Examples and Exercises for *A Primer on Wavelets*

James S. Walker
Department of Mathematics
University of Wisconsin–Eau Claire
Eau Claire, WI 54702–4004

e-mail: walkerjs@uwec.edu

Note: Computer exercises, designed for FAWAV, are indicated by a superscript **c.** For example, problem 2.1.5 below is a computer exercise. A subscript **s** means that a solution is provided. For instance, there is a solution provided for problem 2.1.1(a). Solutions begin on page 47.

Chapter 2

Section 2.1

Example 2.1.1 For the signal $\mathbf{f} = (2, 2, 2, 4, 4, 4)$, find its first level Haar transform.

Solution. The average of the first pair of values is 2, the average of the second pair of values is 3, and the average of the third pair of values is 4. Multiplying these averages by $\sqrt{2}$, we obtain $\mathbf{a}^1 = (2\sqrt{2}, 3\sqrt{2}, 4\sqrt{2})$. To compute $\mathbf{d}^1$, we find that $d_1 = (f_1 - f_2)/\sqrt{2} = 0$, and $d_2 = (f_3 - f_4)/\sqrt{2} = -2/\sqrt{2} = -\sqrt{2}$, and $d_3 = (f_5 - f_6)/\sqrt{2} = 0$. Thus the first level Haar transform of $\mathbf{f}$ is $(2\sqrt{2}, 3\sqrt{2}, 4\sqrt{2} \,|\, 0, -\sqrt{2}, 0)$.

Example 2.1.2 For the signal $\mathbf{f} = (2, 2, 2, 4, 4, 8)$, compute an approximate signal $\widetilde{\mathbf{f}}$ by inverse transforming the *compressed* Haar transform $(\mathbf{a}^1 \,|\, 0, \ldots, 0)$ obtained by setting all the fluctuation values equal to zero. Find the largest error between each value of $\mathbf{f}$ and $\widetilde{\mathbf{f}}$.

Solution. We find that $\mathbf{a}^1 = (2\sqrt{2}, 3\sqrt{2}, 6\sqrt{2})$. Therefore, the inverse Haar transform produces $\widetilde{\mathbf{f}} = (2, 2, 3, 3, 6, 6)$. The largest error between each value of $\mathbf{f}$ and $\widetilde{\mathbf{f}}$ is 2.

Example 2.1.3 [Figure 2.1] To create Figure 2.1 you do the following. First, choose *New 1-dim* from FAWAV's menu, and then choose *Graph/Plot.* Plot the formula[1]

```
20 x^2 (1-x)^4 cos(12 pi x)
```

over the interval of type $[0, L]$ with $L = 1$. That produces the graph plotted in Figure 2.1(a) (after selecting *View/Display style* and choosing *Blank* for the Grid style and *Lines* for the Plot style). To produce Figure 2.1(b), select *Transforms/Wavelet* and choose *Haar* as the Wavelet type with 1 entered for the Levels value. After plotting the transform, change to *Lines* as the plot style to get the graph shown in the figure.

2.1.1 Compute the first trend and first fluctuation for the following signals:

(a)$_{\mathbf{s}}$ $\mathbf{f} = (2, 4, 6, 6, 4, 2)$

(b) $\mathbf{f} = (-1, 1, 2, -2, 4, -4, 2, 2)$

(c)$_{\mathbf{s}}$ $\mathbf{f} = (1, 2, 3, 3, 2, 2, 1, 1)$

(d) $\mathbf{f} = (2, 2, 4, 4, 6, 6, 8, 8)$

2.1.2 Given the following Haar transformed signals, find the original signals $\mathbf{f}$ that correspond to them.

(a)$_{\mathbf{s}}$ $(2\sqrt{2}, -\sqrt{2}, 3\sqrt{2}, -\sqrt{2} \,|\, 0, \sqrt{2}, 0, \sqrt{2})$

(b) $(4\sqrt{2}, 3\sqrt{2}, -\sqrt{2}, 2\sqrt{2} \,|\, \sqrt{2}, -\sqrt{2}, 0, 2\sqrt{2})$

(c)$_{\mathbf{s}}$ $(3\sqrt{2}, 2\sqrt{2}, 2\sqrt{2}, 0 \,|\, 2\sqrt{2}, -\sqrt{2}, \sqrt{2}, 0)$

(d) $(4\sqrt{2}, 5\sqrt{2}, 7\sqrt{2}, -4\sqrt{2} \,|\, \sqrt{2}, 2\sqrt{2}, -2\sqrt{2}, \sqrt{2})$

[1] This formula is saved in the archive `BookFormulas.zip` which can be downloaded from the *Examples&Exercises* link at the book's website. You extract the formulas from this archive and save them in a directory that you can then call up with the *Load* button under the text box in the graphing procedure.

2.1.3 For each of the signals $\mathbf{f}$ given below, compute an approximate signal $\widetilde{\mathbf{f}}$ by inverse transforming the *compressed* Haar transform $(\mathbf{a}^1 \,|\, 0, \ldots, 0)$ obtained by setting all the fluctuation values equal to zero. In each case, find the largest error between each value of $\mathbf{f}$ and $\widetilde{\mathbf{f}}$.

(a)$_\mathbf{s}$ $\mathbf{f} = (2, 2, 3, 3, 4, 5, 6, 6)$

(b) $\mathbf{f} = (1, 2, 3, 3, 2, 1)$

(c)$_\mathbf{s}$ $\mathbf{f} = (2, -2, -2, -2, -2, 0, 2, 2)$

(d) $\mathbf{f} = (4, 4, 4, -1, -1, 1, 2, 2, 4, 6)$

2.1.4 Consider again problem 2.1.3 above. When will there be a difference between a value of $\mathbf{f}$ and a value of the approximate signal $\widetilde{\mathbf{f}}$, and when will the two signals' values be the same?

2.1.5$^\mathbf{c}$ Plot 1-level Haar transforms of the following functions—sampled uniformly over $[0, 1)$ using 1024 points.

(a) $f(x) = x^2(1 - x)$

(b)$_\mathbf{s}$ $f(x) = x^4(1 - x)^6 \cos(64\pi x)$

(c) $(0.2 < x < 0.3) - 3(0.4 < x < 0.5) + 2(0.5 < x < 0.8)$

(d) $f(x) = \operatorname{sgn}(\sin 12\pi x)$

Section 2.2

Example 2.2.1 For the signal $\mathbf{f} = (2, 2, 4, 6, 8, 10)$, find the energies of its trend and fluctuation subsignals and show that their sum equals the energy of $\mathbf{f}$.

Solution. The trend is $\mathbf{a}^1 = (2\sqrt{2}, 5\sqrt{2}, 9\sqrt{2})$ and the fluctuation is $\mathbf{d}^1 = (0, -\sqrt{2}, -\sqrt{2})$. The trend energy $\mathcal{E}_{\mathbf{a}^1}$ is $8 + 50 + 162 = 220$, and the fluctuation energy is $\mathcal{E}_{\mathbf{d}^1} = 0 + 2 + 2 = 4$. Their sum is 224 and the energy of $\mathbf{f}$ is $4 + 4 + 16 + 36 + 64 + 100 = 224$ so they are equal.

Example 2.2.2 Compute the percentage of compaction of the energy of $\mathbf{f} = (2, 2, 4, 6, 8, 10)$ by the 1-level Haar transform, in terms of $\mathcal{E}_{\mathbf{a}^1}/\mathcal{E}_\mathbf{f}$.

Solution. In Example 2.2.1, we found that $\mathcal{E}_{\mathbf{a}^1} = 220$ and that $\mathcal{E}_\mathbf{f} = 224$. Therefore, $\mathcal{E}_{\mathbf{a}^1}/\mathcal{E}_\mathbf{f} = 220/224 = 0.982\ldots$

Example 2.2.3 [Figure 2.2] To graph Figure 2.2(a), you plot (after selecting *Edit/Points used* and selecting 4096 as the number of points):

```
50x^2(1-x)^6cos(12pi x)(0<x<1)+80(1-x)^2(2-x)^8sin(20pi x)(1<x<2)
```

using *Lines* as the Plot style. Figure 2.2(b) is plotted by performing a Haar wavelet transform with 2 for the number of Levels (after changing the Grid style to *Blank* and the Y-interval values to $-1.5, 1.5$). Figure 2.2(c) is generated by right-clicking on the graph for the function [Figure 2.2(a)] and selecting *Energy graph*; the resulting graph is then clipped out by right-clicking and selecting *Clip* and entering 2 for the graph to be clipped. Similarly, Figure 2.2(d) is created by performing the same steps for the Haar transform graph.

Example 2.2.4 For the signal $\mathbf{f} = (2, 2, 4, 6, 8, 8, 12, 10)$, find its 1-level, 2-level, and 3-level Haar transforms.

Solution. The pairwise, successive averages for $\mathbf{f}$ are 2, 5, 8, 11. Hence $\mathbf{a}^1 = (2\sqrt{2}, 5\sqrt{2}, 8\sqrt{2}, 11\sqrt{2})$. In a similar way, we find that $\mathbf{d}^1 = (0, -\sqrt{2}, 0, \sqrt{2})$. Thus, the 1-level Haar transform of $\mathbf{f}$ is

$$(\mathbf{a}^1 \,|\, \mathbf{d}^1) = (2\sqrt{2}, 5\sqrt{2}, 8\sqrt{2}, 11\sqrt{2} \,|\, 0, -\sqrt{2}, 0, \sqrt{2}).$$

By applying the 1-level Haar transform to $\mathbf{a}^1$, we obtain

$$\mathbf{a}^1 \stackrel{\mathbf{H}_1}{\longmapsto} (\mathbf{a}^2 \,|\, \mathbf{d}^2) = (7, 19 \mid -3, -3).$$

Hence the 2-level Haar transform of $\mathbf{f}$ is

$$\mathbf{f} \stackrel{\mathbf{H}_2}{\longmapsto} (7, 19 \mid -3, -3 \,|\, 0, -\sqrt{2}, 0, \sqrt{2}).$$

Applying the 1-level Haar transform to $\mathbf{a}^2$ we obtain $(7, 19) \stackrel{\mathbf{H}_1}{\longmapsto} (13\sqrt{2} \mid -6\sqrt{2})$. Thus, the 3-level transform is

$$\mathbf{f} \stackrel{\mathbf{H}_3}{\longmapsto} (13\sqrt{2} \mid -6\sqrt{2} \mid -3, -3 \,|\, 0, -\sqrt{2}, 0, \sqrt{2}).$$

2.2.1$_\mathbf{s}$ Find the energies of the trend and fluctuation subsignals for the signals given in problems 2.1.1 (a)–(d), and show that their sums are equal to the energies of the signals $\mathbf{f}$.

2.2.2 Compute the percentage of compaction of the energy of $\mathbf{f}$ by the 1-level Haar transform, in terms of $\mathcal{E}_{\mathbf{a}^1}/\mathcal{E}_\mathbf{f}$, for each of the signals $\mathbf{f}$ in problem 2.1.1 (a)–(d).

2.2.3$_\mathbf{s}$ Another way to measure compaction is by the percentage of Haar transform values that are less than some small, preassigned number ϵ (> 0). Compute the percentage of 1-level Haar transform values which are less than $\epsilon = 0.05$ for each of the signals $\mathbf{f}$ in problem 2.1.1 (a)–(d).

2.2.4 For problem 2.2.3, change the value of ϵ to $\epsilon = 0.05\,\mathcal{E}_\mathbf{f}$ (so that ϵ depends on $\mathcal{E}_\mathbf{f}$) and recompute the percentage of compaction of the energy of $\mathbf{f}$ by the 1-level Haar transform, in terms of $\mathcal{E}_{\mathbf{a}^1}/\mathcal{E}_\mathbf{f}$, for each of the signals $\mathbf{f}$ in problem 2.1.1 (a)–(d).

2.2.5$^\mathbf{c}_\mathbf{s}$ Find the energies of the trend and fluctuation subsignals for the signals given in problem 2.1.5 (a)–(d), and show that their sums are equal to the energies of the signals $\mathbf{f}$. [Hints: In FAWAV, use *Analysis/Statistics* to compute energies for discrete signals. To compute $\mathcal{E}_{\mathbf{a}^1}$ you can modify the 1-level transform by plotting the function `g1(x)(x<1/2)` and then calculating the energy of the resulting signal. Similarly, the function `g1(x)(x >= 1/2)` will plot the first fluctuation.]

2.2.6$^\mathbf{c}$ Using $\epsilon = 0.05\,\mathcal{E}_\mathbf{f}$ compute the percentage of compaction of the energy of $\mathbf{f}$ by the 1-level Haar transform, in terms of $\mathcal{E}_{\mathbf{a}^1}/\mathcal{E}_\mathbf{f}$, for each of the signals $\mathbf{f}$ in problem 2.1.5 (a)–(d). [*Hint:* You can also plot a graph indicating where x-values are smaller in magnitude than some number c by using the formula `(abs(g1(x))<c)`]

2.2.7 Find the 1-level, 2-level, and 3-level Haar transforms of the following signals.

(a) $(8, 32, 48, 48, 64, 64, 40, -8)$

(b)$_\mathbf{s}$ $(-16, -16, -16, 32, 48, 48, 96, 96)$

(c) $(16, 16, 32, 48, 64, 72, 72, 72, 72, -16, -64, -32, -32, 40, 64, 16)$

(d) $(8, 8, 0, -8, -16, 24, 24, 24, 24, 16, 8, 0, 8, 8, 8, -16)$

Section 2.3

Example 2.3.1 For $\mathbf{f} = (3, 2, -1, 4)$ and $\mathbf{g} = (2, -2, 2, -2)$, find the scalar product $\mathbf{f} \cdot \mathbf{g}$.

Solution. $\mathbf{f} \cdot \mathbf{g} = 3(2) + 2(-2) - 1(2) + 4(-2) = -8$.

Example 2.3.2 Compute the inverse 1-level Haar transform of

$$(\mathbf{a}^1 \,|\, \mathbf{d}^1) = (0, 1, 0, \ldots, 0 \,|\, 0, 0, \ldots, 0),$$

and of

$$(\mathbf{a}^1 \,|\, \mathbf{d}^1) = (0, 0, \ldots, 0 \,|\, 0, 1, 0, \ldots, 0).$$

Solution. For the inverse transform of $(0, 1, 0, \ldots, 0 \,|\, 0, 0, \ldots, 0)$ We find that $f_3 = (a_2 + d_2)/\sqrt{2} = 1/\sqrt{2}$ and $f_4 = (a_2 - d_2)/\sqrt{2} = 1/\sqrt{2}$ and all other values of $\mathbf{f}$ are 0. Thus the inverse transform is $(0, 0, \sqrt{2}/2, \sqrt{2}/2, 0, 0, \ldots, 0)$.

For the inverse transform of $(0, 0, \ldots, 0 \,|\, 0, 1, 0, \ldots, 0)$ we find that $f_3 = (a_2 + d_2)/\sqrt{2} = 1/\sqrt{2}$ and $f_4 = (a_2 - d_2)/\sqrt{2} = -1/\sqrt{2}$ and all other values of $\mathbf{f}$ are 0. Thus the inverse transform is $(0, 0, \sqrt{2}/2, -\sqrt{2}/2, 0, 0, \ldots, 0)$.

2.3.1 Find the scalar product $\mathbf{f} \cdot \mathbf{g}$ when $\mathbf{f}$ and $\mathbf{g}$ are the following:

(a)$_\mathbf{s}$ $\mathbf{f} = (2, 1, 3, 4)$, $\mathbf{g} = (-1, 2, -2, 1)$

(b) $\mathbf{f} = (3, 2, -1, 2)$, $\mathbf{g} = (1, 3, 1, -1)$

(c)$_\mathbf{s}$ $\mathbf{f} = (1, 1, -1, -1, 2, 2, -2, 2)$, $\mathbf{g} = (0, 1, 0, 1, 2, 2, 1, 1)$

(d) $\mathbf{f} = (1, 1, 3)$, $\mathbf{g} = (1, 0, 2)$

2.3.2 Prove Property 1.

2.3.3 When will a second fluctuation value d_k^2 equal zero?

2.3.4 Is the following statement true or false? *Whenever the fluctuation value $d_1^2 = 0$, then the signal $\mathbf{f}$ is constant over the support of $\mathbf{W}_1^2$.*

2.3.5$_\mathbf{s}$ Compute the inverse 1-level Haar transform of

$$(\mathbf{a}^1 \,|\, \mathbf{d}^1) = (1, 0, \ldots, 0 \,|\, 0, 0, \ldots, 0),$$

and of

$$(\mathbf{a}^1 \,|\, \mathbf{d}^1) = (0, 0, \ldots, 0 \,|\, 1, 0, \ldots, 0).$$

Section 2.4

Example 2.4.1 For $\mathbf{f} = (3, 2, -1, 4)$ and $\mathbf{g} = (2, -2, 2, -2)$, find the sum $\mathbf{f} + \mathbf{g}$, the difference $\mathbf{f} - \mathbf{g}$, and the combination $2\mathbf{f} - 3\mathbf{g}$.

Solution. We calculate that

$$\begin{aligned}
\mathbf{f} + \mathbf{g} &= (3 + 2, 2 - 2, -1 + 2, 4 - 2) = (5, 0, 1, 2) \\
\mathbf{f} - \mathbf{g} &= (3 - 2, 2 + 2, -1 - 2, 4 + 2) = (1, 4, -3, 6) \\
2\mathbf{f} - 3\mathbf{g} &= (6 - 6, 4 + 6, -2 - 6, 8 + 6) = (0, 10, -8, 14).
\end{aligned}$$

Example 2.4.2 For $\mathbf{f} = (2, 2, 4, 6, -2, -2, -2, 0)$ find the first averaged signal $\mathbf{A}^1$ and the first detail signal $\mathbf{D}^1$.

Solution. The trend $\mathbf{a}^1$ and fluctuation $\mathbf{d}^1$ satisfy

$$\begin{aligned}
\mathbf{a}^1 &= (2\sqrt{2}, 5\sqrt{2}, -2\sqrt{2}, -\sqrt{2}) \\
\mathbf{d}^1 &= (0, -\sqrt{2}, 0, -\sqrt{2}).
\end{aligned}$$

Hence

$$\mathbf{A}^1 = (2, 2, 5, 5, -2, -2, -1, -1)$$
$$\mathbf{D}^1 = (0, 0, -1, 1, 0, 0, -1, 1).$$

and, as a check, we observe that $\mathbf{A}^1 + \mathbf{D}^1 = \mathbf{f}$.

Example 2.4.3 For $\mathbf{f} = (2, 2, 4, 6, -2, -2, -2, 0)$ find the second averaged signal $\mathbf{A}^2$ and the second detail signal $\mathbf{D}^2$.

Solution. We found in the previous example that $\mathbf{a}^1 = (2\sqrt{2}, 5\sqrt{2}, -2\sqrt{2}, -\sqrt{2})$. The second trend is then $\mathbf{a}^2 = (7, -3)$, and the second fluctuation is $\mathbf{d}^2 = (-3, -1)$. Therefore,

$$\mathbf{A}^2 = (\frac{7}{2}, \frac{7}{2}, \frac{7}{2}, \frac{7}{2}, \frac{-3}{2}, \frac{-3}{2}, \frac{-3}{2}, \frac{-3}{2})$$
$$\mathbf{D}^2 = (\frac{-3}{2}, \frac{-3}{2}, \frac{3}{2}, \frac{3}{2}, \frac{-1}{2}, \frac{-1}{2}, \frac{1}{2}, \frac{1}{2}).$$

and, as a check, we observe that $\mathbf{A}^2 + \mathbf{D}^2 = \mathbf{A}^1$.

Example 2.4.4 [Figure 2.3] The graphs in Figure 2.3 were created by graphing the function in given in Example 2.1.3, and then choosing *Series/Wavelet* and selecting *Haar* for the wavelet, using 10 for the number of Levels, and choosing *Ascending terms* for the Series type. You then enter successively 1, 2, 4,..., 512 for the number of terms to use (in the text box to the right of *Ascending terms:*). For 1, we get the plot of $\mathbf{A}^{10}$, for 2 we get the plot of $\mathbf{A}^9$, for 4 we get the plot of $\mathbf{A}^8$, ..., for 512 we get the plot of $\mathbf{A}^1$.

2.4.1 For the following signals, $\mathbf{f}$ and $\mathbf{g}$, compute their sum $\mathbf{f} + \mathbf{g}$, their difference $\mathbf{f} - \mathbf{g}$, and the constant multiples $3\mathbf{f}$ and $-2\mathbf{g}$.

(a)$_\mathbf{s}$ $\mathbf{f} = (2, 3, 2, 4)$, $\mathbf{g} = (1, 2, -1, 3)$

(b) $\mathbf{f} = (4, 2, 1, 1, 0, 0)$, $\mathbf{g} = (-1, 2, 1, 2, -1, 3)$

(c)$_\mathbf{s}$ $\mathbf{f} = (-1, -1, 1, 1, 1, 1)$, $\mathbf{g} = (1, 2, 0, 1, -1, 1)$

(d) $\mathbf{f} = (1, 1, 2, 1)$, $\mathbf{g} = (1, 2, -3, -4)$

2.4.2$_\mathbf{s}$ For each of the signals in problem 2.1.1, compute the first averaged signal $\mathbf{A}^1$ and the first detail signal $\mathbf{D}^1$.

2.4.3 Find expressions for the first averaged signal $\mathbf{A}^1$ and the first detail signal $\mathbf{D}^1$ in terms of the values of $\mathbf{f} = (f_1, f_2, \ldots, f_N)$.

2.4.4 For each of the following signals, compute the first averaged signal $\mathbf{A}^1$ and the first detail signal $\mathbf{D}^1$.

(a)$_\mathbf{s}$ $(2, 1, 3, -1, 2, 4, 3, 4)$

(b) $(1, 2, 3, 4, 5, 6, 7, 8)$

(c)$_\mathbf{s}$ $(1, 2, -1, -1, 4, 3, 2, 2)$

(d) $(9, 4, 1, 0, 0, 1, 4, 9)$

2.4.5 Express the 2^{nd} averaged signal $\mathbf{A}^2$ and the 2^{nd} detail signal $\mathbf{D}^2$ in terms of the values of $\mathbf{f} = (f_1, f_2, \ldots, f_N)$.

2.4.6$_\mathbf{s}$ For each of the signals in problem 2.4.4, compute the 2^{nd} averaged signal $\mathbf{A}^2$ and the 2^{nd} detail signal $\mathbf{D}^2$.

Section 2.5

Example 2.5.1 Produce graphs like the ones shown in Figure 2.4 for 1024 samples of the following signal:

$$f(x) = 3(1 < x < 4) - 2(5 < x < 8) + 4(10 < x < 18)$$

over the interval $[0, 20]$. What threshold should be used to retain 99.99% of the energy? What compression ratio does this produce, and what is the maximum error between the original signal and the compressed signal?

Solution. Choosing *New 1-dim* from the FAWAV menu and then choosing *Graph/Plot*, we plot the formula above over the interval of type $[0, L]$ setting $L = 20$. This produces the graph shown in Figure 1(a) (after we choose *View/Display style* and selecting a *Blank* Grid style and a *Dots* Plot style). Selecting *Transform/Wavelet* and choosing *Haar* with 10 levels, we obtain the Haar transform shown in Figure 1(b) (after choosing *View/Display style* and changing the Y-range to $-20, 20$). By *right*-clicking on the Haar transform graph and choosing *Sort magnitudes* and then right-clicking again and choosing *Energy map* and entering 2 for the graph number (leaving the *percentage* box checked), we produce the graph shown in Figure 1(c) (after right-clicking and choosing *Clip* and clipping graph 3). Finally, by choosing *Analysis/Trace* from the menu of the window containing the Haar transform along with its sorted magnitudes and energy map, we find by tracing on the three graphs that 100% of the energy is used when the Index is 45 (i.e., 46 values used because in FAWAV the Index counter for arrays of data is initialized at 0 rather than 1), this represents $1024/46 \approx 22$ to 1 compression.

