $b(k) = \int \phi(t)\phi'(t-k)\,dt$ with $\ell = 1$ and $\mu_1 = 1$ (first derivative).

The eigenvalue problem will be $2Tb = b$. The familiar matrix $T = (\downarrow 2)2HH^T$ has an extra 2 because of the derivative ϕ'. We have $\ell = 1$ and $\mu_1 = 1$ (there is no μ_2). Then b is the eigenvector of T with $\lambda = \frac{1}{2}$. The hat function example has eigenvector $(-0.5, 0, 0.5)$ when normalized by (11.22):

$$Tb = \tfrac{1}{8}\begin{bmatrix} \cdots & 6 & 4 & 1 & & & \\ & 1 & 4 & 6 & 4 & 1 & \\ & & & 1 & 4 & 6 & \cdots \end{bmatrix}\begin{bmatrix} -0.5 \\ 0 \\ 0.5 \end{bmatrix} = \tfrac{1}{2}\begin{bmatrix} -0.5 \\ 0 \\ 0.5 \end{bmatrix} = \tfrac{1}{2}b.$$

The integral of the hat function times its derivative (which is $+1$ then -1) is $b(0) = 0$. The inner products with shifts are $-.5$ and $.5$ in agreement with b.

Example 11.3. Compute $b(k) = \int \phi(t)\phi''(t-k)\,dt = -\int \phi'(t)\phi'(t-k)\,dt$.

We can take $\ell = 1$ and choose $\mu_1 = 2$, or we can take $\ell = 2$ and choose $\phi_0(t) \equiv 1$ and $(\mu_1, \mu_2) = (1, 1)$. In either case b is an eigenvector of T with eigenvalue $\lambda = \frac{1}{4}$. For the hat function matrix above, that eigenvector is $b = (-1, 2, -1)$. It is pleasant to go on to third derivatives (Problem 7). The eigenvalue for $\int \phi(t)\phi'''(t-k)\,dt = -\int \phi'(t)\phi''(t-k)\,dt$ is

$\lambda = \frac{1}{8}$. The matrix T has this eigenvalue. But we must use the 5×5 central submatrix instead of 3×3.

Example 11.4. Integrate the Daubechies D_4 function $\phi(t)$ over each interval $[0, 1]$, $[1, 2]$, $[2, 3]$.

Solution. We integrate $\phi(t)$ times translates of a box function, which has dilation coefficients $\frac{1}{2}$ and $\frac{1}{2}$. The Daubechies coefficients a, b, c, d add to 1 and are convolved with $(\frac{1}{2}, \frac{1}{2})$ to find $\frac{1}{2}(a, a+b, b+c, c+d, d)$. After double shift, the eigenvector $\boldsymbol{b}$ for $\lambda = 1$ gives the integrals of $\phi(t)$ over the three intervals:

$$\begin{bmatrix} 0 & d & c+d \\ c+d & b+c & a+b \\ a+b & a & 0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} \quad \text{yields} \quad \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = \tfrac{1}{12} \begin{bmatrix} 5+3\sqrt{3} \\ 2 \\ 5-3\sqrt{3} \end{bmatrix}.$$

The theory extends directly from a single variable t to d variables $\boldsymbol{x} = (x_1, \ldots, x_d)$. Each dilation equation involves integer vectors $\boldsymbol{k} = (k_1, \ldots, k_d)$:

$$\phi(\boldsymbol{x}) = 2^d \sum \boldsymbol{h}(\boldsymbol{k}) \phi(2\boldsymbol{x} - \boldsymbol{k}). \tag{11.23}$$