We also find by tracing that 99.99% of the energy is used when a threshold of 0.5 is used. By graphing the function $g1(x)(\mathrm{abs}(g1(x)) > 0.5)$ we produce a fourth graph that is a thresholded transform. Then by choosing *Transform/Wavelet*, selecting Haar *with the Inverse box checked*, and entering 4 for the graph number, we plot an approximation of the given step function. We then right-click on the graph of this approximation, choose *Copy*, return to the window with the original function displayed and right-click on its graph followed by selecting *Paste*. This pastes the approximate function data into the window, for comparison with the original graph. Selecting *Analysis/Norm difference* and using the default values (*Sup-norm*, graphs 1 and 2, and *absolute*) we find that the maximum error (in magnitude) between the original signal and the approximation is 0.1094. (Note: since the original signal consists of integer values, by graphing the function `gri(g2(x)+1/2)` we round the approximate signal values to their nearest integers. Thus producing a signal that is equal to the original signal at all values.)

Example 2.5.2 [Figure 2.4] To produce Figure 2.4(a) you graph

```
(2<=x<4)-(6<=x<9)+2[14<=x<18]-2(10<= x< 11)
```

over the interval $[0, 20]$. Figure 2.4(b) is obtained using a 10-level Haar transform. To get Figure 2.4(c) you right-click on the graph of the Haar transform and select *Sort magnitudes* then right-click again and select *Energy graph*. You then plot the energy graph for graph 2 (the sorted magnitude graph). Figure 2.4(d) was obtained by selecting *Analysis/Trace* and using the tracing tool to determine that the 52 highest magnitude transform values account for 100% of the energy of the signal. Then, by returning to the window containing the original signal, selecting *Series/Wavelet* and performing a Haar series[2] with Series type *Highest mag. coefficients*, and entering 52 for the number of coefficients, we obtain the graph shown in Figure 2.4(d).

Example 2.5.3 [Figure 2.5] Figure 2.5 is produced in the same way as Figure 2.4 (see preceding example), except that 4096 points are used and the following function

```
40x^2(1-x)^4cos(12pi x)[0<x<1)+{40(x-1)^2(2-x)^8cos(48pi x)
   +80(x-1)^12[2-x]^2sin(80pi x)}[1<x<2]
```

is graphed over $[0, 2]$ to produce the initial signal.

2.5.1$^{c}_{s}$ Produce graphs like the ones shown in Figure 2.4 for 1024 samples of the following signal:

$$f(x) = 2[1 < x < 2] - 3[3 < x < 5] + [6 < x < 7] + 4[8 < x < 9]$$

[2] See the **Remark** on the next page.

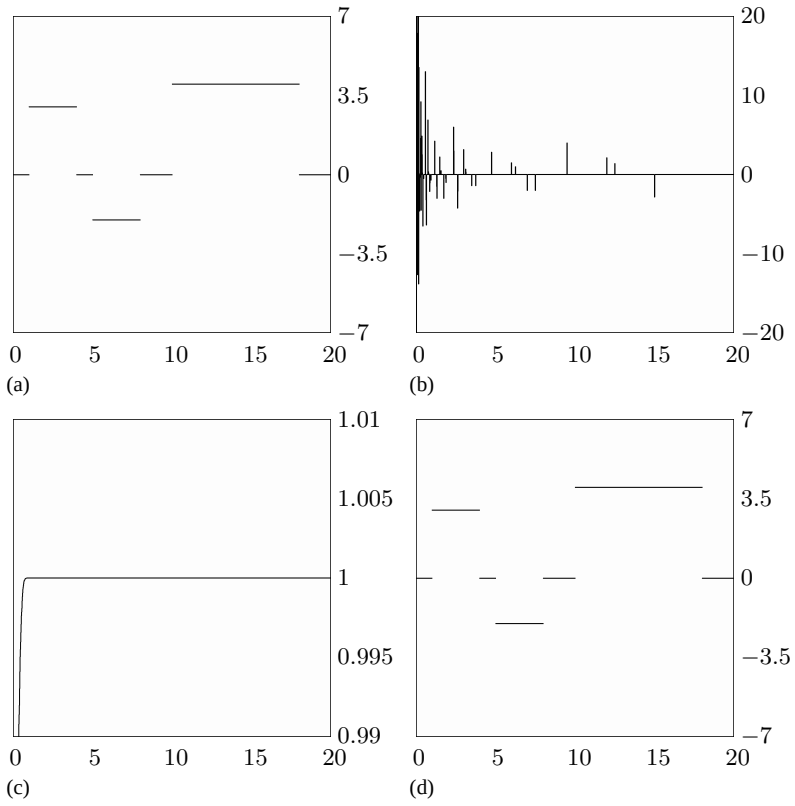


Figure 1
(a) Signal, (b) 10-level Haar transform, (c) energy map of Haar transform, (d) 22:1 compression of Signal, 100% of energy.

over the interval $[0, 10]$. What threshold should be used to retain 99.99% of the energy? What compression ratio does this produce, and what is the maximum error between the original signal and the compressed signal? [Note: maximum error is computed using the *sup-norm* method described below in Example 2.5.4.]

2.5.2$^{\mathbf{c}}$ Repeat problem 2.5.1 for each of the following:

(a)$_{\mathbf{s}}$ $f(x) = x(10 - x)$

(b) $f(x) = 2[2 < x < 4] - 2[5 < x < 7] + 2[8 < x < 9]$

(c) $f(x) = 2[2 < x < 4] - x[5 < x < 7] + 2[8 < x < 9]$

Remark. The next three exercises deal with Haar wavelet series. A *wavelet series* is a convenient term for the following three step process:

1. Compute a transform, either wavelet or wavelet packet.
2. Modify the transform values from Step 1.
3. Compute the inverse transform of the modified values from Step 2.

There are a number of ways to carry out the modification of the wavelet transform values in Step 2. The most common method is to threshold the values. When a threshold method is used, we say that the three step process produces a *thresholded series.* To produce a thresholded series with FAWAV, you begin by selecting *Series* from the menu. You then select either *Wavelet* or *Wavelet packet* to specify which of the two types of transform will be used. A dialog box

will then appear on the right side of the window, and you select *Threshold* as the method for the series. Another type of series is *Highest mag.*, which modifies the transform values by using only a specified number of the highest magnitude transform values (setting all others equal to zero).

Example 2.5.4 For the function in Example 2.5.1, compute the Haar series using the 30 highest magnitude values. What is the sup-norm difference between the function and its wavelet series? What is the *Sup-norm difference* over the interval $[2, 3]$?

Solution. We plot the function as described in the solution of Example 2.5.1. Selecting *Transform/Wavelet* and then *Haar* for the transform and *Highet mag. coefficients* with 30 as the number, we obtain an approximation of the original signal. The *sup-norm difference* between the two signal is 1.98. To find the sup-norm difference over the interval $[2, 3]$ we choose *View/Display style* to change the X-interval to $[2, 3]$. The graphs are then displayed over the interval $[2, 3]$ and computing a sup-norm difference between them yields 0.0117.

2.5.3$^{\mathbf{c}}_{\mathbf{s}}$ For the functions in problem 2.5.2 (a) and (b), compute the Haar series using the 50 highest magnitude values. Which function is best approximated by such a Haar series? Why?

2.5.4$^{\mathbf{c}}$ For the Haar series graphed in problem 2.5.3, what is the maximum error between the original function and the Haar series over the interval $[2.5, 3.5]$, and over the interval $[5.5, 6.5]$? Why are the two errors very similar over $[2.5, 3.5]$, but not over $[5.5, 6.5]$?

2.5.5 Suppose a Haar transform $(\mathbf{a}^k \,|\, \mathbf{d}^k \,|\, \ldots \,|\, \mathbf{d}^1)$ is thresholded, producing a signal $(\widetilde{\mathbf{a}}^k \,|\, \widetilde{\mathbf{d}}^k \,|\, \ldots \,|\, \widetilde{\mathbf{d}}^1)$, and then an inverse Haar transform is performed on the thresholded signal, producing a Haar series. Find an expression for this Haar series in terms of scaling signals and wavelets.

Section 2.6

Example 2.6.1 [Figure 2.6] To get Figure 2.6(a), you graph

```
.1ran(x)+(2<=x<4)-(6<=x<9)+2[14<=x<18]-2(10<= x< 11)
```

over $[0, 20]$. Figure 2.6(b) was obtained in the following way. First, a Haar transform with 10 levels was performed and then the Y-interval changed to $[-4, 4]$. Second, by using the *Trace* tool, or by right-clicking on the Haar transform and selecting *Display cursor coordinates* to use the mouse to scan over the graph box with a readout of x- and y-values, we determined that 0.2 was a good threshold for removing noise. (The horizontal graphs in Figure 2.6(b) were obtained by plotting `0.2` and `-0.2` with the *Auto-fit* option unchecked.) To get the thresholded transform shown in Figure 2.6(c), we plotted the function

```
g1(x)(abs(g1(x))>c) \c=0.2
```

Finally, to obtain the denoised signal in Figure 2.6(d) we performed an inverse Haar transform on the thresholded transform.

Example 2.6.2 [Figure 2.7] The graphs in Figure 2.7 were obtained in the same way as Figure 2.6, except that the initial noisy signal was graphed using the formula

```
.1ran(x)+40x^2(1-x)^4cos(12pi x)[0<x<1)
+{40(x-1)^2(2-x)^8cos(48pi x)+80(x-1)^12[2-x]^2sin(80pi x)}[1<x<2]
```

over the interval $[0, 2]$ using 4096 points, and 12 levels were used for the Haar transform.

2.6.1$^{\mathbf{c}}$ Graph the random noise signal, $f(x) = \mathrm{rang}(x)$, over the interval $[0, 1]$ using 8192 points. Then play the sound generated by this signal, using *Graph/Audio* in FAWAV, with a sampling rate of 8820, a bit rate of 16 bits, and volume level of 32000. What does this random noise sound like?

2.6.2$^{\mathbf{c}}_{\mathbf{s}}$ Perform a 1-level Haar transform of the signal in problem 2.6.1. What does this Haar transform look like, and what does it sound like (when it is played using the same parameter values as in 2.6.1)?

2.6.3$^{\mathbf{c}}$ Using the Threshold Method, denoise each of the following signals (in each case, use 1024 points and $[0, 10]$ as interval):

(a)$_{\mathbf{s}}$ $f(x) = 40[2 < x < 4] - 60[5 < x < 7] + 80[8 < x < 9] + 10\,\mathrm{rang(x)}$

(b) $f(x) = 4\sin(2\pi x) + 10\,\mathrm{rang(x)}$

(c) $f(x) = 40[2 < x < 4] + 8x[5 < x < 7] + 40[8 < x < 9] + 10\,\mathrm{rang(x)}$

(d)$_{\mathbf{s}}$ $f(x) = [40\cos(2\pi x)](2 < x < 6) + 10\,\mathrm{rang(x)}$

Which denoisings would you deem to be the most successful, and why?

2.6.4 Explain the cause of the very ragged, jumpy appearance of the denoised signal in Figure 2.7(d).

Chapter 3

Section 3.1

Example 3.1.1 [Figure 3.1] The graphs of $\mathbf{V}_1^5$, $\mathbf{V}_8^5$ and $\mathbf{V}_{16}^5$ were produced in the following way. To produce $\mathbf{V}_1^5$ we applied the 5^{th} level inverse Daub4 transform to the signal $(1, 0, 0, \ldots, 0)$. This signal was produced by plotting the formula `del(x)` over the interval $[0, 1024]$ with 1024 points. We then chose *Transform/Wavelet* and selected the *Daub 4* option with 5 levels. To produce $\mathbf{W}_1^5$ we applied the 5^{th} level inverse Daub4 transform to the signal $(0, \ldots, 0, 1, 0, \ldots, 0$ where the 1 is in the 33^{rd} position (that signal was produced by plotting the formula `del(x-32)` over the interval $[0, 1024]$ with 1024 points). The reason this works is explained in detail in the subsection **Daub 5/3 transform, multiple levels** in section 3.7. As explained in that subsection, we produce $\mathbf{V}_k^m$ by applying the m-level inverse Daub4 transform to the signal $(0, \ldots, 0, 1, 0, \ldots, 0)$ where the 1 is in the k^{th} position, $k = 1,\ldots, N/2^m$; and we produce $\mathbf{W}_k^m$ by applying the m-level inverse Daub4 transform to the signal $(0, \ldots, 0, 1, 0, \ldots, 0)$ where the 1 is in the $k + N/2^m$ position, $k = 1,\ldots, N/2^m$.

Example 3.1.2 [Figure 3.2] To produce Figure 3.2(a) we plotted the formula `20x^2(1-x)^4cos(12pi x)` over the interval $[0, 1]$ using 1024 points. Figure 3.2 was then created by choosing *Transform/Wavelet*, and selecting the *Daub 4* option with 2 levels. Figures 3.2(c) and (d) were plotted by selecting *View/Display style* from the menu for the graph of the original signal and changing the X and Y intervals to the ones shown.

Example 3.1.3 [Figure 3.3] Figure 3.3 was produced in the same way as Figure 2.3 (see Example 2.4.4), except that Daub 4 was used as the choice of wavelet.

3.1.1$_{\mathbf{s}}$ Explain why $\mathbf{V}_m^2$ has a support of 10 time-units and is a translate of $\mathbf{V}_1^2$ by $4(m-1)$ time-units. (Ignore wrap-around.)

3.1.2 The 3^{rd} level Daub4 scaling signals and wavelets have supports of how many time-units? Are they all shifts of $\mathbf{V}_1^3$ and $\mathbf{W}_1^3$?

3.1.3$_{\mathbf{s}}$ Does $0\alpha_1 + 1\alpha_2 + 2\alpha_3 + 3\alpha_4 = 0$ hold for the Daub4 scaling numbers?

3.1.4 Show that Property I holds for 2-level wavelets, i.e., *if a signal* $\mathbf{f}$ *is (approximately) linear over the support of a* 2*-level Daub4 wavelet* $\mathbf{W}_m^2$*, then the* 2*-level fluctuation value* $\mathbf{f} \cdot \mathbf{W}_m^2$ *is (approximately) zero.*

3.1.5$^{\mathbf{c}}$ Plot 1-level Daub4 transforms of the following functions—sampled uniformly over $[0, 1]$ using 1024 points. [*Note:* These are the same functions considered in problem 1.1.5.]

(a) $f(x) = x^2(1-x)$

(b)$_{\mathbf{s}}$ $f(x) = x^4(1-x)^6\cos 64\pi x$

(c) $(0.2 < x < 0.3) - 3(0.4 < x < 0.5) + 2(0.5 < x < 0.8)$

(d) $f(x) = \operatorname{sgn}(\sin 12\pi x)$

Remark. In order to use FAWAV to graph the averaged signals—$\mathbf{A}^1$, $\mathbf{A}^2$, $\mathbf{A}^3$, etc.—you proceed as follows. After plotting the signal $\mathbf{f}$, you then choose *Series/Wavelet* and select *Ascending terms* as the series choice. If you specify $N/2$ number of terms (where N is the number of points), then the first averaged signal $\mathbf{A}^1$ will be plotted. Or, if you select $N/4$ number of points, then $\mathbf{A}^2$ will be plotted. Or, by selecting $N/8$ number of points, then $\mathbf{A}^3$ will be plotted. (This method of plotting averaged signals is equivalent to taking a wavelet transform, then setting all values of the transform to 0 for indices above $N/2$, $N/4$, or $N/8$, and then inverse transforming.)

3.1.6$^{\mathbf{c}}$ Compute the Daub4 averaged signals $\mathbf{A}^1$, $\mathbf{A}^2$, $\mathbf{A}^3$, and $\mathbf{A}^4$ for the function

$$g(x) = 20x^4(1-x)^6 \cos 48\pi x$$

over the interval $[0, 1]$ using 1024 points.

3.1.7$^{\mathbf{c}}_{\mathbf{s}}$ What is the maximum error (over all points) between each of the averaged signals in problem 3.1.6 and the given signal?

3.1.8$^{\mathbf{c}}$ Repeat problems 3.1.6 and 3.1.7 for the signal

$$g(x) = 20x^2(1-x)^2 \cos 64\pi x + 30x^2(1-x)^4 \sin 30\pi x.$$

3.1.9 Show that $\mathbf{f} \cdot \mathbf{W}_m^1 = \mathrm{O}(h)$ when $\mathbf{W}_m^1$ is a 1-level Haar wavelet. [*Hint:* Use Formula (3.15).]

3.1.10$^{\mathbf{c}}_{\mathbf{s}}$ Repeat problem 1.5.2, but use a Daub4 transform instead of a Haar transform.

3.1.11$^{\mathbf{c}}$ Repeat problem 1.5.3, but use a Daub4 series instead of a Haar series. Which function is best approximated by a Daub4 series? Why?

Section 3.2

Example 3.2.1 [Figure 3.4] Figure 3.4 was created by first plotting the function

```
50x^2(1-x)^6cos(12pi x)(0<x<1)+80(1-x)^2(2-x)^8sin(20pi x)(1<x<2)
```

over the interval $[0, 2]$ using 4096 points. Then, the graph in (a) was created by selecting *Transform/Wavelet* and choosing a Haar wavelet with 2 levels. The corresponding cumulative energy profile in (c) was created by right-clicking on the Haar transform graph and selecting *Energy graph.* The graphs in (b) and (d) were created in a similar way, except that a Daub4 transform was used.

3.2.1 Verify Equations (3.17a)–(3.17c).

3.2.2$^{\mathbf{c}}_{\mathbf{s}}$ Compute the 3-level Daub4 transform of the signal $\mathbf{f}$ obtained from 1024 uniformly spaced samples of

$$g(x) = x^2(4-x)^4 \sin 12\pi x$$

over the interval $[0, 4]$. Compute the energy of $\mathbf{f}$ and of its transform and check that conservation of energy holds.

3.2.3$^{\mathbf{c}}$ Using $\epsilon = 0.0001\,\mathcal{E}_{\mathbf{f}}$, compute the percentage of 1-level Daub4 transform values which are less than ϵ for each of the signals in problem 1.1.5 (a)–(d). Compare your results with a 1-level Haar transform [see problem 2.2.6].

3.2.4$^{\mathbf{c}}_{\mathbf{s}}$ Repeat problem 3.2.3, but use a 2-level Daub4 transform instead. Compare your results with a 2-level Haar transform.

Section 3.3

Example 3.3.1 [Figure 3.5] The signal analyzed in Figure 3.5 was created by plotting the formula

```
50x^2(1-x)^6cos(12pi x)(0<x<1)+80(1-x)^2(2-x)^8sin(20pi x)(1<x<2)
```

over the interval $[0, 2]$ using 4096 points. To compute 1000 times its fluctuation $\mathbf{d}^1$, we plotted the 1-level Daub4 transform and then graphed

```
1000g1(x/2+1)
```

To compute 30 times the fluctuation of its fluctuation $\mathbf{d}^3$ we plotted its 3-level Daub4 transform and then graphed

```
30g1(x/8+1/4)
```

Similar computations were done for the Daub6 case.

Example 3.3.2 [Figure 3.6] Figure 3.6 was produced in the same way as Figure 2.3 (see Example 2.4.4), except that Daub 20 was used as the choice of wavelet.

Example 3.3.3 [Figure 3.7] Figure 3.7 was produced in the same way as Figure 3.1 (see Example 3.1.1), except that an inverse Coif 6 transform was used.

3.3.1$_{\mathbf{s}}$ Show that if $\mathbf{f}$ is obtained from samples of a 3-times continuously differentiable function g over the support of a 1-level Daub6 wavelet $\mathbf{W}_m^1$, then the fluctuation value $\mathbf{f} \cdot \mathbf{W}_m^1$ satisfies $\mathbf{f} \cdot \mathbf{W}_m^1 = \mathrm{O}(h^3)$.

3.3.2$^{\mathbf{c}}$ Compute the Daub6 averaged signals $\mathbf{A}^1$, $\mathbf{A}^2$, $\mathbf{A}^3$, and $\mathbf{A}^4$ for the function

$$g(x) = 20x^4(1-x)^6 \cos 48\pi x$$

over the interval $[0, 1]$ using 1024 points.

3.3.3$^{\mathbf{c}}_{\mathbf{s}}$ What is the maximum error (over all points) between each of the averaged signals in problem 3.3.2 and the original signal.

3.3.4$^{\mathbf{c}}$ Repeat problems 3.3.2 and 3.3.3 for the signal

$$g(x) = 20x^2(1-x)^2 \cos 64\pi x + 30x^2(1-x)^4 \sin 30\pi x.$$

3.3.5$^{\mathbf{c}}$ Repeat problem 2.5.2, but use a Daub6 wavelet series instead of a Haar series.

3.3.6$^{\mathbf{c}}$ Repeat problem 2.5.3, but use a Daub6 wavelet series instead of a Haar series. Which function is best approximated by a Daub6 series? Why?

3.3.7$^{\mathbf{c}}$ Repeat problem 2.5.2, but use a Daub8 wavelet series instead of a Haar series.

3.3.8$^{\mathbf{c}}$ Repeat problem 2.5.3, but use a Daub8 wavelet series instead of a Haar series. Which function is best approximated by a Daub8 series? Why?

3.3.9$^{\mathbf{c}}_{\mathbf{s}}$ For the following function, compute the minimum number of terms needed to capture 99.99% of the energy in a Daub$J$ series for each $J = 4, 6, 8$ and for each level $L = 1, 2, \ldots, 6$ (using 1024 points over the interval $[0, 1]$):

$$g(x) = x^2(1-x)^6 \cos 25\pi x.$$

Note: To determine the minimum number of terms you use the option *Energy fraction* for a wavelet series and enter the value 0.9999 to require 99.99% of the energy. A report is displayed that indicates the number of terms used.

3.3.10$^{\mathbf{c}}$ Repeat problem 3.3.9 for the Daub$J$ series, $J = 10, 12, 14$.

3.3.11$^{\mathbf{c}}$ Repeat problem 3.3.9 for each of the following functions (and use Haar series as well as Daub$J$ series):

(a) $g(x) = [0.1 < x < 0.2] - 3[0.3 < x < 0.5] + 9[0.7 < x < 0.9]$

(b) $g(x) = x[0.1 < x < 0.2] - x^2[0.3 < x < 0.5] + 9[0.7 < x < 0.9]$

(c) $g(x) = [0.1 < x < 0.2]\cos 40\pi x - [0.3 < x < 0.5]\sin 30\pi x + [0.7 < x < 0.9]\cos 40\pi x$

(d) $g(x) = [0.1 < x < 0.2]\sin 40\pi x + [0.3 < x < 0.5]2\sin 30\pi x + [0.7 < x < 0.9]\sin 60\pi x$

3.3.12$^{c}_{s}$ Repeat problem 3.3.9, but use Coif$I$ series for $I = 6, 18, 30$.

3.3.13^{c} Repeat problem 3.3.11, but use Coif$I$ series for $I = 6, 18, 30$.

3.3.14^{c} Compare 3-level Coif6 trend values with $2\sqrt{2}g(8x)$ over the interval $[0, 0.125]$, where $g(x)$ is defined by Equation (3.35) [use 2^{14} samples over the interval $[0, 1]$ as in the text]. Do the same for the Daub4 transform. *Note:* By "compare" we mean determine the maximum error.

3.3.15$^{c}_{s}$ For the following function $g(x)$ over the interval $[0, 1]$:

$$g(x) = 40x^4(1-x)^6\cos 24\pi x$$

compare $\sqrt{2}g(2x)$ over the interval $[0, 0.5]$ with the 1-level Coif$I$ trend values for $I = 6, 12, 18, 24, 30$. [Use 2^{12} samples, and by "compare" we mean find the maximum error.] Do the same for Daub$J$ trend values for $J = 6, 12, \ldots, 18$.

3.3.16$^{c}_{s}$ Repeat problem 3.3.15, but compare $2g(4x)$ over $[0, 0.25]$ with Coif$I$ and Daub$J$ 2-level trend values.

Section 3.4

Example 3.4.1 [Figure 3.8] Figure 3.8 is produced in the same way as Figure 2.5 (see Example 2.5.3), except that a Coif30 transform is used.

3.4.1^{c} Produce graphs like the ones shown in Figures 2.4 and 3.8 for 1024 samples of the following signals over the interval $[0, 10]$:

(a) $f(x) = x$

(b) $f(x) = 2[2 < x < 4] - 2[5 < x < 7] + 2[8 < x < 9]$

(c) $f(x) = 2[2 < x < 4] - x[5 < x < 7] + 2[8 < x < 9]$

(d)$_{s}$ $f(x) = 0.001x^4(10-x)^2$

What thresholds should be used to capture 99.99% of the energy using a 10-level Daub4 wavelet transform? What compression ratios do these produce, and what are the maximum errors between the original signals and the compressed signals?

3.4.2^{c} Repeat problem 3.4.1, but use a 10-level Daub18 transform.

3.4.3^{c} Repeat problem 3.4.1, but use a 10-level Coif12 transform.

3.4.4^{c} Repeat problem 3.4.1, but use a 10-level Coif30 transform.

3.4.5$^{c}_{s}$ For each of the functions in problem 3.4.1, find the level (from 1 to 10) which uses the least number of Daub4 transform values to capture 99.99% of each signal's energy. [*Hint:* Use the *energy percentage* method for forming wavelet series.]

3.4.6^{c} Repeat problem 3.4.5, but use a Daub12 transform.

3.4.7$^{c}_{s}$ Repeat problem 3.4.5, but use a Coif18 transform.

3.4.8^{c} Repeat problem 3.4.5, but use a Coif30 transform.

3.4.9^{c} Record your own voice saying the word "alfalfa" at 8000 Hz and 8 bpp. Using a Coif30 series, determine the level that uses the least number of transform values that capture 99.99% of the energy of the audio signal.

3.4.10^{c} Repeat problem 3.4.9, but with a recording of "alfalfa" at 16 bpp.

3.4.11^{c} Repeat problem 3.4.9, but with a recording of "alfalfa" at $22,050$ Hz and 8 bpp.

3.4.12^{c} Repeat problem 3.4.9, but with a recording of "alfalfa" at $22,050$ Hz and 16 bpp.

Section 3.5

Example 3.5.1 [Figure 3.10] To graph the signal in (a) you select *New 1-dim* from the menu, right-click on the graph area and select *Load/Sound file*, and select `greasy.wav` as the sound file to load. To plot the histogram in (b) you right-click on the graph and select *Histogram.* You then choose an 8-bit histogram with the both the choices *Include zero values* and *Include sign bit* checked. To obtain the graph in (c) you perform a 14-level Coif30 transform of the signal in (a). Finally, to get the graph in (d) you compute a histogram of the transform, but uncheck the choice *Include zero values* and make sure that the choice *Include zero values* is checked. (We call this a *dead-zone histogram.*)

Example 3.5.2 [Figure 3.11] To obtain the plot of the fourth trend of in (a), you plot a 4-level Coif30 transform of the `greasy.wav` signal, select *View/Display*, and plot over the new X-interval: `0,.743038548752834/2^4` To obtain the graph in (b) you plot a histogram with the *Include zero values* unchecked. To obtain the graph in (c), you change the X-interval of the Coif30 transform to `.743038548752834/2^4,.743038548752834`. To obtain the histogram in (d) you compute a dead-zone histogram of the graph from (c).

3.5.1^{c} Draw a plot of the *entropy function*, $f(x) = x\log_2(1/x)$ over the interval $[0,1]$. [*Hint:* In FAWAV, there is a built-in function, $\mathrm{entr}(x)$, which calculates $x\log_2(1/x)$.] Find the point where the maximum of this function lies, either by numerical estimation (using the *Analysis/Trace* procedure) or by calculus.

3.5.2$_{s}$ Show that the entropy of the uniform probability sequence, $p_k = 1/N$ for $k = 1, 2, \ldots, N$, is $\log_2 N$.

3.5.3 Find the entropy of the sequence $p_k = c\,2^{-k}$, $k = 1, 2, \ldots, 16$ (where $c = 1/\sum_{k=1}^{16} 2^{-k}$).

3.5.4$^{c}_{s}$ Make a recording of the word "alfalfa" at 8000 Hz and 16 bpp. Estimate—as was done in the text for *greasy*—the number of bpp needed for an optimal, entropy-based compression of this recording of "alfalfa" (using 16 bpp), then compare this with a maximum-level Coif30 transform at 16 bpp and with a 4-level Coif30 transform at 16 bpp for the trend and 12 bpp for the fluctuations.

3.5.5^{c} Repeat problem 3.5.4, but using $22\,050$ Hz and 16 bpp.

3.5.6$^{c}_{s}$ For problems 3.5.4 and 3.5.5, calculate the Sup-norm and relative 2-norm errors between the original and compressed signals. [*Note:* The compressed signals can be plotted using *Thresholded wavelet series*, when the settings are adjusted (press the button labeled *Edit settings*) to *Quantized thresholding* and either *Uniform threshold* or *Multiple thresholds* are selected.]

Section 3.6

Example 3.6.1 [Figure 3.12] To produce the graph in Figure 3.12(a), we plotted the function

```
.1ran(x)+40x^2(1-x)^4cos(12pi x)[0<x<1)+{40(x-1)^2(2-x)^8cos(48pi x)
    +80(x-1)^12[2-x]^2sin(80pi x)}[1<x<2]
```

over the interval $[0, 2]$ using 4096 points. The graph in (b) was obtained by a 12-level Coif30 transform of this signal, and then changing to the Y-interval $[-2, 2]$. By right-clicking on the graph and selecting *Display cursor coordinates* we were able to scan over the graph and obtain 0.2 as a threshold for eliminating noisy transform values. To obtain the plot in (c) we plotted the function

```
g1(x)(abs(g1(x))>0.2)
```

where `g1(x)` stands for the transform values. Then by applying an inverse Coif30 transform to the graph in (c) we obtained the graph shown in (d), the denoised signal.

Example 3.6.2 [Figure 3.13] To produce the graph in Figure 3.13(a), we plotted the function `0.5rang(0)` over $[0, 1]$ using 4096 points. The mean and standard deviation of this noise were found by selecting *Analysis/Statistics* from the menu. To graph the histogram in (b) we calculated an 8-bit histogram with both options (*Include zero values* and *Include sign bit*) checked. To obtain the graph in (c) we performed a 12-level Coif30 transform of the noise signal from (a). We then calculated a histogram [the same type as we did with (a)] to get the graph in (d).

Example 3.6.3 [Figure 3.14] To get the graph in (a) we loaded the sound file `noisy_wolf_whistle.wav`. The graph in (b) was obtained by performing a 15-level Coif18 transform of this sound file. To obtain the graph in (c) we then processed this transform by selecting *Graph/Plot* and plotting the formula contained in the file `fig_3_14_(c).uf1` (by clicking on the *Load* button under the formula text area). This is one of the files that is contained in the zip file `BookFigures.zip` that accompanies these exercises. After producing the processed transform in (c) we then performed an inverse transform on it to produce the denoised signal in (d).

3.6.1[c] Using the threshold method, denoise each of the following signals (each of which is defined over the interval $[0, 10]$ using 1024 points). Use a 10-level Coif30 transform.

(a) `20(2 < x < 4) - 30(5<x<7) + 40(8<x<9) + 5rang(0)`

(b)$_s$ `40cos(4pi x) +  5rang(0)`

(c) `20[2< x < 4] + 5x[5 < x < 7] + 20[8 < x < 9]+ 5rang(0)`

(d) `40cos(4pi x)(2 < x < 6) + 5rang(0)`

Which denoisings would you deem to be the most successful, and why?

3.6.2[c] Repeat problem 3.6.1, but use a 4-level Coif30 transform.

3.6.3[c] Repeat problem 3.6.1, but use a 10-level Coif18 transform and also a 4-level Coif18 transform.

Remark In Exercises 3.6.4 to 3.6.6, you should use the built-in wavelet denoising method obtained by selecting *Denoise/Wavelet* and checking the box labelled *Average.* Try 5-levels for the wavelet transform. This procedure automatically selects the threshold and performs an average of denoisings of shiftings of the signal (this further reduces noise).

3.6.4[c] Accompanying these exercises is a data file, `noisy word 1.wav` which is a noisy version of the audio file `alfalfa_2.wav`. Use wavelet-based denoising to denoise this signal. What percentage reduction of RMS do you obtain?

3.6.5[c] Repeat problem 3.6.4, but for the audio file `noisy word 2.wav.`

3.6.6[c] Repeat problem 3.6.4, but for the audio file `noisy word 3.wav.`

Section 3.7

Example 3.7.1 For the signal $\mathbf{f} = (8, 16, 8, -8, 0, 16)$, compute its 1-level Daub 5/3 transform.

Solution. Using the formulas for the analysis scaling vectors $\{\mathbf{V}_k\}$ and the equation $a_k = \mathbf{f} \cdot \mathbf{V}_k$ we obtain

$$\begin{aligned} a_1 &= (8,16,8,-8,0,16) \cdot (\frac{3}{4}, \frac{1}{2}, \frac{-1}{4}, 0, 0, 0) \ = 12 \\ a_2 &= (8,16,8,-8,0,16) \cdot (\frac{-1}{8}, \frac{1}{4}, \frac{3}{4}, \frac{1}{4}, \frac{-1}{8}, 0) \ = 7 \\ a_3 &= (8,16,8,-8,0,16) \cdot (0, 0, \frac{-1}{8}, \frac{1}{4}, \frac{5}{8}, \frac{1}{4}) \ = 1. \end{aligned}$$

Similarly, we compute d_k from $d_k = \mathbf{f} \cdot \mathbf{W}_k$ where $\{\mathbf{W}_k\}$ are the analysis wavelets:

$$\begin{aligned} d_1 &= (8,16,8,-8,0,16) \cdot (\frac{-1}{2}, 1, \frac{-1}{2}, 0, 0, 0) \ = 8 \\ d_2 &= (8,16,8,-8,0,16) \cdot (0, 0, \frac{-1}{2}, 1, \frac{-1}{2}, 0) \ = -12 \\ d_3 &= (8,16,8,-8,0,16) \cdot (0, 0, 0, 0, -1, 1) \ = 16. \end{aligned}$$

So the 1-level Daub 5/3 transform is $(\mathbf{a}^1 \,|\, \mathbf{d}^1) = (12, 7, 1 \,|\, 8, -12, 16)$.

Example 3.7.2 [Figure 3.15] As explained in the subsection, **Daub 5/3 transform, multiple levels,** we produce $\widetilde{\mathbf{V}}_k^m$ by applying the m-level inverse Daub 5/3 transform (select `DD 5/3 (2,2)` as the transform type) to the signal $(0, \ldots, 0, 1, 0, \ldots, 0)$ where the 1 is in the k^{th} position, $k = 1,\ldots, N/2^m$; and we produce $\widetilde{\mathbf{W}}_k^m$ by applying the m-level inverse Daub4 transform to the signal $(0, \ldots, 0, 1, 0, \ldots, 0)$ where the 1 is in the $k + N/2^m$ position, $k = 1,\ldots, N/2^m$.

3.7.1$^{\mathbf{c}}$ Compute the Daub 5/3 averaged signals $\mathbf{A}^1$, $\mathbf{A}^2$, $\mathbf{A}^3$, and $\mathbf{A}^4$ for the function

$$g(x) = 20x^4(1-x)^6 \cos 48\pi x$$

over the interval $[0, 1]$ using 1024 points.

3.7.2$^{\mathbf{c}}_{\mathbf{s}}$ What is the maximum error (over all points) between each of the averaged signals in problem 3.7.1 and the original signal.

3.7.3$^{\mathbf{c}}$ Repeat problems 3.7.1 and 3.7.2 for the signal

$$g(x) = 20x^2(1-x)^2 \cos 64\pi x + 30x^2(1-x)^4 \sin 30\pi x.$$

3.7.4$^{\mathbf{c}}$ Repeat problem 2.5.3, but use a Daub 5/3 wavelet series instead of a Haar series. Which function is best approximated by a Daub 5/3 series? Why?

3.7.5$^{\mathbf{c}}_{\mathbf{s}}$ For the following function, compute the minimum number of terms needed to capture 99.99% of the energy in a Daub 5/3 series for each level $L = 1, 2, \ldots, 6$ (using 1024 points over the interval $[0, 1]$):

$$g(x) = x^2(1-x)^6 \cos 25\pi x.$$

3.7.6$^{\mathbf{c}}_{\mathbf{s}}$ Compare 1-level Daub 5/3 trend values with $g(2x)$ over the interval $[0, 0.5]$, where $g(x)$ is defined by Equation (3.35). [Use 2^{14} samples over the interval $[0, 1]$ as in the text.] *Note:* By "compare" we mean determine the maximum error.

3.7.7$^{\mathbf{c}}$ Compare 2-level Daub 5/3 trend values with $g(4x)$ over the interval $[0, 0.25]$, where $g(x)$ is defined by Equation (3.35). Also compare 3-level Daub 5/3 trend values with $g(8x)$ over the interval $[0, 0.125]$. [Use 2^{14} samples over the interval $[0, 1]$ as in the text.]

Section 3.8

Example 3.8.1 [Figure 3.16] As explained in the subsection, **Daub 5/3 transform, multiple levels,** we produce $\widetilde{\mathbf{V}}_k^m$ by applying the m-level inverse Daub 9/7 transform (select `Daug 9/7` as the transform type) to the signal $(0,\ldots,0,1,0,\ldots,0)$ where the 1 is in the k^{th} position, $k = 1,\ldots, N/2^m$; and we produce $\widetilde{\mathbf{W}}_k^m$ by applying the m-level inverse Daub4 transform to the signal $(0,\ldots,0,1,0,\ldots,0)$ where the 1 is in the $k + N/2^m$ position, $k = 1,\ldots, N/2^m$.

3.8.1$^{\mathbf{c}}$ Compute the Daub 9/7 averaged signals $\mathbf{A}^1$, $\mathbf{A}^2$, $\mathbf{A}^3$, and $\mathbf{A}^4$ for the function

$$g(x) = 20x^4(1-x)^6\cos 48\pi x$$

over the interval $[0, 1]$ using 1024 points.

3.8.2$^{\mathbf{c}}_{\mathbf{s}}$ What is the maximum error (over all points) between each of the averaged signals in problem 3.8.1 and the original signal.

3.8.3$^{\mathbf{c}}$ Repeat problems 3.8.1 and 3.8.2 for the signal

$$g(x) = 20x^2(1-x)^2\cos 64\pi x + 30x^2(1-x)^4\sin 30\pi x.$$

3.8.4$^{\mathbf{c}}$ Repeat problem 2.5.3, but use a Daub 9/7 wavelet series instead of a Haar series. Which function is best approximated by a Daub 9/7 series? Why?

3.8.5$^{\mathbf{c}}_{\mathbf{s}}$ For the following function, compute the minimum number of terms needed to capture 99.99% of the energy in a Daub 9/7 series for each level $L = 1, 2, \ldots, 6$ (using 1024 points over the interval $[0, 1]$):

$$g(x) = x^2(1-x)^6\cos 25\pi x.$$

3.8.6$^{\mathbf{c}}_{\mathbf{s}}$ Compare 1-level Daub 9/7 trend values with $\sqrt{2}g(2x)$ over the interval $[0, 0.5]$, where $g(x)$ is defined by Equation (3.35). [Use 2^{14} samples over the interval $[0, 1]$ as in the text.] *Note:* By "compare" we mean determine the maximum error.

3.8.7$^{\mathbf{c}}$ Compare 2-level Daub 9/7 trend values with $2g(4x)$ over the interval $[0, 0.25]$, where $g(x)$ is defined by Equation (3.35). Also compare 3-level Daub 5/3 trend values with $g(8x)$ over the interval $[0, 0.125]$. [Use 2^{14} samples over the interval $[0, 1]$ as in the text.]

Chapter 4

Section 4.1

Example 4.1.1 For the array

$$\begin{pmatrix} 4 & 8 & 8 & 4 \\ 4 & 8 & 6 & 6 \\ 4 & 0 & 0 & 4 \\ 8 & 8 & 4 & 4 \end{pmatrix}$$

we compute its 1-level Haar transform in two steps. First, we find the 1-level Haar transform along each row:

$$\begin{pmatrix} 6\sqrt{2} & 6\sqrt{2} & -2\sqrt{2} & 2\sqrt{2} \\ 6\sqrt{2} & 6\sqrt{2} & -2\sqrt{2} & 0 \\ 2\sqrt{2} & 2\sqrt{2} & 2\sqrt{2} & -2\sqrt{2} \\ 8\sqrt{2} & 4\sqrt{2} & 0 & 0 \end{pmatrix}$$

Second, we find the 1-level Haar transform of each column of this new array (indexing from the bottom up):

$$\begin{pmatrix} 0 & 0 & 0 & -2 \\ 6 & 2 & -2 & 2 \\ 12 & 12 & -4 & 2 \\ 10 & 6 & 2 & -2 \end{pmatrix}$$

which is the 1-level Haar transform.

Example 4.1.2 [Figure 4.1] To graph Figure 4.1(a), you select *New 2 dim* from the menu and plot the function

```
(y-x-c<=0)(x+y-c<=0)(x-y-c<=0)
(x+y+c>=0)(abs(y)<=b)(abs(x)<=b)
 \c=12/5\b=8.5/5\Rem Use L = 4
```

over $[-L, L] \times [-L, L]$ using $L = 4$. You then right-click on the graph and select *Graph style* which allows you to replot the graph using the *Grey (+/-)* option. To get the 1-level Coif6 transform in Figure 4.1(b), you plot a Coif6 transform of the octagon graph in (a), and then change the *Graph style* of the transform to *Grey (+/-)* with the `LinLog` option selected and with a threshold of $.0000000001$. The graphs in (c) and (d) are obtained in a similar way, except that 2-level and 3-level Coif6 transforms are performed, respectively.

Example 4.1.3 [Figure 4.2] To graph Figure 4.2(a), you select *New 2 dim* from the *File* menu and then select *Points/128* from the *Edit* menu. You then graph the formula

```
del(x+64-2)del(y+32-4)
```

over $[-L, L] \times [-L, L]$ using $L = 64$. That produces an image with all values 0, except one pixel of value 1 (an element of the standard basis for 128 by 128 matrices, which when an inverse Haar transform is applied produces a Haar wavelet. In fact, you perform a 2-level *inverse* Haar transform, and right-click on the transform's graph in order to select *Graph style* which allows you to replot the transform using the *Grey (+/-)* option. That completes the construction of the image in (a). The images in Figures (b) to (d) were obtained in a similar way through modifying the formula above (use the formulas in the archive `BookFigures.zip`).

4.1.1 For each of the following arrays, compute its 1-level Haar transform.

(a)$_{\mathbf{s}}$ $\begin{pmatrix} 2 & 4 & 2 & 0 \\ -2 & 0 & 2 & 2 \\ 4 & 0 & 2 & 4 \\ 6 & 8 & 12 & 12 \end{pmatrix}$

(b) $\begin{pmatrix} 1 & 3 & 3 & 1 \\ 2 & 5 & 5 & 2 \\ 4 & 7 & 7 & 4 \\ 2 & 7 & 7 & 2 \end{pmatrix}$

(c) $\begin{pmatrix} 3 & 1 & 2 & 2 \\ 1 & 1 & 1 & 1 \\ 3 & 5 & 7 & 9 \\ 9 & 7 & 5 & 3 \end{pmatrix}$

(d) $\begin{pmatrix} 4 & 6 & 8 & 12 \\ 4 & 16 & 24 & 32 \\ 8 & 12 & 6 & 6 \\ 4 & 4 & 2 & 2 \end{pmatrix}$

4.1.2$_s$ For each of the arrays in problem 4.1.1, compute its 2-level Haar transform.

4.1.3 Compute $\mathbf{W}_1^1 \otimes \mathbf{W}_1^1$, where $\mathbf{W}_1^1$ is a Haar wavelet.

4.1.4$_s$ Compute $\mathbf{W}_1^1 \otimes \mathbf{V}_1^1$, where $\mathbf{W}_1^1$ and $\mathbf{V}_1^1$ are a Haar wavelet and Haar scaling signal, respectively.

4.1.5^c Compute $\mathbf{V}_3^2 \otimes \mathbf{V}_5^2$, $\mathbf{V}_3^2 \otimes \mathbf{W}_5^2$, $\mathbf{W}_3^2 \otimes \mathbf{V}_5^2$, and $\mathbf{W}_3^2 \otimes \mathbf{W}_5^2$, when $\mathbf{V}_j^2$ and $\mathbf{W}_k^2$ are Daub 9/7 scaling signals and Daub 9/7 wavelets, respectively.

4.1.6 Suppose a 1-level wavelet transform of an image $\mathbf{f}$ has $\mathbf{h}^1 = 0$, $\mathbf{a}^1 = 0$, and $\mathbf{v}^1 = 0$ (i.e., all the values in these sub-arrays are zero), and suppose that $d_{2,3}^1 = 1$ while all other values of $\mathbf{d}^1$ are zero. Show that the inverse transform yields $\mathbf{f} = \mathbf{W}_2^1 \otimes \mathbf{W}_3^1$. How does this result generalize?

Note. For subsequent examples and exercises, you may need to download images that are not installed by your initial installation of FAWAV. These additional image files can be downloaded from the *Images* link at the FAWAV website.

Section 4.2

Example 4.2.1 [Figure 4.3] To generate the 4:1 compression in the figure you do the following. First, right-click on an image square on the 2D-form and select *Load/Image* and load the PGM file `barb.pgm` You then select *File/Save options (images)* and enter a *target rate* of 2 bpp. Since the original image is 8 bpp that represents a 4:1 compression ratio. After saving a compressed image (right-click on the original image, choose *Save/Image*, and select `*.wc2` as file format), you then load this compressed file (*Load/Image* with `*.wc2` as format) and that produces the reconstruction shown in part (b) of Figure 4.3. The image in (c) was produced by plotting the function `g1 - g2` where `g1` stands for graph 1 (the original image) and `g2` stands for graph 2 (the reconstruction of the 4:1 compression). This third image is displayed with a *Graph style* of *Grey (+/-)* selected. Finally, the fourth graph in (d) was obtained by right-clicking on the graph in (c), and then plotting a 9-bit histogram.

Example 4.2.2 [Figure 4.4] To generate the 16:1 compression in the figure you load the image `dog_head.pgm` then select *File/Save options (images)* and enter a *target rate* of 0.5 bpp. (Since the original image is 8 bpp that represents a 16:1 compression ratio.) After saving a compressed image (right-click on the original image, choose *Save/Image*, and select `*.wc2` as file format), you then load this compressed file (*Load/Image* with `*.wc2` as format) and that produces the reconstruction shown in part (b) of Figure 4.4.

Example 4.2.3 [Figure 4.5] The image in (a) was obtained by loading the file `boat.bmp`. To obtain the reconstruction of the 64:1 JPEG compression shown in (b) we used the program IMAGE ANALYZER[3]. We opened the image file `boat.bmp` with `Image Analyzer` and then selected *File format options* from the *File* menu. Clicking on the *JPEG* tab and entering 4 kb for the compressed file size, we then saved the image as a **.jpg* file. Loading that compressed **.jpg* file produced the reconstruction shown in (b). The image in (c) was produced in a similar way, except that the *JPEG 2000* tab was clicked and we entered 1.5625 for the *% of compressed size*, and the *extra compression option* of `mode=real` was entered (that specifies that a Daub 9/7 transform is to be used). The image was then saved and opened as a **.jp2* file. To obtain the image in (d) we used FAWAV. After loading the `boat.bmp` image, we selected *Save options (images)* from the file menu and entered 0.125 for the bpp rate. We then saved the image (right-clicking on the image and selecting *Load/Save*) as a **.wc2* file. Loading that saved **.wc2* file produced the image shown in (d).