The convention is still $\sum \boldsymbol{h}(\boldsymbol{k}) = 1$. The ordinary derivatives $D^{\mu}\phi(t)$ become mixed partial derivatives $D^{\boldsymbol{\mu}}\phi(\boldsymbol{x}) = (\partial/\partial x_1)^{\mu_1} \cdots (\partial/\partial x_d)^{\mu_d}\phi(\boldsymbol{x})$ of order $|\boldsymbol{\mu}| = \mu_1 + \cdots + \mu_d$. Substitute the dilation equations into the desired integral, which is

$$\boldsymbol{b}(\boldsymbol{k}_1, \ldots, \boldsymbol{k}_\ell) = \int \cdots \int \phi_0(\boldsymbol{x})\, D^{\boldsymbol{\mu}_1}\phi_1(\boldsymbol{x} - \boldsymbol{k}_1) \cdots D^{\boldsymbol{\mu}_\ell}\phi_\ell(\boldsymbol{x} - \boldsymbol{k}_\ell)\, d\boldsymbol{x}. \tag{11.24}$$

Each integer vector $\boldsymbol{k}_1, \ldots, \boldsymbol{k}_\ell$ has d components. When $2\boldsymbol{x}$ in the dilation equation is replaced by $\boldsymbol{x}$, the multiplying factor 2^α has $\alpha = \ell d + |\boldsymbol{\mu}_1| + \cdots + |\boldsymbol{\mu}_\ell|$. The two-scale equation for the integrals is

$$\boldsymbol{b}(\boldsymbol{k}_1, \ldots, \boldsymbol{k}_\ell) = 2^\alpha \sum \boldsymbol{g}(\boldsymbol{n}_1, \ldots, \boldsymbol{n}_\ell)\, \boldsymbol{b}\,(2\boldsymbol{k}_1 - \boldsymbol{n}_1, \ldots, 2\boldsymbol{k}_\ell - \boldsymbol{n}_\ell)\,. \tag{11.25}$$

The coefficients $\boldsymbol{g}$ follow the model (11.16). Now $\boldsymbol{k}$ is $(k_1, \ldots, k_d)$ and each vector $\boldsymbol{n}_1, \ldots, \boldsymbol{n}_\ell$ has d components:

$$\boldsymbol{g}(\boldsymbol{n}_1, \ldots, \boldsymbol{n}_\ell) = \sum_{\boldsymbol{k}} \boldsymbol{h}_0(\boldsymbol{k})\boldsymbol{h}_1(\boldsymbol{k} - \boldsymbol{n}_1) \cdots \boldsymbol{h}_\ell\,(\boldsymbol{k} - \boldsymbol{n}_\ell)\,. \tag{11.26}$$

Integrals with Wavelets

Integrals involving wavelets come directly from the integrals $\boldsymbol{b}(\boldsymbol{k})$ involving scaling functions. The user must provide the highpass coefficients for any wavelets $w_0(t)$, $w_1(t), \ldots, w_\ell(t)$ that are in the integrals (staying for now at unit scale $j = 0$). Make those highpass replacements in $\boldsymbol{g}(\boldsymbol{n})$ to get $\boldsymbol{g}_w(\boldsymbol{n})$, leaving the lowpass coefficients whenever $\phi_i(t)$ is left in the integral. Then the wavelet integrals are $\boldsymbol{b}_w = 2^\alpha(\downarrow 2)\boldsymbol{g}_w * \boldsymbol{b}$.

To emphasize: Wavelets bring no new eigenvalue problem, just a trivial matrix multiplication (trivial if you manage all the indices). Start with a basic example to see the pattern. When $w(t)$ replaces $\phi(t)$, the matrix $\boldsymbol{H}_1$ replaces $\boldsymbol{H}$:

Theorem 11.2 *The wavelet integrals* $a_w(k) = \int \phi(t)w(t+k)\,dt$ *and* $a_{ww}(k) = \int w(t)w(t+k)\,dt$ *are computed from* $\boldsymbol{a}$ *by the equations*

$$\boldsymbol{a}_w = \boldsymbol{T}_w\boldsymbol{a} = (\downarrow 2)\,2\boldsymbol{H}_1\boldsymbol{H}^T\boldsymbol{a} \quad \textit{and} \quad \boldsymbol{a}_{ww} = \boldsymbol{T}_{ww}\boldsymbol{a} = (\downarrow 2)2\boldsymbol{H}_1\boldsymbol{H}_1^T\boldsymbol{a}. \tag{11.27}$$