Note: The PSNR values cited in the text were obtained by selecting *Analysis/Norm Difference* and then choosing the option *PSNR*. For example, to obtain the PSNR between image 2 (Gr 2) and image 3 (Gr 3), where image 2 is the original being compared with, you would enter **2** for the *Graph 1* number and **3** for the *Graph 2* number, select option *PSNR*, and click on the *Compute* button. Unfortunately, FAWAV does not support the **.jp2* file format. Therefore, to obtain PSNRs for **.jp2* reconstructions, you must first save a reconstructed (decompressed) **.jp2* image in a file format that FAWAV can read (such as **.bmp*).

[3]This program can be obtained as a free download from the *Software* link at the book's website: `www.uwec.walkerjs/Primer`.

4.2.1$^{c}_{s}$ Using FAWAV for the ASWDR compressions (format `*.wc2`) and IMAGE ANALYZER for the JPEG and J2K compressions, find the PSNR values for 8:1, 16:1, and 32:1 compressions of the `Airfield.bmp` image.

4.2.2^{c} Using FAWAV for the ASWDR compressions (format `*.wc2`) and IMAGE ANALYZER for the JPEG and J2K compressions, find the PSNR values for 8:1, 16:1, and 32:1 compressions of the `goldhill.bmp` image.

4.2.3^{c} Using FAWAV for the ASWDR compressions (format `*.wc2`) and IMAGE ANALYZER for the JPEG and J2K compressions, find the PSNR values for 8:1, 16:1, and 32:1 compressions of the `peppers.bmp` image.

4.2.4^{c} Using FAWAV for the ASWDR compressions (format `*.wc2`) and IMAGE ANALYZER for the JPEG and J2K compressions, find the PSNR values for 8:1, 16:1, and 32:1 compressions of the `zelda.bmp` image.

Section 4.3

Example 4.3.1 [Figure 4.6] To obtain the image in part (a) of Figure 4.6, we loaded the image

```
fingerprint 1.bmp
```

and then right-clicked on a point with coordinates $(227, 158)$ within the image and selected *Zoom.* Images (b) to (d) were obtained by reconstructing 20:1 compressions by the JPEG, J2K, and ASWDR methods, respectively and then zooming around the same coordinates as for image (a).

Example 4.3.2 [Figure 4.7] To obtain the image in part (a) of Figure 4.7, we loaded the image

```
fingerprint 1.bmp
```

The image in (b) was obtained from reconstructing a 0.8 bpp ASWDR compression of the image in (a). To obtain the images in (c) and (d) we zoomed around the coordinates $(157, 233)$ for images (a) and (b), respectively. *Note:* If PSNR is calculated for these zoomed images (by selecting *Analysis/Norm difference* and entering the graph numbers for the zoomed images), we find that it is 31.1 dB (which is lower than the PSNR for the full images, but still above the rule-of-thumb value of 30 dB).

4.3.1^{c} Using FAWAV for the ASWDR compressions (format `*.wc2`) and IMAGE ANALYZER for the JPEG and J2K compressions, find the PSNR values for 8:1, 16:1, and 32:1 compressions of the `fingerprint 1.bmp` image.

4.3.2^{c} In problem 4.3.1, find PSNRs for zoomed images around the coordinates $(157, 233)$.

4.3.3^{c} Repeat problem 4.3.1, but use the image `fingerprint_2.bmp`.

4.3.4^{c} Using the image `fingerprint_2.bmp`, repeat problem 4.3.1, but for zoomed images around the coordinates $(157, 233)$.

Section 4.4

Example 4.4.1 [Figure 4.10] To produce Figure 4.10(a) you load the image `boat.pgm`, select *Transform/Wavelet*, and then plot a 5-level Daub 9/7 wavelet transform of the image. You then select *Graph/Plot* and plot the formula `abs(g2)>=64` where `g2` stands for the wavelet transform (graph number 2). That produces the image shown in (a), the significance map for threshold 64. To produce the image in (d), the approximate image reconstructed for threshold 64, you plot the graph `g2(abs(g2)>=64)` and compute the 5-level inverse Daub 9/7 wavelet transform of the image produced by that plot. The images in (b), (e), (c), and (f) are produced by obvious modifications of the methods just described.

Example 4.4.2 [Table 4.4] To produce the PSNR values in the *WDR (No AC)* column for the BARBARA IMAGE portion of the table, we used the program IMAGECOMPRESSOR located in the main FAWAV directory. Using the

choice *Get Image* on the *File* menu of IMAGECOMPRESSOR we loaded the image `Barb.pgm` and then performed compressions and decompressions in the following way. After selecting *Compress*, to perform *WDR (No AC)* we selected the options *Binary* and *WDR*. We also selected a Daub 9/7 transform with 5 levels and entered 1 for the *Bit rate (bpp)* value. By clicking the *Compress* button, we were then able to save to an 8:1 compressed file. By selecting *Decompress* from the file menu, we then decompressed this file and then selected *Error measures* to compute a PSNR value between the original and reconstructed image. To get the 16:1 and 32:1 PSNR values, we *did not do further compressions.* Instead, we decompressed the 1 bpp compressed file at 0.5 bpp (for 16:1) and 0.25 bpp (for 32:1). To find the PSNR values for *WDR* we proceeded as just described, except that we checked the option *Arithmetic* when performing the 1 bpp compression. For *ASWDR (No AC)*, you select *ASWDR* and *Binary.* For *ASWDR*, you select *ASWDR* and *Arithmetic*. The PSNR values for the BOATS IMAGE and X-RAY IMAGE parts of the table were computed by repeating all this work for the images `Boat.pgm` and `dog_head.pgm` respectively.

4.4.1$_s$ Find the quantized transform for the wavelet transform in Figure 4.8(b), when the threshold is 2. Also find the analog of the last stage, half-threshold, array shown in Figure 4.9(b) when the threshold is 2.

4.4.2 Find the quantized transform for the wavelet transform in Figure 4.8(b), when the threshold is 1. Why is the half-threshold, last stage approximation unnecessary when the threshold is 1?

4.4.3 For the 2-level wavelet transform shown in Figure 2, find the quantized wavelet transforms for thresholds 16, 8, and 4.

-4	-4	-12	-12	3	-4	-2	6
10	10	10	12	1	2	2	-4
-6	-6	-6	-8	4	-2	5	-8
4	4	6	4	2	4	-1	-2
4	4	5	-3	4	5	4	4
6	6	-3	5	-9	-8	-8	-7
16	20	-10	8	-9	-9	-10	-10
18	18	-10	8	8	6	6	8

Figure 2
2-level wavelet transform for Exercise 4.4.4.

4.4.4$_s$ Compute the wavelet difference reduction encoding (using symbols $+, -, 0, 1$) for the first pass (threshold 16) of the wavelet transform on the right of Figure 3, using the scan order on the left.

4.4.5 Compute the wavelet difference reduction encoding (using symbols $+, -, 0, 1$) for the Significance Pass and Refinement Pass when the threshold is 8 for the wavelet transform shown in Figure 3(b).

4.4.6 Produce figures like Figure 4.10 for the image `Barb.pgm`.

4.4.7 Produce figures like Figure 4.10 for the image `Airfield.pgm`.

4.4.8 Verify all entries in Table 4.4.

4.4.9 Add a new part to Table 4.4, labelled FINGERPRINT, by computing PSNR values for each of the three compression ratios using each of the four compression methods for the `fingerprint 1.pgm` image.

4.4.10 Add a new part to Table 4.4, labelled PEPPERS, by computing PSNR values for each of the three compression

32	31	30	29	55	61	62	64
25	26	27	28	54	56	60	63
24	23	22	21	50	53	57	59
17	18	19	20	49	51	52	58
8	7	14	16	36	37	44	45
5	6	13	15	35	38	43	46
2	4	9	12	34	39	42	47
1	3	10	11	33	40	41	48

(a) Scan order

−4	−4	−12	−12	3	−4	−2	6
10	10	10	12	1	2	2	−4
−6	−6	−6	−8	4	−2	5	−8
4	4	6	4	2	4	−1	−2
4	4	5	−3	4	5	4	4
6	6	−3	5	−9	−8	−8	−7
16	20	−10	8	−9	−9	−10	−10
18	18	−10	8	8	6	6	8

(b) Wavelet transform

Figure 3
Data for Exercise 4.4.5. (a) 2**-level scan order. (b)** 2**-level wavelet transform.**

ratios using each of the four compression methods for the `peppers.pgm` image.

Section 4.5

Example 4.5.1 [Figure 4.11] To produce Figure 4.11(a), we performed a 2-level Daub 9/7 wavelet transform of the `Boat.pgm` image. We then zoomed in on $\mathbf{v}_2$ by right-clicking on the lower left corner of the wavelet transform image and selecting *Zoom* (and then clicking on the *Zoom* button). After that first zoom, we then right-clicked on the lower right corner and zoomed once more. The resulting displayed image is $\mathbf{v}_2$. We then right-clicked on it and selected *Clip*, which produced a clipped out image of $\mathbf{v}_2$ only. In this new active window, we then graphed the function `(abs(g1)>=16)` to produce an image where the white pixels are parent values significant at threshold 16 (and the black pixels are insignificant parents). We then select *Graph/Interpolate/Haar* in order to produce a new image that is twice as large in each dimension, and each of the pixels (white and black) gives rise to a 2 by 2 child matrix of either all 1 values (if the parent pixel is white) or all 0 values (if the parent pixel is black). This image displays the locations (in white) of all child values in $\mathbf{v}_1$ whose parent values are significant at threshold 16. To get an image of only those children who are predicted to be *newly significant* at threshold 16, we return to the first window containing the wavelet transform of the `Boat.pgm` image and clip out the $\mathbf{v}_1$ subimage. This is done by right-clicking on the wavelet transform image and selecting *Restore full image* from the popup menu, then right-clicking on the lower-right corner and zooming once, and then clipping out this zoomed subimage. In the resulting new window, we graph the function `(32 > abs(g1) >= 16)` and then copy this image (right-click on it and select *Copy graph* from the popup menu). Returning to the window containing the child values in $\mathbf{v}_1$ with significant parents, we right-click on its image and select *Paste.* Finally, we graph `g1 g2` and that produces a black and white version of the image in (a) [to change it to the gray and white image displayed in the *Primer*, right-click on it and select *Graph style*, and then select the option *Grey (+/-)*]. To produce Figure 4.11(b) we return to the window with the clipping of $\mathbf{v}_1$ and plot `(16>abs(g1)>= 8)` to produce a black and white image of the new significant values in $\mathbf{v}_1$ [which one can then convert to a gray and white image identical to (b) as we did above for (a)].

To get the percentage of correct predictions of 41.2% given in the caption of Figure 4.11, we proceed as follows. First, open a new window by selecting *File/New 2-dim.* Second, copy and paste the image for (a) into this new window, then copy and paste the image for (b) into the window as well. Select *Analysis/Statistics* and compute statistics for graphs 1 and 2. The fraction of energy of graph 1 divided by the energy of graph 2 yields the required percentage.

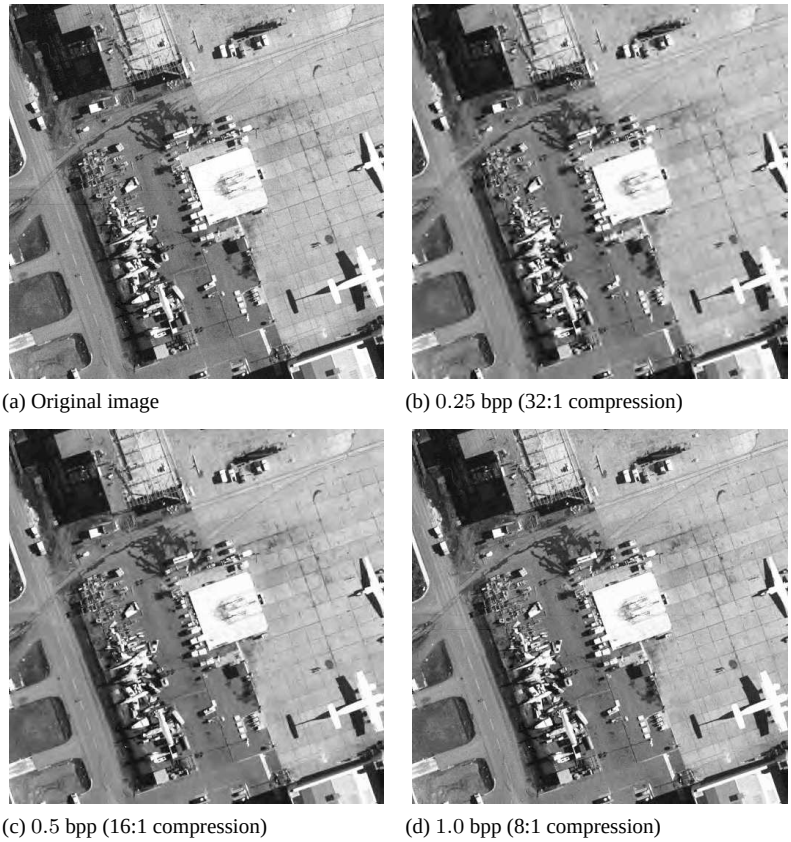
For Figures (c) and (d), we did similar work. The only differences were that we performed a 3-level transform, and to clip $\mathbf{h}_3$ we zoomed twice in a row after right-clicking on the lower left corner of the wavelet transform, then once on the upper right corner; while to clip $\mathbf{h}_2$ we zoomed once on the lower left corner, followed by once once on the upper

right corner.

Example 4.5.2 [Figure 4.12] The image used was `Barb.bmp` and we applied a 5-level Daub 9/7 transform for the WDR and ASWDR methods and the `real` mode for J2K. (See Example 4.4.2 for more details on using IMAGECOMPRESSOR and IMAGE ANALYZER) for doing these compressions.) The decompressed images were all saved as `.bmp` files for loading into the FAWAV program. Then each of these `.bmp` images, and the original `.bmp` image, were loaded into a 2-dim form in FAWAV. We then zoomed in on the same pixel for each of the four graphs (near the tip of Barb's nose).

Example 4.5.3 [Figure 4.13] These images were generated in the same way as the last example, except we zoomed on a different pixel.

Example 4.5.4 In Figure 4 we illustrate the progressive reconstruction property of ASWDR. To produce these images you select *New Image Processor* from the *File* menu of FAWAV. You then select *Get image* from the *File* menu of the *Image Processor* window. By loading the image `Airfield.pgm` you get the original (uncompressed) image shown in (a). You then select *Compress* from the main menu, and select *Daub 9/7* for the wavelet and enter 1 for the Bit rate (bpp). Click on the *Go* button to save the image in compressed form at 1 bpp (8:1 compression). From that one file, you can generate each of the decompressions in Figure 4. You select *Decompress* from the main menu, and enter 0.25 for the bit rate and then click on the *Go* button and select the compressed file for decompression. In that way, you produce the image shown in (b). The images in (c) and (d) are generated by decompressing this same compressed file, using bit rates of 0.5 and 1, respectively. Because we are only using one compressed file, this illustrates progressive reconstruction.



(a) Original image (b) 0.25 bpp (32:1 compression)

(c) 0.5 bpp (16:1 compression) (d) 1.0 bpp (8:1 compression)

Figure 4
Illustration of Progressive Reconstruction. Each image in (b) to (d) was computed from a single compressed file (saved at 1.0 bpp). First, (b) is reconstructed, then (c), then (d).

Example 4.5.5 [Table 4.5] The data shown in Table 4.5 was generated in the following way. The percentages for WINZIP were generated by using this program[4] to create an archive of compressions of `Barb.pgm` and `Boat.pgm` and `dog_head.bmp` and reading off the compression percentages provided by WINZIP. To generate the values for ASWDR we used IMAGECOMPRESSOR. By selecting *Get image* from the file menu of IMAGECOMPRESSOR we loaded in an image file. When we loaded an image, we first recorded its file size (obtained by right-clicking on it and selecting *Properties*). We then selected *Compress* from the IMAGECOMPRESSOR menu and entered 9 for the bit rate (that indicates lossless compression because the original image uses 8 bpp), and we also made sure that the wavelet was Daub 5/3 and that the options *arithmetic* for Symbol encoding and *ASWDR* for Method were checked. Clicking on the *Go* button, we saved the image in a compressed format. When decompressing we again noted the file size of the compressed file before we selecting it for decompression. Using the uncompressed and compressed file sizes for each image, we calculated the percentages shown in the ASWDR column of the table.

Example 4.5.6 [Figure 4.14] The decompressions were generated as follows. The `Airfield.pgm` image was opened using an *Image Processor* window in FAWAV. Then we compressed at 200:1 (0.04 bpp as bit rate) using a 4-level Daub 5/3 transform. After decompressing that compressed file we obtained the image in (b). To obtain the image in (d) we right-clicked on the original image (on the left of the window) and drew a rectangle enclosing the image that we want to have losslessly compressed [the airplane at lower left indicated in (c)]. We then compressed the image again. Upon decompression, the selected region of the original image is reconstructed exactly (while some error remains in the rest of the reconstructed image).

4.5.1$^{\mathbf{c}}$ Produce images like the ones in Figure 4.11, but for the `Barb.pgm` image. Calculate the percentages of correct predictors as well.

4.5.2$^{\mathbf{c}}$ Produce images like the ones in Figure 4.11 for the `Boat.pgm` image, except this time show children of $\mathbf{h}^2$ as predictors of new significant values for $\mathbf{h}^1$, and children of $\mathbf{v}^3$ as predictors of new significant values for $\mathbf{v}^2$. Calculate the percentage of correct predictors as well.

4.5.3$^{\mathbf{c}}$ Produce images like the ones in Figure 4.11, but for the `Barb.pgm` image. Calculate the percentages of correct predictors as well.

4.5.4$^{\mathbf{c}}$ Produce images like the ones in Figure 4.11, but in this case use the `Barb.pgm` image and show children of $\mathbf{h}^2$ as predictors of new significant values for $\mathbf{h}^1$, and children of $\mathbf{v}^3$ as predictors of new significant values for $\mathbf{v}^2$. Calculate the percentage of correct predictors as well.

4.5.5$^{\mathbf{c}}_{\mathbf{s}}$ Create images like Figure 4 for the `boat.pgm` image.

4.5.6$^{\mathbf{c}}$ Create images like Figure 4 for the `peppers.pgm` image.

4.5.7$^{\mathbf{c}}$ Create images like Figure 4 for the `goldhill.pgm` image.

4.5.8$^{\mathbf{c}}$ Add entries to Table 4.5 for the `peppers.pgm` and `zelda.pgm` and `airfield.pgm` images. Which of the images (now six in number) shows the least compression? Why do you think it compresses the least?

4.5.9$^{\mathbf{c}}_{\mathbf{s}}$ Produce images like in Figure 4.14, but this time select a region of interest that contains the swept-wing aircraft just to the left of center. How much savings in file size do you get over sending a lossless compression of the entire image?

4.5.10$^{\mathbf{c}}$ Produce images like in Figure 4.14, but this time use the `dog_head.bmp` image and select a region of interest that contains just the jaw region of the dog. How much savings in file size do you get over sending a lossless compression of the entire image?

[4] WINZIP is available from `http://www.winzip.com/`

Section 4.6

In the following example and exercises for section 4.6, we consider the *rate-distortion* curves (R-D curves) for J2K. An R-D curve is a graph of PSNR versus bit-rate for a given image: e.g., bit-rate in increments of 0.1 bpp along the horizontal and PSNR in dBs along the vertical.

Example 4.6.1 In Figure 5 we show the R-D curve for J2K for the `Goldhill.bmp` image, and for comparison the R-D curve for ASWDR. To obtain the J2K compressions and PSNR values we used IMAGE ANALYZER in conjunction with FAWAV. (See Example 4.2.3 and the Note that follows it.) To obtain the ASWDR compressions and PSNR values we used IMAGECOMPRESSOR. (Note: We used 7 levels of Daub 9/7 transform and selected the *ASWDR* and *Arithmetic* coding options, and compressed to a 1 bpp file, which we then decompressed at 0.1, 0.2, ..., 1.0 bpps, and selected *Error* with *PSNR* option to measure the PSNR for each decompression.)

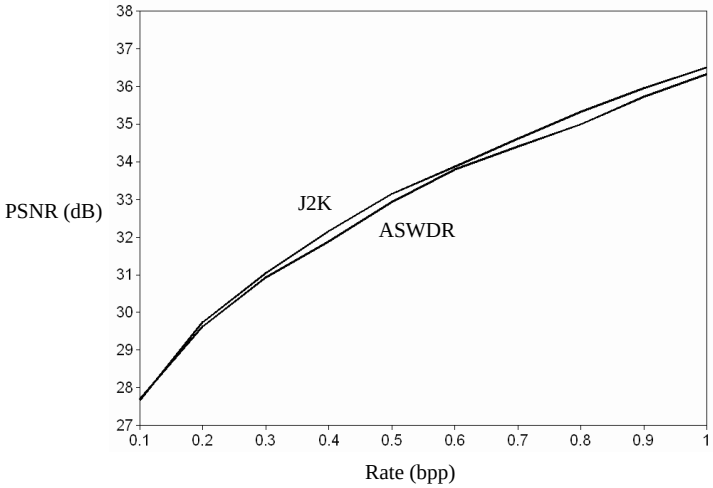


Figure 5: Rate-Distortion curves for the `Goldhill.bmp` image.

4.6.1$^{\mathbf{c}}$ Graph RD-curves for J2K and ASWDR using the `Airfield.bmp` image.

4.6.2$^{\mathbf{c}}$ Graph RD-curves for J2K and ASWDR using the `Barb.bmp` image.

4.6.3$^{\mathbf{c}}_{\mathbf{s}}$ Graph RD-curves for J2K and ASWDR using the `Boat.bmp` image.

4.6.4$^{\mathbf{c}}$ Graph RD-curves for J2K and ASWDR using the `Mountain.bmp` image.

4.6.5$^{\mathbf{c}}$ Graph RD-curves for J2K and ASWDR using the `Peppers.bmp` image.

Section 4.7

Example 4.7.1 [Figure 4.15] To create the image in (a), the image `Boat.bmp` was loaded into FAWAV. The noisy image in (b) was created as follows: random noise with $\sigma = 20$ was added to the image by plotting the graph of `g1+20rang(0)` and then the resulting image was converted to an 8 bpp gray-scale image by selecting *Graph/Quantize (8-bit)*. *After deleting the second graph (the unquantized noisy image)*, we created the image in (c) by first performing a 5-level Daub $9/7$ wavelet transform of the noisy image, and then plotting the function

```
g3(abs(g3)>= 20sqr(2log(512)))
```

to produce a denoised transform. *After deleting graph 3 (the noisy transform),* we then performed an inverse 5-level Daub 9/7 transform on graph 3 to produce an (unquantized) denoised image. Finally, we deleted graph 3 (the denoised transform) and selected *Graph/Quantize (8-bit)* to create an 8 bpp gray-scale image. This final image is the base threshold denoising shown in (c). To create the image in (d), we deleted the unquantized denoising (graph 3 in the window), and then selected *Graph/Denoise (wavelet)*. By clicking the *Plot* button (with graph 2 and 5 levels and a Daub 9/7 wavelet specified), we performed a TAWS denoising of the noisy image (which is automatically quantized to an 8-bit gray-scale image), producing the image shown in (d). We also computed PSNR values for each of the images in (b) to (d), in comparison to the original image in (a). We obtained the following results: (b) 22.2 dB, (c) 27.0 dB, (d) 29.6 dB; which show that, for this example, TAWS provides a 2.6 dB improvement over the base threshold method.

Example 4.7.2 [Table 4.6] We describe how to obtain the results in the third row, for the *Boats* case with $\sigma = 8$. The other rows of the table are obtained in a similar way. First, we loaded the image `boat.bmp` into FAWAV. We then added noise with $\sigma = 8$ to the image by plotting the graph `g1 + 8rang(0)` and then quantizing to 8 bpp (as explained in the previous example). We then deleted graph 2 (the unquantized, noisy image) and saved the noisy gray-scale image to the file:

```
c:\fawave\images\noisy_boat_8.bmp
```

To obtain the Wiener denoising, using MATLAB, we executed the following three MATLAB commands:

```
I=imread('c:\fawave\images\noisy_boat_8.bmp');
J=wiener2(I);
imwrite(J,'c:\fawave\images\noisy_boat_8_wiener.png';
```

The last command saves the wiener2 denoised image to a `png` file (a simpler image format for MATLAB syntax) rather than a `bmp` file. Loading the wiener2 denoised image into FAWAV, we then computed a PSNR value for it in comparison to the original image. That gave the result shown in the table under *Wiener* (your results might differ slightly due to the random nature of the noise). To obtain the results under *TAWS* we performed a TAWS denoising (as described in the previous example) of the noisy image, and then computed a PSNR value of the TAWS denoising in comparison to the original image (again, your results might vary slightly due to randomness).

Remark As one can see from this last example, MATLAB is very simple to operate, but also very powerful. Unfortunately, it also *very expensive* (especially if one has to also buy the IMAGE PROCESSING and SIGNAL PROCESSING toolkits which are needed for the examples described in this book). Because of MATLAB's cost, I decided to concentrate on examples using FAWAV (which although less powerful than MATLAB, has the advantage of being free).

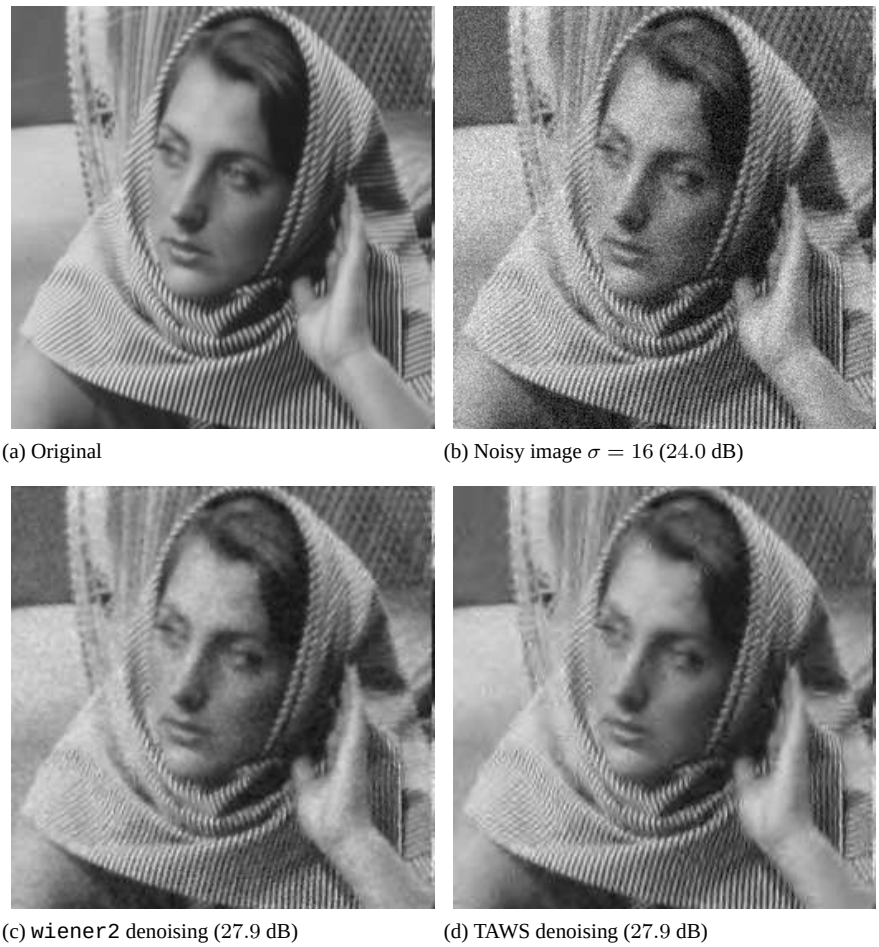
Example 4.7.3 [Figure 4.16] The images in the figure were produced using the method described in the previous example (using the `Barb.bmp` image, and $\sigma = 16$). If, in addition, we zoom in on the upper right corner of all four images, then we obtain the images shown in Figure 6.

Example 4.7.4 The TAWS denoising in Figure 6(d) suffers from some annoying noise residuals (which appear as small pixel-size blemishes). An improved TAWS denoising option is available in FAWAV. It is called TAWS-SPIN and is described in detail in a paper of the author's, *Tree-adapted wavelet shrinkage,* which can be downloaded from

```
http://www.uwec.edu/walkerjs/media/TAWSsurv.pdf
```

Here we shall illustrate TAWS-SPIN. By selecting the *Denoise (wavelet)* option, and selecting a Daub 9/7 wavelet, you then check the box labeled *Avg.* and select the *2D* option. That performs the TAWS-SPIN algorithm with the parameters recommended in the paper. For instance, performing it on the noisy Barbara image from the previous example, we obtain the image shown in Figure 7(d). Compared to Figure 6(a), there is a slight improvement in PSNR and the pixel-size blemishes are gone.

Example 4.7.5 In Figure 8 we compare TAWS-SPIN with the Wiener2 method for the `Boat.bmp` image contaminated with $\sigma = 32$ Gaussian random noise (the noisy image is at the FAWAV webpage as `Noisy_boat_32.bmp`). The TAWS-SPIN denoising shows a much higher PSNR than the Wiener2 image and is much less free of noise artifacts.



(a) Original (b) Noisy image $\sigma = 16$ (24.0 dB)

(c) `wiener2` denoising (27.9 dB) (d) TAWS denoising (27.9 dB)

Figure 6
Zooms of two denoisings of Barbara image (PSNR values in parentheses).

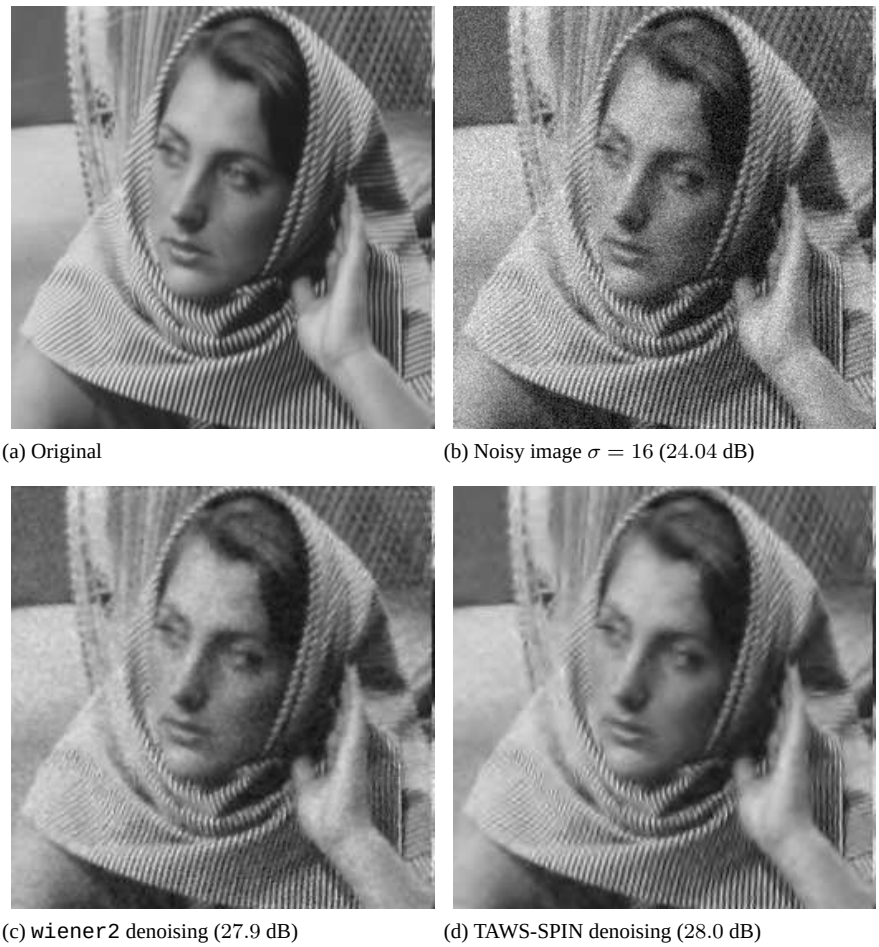
Example 4.7.6 [Figure 4.18] We loaded the image `STM_Si_111_a.bmp` into FAWAV to produce the image in (a). To denoise it, we performed a 5-level Daub 9/7 transform and then plotted the following function

```
g2 (x<c)(y<c)
+ g2(abs(g2)<350)(1-(x<c)(y<c))(x<a)(y<a)
+ g2 (abs(g1)<t)(1-(x<a)(y<a))
\c = -.5+1/2^5 \a=-.5+1/2^4
\t =20
```

to process the transform. We then performed an inverse 5-level Daub 9/7 transform on the processed transform to produce the denoised image in (d). [The images in (b) and (c) were produced by zooming in on portions of the transform of (a), after selecting *Graph style* and choosing the options *LinLog* and *Grey (+/-).* For (b), we also entered 20 for a threshold.]

Example 4.7.7 [Figure 4.19] We loaded the image `STM_Si_111_b.bmp` into FAWAV to produce the image in (a). To denoise it, we performed a 5-level Daub 9/7 transform and then plotted the following function

```
g2 (x<c)(y<c) + g2 (abs(g2)<t)(1-(x<c)(y<c))
\c = -.5+1/2^4
```



(a) Original (b) Noisy image $\sigma = 16$ (24.04 dB)

(c) `wiener2` denoising (27.9 dB) (d) TAWS-SPIN denoising (28.0 dB)

Figure 7
Zooms of two denoisings of Barbara image (PSNR values in parentheses).

```
\t = 30
```

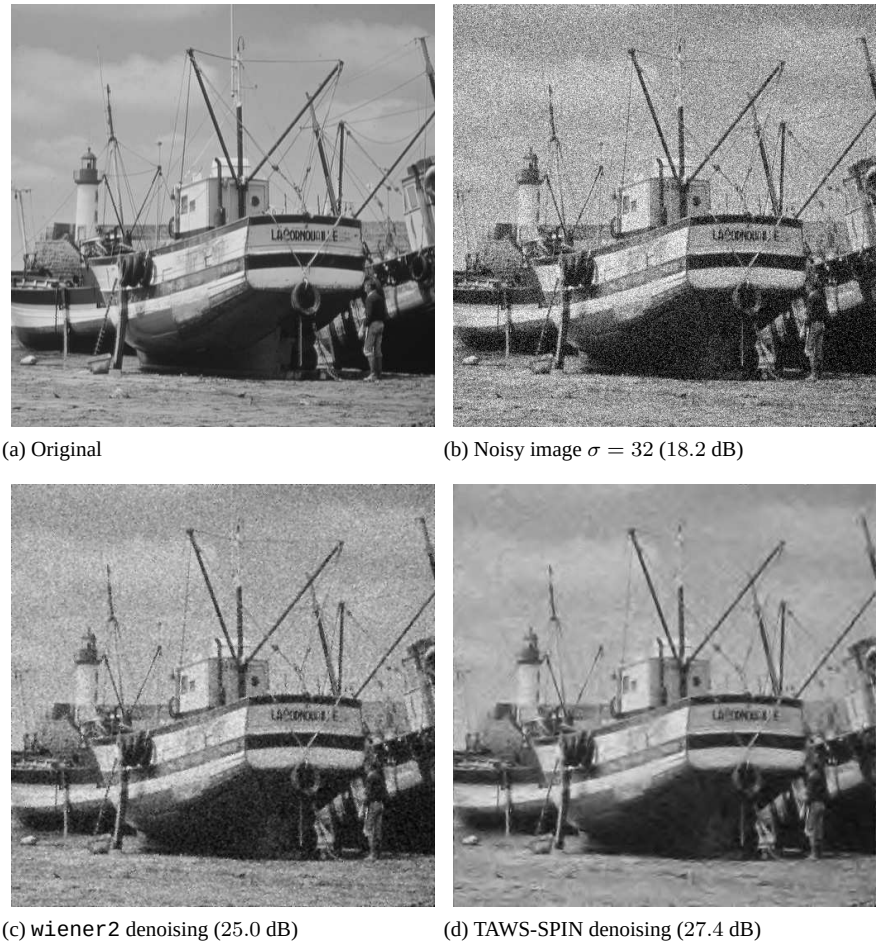
to process the transform. We then performed an inverse 5-level Daub 9/7 transform on the processed transform to produce the denoised image in (b).

4.7.1[c] Denoise the noisy images `Lena_16.bmp` and `Lena_24.bmp` and `Lena_32.bmp` (noisy images with $\sigma = 16, 24, 32$, respectively) using TAWS, and TAWS-SPIN, and (if available) Wiener2. Which method appears best (both objectively, in terms of PSNR, and subjectively, in terms of how it appears to you visually)?

4.7.2[c] Denoise the noisy images `Goldhill_16.bmp` and `Goldhill_24.bmp` and `Goldhill_32.bmp` (noisy images with $\sigma = 16, 24, 32$, respectively) using TAWS, and TAWS-SPIN, and (if available) Wiener2. Which method appears best (both objectively, in terms of PSNR, and subjectively, in terms of how it appears to you visually)?

4.7.3[c] Denoise the noisy images `Boat_16.bmp` and `Boat_24.bmp` and `Boat_32.bmp` (noisy images with $\sigma = 16, 24, 32$, respectively) using TAWS, and TAWS-SPIN, and (if available) Wiener2. Which method appears best (both objectively, in terms of PSNR, and subjectively, in terms of how it appears to you visually)?

4.7.4[c] Denoise the noisy images `Peppers_16.bmp` and `Peppers_24.bmp` and `Peppers_32.bmp` (noisy images with $\sigma = 16, 24, 32$, respectively) using TAWS, and TAWS-SPIN, and (if available) Wiener2. Which method



(a) Original (b) Noisy image $\sigma = 32$ (18.2 dB)

(c) `wiener2` denoising (25.0 dB) (d) TAWS-SPIN denoising (27.4 dB)

Figure 8
Two denoisings of Boats image (PSNR values in parentheses).

appears best (both objectively, in terms of PSNR, and subjectively, in terms of how it appears to you visually)?

4.7.5$^{c}_{s}$ The image `Elaine.bmp` has some noise (try zooming a few times on the center pixel to see it more clearly). Denoise this image using TAWS, and TAWS-SPIN, and (if available) Wiener2. Which method appears best subjectively, in terms of how it appears to you visually?

4.7.6^{c} Try to improve the denoising of the `STM_Si_111_a.bmp` image, obtaining a more sharply focused image than the example given in the text (see Example 4.7.6).

4.7.7^{c} Try to improve the denoising of the `STM_Si_111_b.bmp` image, obtaining a more sharply focused image than the example given in the text (see Example 4.7.7).

Section 4.8

Example 4.8.1 [Figure 4.20] To produce (a) we first graphed an octagon shaped figure by plotting the following function

```
(y-x-c<=0)(x+y-c<=0)(x-y-c<=0)
```

```
(x+y+c>=0)(abs(y)<=b)(abs(x)<=b)
\c=12/5\b=8.5/5
```

over the region $[-4,4] \times [-4,4]$ using 128 as the choice for *Points.* We then computed a 1-level Coif6 transform and plotted

```
10g2(1-(x<0)(y<0))
```

After changing the display style of the resulting graph to *Log* and setting a threshold of 0.1 we obtained the image shown in (a). To obtain the image in (b) we then performed an inverse 1-level Coif6 transform on image (a) and changed its graph style to *Grey (+/-).*

Example 4.8.2 [Figure 4.21] To obtain the images in (a) and (b) we first loaded the image `house.pgm` and that gave us (a). We then performed a 1-level Daub4 transform on this image and plotted the function

```
g2+2g2(1-(x<0)(y<0))
```

to obtain a processed transform. We then performed an inverse 1-level Daub4 transform of this processed transform. By quantizing to get an 8-bit gray-scale image, we obtained the image (b).

Example 4.8.3 [Figure 4.22] The image in (a) was obtained by successively loading the images `goldhill.pgm` and `boat.pgm` and `airfield.pgm` and `peppers.pgm` into a 2-dim form. The image in (b) were obtained by performing 2-level Coif18 transforms of these images, copying and pasting them into a single 2-dim form, and changing their graph styles to *Lin-Log* 128 and choosing a threshold of 4.

Example 4.8.4 [Table 4.7] We explain how the entries for the column labeled *Second* were obtained (the entries for the other columns were obtained in a similar way). First we obtained the denoised image of the noisy boats image (see Example 4.7.1). We copied and pasted this denoised image into a new 2-dim form. We then produced a 2-level Coif18 transform of this denoised image, and displayed it with the same graph style as the images in Figure 4.21(b) discussed in Example 4.8.2. The next step is to compute the relative 2-norm differences between the trend subimages. To do that for the Gr 1 entry (corresponding to the Goldhill image) we zoomed in on the lower left corner of the Coif18 transform of the Goldhill image, so that just the trend subimage was displayed. We then clipped this subimage to obtain a new 2-dim form containing a display of just the trend subimage. and repeated this process with the denoised Boats image transform. We then copied the trend subimage from the Goldhill image and pasted it into the form containing the trend subimage from the denoised Boats image. Finally, we computed a 2-norm difference between graphs 1 and 2, with the option *Relative* selected. That gave us the value 0.455 shown in the table. We then repeated this process with the other images to get the remaining entries. *Note:* The values you obtain may differ slightly from the ones reported in Table 4.7 due to the random nature of the additive noise.

4.8.1$^{\mathbf{c}}$ Produce an image of the edges of `Peppers.bmp`.

4.8.2$^{\mathbf{c}}_{\mathbf{s}}$ Edge enhance the image `cathedra.pgm`.

4.8.3$^{\mathbf{c}}$ Edge enhance the image `Peppers.bmp`.

4.8.4$^{\mathbf{c}}$ Add additional data to Table 4.7, using the image `Elaine.pgm`.

4.8.5$^{\mathbf{c}}$ Add additional data to Table 4.7, using the image `Zelda.pgm`.

4.8.6$^{\mathbf{c}}_{\mathbf{s}}$ Construct a table analogous to Table 4.7, using a 32:1 compression of *Lena.pgm* in comparison to the four images: `Barb.bmp`, `Zelda.bmp`, `Lena.pgm`, and `Peppers.bmp`.

Chapter 5

Section 5.1

Example 5.1.1 [Figure 5.1] To create (a) we used *Graph/Plot* and plotted

```
2cos(4pi x) + 0.5sin(24pi x)
```

over the interval $[-16, 16]$. For (b) we then selected *Transform/Fourier* and pressed the *Plot* button. For (c) we opened a new 1-D form, and plotted the function

```
(1 + cos(24 pi x))/(1 + 4xˆ2)
```

over the interval $[-16, 16]$, after which we obtained (d) by the method used to get (b).

Example 5.1.2 [Figure 5.2] To get (a) we plotted `del(x)` over the interval $[0, 1024]$ using 1024 points. We then computed a 1-level Coif12 inverse wavelet transform. That produced a plot of the scaling signal $\mathbf{V}_1^1$. We then chose *Transform/Fourier* and selected the options *Power sp.* and $[0, L] \to [-A, A]$, and unchecked the box labelled *Periodic, endpoint averaged,* to plot the spectrum of $\mathbf{V}_8^1$. For (b) we did the same work, except we first plotted `del(x-512)`.

Example 5.1.3 We show in Figure 9 the graphs of the spectra of the Coif12 scaling signal $\mathbf{V}_{17}^1$ and wavelet $\mathbf{W}_{17}^1$. These spectra were produced as in Example 5.1.2, using `del(x-16)` and `del(x-512-16)`. Notice how they match the spectra of $\mathbf{V}_1^1$ and $\mathbf{W}_1^1$ shown in Figure 5.2 in the Primer.

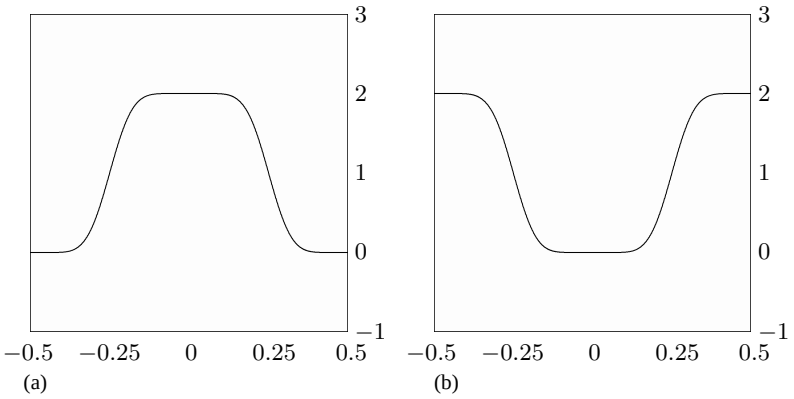


Figure 9
(a) Spectrum of Coif12 scaling signal $\mathbf{V}_{17}^1$. (b) Spectrum of Coif12 wavelet $\mathbf{W}_{17}^1$.

5.1.1$^{c}_{s}$ Produce a graph of

$$3\sin(8\pi x) - 2\cos(16\pi x)$$

over the interval $[-16, 16]$ using 1024, and then produce a plot that displays the frequency content of this function.

5.1.2^{c} Produce a graph of

$$2\cos(12\pi x) + 8\sin(24\pi x)$$

over the interval $[-16, 16]$ using 1024, and then produce a plot that displays the frequency content of this function.

5.1.3$^{c}_{s}$ Plot spectra of the Coif12 scaling signal $\mathbf{V}_{20}^1$ and wavelet $\mathbf{W}_{20}^1$.

5.1.4^{c} Plot spectra of the Coif18 scaling signal $\mathbf{V}_{20}^1$ and wavelet $\mathbf{W}_{20}^1$.

5.1.5^{c} Plot spectra of the Daub12 scaling signal $\mathbf{V}_{20}^1$ and wavelet $\mathbf{W}_{20}^1$.

5.1.6^{c} Plot spectra of the Haar scaling signal $\mathbf{V}_{20}^1$ and wavelet $\mathbf{W}_{20}^1$.

Section 5.2

5.2.1$_s$ Prove the linearity property of the DFT.

5.2.2$_s$ Prove the periodicity property of the DFT.

5.2.3$_s$ Prove the inversion property of the DFT.

5.2.4 Prove Parseval's Equality for the DFT.

5.2.5$_s$ Find the z-transform of $\mathbf{f} = (1, 1, 1, 1)$. What are its roots?

5.2.6 Find the z-transform of $\mathbf{f} = (1, 0, 1, 0, 1, 0, 1, 0)$. What are its roots?

5.2.7 Prove that $\mathcal{T}_k \circ \mathcal{T}_m = \mathcal{T}_{k+m}$. Also prove that $\mathcal{T}_{-k} = \mathcal{T}_k^{-1}$.

Section 5.3

Example 5.3.1 [Figure 5.3] To produce the graph in (a) we first plotted

```
(1 + cos(24 pi x))/(1 + 4xˆ2)
```

over the interval $[-16, 16]$ using 1024 points. We then selected *Series/Wavelet* and chose a 1-level Coif12 wavelet with option *Ascending terms* and entered 512 for the number of terms. Plotting that series produced the graph shown in (a). To produce the graph in (b) we performed a Fourier transform of the graph in (a). We obtained the graph in (c) by subtracting the graph in (a) from the original function's graph. The graph in (d) is the Fourier transform of the graph in (c).