Proof: The wavelet equation in vector form is $\boldsymbol{W}(t) = (\downarrow 2)2\boldsymbol{H}_1\boldsymbol{\Phi}(2t)$, where

$$\boldsymbol{\Phi}(t) = \begin{bmatrix} \cdot \\ \phi(t-1) \\ \phi(t) \\ \phi(t+1) \\ \cdot \end{bmatrix} \quad \text{and} \quad \boldsymbol{W}(t) = \begin{bmatrix} \cdot \\ w(t-1) \\ w(t) \\ w(t+1) \\ \cdot \end{bmatrix}. \tag{11.28}$$

This vector form leads directly to the inner product vectors:

$$\boldsymbol{a}_w = \int_{-\infty}^{\infty} \phi(t)\boldsymbol{W}(t)\,dt \quad \text{and} \quad \boldsymbol{a}_{ww} = \int_{-\infty}^{\infty} w(t)\boldsymbol{W}(t)\,dt.$$

To compute $\boldsymbol{a}_w$ substitute the dilation and wavelet equations:

$$\boldsymbol{a}_w = \int_{-\infty}^{\infty} \Big(\sum 2\boldsymbol{h}(k)\boldsymbol{\phi}(2t-k)\Big)\,(\downarrow 2)2\boldsymbol{H}_1\boldsymbol{\Phi}(2t)dt. \tag{11.29}$$

For the kth term in the sum, change variables to $u = 2t - k$ (so that $du = 2dt$). That kth term becomes

$$2\boldsymbol{h}(k)(\downarrow 2)\boldsymbol{H}_1 \int_{-\infty}^{\infty} \phi(u)\boldsymbol{\Phi}(u+k)\,du = 2\boldsymbol{h}(k)(\downarrow 2)\boldsymbol{H}_1\boldsymbol{S}^{-k}\boldsymbol{a}. \tag{11.30}$$

The k-step shift gave $\boldsymbol{\Phi}(u+k) = \boldsymbol{S}^{-k}\boldsymbol{\Phi}(u)$, and then $\int \phi(t)\boldsymbol{\Phi}(t)\,dt$ is exactly $\boldsymbol{a}$. Now sum on k to complete the formula:

$$\boldsymbol{a}_w = \sum_k 2\boldsymbol{h}(k)(\downarrow 2)\boldsymbol{H}_1\boldsymbol{S}^{-k}\boldsymbol{a} = (\downarrow 2)2\boldsymbol{H}_1\boldsymbol{H}^T\boldsymbol{a}. \tag{11.31}$$

Notice that $\sum \boldsymbol{h}(k)\boldsymbol{S}^{-k}$ is the *upper triangular Toeplitz matrix* $\boldsymbol{H}^T$. The same steps for $\boldsymbol{a}_{ww}$, wavelet times wavelet, produce the double-shifted Toeplitz matrix $\boldsymbol{T}_{ww} = (\downarrow 2)2\boldsymbol{H}_1\boldsymbol{H}_1^T$.

Example 11.5. For $\boldsymbol{h}(k) = \left(\frac{1}{4}, \frac{2}{4}, \frac{1}{4}\right)$ the piecewise linear hat function on $[0, 2]$ is $\phi(t)$. Its inner products $\boldsymbol{a}(k)$ come from $\boldsymbol{T}$:

$$\boldsymbol{T} = \frac{1}{8}\begin{bmatrix} 4 & 1 & 0 \\ 4 & 6 & 4 \\ 0 & 1 & 4 \end{bmatrix} \text{ has eigenvector } \boldsymbol{a} = \frac{1}{6}\begin{bmatrix} 1 \\ 4 \\ 1 \end{bmatrix}.$$