Example 5.3.2 [Figure 5.4] To obtain the graphs in (a) we computed power spectra of scaling signals $\mathbf{V}_1^k$ and multiplied each power spectrum by 2^{-k} for $k = 1$ to $k = 4$. For example, to plot $\mathbf{V}_1^3$ we graphed `del(x)` over $[0, 1024]$ using 1024 points, and then computed a 3-level inverse Coif12 transform. Plotting the Power spectrum and multiplying by 2^{-3} produced the 3rd graph from the top in (a). Similar work was done to produce the graphs in (b) except that wavelets $\mathbf{W}_1^k$ were used. For instance, to plot the graph of $\mathbf{W}_1^3$ we plotted `del(x-128)` over $[0, 1024]$ using 1024 points and then computed an inverse 3-level Coif12 transform.

5.3.1$_s^c$ In Figure 10(a) we show the graph of the function

$$\frac{1 + \cos(16\pi x)}{1 + 4x^2} + \frac{1 + \sin(24\pi x)}{1 + 4(x - 8)^2} + \frac{1 - \cos(8\pi x)}{1 + 4(x + 8)^2}$$

over the interval $[-16, 16]$ using 1024 points, and in (b) we show the graph of its DFT. Reproduce these graphs and identify the portions of the DFT graph that correspond to transforms of the 3-level Coif12 averaged signal $\mathbf{A}^3$, and detail signals $\mathbf{D}^3$, $\mathbf{D}^2$, $\mathbf{D}^1$.

5.3.2^c Graph the function

$$\sum_{k=1}^{4} \frac{1 + 4\cos(2\pi 3 \cdot 2^{k-1}x)}{1 + 4(x - (-40 + 16k))^2}$$

over the interval $[-32, 32]$ using 4096 points, and plot its DFT. For a Coif12 wavelet, identify which portions of the DFT correspond to transforms of the signals $\mathbf{A}^4$, $\mathbf{D}^4$, $\mathbf{D}^3$, $\mathbf{D}^2$, $\mathbf{D}^1$.

5.3.3 Prove formula (5.29).

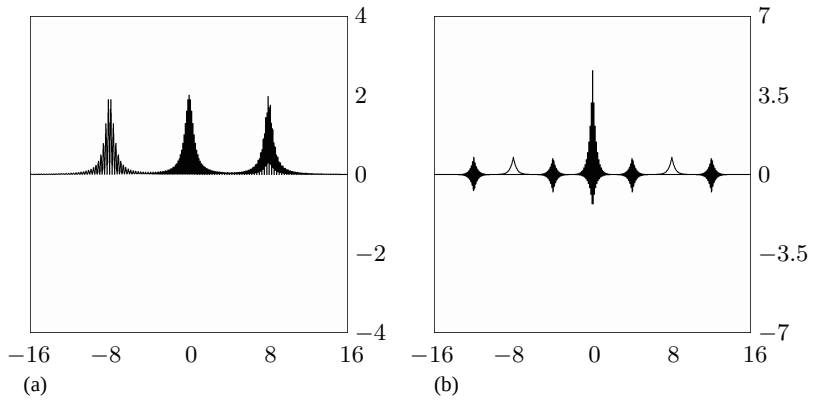


Figure 10
(a) Graph of function from Exercise 5.3.1. (b) DFT of that function.

Section 5.4

Example 5.4.1 [Figure 5.5] The graph of the top signal was obtained by plotting

```
sumk(100u(u-.1)(u+.2)(-.2<u<.1)(abs(k-1)>.5))
+.1sin(12pi v)(-.2<v<.1)
\u = x-2k \k=-2,2 \v=x-2
```

over the interval $[-5, 5]$ using 1024 points. The middle signal was obtained by then plotting

```
100u(u-.1)(u+.2)(-.2<u<.1) \u=x
```

over the same interval with the same number of points. To obtain the bottom graph we selected *Convolve* and *Pair Correlation*, and entered 2 for *Graph 1* and 1 for *Graph 2* with the *Normalize* option selected, and after plotting this normalized correlation, we then plotted the graph of `g3(x)(g3(x)>.9)` to show those correlation values that exceeded 0.9.

5.4.1[c] Compute a graph showing the values of the normalized correlation of the following two graphs, over the interval $[-5, 5]$ using 1024 points:

$$\mathbf{g}: \quad 2(-3 < x < -2) + (1 - \text{abs}(x))(\text{abs}(x) < 1) - 1(3 < x < 4)$$

$$\mathbf{f}: \quad 2(-.5 < x < .5)$$

Explain why the position of the maximum of 1 for the normalized correlation occurs where it does, and why the minimum of -1 occurs where it does.

5.4.2[c] Compute a graph showing the values that exceed 0.90 of the normalized correlation of the following two graphs, over the interval $[-5, 5]$ using 1024 points:

$$\mathbf{g}: \quad \text{sumk}(100u(u-.1)(u+.2)(-.2<u<.1)(\text{abs}(k-1)>.5)) + .1\cos(14\pi v)(-.2<v<.1)$$
$$\backslash u = x - 2k \backslash k = -2, 2 \backslash v = x - 2$$

$$\mathbf{f}: \quad 100u(u-.1)(u+.2)(-.2<u<.1) \backslash u = x$$

Section 5.5

Example 5.5.1 [Figure 5.6] To produce this figure you first load the image `P.bmp` and then the image `PQW.bmp` and that displays Gr 1 and Gr 2. To produce Gr 3 you select *Convolve* and *Pair correlation* and check the box entitled *Normalize.* You also enter 1 for Graph 1, and 1 for Graph 2. After plotting the normalized correlation as the third graph and then plotting `(g3 > .9)` you obtain the tiny white pixel at the center shown in Gr 3 in the figure. To obtain the image in Gr 4 you proceed in a similar way, except you enter 1 for Graph 1 and 2 for Graph 2 when computing the normalized correlation.

Example 5.5.2 [Figure 5.7] To obtain the graphs in (a) you load successively the images `Elaine_face.bmp`, `boat.bmp`, `Elaine.bmp` and `Zelda.bmp`. To obtain the images in (b) you perform normalized correlations with `Elaine_face.bmp` as Graph 1, and each of the images in (a) as Graph 2, and you also plot the values exceeding 0.9 for each of these normalized correlations.

Example 5.5.3 [Figure 5.8] To obtain the graphs in (a) you proceed as follows. Perform a 1-level Coif12 transform of `Elaine_face.bmp`, then graph `g2(1-(x<0)(y<0))` and then inverse transform on Gr 3 to get $\mathbf{D}^1$. You then plot

```
g4((x^2 + y^2) < c^2) \c = 0.15
```

to remove extraneous edges at the boundary of the disc containing edges of Elaine's face. Then repeat this work (without the boundary removal step) for each of the images `boat.bmp`, `Elaine.bmp`, and `Zelda.bmp`, copying and pasting the $\mathbf{D}^1$ images into the form containing the face version of $\mathbf{D}^1$, and computing normalized correlations with thresholding at 0.9.

5.5.1$^{\mathbf{c}}$ Do a normalized correlation, and retain only values exceeding 0.9, for the image `2DCorr_a.bmp` of the letter *a* within the portion of text in the image `2DCorr_text.bmp`. Verify that all instances of the letter *a* and their positions are detected correctly.

5.5.2$^{\mathbf{c}}$ Do a normalized correlation, and retain only values exceeding 0.9, for the image `2DCorr_u.bmp` of the letter *u* within the portion of text in the image `2DCorr_text.bmp`. Verify that all instances of the letter *u of the same size* and their positions are detected correctly. **Note:** Some larger size versions of `u` are not detected. This provides evidence that our brains use much more sophisticated methods of pattern matching, valid across a range of sizes, not just a single fixed size.

5.5.3$_{\mathbf{s}}$ In the previous two exercises, the text and the individual letters were white on a black background. Explain how to handle letter detection if the text and letter are both black on a white background. Do you see any relation to this problem and vision?

5.5.4$^{\mathbf{c}}$ Apply the edge correlation method to detecting the presence of `Barb_face` within the image `Barb.bmp` and its lack of presence within the images `Zelda.bmp` and `Elaine.bmp`.

5.5.5$^{\mathbf{c}}$ Repeat the previous exercise, but use edge correlations based on the 3^{rd} level Coif12 trends for the images.

Section 5.6

5.6.1 Show that for $P(\theta) = e^{i\pi}\cos\pi\theta$, we obtain the Haar scaling numbers $\alpha_1 = \alpha_2 = 1/\sqrt{2}$ and wavelet numbers $\beta_1 = 1/\sqrt{2}$, $\beta_2 = -1/\sqrt{2}$.

5.6.2 Show that (5.67) implies (5.68).

5.6.3$_{\mathbf{s}}$ Use the method of this section to derive the Daub6 scaling numbers and wavelet numbers.

5.6.4 Generalize the method of this section to the biorthogonal case.

Section 5.7

Example 5.7.1 [Figure 5.9] To produce (a) we plotted the function

$$e^{-400(t-0.2)^2}\sin 1024\pi t + e^{-400(t-0.5)^2}\cos 2048\pi t + e^{-400(t-0.7)^2}(\sin 512\pi t - \cos 3072\pi t)$$

over the interval $[0, 1]$ using 8192 points. The DFT in (b) was then produced by choosing *Transform/Fourier* and selecting `[0,L]->[-A,A]` for *Interval Type.* To produce (c) we selected *Analysis/Spectrograph*, from the window containing the graph in (a), and when the spectrogram window opened we then plotted the spectrogram with *Blackman* specified as the filter.[5] We produced (d) by selecting *None* for the filter in the spectrogram (this specifies a Boxcar window).

Example 5.7.2 [Figure 5.10] The graph in (a) was created by opening a new 1D-form and then right-clicking on the graph region followed by selection of *Load/Sound file.* We then selected the sound file `flute_clip.wav`. The graph in (b) was created by plotting several shifts of the Blackman window defined in the Primer, and (c) was obtained by multiplying the signal in (a) by the central window displayed in (b).

Example 5.7.3 [Figure 5.11] This figure was obtained by plotting the formulas for the Hanning and Blackman window functions, using $\lambda = 1$, over the interval $[-0.5, 0.5]$ using 1024 points.

5.7.1$^{c}_{s}$ Plot the following function

$$e^{-400(t-0.2)^2}\sin 2048\pi t + e^{-400(t-0.5)^2}\cos 512\pi t + e^{-400(t-0.7)^2}(\sin 1024\pi t - \cos 3072\pi t)$$

over the interval $[0, 1]$ using 8192 points and then compute its Hanning and Blackman windowed spectrograms. Do you observe any differences?

5.7.2^{c} Plot the following function

$$e^{-400(t-0.2)^2}\sin 512\pi t + e^{-400(t-0.5)^2}\cos 2048\pi t + e^{-400(t-0.7)^2}(\sin 512\pi t + 0.5\cos 1024\pi t - \cos 3072\pi t)$$

over the interval $[0, 1]$ using 8192 points and then compute its Hanning and Blackman windowed spectrograms.

5.7.3^{c} Load the signal `greasy.wav` and compute its Blackman windowed spectrogram.

5.7.4^{c} Load the signal `Chong's 'Bait'.wav` and compute its Blackman windowed spectrogram.

Section 5.8

Example 5.8.1 [Figure 5.12] The spectrogram in (a) was generated by loading the sound file `piano_clip.wav` and then computing a Blackman windowed spectrogram. For (b) we plotted the formula `sin[8192(pi/3)x^3]` over the interval $[0, 1]$ using 8192 points and then computed a Blackman windowed spectrogram.

Example 5.8.2 [Figure 5.13] The spectrum in (a) was computed in the following way. In the *spectrogram* window containing the Blackman windowed spectrogram of the piano notes (see last example), we selected *View* and then chose *View cursor coordinates.* We then moved the mouse cursor over the spectrogram until the tip of the mouse cursor displayed 0.6 as its first coordinate, and then right-clicked and chose *Vertical slice* from the popup menu. That produced the real and imaginary parts of the FFT corresponding to the vertical slice along $t = 0.6$ in the spectrogram. To get the spectrum's plot, we graphed the function

[5] A time-domain filter is usually called a *window;* we used the term *window* in the Primer.

```
sqr(g1(x)^2 + g2(x)^2)
```

and then clipped out graph 3, and changed the display style to *Lines* with the X and Y ranges shown in the figure. To produce (b) we used the same steps, except that we right-clicked on the spectrogram when the first coordinate was 1.115.

Example 5.8.2 [Figure 5.14] The spectrogram in this figure was created by juxtaposing two spectrograms. We began by loading the sound file `firebird_clip2.wav`, which loaded into an *Audio editor* window. We then left-clicked at about a quarter of the way from the left end of the signal (at about 54000 for the *Line 1* reading) to create a beginning clip-line, followed by a click toward the middle of the signal to create a right-click line. We then right-clicked on the selected region and chose *Clip.* By clicking on the *Analyze* button we opened the clipped signal within a new 1D-form and then computed a Blackman windowed spectrogram. This spectrogram is the left half of the spectrogram shown in the figure. To produce the right half, we right-clicked on the right clip line and selected *Move clip region.* The clipped signal shown in the small box on the right of the *Audio editor* is automatically updated with this new clipping. We then clicked the *Analyze* button and created a Blackman windowed spectrogram, which is the right half of the spectrogram shown in the figure.

Example 5.8.3 [Figure 5.15] To create the spectrogram in (a) we loaded the sound file

```
Chinese_folk_Music.wav
```

and computed a Blackman windowed spectrogram. The zooming in shown in (b) was computed in the following way (the explanation of the mathematics underlying the following procedures is discussed in sections 6.3 and 6.4 of the Primer). From the menu for the original sound signal we selected *Analysis* and chose *Scalogram.* In the *scalogram* window that opens, we chose to compute a *Gabor (complex)* scalogram, using the following settings:

```
Octaves: 2      Voices: 128

Width: 0.25     Freq.: 125
```

and then clicked on the *Plot* button to compute the scalogram. Once the scalogram was plotted, we then selected *View/Display style* and selected *Log (global)* for the *Magnitude* setting.

Example 5.8.4 [Figure 5.16] These spectrograms were created by loading the sound files `osprey_song.wav` and `oriole_song.wav` and computing Blackman windowed spectrograms.

5.8.1$^{\mathbf{c}}$ Analyze the oriole's song from the recording `oriole_song.wav`.

5.8.2$^{\mathbf{c}}_{\mathbf{s}}$ Use AUDACITY to create a spectrogram of the passage from the classical Chinese folk song recorded in the file `Happiness_clip.wav`, and use the Multiresolution Principle to analyze its musical qualities. Details on how to use AUDACITY can be obtained by selecting the link

Document on using *FAWAV* and Audacity

available at the following website

```
http://www.uwec.edu/walkerjs/tfam/
```

5.8.3$^{\mathbf{c}}$ Analyze the passage from the song *Buenos Aires* in the recording `BA_passage.wav`.

Section 5.9

Example 5.9.1 [Figure 5.16] To create (a) we loaded the sound file `oriole_whistle.wav` and performed a Blackman windowed spectrogram. We create (b) by selecting *Graph/Plot* from the spectrogram window where (a) is

displayed, and then clicking on the *Load* button to load the formula file `synthetic_oriole_whistle.uf2`, and then plotting this formula. (The sound signal corresponding to this spectrogram is generated by selecting *Graph/Inverse*. It should be played at the same sampling rate and bit-rate as the oriole whistle.) For (c), you select *Graph/Restore* from the spectrogram menu, then proceed as for (b) except that you plot the formula from the file:

```
a_synthetic_bird_song.uf2
```

5.9.1$^{\mathbf{c}}$ Model the spectrogram of `house_wren_chirp.wav` and compute a synthesized mimic of this bird call.

5.9.2$^{\mathbf{c}}_{\mathbf{s}}$ Compute a spectrogram of the recorded sound `warbler.wav`. Analyze this spectrogram for its musical qualities using the principles discussed in the previous section.

5.9.3$^{\mathbf{c}}$ Model the spectrogram of `warbler.wav` and compute a synthesized mimic of this bird song.

5.9.4$^{\mathbf{c}}_{\mathbf{s}}$ Selectively amplify the harp glissando in the signal `original clip from Firebird Suite.wav`.

5.9.5$^{\mathbf{c}}$ Using the Multiresolution Principle, synthesize your own bird song and/or musical passage.

Section 5.10

Example 5.10.1 [Figure 5.18] To perform the denoising illustrated in the figure we proceeded as follows. First, we graphed the following multiple of the chirp formula given in the text:

```
c sin(8192 (pi/3)x^3) \c = 7/.695750610959927
```

over the interval $[0, 1]$ using 8192 points. The constant c is chosen so that the standard deviation of the values of this signal is 7 (that is the standard deviation set by Donoho for the test signals he created, which are widely used in benchmarking denoising algorithms). After graphing this chirp signal, we then added Gaussian random noise of standard deviation $\sigma = 1$ to it by plotting the formula

```
c sin(8192 (pi/3)x^3) + rang(0)
\c = 7/.695750610959927
```

The MSE for the noisy chirp compared to the original chirp is then calculated by choosing *Norm difference* and selecting the *Power norm* option with power 2. That gives the Root Mean Square (RMS) error, the MSE is the square of that RMS value.

The Blackman windowed spectrogram in (a) was then created from this noisy chirp signal. Its thresholded spectrogram in (b) was created by choosing *Graph/Denoise* from the spectrogram menu and plotting the formula that FAWAV automatically supplies. [The theory that explains how that denoising formula is created is discussed in the paper *Denoising Gabor transforms* (reference [9] in Chapter 5 of the Primer).] To get the MSE error for denoising, you then select *Graph/Invert* from the spectrogram menu and copy and paste the resulting signal in to the window containing the original chirp signal, and square the RMS error (computed between graphs 1 and 3).

Example 5.10.2 [Figure 5.19] To create the graph shown in (a) you plot the formula `bumps.uf1`, available from the `Exercise_formulas.zip` archive from the Primer webpage. The plotting is done over the interval $[0, 1]$ using 8192 points. You then calculate that signal's standard deviation (which my computer gave as 1.53952824076645) and plot the following signal

```
7g1(x)/c \c = 1.53952824076645
```

After removing graph 1 (right-click on the graph region and select *Remove graph* and specify graph 1 for removal), you are left with the plot of Donoho's test function *Bumps* shown in (a). The noisy signal in (b) is plotted as in the previous example (using `g1(x)+rang(0)`). The denoising in (c) is obtained by selecting *Graph* and then *Denoise (Gabor)* and specifying graph 2. A spectrogram window opens up with a formula for plotting the thresholded spectrogram

(it is not automatically computed because you are able to modify the formula for more advanced denoising such as garotte shrinkage, see Example 5.10.4 below). After plotting the thresholded spectrogram and selecting *Graph/Invert* you obtain the denoised signal shown in (c). The MSEs were computed as described in the previous example.

Example 5.10.3 [Figure 5.20] To produce the spectrograms in Figure 5.20 you load the sound file

```
oriole_song.wav
```

and compute its Blackman windowed Gabor transform to get (a), and then select *Graph/Denoise* to plot the thresholded spectrogram (b). As in previous examples, the denoised signal was produced by then selecting *Graph/Invert*. We then saved this signal as `oriole_song_denoised.wav` by right-clicking on it and selecting *Save/Sound file*.

Example 5.10.4 [Figure 5.21] To obtain the spectrogram in (a) you load the audio file `noisy_thrush.wav` and plot a Blackman windowed spectrogram. If you select *Graph/Denoise* then the automatically generated formula for thresholding is

```
(g1)(g1 > c)
\c=.55*sqr(2log(1024))*(187.910060820749)*sqr(1024)
```

If you plot this formula and then select *Graph/Invert* you will produce a denoising that suffers from high-pitched artifacts and low-pitch rumbling and thumping. The garotte shrinkage shown in (b) is performed by selecting *Graph/Restore*, followed by *Graph/Denoise*, and then modifying the automatically generated function to obtain the following formula:

```
(g1)(g1 > c)(1-(c/g1)^2)
\c=.55*sqr(2log(1024))*(187.910060820749)*sqr(1024)
```

Plotting this formula produces (b). If you then select *Graph/Invert* you will create a sound signal that no longer suffers from high-pitched artifacts, but still has low-pitch rumbling and thumping. These latter noises are removed by a high-pass filtering, illustrated in (c). To obtain the spectrogram in (c), you select *Graph/Restore* from the spectrogram's menu, followed by *Graph/Denoise*. You then modify the automatically generated formula to obtain:

```
(g1)(g1 > c)(1-(c/g1)^2)(y > 1300)
\c=.55*sqr(2log(1024))*(187.910060820749)*sqr(1024)
```

Plotting this formula produces (c). If you then select *Graph/Invert* you will create a sound signal that is relatively noise-free, no longer suffering from either high-pitched artifacts or low-pitch rumbling and thumping.

Example 5.10.5 In this example we consider a challenging denoising of a real audio signal. The noisy recording is the audio file:

```
Chinese_Folk_Music.wav
```

We show its Blackman windowed spectrogram in Figure 11(a). Its thresholded spectrogram, is graphed by plotting the following function (obtained by selecting *Graph/Denoise* from the spectrogram menu):

```
(g1)(g1 > c)
\c=.55*sqr(2log(1024))*(355.195793541264)*sqr(1024)
```

See Figure 11(b). All of the noise has been removed. Unfortunately, however, some signal values are lost (especially in the higher frequencies). When the denoised signal obtained from the thresholded Gabor transform is played it sounds "muddy" due to loss of some of the high frequency content of the music.

To fix this problem, which is due to the rather high estimate of the standard deviation ($\sigma \approx 355$), we need a different estimate of the standard deviation. We find this new estimate as follows. By examining the spectrogram in Figure 11(a) we see that there is a region of the time-frequency plane, shown in Figure 11(c), that is mostly noise. To obtain the plot in Figure 11(c) we used the following function:

```
g1(5.302<x<5.746)(2680<y<3263)
```

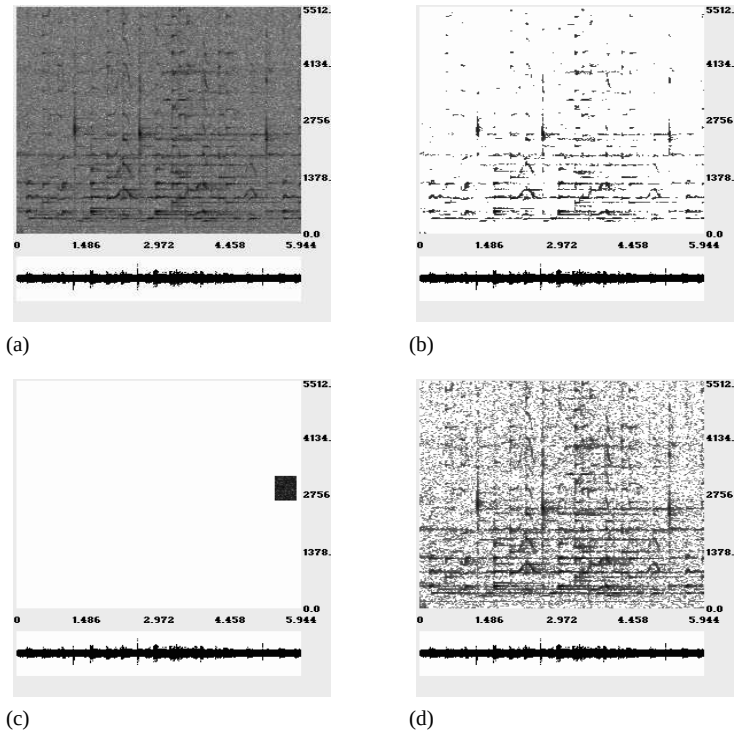


Figure 11
Denoising Chinese folk music. (a) Noisy signal spectrogram. (b) Spectrogram of thresholded Gabor transform. (c) Small region of just noise from (a). (d) Garrote shrinkage applied to Gabor transform for (a) with a new standard deviation.

After performing an inverse Gabor transform, we then changed the X-range to $5.302, 5.746$ and calculated the standard deviation of the noisy signal. We got a new estimate of the standard deviation: $\sigma \approx 93$. Returning to the spectrogram window and restoring the noisy spectrogram of the full signal, we then plotted the following garotte shrinkage

```
(g1)(g1 > c)(1-(c/g1)^2)
\c=.55*sqr(2log(1024))*(93)*sqr(1024)
```

which makes use of this new standard deviation. The resulting plot is shown in Figure 11(d). Although this spectrogram appears noisy, when it is inverted the resulting denoised signal sounds noise-free, and more sharply defined due to better preservation of the high-frequency content of the music. We have saved this denoised audio file as

```
denoised_Chinese_music_garotte.wav
```

5.10.1[c] Denoise the noisy recording `Chinese_Folk_Song_Clip_b.wav`.

5.10.2[c] Denoise the noisy recording `Dan's 'Bait'.wav`.

5.10.3[c] Denoise the noisy recording `bobolink.wav`. [Note: garotte shrinkage will produce a *Run error* due to division by 0 in `(g1>c)(1-(c/g1)^2)`. Use the modified expression `(g1>c)(1-(c/(g1+(g1<c/2))^2)` to avoid division by 0.]

Chapter 6

Section 6.1

Example 6.1.1 In this example we find the 2-level Walsh transform of the the signal $\mathbf{f} = (-2, -4, 2, 6, 8, 4, 4, 2)$. First, a 1-level Haar transform is computed:

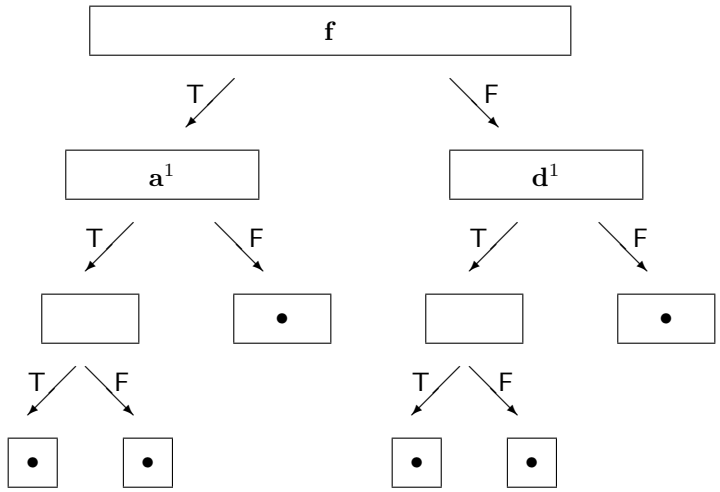
$$(\mathbf{a}^1 \,|\, \mathbf{d}^1) = (-3\sqrt{2}, 4\sqrt{2}, 6\sqrt{2}, 3\sqrt{2} \,|\, \sqrt{2}, -2\sqrt{2}, 2\sqrt{2}, \sqrt{2}).$$

Then we compute 1-level Haar transforms of both $\mathbf{a}^1$ and $\mathbf{d}^1$, obtaining

$$(1, 9 \,|\, -7, 3 \,|\, -1, 3 \,|\, 3, 1)$$

which is the 2-level Walsh transform of $\mathbf{f}$.

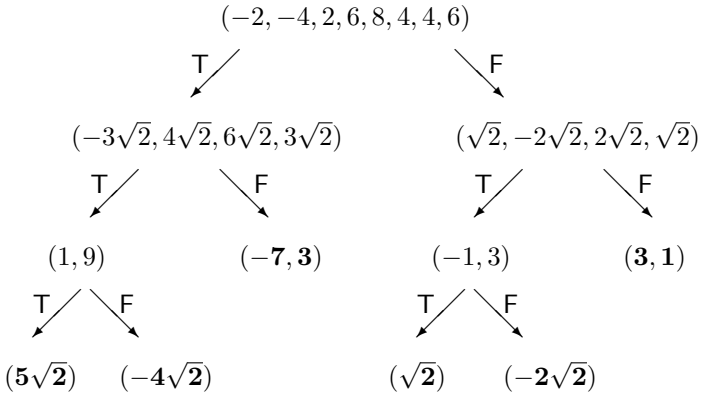
Example 6.1.2 [Other wavelet packet transforms] Wavelet packet transforms in general are defined as signals that result at the end nodes of a tree-diagram describing applications of a trend calculation T and a fluctuation calculation F. For example, consider the following tree diagram (where the symbol • indicates an end node signal):



which can also be expressed more succinctly as

$$(\mathsf{TTT}(\mathbf{f}) \,|\, \mathsf{FTT}(\mathbf{f}) \,|\, \mathsf{FT}(\mathbf{f}) \,|\, \mathsf{TTF}(\mathbf{f}) \,|\, \mathsf{FTF}(\mathbf{f}) \,|\, \mathsf{FF}(\mathbf{f}))\,.$$

For instance, if $\mathbf{f} = (-2, -4, 2, 6, 8, 4, 4, 6)$ then this wavelet packet transform is computed as follows (with the end-node signals given in bold-face):



and we can write this wavelet packet transformed signal as:

$$(5\sqrt{2}\,|\,-4\sqrt{2}\,|\,-7,3\,|\,\sqrt{2}\,|\,-2\sqrt{2}\,|\,3,1).$$

Example 6.1.3 In this example we show how to use FAWAV to compute a wavelet packet series of the kind described in the text, where each subsignal is transformed at every level, for a Coif30 wavelet. Suppose that our signal is obtained from plotting the formula

$$40\sin(12\pi x^2)$$

over the interval $[0, 4]$ using 4096 points. To compute a 6-level Coif30 wavelet packet series, transforming each subsignal at every level, we select *Series* and *Wavelet Packet* and then specify 6 levels and a Coif30 wavelet. If we also select *Energy fraction* and specify 0.9999, then FAWAV uses 250 transform values (which it reports as "250 coefficients") to obtain a wavelet packet approximation of the signal that captures 99.99% of the signal's energy. To see the advantage for this signal of computing a wavelet packet series, if we compute a 6-level Coif30 wavelet series with energy fraction 0.9999, then FAWAV uses 397 transform values, a far greater number.

6.1.1$_s$ Given $\mathbf{f} = (2, 4, 8, 6, 2, 4, 6, 8)$, find its 2-level Walsh transform.

6.1.2 Given $\mathbf{f} = (10, 8, 4, -2, 4, 8, 10, 18)$, find its 3-level Walsh transform.

6.1.3^c_s Given the signal obtained by plotting

$$\sin(24\pi x^2) - \sin(12\pi x^2)$$

over $[0, 2]$ using 4096 points. How many transform values (coefficients) are needed in a 6-level Coif30 wavelet packet series to capture 99.99% of the signal's energy? How many for a Coif30 wavelet series?

6.1.4^c Given the signal obtained by plotting

$$\sin[24\pi(2 - x)^2] + 4\sin(12\pi x^2)$$

over $[0, 2]$ using 4096 points. How many transform values (coefficients) are needed in a 6-level Coif30 wavelet packet series to capture 99.99% of the signal's energy? How many for a Coif30 wavelet series?

6.1.5 Given the signal

$$\mathbf{f} = (2, 4, 6, 8, 16, 20, 22, 22).$$

Compute its wavelet packet transform defined by

$$(\mathsf{TT}(\mathbf{f}), \mathsf{TFT}(\mathbf{f}), \mathsf{FFT}(\mathbf{f}), \mathsf{TF}(\mathbf{f}), \mathsf{TFF}(\mathbf{f}), \mathsf{FFF}(\mathbf{f}))$$

and draw the tree diagram corresponding to this transform.

6.1.6 Given the signal

$$\mathbf{f} = (2, 0, 4, 4, 4, 2, 4, 6, 8, 12, 12, 10, 8, 8, 8, 6).$$

Compute its wavelet packet transform defined by the following transforms applied to $\mathbf{f}$

$$\mathsf{TTT}, \mathsf{TFTT}, \mathsf{FFTT}, \mathsf{FT}, \mathsf{TTF}, \mathsf{TFTF}, \mathsf{FFTF}, \mathsf{TFF}, \mathsf{FFF}$$

and draw the tree diagram corresponding to this wavelet packet transform.

Section 6.2

Example 6.2.1 [Table 6.1] The values in the first row were obtained as follows. We began by loading the sound file `greasy.wav` and then selected *Series/Wavelet*. After specifying 4 levels, and a Coif18 wavelet, and choosing the *Threshold* option, we then clicked on the *Edit Settings* button. In the edit window that opens we selected the *Multiple Thresholds* option and entered `1/2^7:1/2^5` for the thresholds. After clicking *Apply* we then plotted the wavelet series. The entries *Sig. values* and *Bpp* were then read off from the report of the number of coefficients and the bits per point, respectively. The RMS error was obtained from selecting *Analysis/Norm difference* and using the default choices (Normalized, absolute Power 2 norm). Similar work was done to get the second row's values, except that we used the choice *Series/Wavelet packet.* Note: your results may differ slightly from ours due to differences in floating point arithmetic for various CPUs.

Example 6.2.2 [Table 6.2] To obtain the data in the first row of the table, we loaded the image `Barb.bmp` and performed a 4-level Daub 9/7 wavelet transform. We then right-clicked on the transform image and selected *Save graph/Graph.* We then saved our transform data to a file `Barb_4_tr.fb2` in the *Data* subdirectory of the FAWAV directory. After that, we selected *Image compression workshop* from the *Graph* menu, and proceeded as follows (Note: we go through this procedure of using *Image compression workshop* for consistency, since it is the only way that FAWAV has for performing wavelet packet compression.). Within the *Image compression workshop* window we specified 4 for the number of levels, and 0.5 for the target bpp rate, and then clicked the button labelled *Encode Transform* and selected the file `Barb_4_tr.fb2` for encoding. The *Sig. values* entry was then read off from the report generated by the encoding process when it completed its encoding at the rate of almost exactly the target rate of 0.5 bpp. We then clicked on the button *Decode Transform* and selected the file `Barb_4_tr.wic` from the *Compress/Data* subdirectory of the FAWAV directory. (Note: when selecting this file, you might first right-click on it and check its *Properties* to verify that it has a file size of 16 kB, which is a 16:1 compression of the 256 kB file `Barb.bmp`.) The decoded transform was saved to the file `Barb_4_tr_quant.fb2` in the *Compress/Data* subdirectory. Finally, we returned to the 2D-form containing the images and right-clicked on one of the images, followed by selecting *Load/Graph* and loaded the file `Barb_4_tr_quant.fb2`. After inverse transforming this image, and 8-bit quantizing the resulting image [by deleting one of the transform images, and then using *Graph/Quantize (8-bit)*], we had our decompressed image. We obtained the PSNR by selecting *Analysis/Norm difference* and choosing the *PSNR* option. The second row of the table was obtained by repeating this work, but using a wavelet packet transform and its inverse. The third and fourth rows were obtained in the same way as the first and second rows, except that 0.25 bpp was used as the target rate for compression.

Example 6.2.3 [Figure 6.1] The images in the Figure were obtained during the process of creating the decompressions of the 16:1 compressions of `Barb.bmp` described in the previous example, and zooming in twice on a region around the fold in Barb's scarf (using the same center pixel values for each zooming). The PSNRs computed by FAWAV are obtained just for the region selected in the zooms (using the first specified graph to determine the zoomed region, so be sure that you have zoomed on exactly the same pixel coordinates for each image).

Example 6.2.4 [Figure 6.2] The image in (a) was created by performing a 2-level Daub $9/7$ transform of the image `Barb.bmp` and then plotting

```
g2 a + 950(abs(g2)>8)(1-a)
+ 475(abs(g2)<8)(1-a)
\a = (x<c)(y<c)\c = -.5 + 1/2^2
```

The value 950 is the maximum magnitude for the transformed image (obtained using *Analysis/Statistics*), so this formula displays the brightest white for transform values that exceed 8 in magnitude, and a dull gray for those which do not, and it leaves the trend image values unchanged from their original values. The image in (b) was produced in the same way, except a wavelet packet transform was used.

6.2.1$^{\mathbf{c}}_{\mathbf{s}}$ Compute 5-level Coif30 compressions of `alfalfa_22050.wav` using both wavelet and wavelet packet transforms, and complete a table of data similar to Table 6.1. Can you hear any differences between the compressed

signals and the original?

6.2.2^{c} Compute 5-level Coif30 compressions of `denoised_thrush.wav` using both wavelet and wavelet packet transforms, and complete a table of data similar to Table 6.1.

6.2.3^{c} Compute 5-level Coif30 compressions of `Call(Jim).wav` using both wavelet and wavelet packet transforms, and complete a table of data similar to Table 6.1.

6.2.4$^{c}_{s}$ The sound file `noisy_osprey_clip.wav` is a noisy version of `osprey_clip.wav`. Compute RMS errors for threshold denoisings with both wavelet and wavelet packet transforms. How much reduction of RMS error is obtained through this denoising? Which denoising sounds better, and why?

6.2.5^{c} Load `Call(Jim).wav` and simulate a noisy version by plotting

```
g1(x) + rang(0)
```

Perform threshold denoising using a wavelet transform and a wavelet packet denoising. Which denoising has smaller RMS, and which sounds better? Explain why.

6.2.6$^{c}_{s}$ Produce a table like Table 6.2, but use the `Boat.bmp` image.

6.2.7 Produce a table like Table 6.2, but use the `Airfield.bmp` image.

6.2.8 Produce a table like Table 6.2, but use the `Peppers.bmp` image.

6.2.9 Compare J2K, ASWDR, and WSQ methods for compressing `fingerprint_1.bmp` at $10:1$ compression ratio. By compare we mean in terms of PSNR and subjective visual inspection of both the whole images, and zooms about the coordinates $(256, 256)$.[6] Which method produces the best results?

6.2.10^{c} Repeat the previous exercise, but use 20:1 compression.

6.2.11^{c} Repeat Exercise 6.2.9, but use the `Fingerprint_2.bmp` image and $(128, 128)$ as zoom coordinates.

6.2.12^{c} Repeat Exercise 6.2.10, but use the `Fingerprint_2.bmp` image and $(128, 128)$ as zoom coordinates.

Section 6.3

Example 6.3.1 [Figure 6.3] The graph in (a) was obtained by plotting

```
2pi(1-2pi(u/w)^2)e^{-pi(u/w)^2}a/w
\w=1/16\u=ax\a=2^(m/6)
\m=0
```

over the interval $[-0.5, 0.5]$ using 4096 points. The top graph in (b) was then obtained by selecting *Transform/Fourier,* clicking the *Plot* button, and retaining only the first graph plotted (the real part of the DFT, since the imaginary part is essentially zero) and changing the X-interval to $[-75, 75]$. The rest of the graphs in (b) were obtained by successively plotting the formula above (changing `\m=0` to `\m=2`, `\m=4`,..., `\m=8`), and then plotting Fourier transforms.

Example 6.3.2 [Figure 6.4] The image in (a) was obtained by plotting

```
sin(40pi x)e^{-100pi (x-.2)^2}+
[sin(40pi x)+2cos(160pi x)]e^{-50pi (x-.5)^2}
+ 2sin(160pi x)e^{-100pi(x-.8)^2}
```

[6] The J2K compression can be done with IMAGEANALYZER, and its decompression saved as a `bmp` file for computing PSNR with FAWAV. Similarly, the WSQ compression can be done with WSQ VIEWER.

over the interval $[0, 1]$ using 2048 points, and then selecting *Scalogram/Analysis* and computing a Mexican hat scalogram with *Width* specified as $1/16$. The image in (b) was obtained by plotting

```
sumk(100u(u-.1)(u+.2)(-.2<u<.1)(abs(k-1)>.5))
+.1sin(12pi v)(-.2<v<.1)
\u = x-2k \k=-2,2 \v=x-2
```

over the interval $[-5, 5]$ using 1024 points, changing the plotting style to *Lines*, and then computing a Mexican hat scalogram with *Width* specified at 2.

6.3.1$^{\mathbf{c}}_{\mathbf{s}}$ Compute Blackman windowed spectrograms for the following three shifted and scaled Mexican hat wavelets:

```
2pi(1-2pi(u/w)^2)e^{-pi(u/w)^2}a/w \w=1/16\u=a(x+.25)\a=2^(m/6) \m=8

2pi(1-2pi(u/w)^2)e^{-pi(u/w)^2}a/w \w=1/16\u=ax\a=2^(m/6) \m=32

2pi(1-2pi(u/w)^2)e^{-pi(u/w)^2}a/w \w=1/16\u=a(x-.25)\a=2^(m/6) \m=16
```

over the interval $[-0.5, 0.5]$ using 1024 points [and displaying the spectrograms with *Display style* setting *Linear (global)*]. How do these spectrograms relate to the time and frequency decomposition given by a Mexican hat CWT?

6.3.2$^{\mathbf{c}}$ For the following test signal

```
[sin(80pi x)-cos(40pi x)]e^{-100pi (x-.2)^2}+
[sin(160pi x) + cos(80pi x)]e^{-50pi (x-.5)^2}
+ sin(80pi x)e^{-100pi(x-.8)^2}
```

graphed over $[0, 1]$ using 4096 points, plot its Mexican hat scalogram using 6 octaves, 42 voices, and a width of 0.05. Explain the relationship between the features of the scalogram and the frequencies of the sine and cosine functions in the signal's formula.

6.3.3$^{\mathbf{c}}$ For the following simulated ECG signal

```
sumk(100u(u-.1)(u+.2)(-.2<u<.1)(abs(k+1)>.5))
+.1sin(16pi v)(-.2<v<.1)
\u = x-2k \k=-2,2 \v=x+2
```

graphed over the interval $[-5, 5]$ using 8192 points, plot its Mexican hat scalogram using 8 octaves, 16 voices, and width 2.

6.3.4$^{\mathbf{c}}$ For the following simulated ECG signal

```
sumk(100u(u-.1)(u+.2)(-.2<u<.1))
+.2sin(16pi v)(-.2<v<.1)
\u = x-2k \k=-2,2 \v=x+2
```

graphed over the interval $[-5, 5]$ using 8192 points, plot its Mexican hat scalogram using 8 octaves, 16 voices, and width 2.

Section 6.4

Example 6.4.1 [Figure 6.5] The image in (a) was produced by graphing the same function as in Example 6.3.2, then selecting *Analysis/Scalogram*, and choosing a *Gabor (complex)* scalogram. The scalogram was plotted using 8 octaves and 16 voices, and a width parameter of 1 and freq. parameter of 5. The image in (b) was obtained by computing a Blackman windowed spectrogram of the signal.

Example 6.4.2 [Figure 6.6] The image in (a) was obtained by loading the sound file `Call(Jim).wav` and computing a Blackman windowed spectrogram. The image in (b) was computed using a *Gabor (complex)* scalogram using 4 octaves and 16 voices, and a width parameter of $1/8$ and freq. parameter of 10.

Example 6.4.3 [Figure 6.7] The graph in (a) was obtained by loading the sound file `Call(Jim).wav`, then changing the X-interval to $.09, .09 + .371519274376417/2$, and then clipping out the displayed graph (right-clicking and selecting *Clip* from the popup menu). The spectrum in (b) was obtained from (a) by selecting *Transforms/Fourier,* and choosing the options *Amp/Phase* and interval type $[0, L] \to [-A, A]$. Removing the phase graph (graph 2) from the resulting transform, and changing the X-interval to $[0, 992]$ and Y-interval to $[-1, 3]$.

Example 6.4.4 [Figure 6.8] The image in (a) was obtained by computing a Blackman windowed spectrogram of the sound file

`Buenos_aires_Madonna_lyrics.wav`

The image in (b) was obtained by computing a scalogram of type *Gabor (complex)* using 3 octaves and 85 voices, and a width parameter of 0.1 and freq. parameter of 20, and changing the display style to *Log (global).*

6.4.1[c] Compute spectrograms and scalograms for the word "call" from each of the following 10 recordings:

`call back 1.wav, call back 2.wav, ... , call back 10.wav.`

Can you formulate any conjectures about formants of various speakers (e.g. male/female, or Native English/Foreign, etc.)?

6.4.2 Do a time-frequency analysis of the sound clip `chaiya chaiya clip.wav`. Does the Multiresolution Principle apply here? [The spectrogram is best viewed with AUDACITY. The *Gabor (complex)* scalograms from FAWAV, will need to be constructed from clips of the audio file using the following settings: 3 Octaves, 32 voices, width 0.05, and freq. 10.]

Section 6.5

Example 6.5.1 [Figure 6.9] The spectrogram in part (a) of the figure was generated by loading the sound file `el_matador_percussion_clip.wav` and then selecting *Analysis/Spectrogram* and performing a Blackman windowed spectrogram with the default settings. To produce (b) the sound file

`Buenos Aires percussion clip.wav`

was processed in a similar way.

Example 6.5.2 [Figure 6.10] The processed spectrogram at the top of the figure was produced from the spectrogram shown in Figure 6.9(a) of the Primer (see the example just discussed) by selecting *Graph/Plot* and then plotting the formula

`g1(2500 < y < 4500)`

The pulse train shown at the bottom was then generated by selecting *Graph* and then choosing *Percussion scalogram.*

Example 6.5.3 [Figure 6.11] The percussion scalogram was generated from the pulse train generated by the method described in the previous example. You select a *Gabor (complex)* type of scalogram and then enter the following data:

`Octaves:  4` `Voices:  64`

`Width:  0.5` `Freq.:  0.5`

and plot the scalogram.

Example 6.5.4 [Figure 6.12] The percussion scalogram in this figure was generated in the following way. First, the Blackman-windowed spectrogram of the sound file `Buenos Aires percussion clip.wav` was produced. Second, from the spectrogram menu select *Graph/Plot* and plot the following function:

```
g1(2000 < y < 3000)
```

Third, select *Graph/Percussion scalogram* from the spectrogram menu. Finally, in the scalogram form that opens up, you specify a *Gabor (complex)* type of scalogram, enter the following data

```
Octaves:  5          Voices:  51

Width:  2            Freq.:  1
```

and plot the scalogram.

6.5.1[c] Use the percussion scalogram method to analyze the rhythm in the audio file `Conga_solo_clip.wav`.

6.5.2[c] Use the percussion scalogram method to analyze the rhythm in the audio file `brazil_clip.wav`.

6.5.3[c] Use the percussion scalogram method to analyze the rhythm in the audio file `brazil_medley_clip.wav`.

Solutions to selected exercises

Chapter 2

2.1.1 (a) $(\mathbf{a}^1 \,|\, \mathbf{d}^1) = (3\sqrt{2}, 6\sqrt{2}, 3\sqrt{2} \,|\, -\sqrt{2}, 0, \sqrt{2})$ (c) $(\mathbf{a}^1 \,|\, \mathbf{d}^1) = (1.5\sqrt{2}, 3\sqrt{2}, 2\sqrt{2}, \sqrt{2} \,|\, -0.5\sqrt{2}, 0, 0, 0)$

2.1.2 (a) $\mathbf{f} = (2, 2, 0, -2, 3, 3, 0, -2)$ (c) $\mathbf{f} = (5, 1, 1, 3, 3, 1, 0, 0)$

2.1.3 (a) $\widetilde{\mathbf{f}} = (2, 2, 3, 3, 4.5, 5.5, 6, 6)$, largest error = 0.5. (c) $\widetilde{\mathbf{f}} = (0, 0, -2, -2, -1, -1, 2, 2)$, largest error = 2.

2.1.5 (b) The original signal and its 1-level Haar transform are shown in Figure 12.