For the wavelets, we choose the highpass coefficients $\boldsymbol{h}_1(k) = \frac{1}{8}(-1, -2, 6, -2, -1)$. Then $H(z)H_1(-z)$ is the 6th degree Daubechies halfband filter and the synthesis functions $\widehat{\phi}(t)$ are biorthogonal to the hats. From the wavelet equation, $w(t)$ is also piecewise linear and it is supported on $[0, 3]$. The inner products with wavelet use matrix entries from $H_1(z)H(z^{-1})$ and $H_1(z)H_1(z^{-1})$:

$$\boldsymbol{a}_w = \frac{1}{16}\begin{bmatrix} -2 & -1 & & & \\ -2 & 6 & -2 & -1 & \\ & -1 & -2 & 6 & -2 \\ & & & -1 & -2 \end{bmatrix}\begin{bmatrix} 1 & & \\ 2 & 1 & \\ 1 & 2 & 1 \\ & 1 & 2 \\ & & 1 \end{bmatrix}\begin{bmatrix} 1/6 \\ 4/6 \\ 1/6 \end{bmatrix} = \begin{bmatrix} -1/12 \\ 1/12 \\ 1/12 \\ -1/12 \end{bmatrix}$$

$$a_{ww} = \frac{1}{32}\begin{bmatrix} 4 & 1 & \\ -20 & -8 & 4 \\ -20 & 46 & -20 \\ 4 & -8 & -20 \\ & 1 & 4 \end{bmatrix}\begin{bmatrix} 1/6 \\ 4/6 \\ 1/6 \end{bmatrix} = \begin{bmatrix} 1/24 \\ -6/24 \\ 18/24 \\ -6/24 \\ 1/24 \end{bmatrix}.$$

Integrals with multiple scales are also possible. For $a_1(k) = \int \phi(t)\phi(2t-k)\,dt$ we get a different eigenvalue problem $a_1 = (\downarrow 2)2UH^T a_1$. The reason is that the dilation equation for $\phi(2t)$ involves $4t$. On the right side is $h(k)\phi(4t-k)$ which is $u(2k)\phi(4t-k)$, provided $u = (\uparrow 2)h$. The matrix U with diagonals $h(0), 0, h(1), 0, h(2), 0, \ldots$ replaces H when the integral involves $\phi(2t)$.

Problem Set 11.6

1. Integrate the hat function on [0, 2] times the box function on [0, 1], using their dilation coefficients $(\frac{1}{4}, \frac{2}{4}, \frac{1}{4})$ and $(\frac{1}{2}, \frac{1}{2})$.
2. For the quadratic spline that comes from $h = \frac{1}{8}(1, 3, 3, 1)$, find the integrals over [0, 1] and [1, 2] and [2, 3].

Compute the integrals in **3.–6.** by creating the coefficients $g(n)$ from h_0, h_1, h_2 and solving the eigenvalue problem $(\downarrow 2)Gb = 2^{-\alpha}b$. Normalize by (11.22) and check by direct integration.

3. $\displaystyle\int_{-\infty}^{\infty} (\text{hat function})^3\,dt$
4. $\displaystyle\int_{0}^{1} (\text{hat function})^2\,dt$
5. $\displaystyle\int_{-\infty}^{\infty} (D_4 \text{ function } \phi(t))^2\,dt$
6. $\displaystyle\int_{0}^{1} (D_4 \text{ function } \phi(t))^2\,dt$

7. Follow Example 3 to integrate $\phi(t)\,\phi'''(t-k)$ for $\phi(t) =$ hat function.
8. The bilinear hat function in two variables $x = (x_1, x_2)$ has what formula and what coefficients $h(n)$? Find its inner products with its translates.
9. The frequency response $\sum h(k)\,e^{-ik\omega}$ gives the moments $H(0) = m_0 = 1$ and $H'(0) = -im_1$ and $H''(0) = -m_2$. Prove that an orthonormal filter with $H(\pi) = H'(\pi) = 0$ has $m_2 = m_1^2$, by setting $\omega = 0$ in the second derivative of $|H(\omega)|^2 + |H(\omega+\pi)|^2 = 1$.
10. Put $m_2 = m_1^2$ in equation (11.13) to show that $M_2 = M_1^2$ for orthonormal scaling functions with accuracy $p > 1$. Then the one-point rule $\int f(t)\phi(t)dt \approx f(M_1)$ has second-order accuracy.