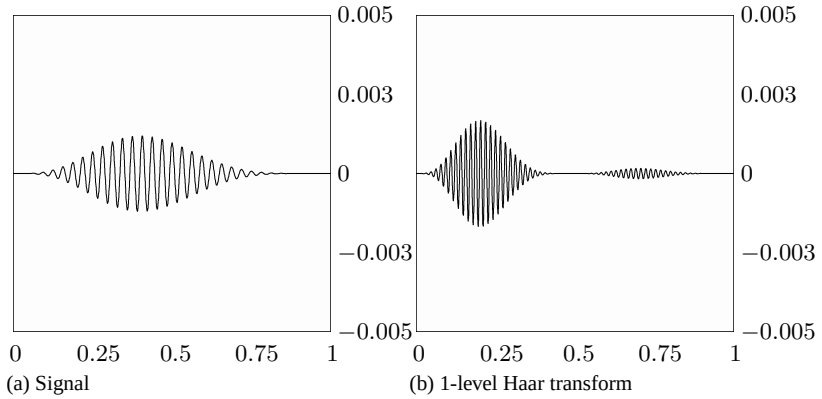


Figure 12
Solution to 2.1.5(b).

2.2.1 (a) $\mathcal{E}_{\mathbf{a}^1} = 108$, $\mathcal{E}_{\mathbf{d}^1} = 4$, $\mathcal{E}_{\mathbf{f}} = 112 = 108 + 4$. (c) $\mathcal{E}_{\mathbf{a}^1} = 32.5$, $\mathcal{E}_{\mathbf{d}^1} = 0.5$, $\mathcal{E}_{\mathbf{f}} = 33 = 32.5 + 0.5$.

2.2.3 (a) 12.5% (c) 37.5%

2.2.5 (b) $\mathcal{E}_{\mathbf{a}^1} = 1.91685\cdots \times 10^{-4}$, $\mathcal{E}_{\mathbf{d}^1} = 1.86057\cdots \times 10^{-6}$, $\mathcal{E}_{\mathbf{f}} = 1.935457 \times 10^{-4} = \mathcal{E}_{\mathbf{a}^1} + \mathcal{E}_{\mathbf{d}^1}$ to an accuracy slightly better than 6×10^{-19}.

2.2.7 (b) The 1-level, 2-level, and 3-level transforms are respectively:

$$(\mathbf{a}^1 \,|\, \mathbf{d}^1) = (-16\sqrt{2}, 8\sqrt{2}, 48\sqrt{2}, 96\sqrt{2} \,|\, 0, -24\sqrt{2}, 0, 0)$$
$$(\mathbf{a}^2 \,|\, \mathbf{d}^2 \,|\, \mathbf{d}^1) = (-8, 72 \,|\, -24, -48 \,|\, 0, -24\sqrt{2}, 0, 0)$$
$$(\mathbf{a}^3 \,|\, \mathbf{a}^2 \,|\, \mathbf{d}^2 \,|\, \mathbf{d}^1) = (32\sqrt{2} \,|\, -40\sqrt{2} \,|\, -24, -48 \,|\, 0, -24\sqrt{2}, 0, 0).$$

2.3.1 (a) -2 (c) 8

2.3.5 $(0.5\sqrt{2}, 0.5\sqrt{2}, 0, 0, \ldots, 0)$ and $(0.5\sqrt{2}, -0.5\sqrt{2}, 0, 0, \ldots, 0)$

2.4.1 (a) $\mathbf{f} + \mathbf{g} = (3, 5, 1, 7)$, $\mathbf{f} - \mathbf{g} = (1, 1, 3, 1)$, $3\mathbf{f} = (6, 9, 6, 12)$, $-2\mathbf{g} = (-2, -4, 2, -6)$
(c) $\mathbf{f} + \mathbf{g} = (0, 1, 1, 2, 0, 2)$, $\mathbf{f} - \mathbf{g} = (-2, -3, 1, 0, 2, 0)$, $3\mathbf{f} = (-3, -3, 3, 3, 3, 3)$, $-2\mathbf{g} = (-2, -4, 0, -2, 2, -2)$

2.4.2 (a) $\mathbf{A}^1 = (3, 3, 6, 6, 3, 3)$, $\mathbf{D}^1 = (-1, 1, 0, 0, 1, -1)$
(c) (a) $\mathbf{A}^1 = (1.5, 1.5, 3, 3, 2, 2, 1, 1)$, $\mathbf{D}^1 = (-0.5, 0.5, 0, 0, 0, 0, 0, 0)$

2.4.4 (a) $\mathbf{A}^1 = (1.5, 1.5, 1, 1, 3, 3, 3.5, 3.5)$, $\mathbf{D}^1 = (0.5, -0.5, 2, -2, -1, 1, -0.5, 0.5)$
(c) $\mathbf{A}^1 = (1.5, 1.5, -1, -1, 3.5, 3.5, 2, 2)$, $\mathbf{D}^1 = (-0.5, 0.5, 0, 0, 0.5, -0.5, 0, 0)$

2.4.6 (a) $\mathbf{A}^2 = (1.25, 1.25, 1.25, 1.25, 3.25, 3.25, 3.25, 3.25)$, $\mathbf{D}^2 = (.25, .25, -.25, -.25, -.25, -.25, .25, .25)$
(c) $\mathbf{A}^2 = (.25, .25, .25, .25, 2.75, 2.75, 2.75, 2.75)$, $\mathbf{D}^2 = (1.25, 1.25, -1.25, -1.25, 0.75, 0.75, -0.75, -0.75)$

2.5.1 The graphs are shown in Figure 13 below, 52 transform values were used to produce the signal in Figure 13(d), hence a $19.7:1$ compression ratio. Any threshold less than 0.5 will produce at least 99.99% of energy, using 0.49 we obtained a signal that has a maximum error of 0.125. [Note: since the signal is integer-valued, this error could be completely eliminated by rounding to nearest integer.]

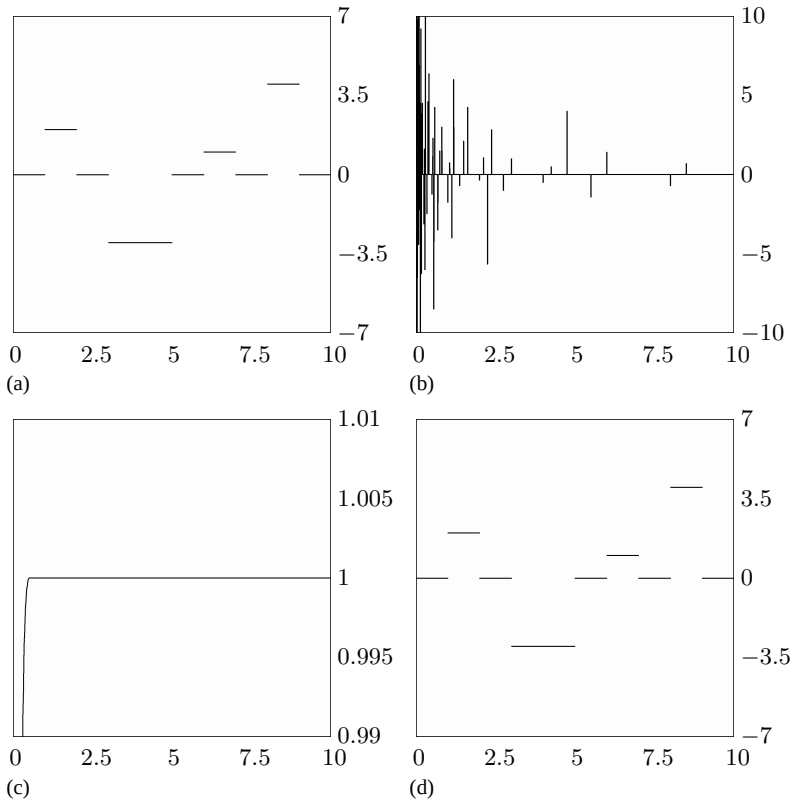


Figure 13
(a) Signal(b) 10-level Haar transform, (c) energy map of Haar transform, (d) 19.7:1 compression of Signal, 100% of energy.

2.5.2 The graphs are shown in Figure 14 below. To get 100% energy requires *all* of the coefficients (no compression) so we did not graph it. Instead, we found that a threshold of 1.245 will retain 65 transform values (a compression ratio of 15.8:1) and produces the graph shown in Figure 14(d). The maximum error between this signal and the original signal is 0.604.

2.5.3 For (a) we get a sup-norm difference of 0.74, while for (b) we get 2.66×10^{-15}. Clearly the series for (b) performs the best. The reason is that the function in (b) is a step function so all of the Haar transform values are 0 except for a small number corresponding to Haar wavelets whose supports overlap the jumps in the step function. Those relatively few high-magnitude coefficients are all included by specifying the highest 50, hence the extremely small error (due to the roundoff error that always occurs with digital calculations). The function in (a) is not a step function so it has many more non-zero transform values.

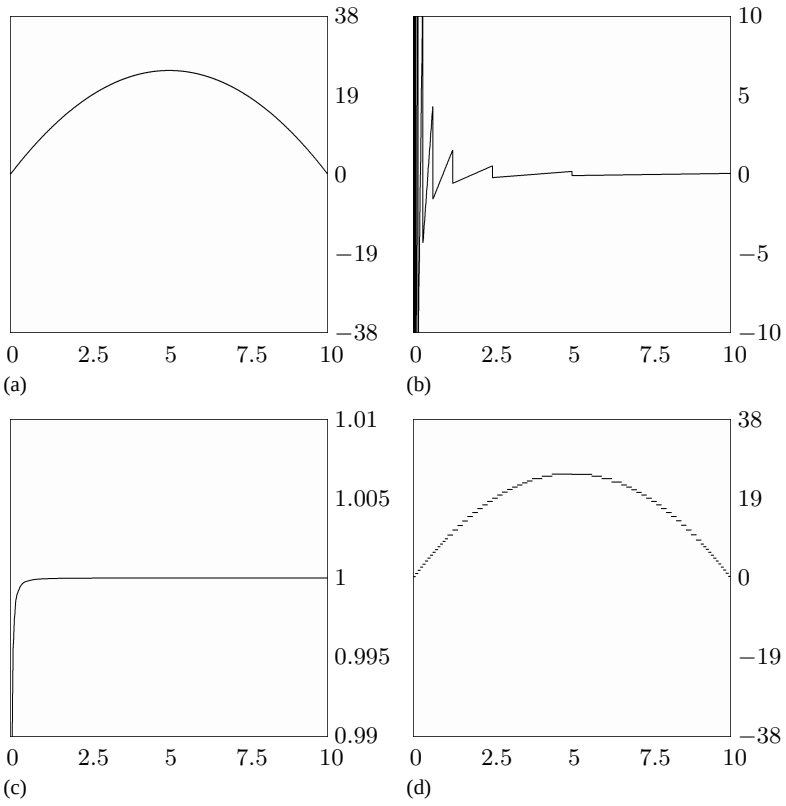


Figure 14
(a) Signal(b) 10-level Haar transform, (c) energy map of Haar transform, (d) 15.8:1 compression of Signal, 99.99% of energy.

2.6.2 It looks like a sequence of random numbers (just as the signal from problem 2.6.1 looks random) and, again just like the signal from 2.6.1, it sounds like the static one hears in radio transmissions.

2.6.3 Noisy signals are shown for parts (a) and (d) in Figure 15. To denoise these signals a threshold of 35 was used in both cases. The denoised signals are shown in the Figure. The denoising for the step function in (a) was the best (most representative of the original signal). The main reason is that the underlying signal for (a) is a step function which is best for Haar transforms.

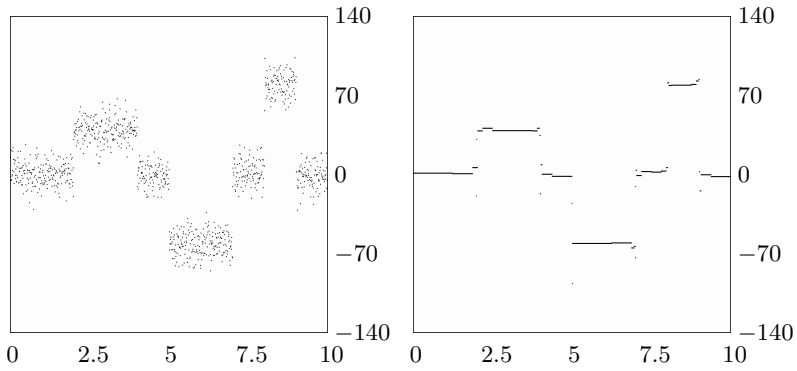
Chapter 3

3.1.1 $\mathbf{V}_1^2 = \alpha_1\mathbf{V}_1^1 + \alpha_2\mathbf{V}_2^1 + \alpha_3\mathbf{V}_3^1 + \alpha_4\mathbf{V}_4^1$ where each V_k^1, $k = 2, 3, 4$, is a translate of $\mathbf{V}_1^1$ by $2*(k-1)$ time-units. It follows that V_4^1 is a translate by 6 units with a support of 4 units, and therefore $\mathbf{V}_1^2$ has a support of 10 time-units. $\mathbf{V}_2^2 = \alpha_1\mathbf{V}_3^1 + \alpha_2\mathbf{V}_4^1 + \alpha_3\mathbf{V}_5^1 + \alpha_4\mathbf{V}_6^1$ and it therefore has a support of 10 time-units. Because its support begins with the support of $\mathbf{V}_3^1$, which is a translation of $\mathbf{V}_1^1$ by 4 units, it follows that V_2^2 is a translation of V_1^2 by 4 units. Similar calculations show that, in general, $\mathbf{V}_m^2$ has a support of 10 time-units and is a translation of $\mathbf{V}_1^2$ by $4(m-1)$ time-units.

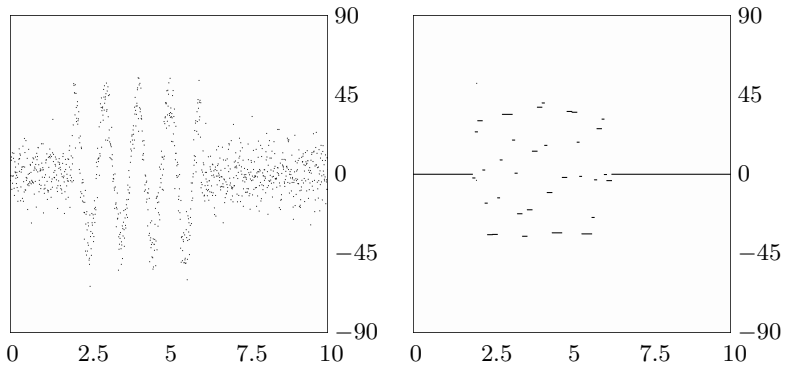
3.1.3 No, it equals $0.896575\ldots$

3.1.5 The original signal for (b) and its 1-level Daub4 transform are shown in Figure

3.1.7 Maximum errors: $\mathbf{A}^1:\ 2.259\times10^{-4}$, $\mathbf{A}^2:\ 1.389\times10^{-3}$, $\mathbf{A}^3:\ 5.877\times10^{-3}$, $\mathbf{A}^4:\ 1.572\times10^{-2}$.

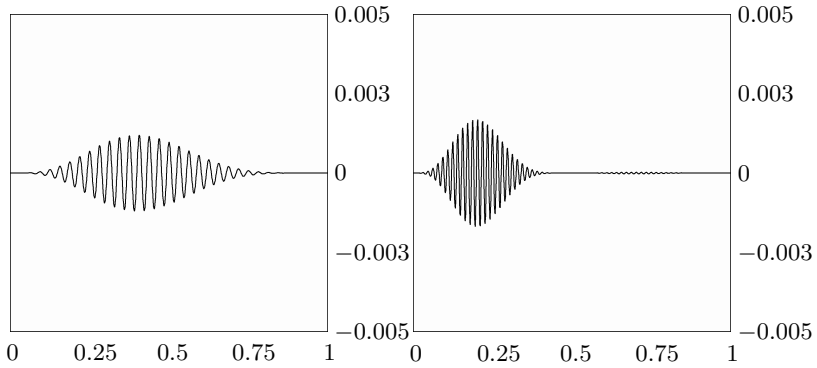


(a) Noisy signal for part (a) (b) Denoising



(c) Noisy signal for part (d) (d) Denoising

Figure 15
Denoising for Exercise 2.6.3.



(a) Signal (b) 1-level Daub4 transform

Figure 16
Solution to 3.1.5(b).

3.1.10 The graphs are shown in Figure 17 below. A threshold of 3.5 retains 17 transform values (a compression ratio of 60:1) and produces the graph shown in Figure 14(d). The maximum error between this signal and the original signal is 0.9357. [Note: Using 65 transform values, as with the Haar case, we obtained a maximum error of 0.0874, which is several times smaller than the maximum error of 0.604 for the Haar case.]

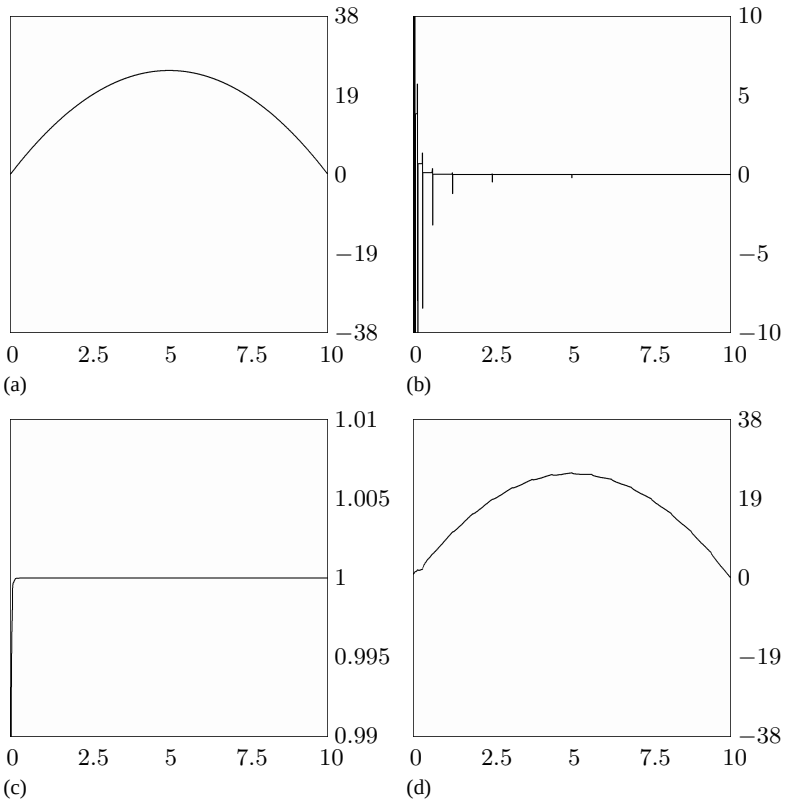


Figure 17
(a) Signal(a) 10-level Daub4 transform, (c) energy map of Daub4 transform, (d) 60:1 compression of Signal, 99.99% of energy.

3.2.2 We found that 1334877.16 equals both the energy of $\mathbf{f}$ and its 3-level Daub4 transform. [Note: energies of signals are found by selecting *Analysis/Statistics* and computing statistics (which includes the energy) for the desired signal.]

3.2.4 We obtained the following results

	Daub4	*Haar*
(a)	75.6%	72.1%
(d)	85.4%	86.7%

3.3.1 We have

$$g(t_{2m-1+k}) = g(t_{2m-1}) + g'(t_{2m-1})(kh) + g"(t_{2m-1})(k^2h^2) + \mathrm{O}(h^3).$$

Hence,

$$\begin{aligned}\mathbf{f}\cdot\mathbf{W}_m^1 &= g(t_{2m-1})\{\beta_1+\beta_2+\beta_3\cdots+\beta_6\}\\ &\quad + g'(t_{2m-1})\{0\beta_1+1\beta+2+2\beta_3+\cdots+5\beta_6\}\\ &\quad + g"(t_{2m-1})\{0^2\beta_1+1^2\beta_2+2^2\beta_3+\cdots+5^2\beta_6\}+\mathrm{O}(h^3)\\ &= \mathrm{O}(h^3).\end{aligned}$$

3.3.3 The maximum errors are $\mathbf{A}^1 : 3.0078 \times 10^{-5}$, $\mathbf{A}^2 : 3.5018 \times 10^{-4}$, $\mathbf{A}^3 : 2.7638 \times 10^{-3}$, $\mathbf{A}^4 : 1.4618 \times 10^{-2}$.

3.3.9 The mininum number of terms for 99.99% of energy are

Levels:	1	2	3	4	5	6
Daub4:	335	192	162	161	161	160
Daub6:	334	167	99	90	90	92
Daub8:	333	167	85	68	70	70

3.3.12 The maximum errors are $\mathbf{A}^1 : 3.0078 \times 10^{-5}$, $\mathbf{A}^2 : 3.5018 \times 10^{-4}$, $\mathbf{A}^3 : 2.7638 \times 10^{-3}$, $\mathbf{A}^4 : 1.4618 \times 10^{-2}$.

3.3.12 The mininum number of terms for 99.99% of energy are

Levels:	1	2	3	4	5	6
Coif6:	334	188	160	156	155	157
Coif18:	333	168	85	52	55	54
Coif30:	333	168	85	42	44	38

3.3.16 The Coif$J$ maximum errors are: *Coif6:* 2.4296×10^{-7}, *Coif12:* 2.4842×10^{-10}, *Coif18:* 5.5743×10^{-13}, *Coif24:* 3.8137×10^{-13}, *Coif30:* 4.1636×10^{-13}. The Duab$J$ maximum errors are: *Daub4:* 3.3445×10^{-3}, *Daub6:* 4.3118×10^{-3}, *Daub8:* 5.3033×10^{-3}, *Daub10:* 6.2972×10^{-3}, *Daub12:* 7.2894×10^{-3}, *Daub14:* 8.2789×10^{-3}, *Daub16:* 9.2653×10^{-3}, *Daub18:* 1.0249×10^{-2}, *Daub20:* 1.1230×10^{-2}.

3.4.1(d) The graphs for part (d) are shown in Figure 18. The threshold used was 2.0, which retained only 19 coefficients of the transform, hence a $1024 : 19 \approx 54 : 1$ compression ratio. The maximum error was 0.4411.

3.4.5(d) With 6 levels there was a minimum number of 17 transform values.

3.4.7(d) With 7 levels there was a minimum number of 9 transform values.

3.5.2 The entropy is $\displaystyle\sum_{k=1}^{N} p_k \log_2(1/p_k) = \sum_{k=1}^{N} (1/N) \log_2 N = \log_2 N$.

3.5.4 Using the rule-of-thumb of $(\text{entropy}) + 0.5$ we obtained these estimates for the signal `alfalfa_2.wav` (which is available from the Exercises webpage of the FAWAV website): Original signal: ≈ 11.9 bpp. 13-level Coif30 transform (dead-zone histogram): ≈ 10.3 bpp. 4-level Coif30 transform (with different quantization on the trend and fluctuation): ≈ 7.7 bpp.

3.5.6 For problem 3.5.4, using the signal `alfalfa_2.wav` we obtained these errors.

	Sup-norm difference	Rel. 2-Norm difference
13-level (Uniform threshold Value `1/2^16`)	0.622	5.66×10^{-5}
4-level (Multiple thresholds Value `1/2^16, 1/2^12`)	11.68	1.02×10^{-3}

Note: for both compressions the sound of the signals was indistinguishable from the original recording.

3.6.1(b) The signal and denoised signal (using a threshold of 16) are shown in Figure 19. This was the best of the denoisings, due to the lack of any large jumps in the signal values. (The denoising for (d) with a threshold of 16 was the next best.)

3.6.4 The RMS error (2-norm difference) for the noisy signal was 501.87 while for the denoised signal it was 379.60, a 24% reduction.

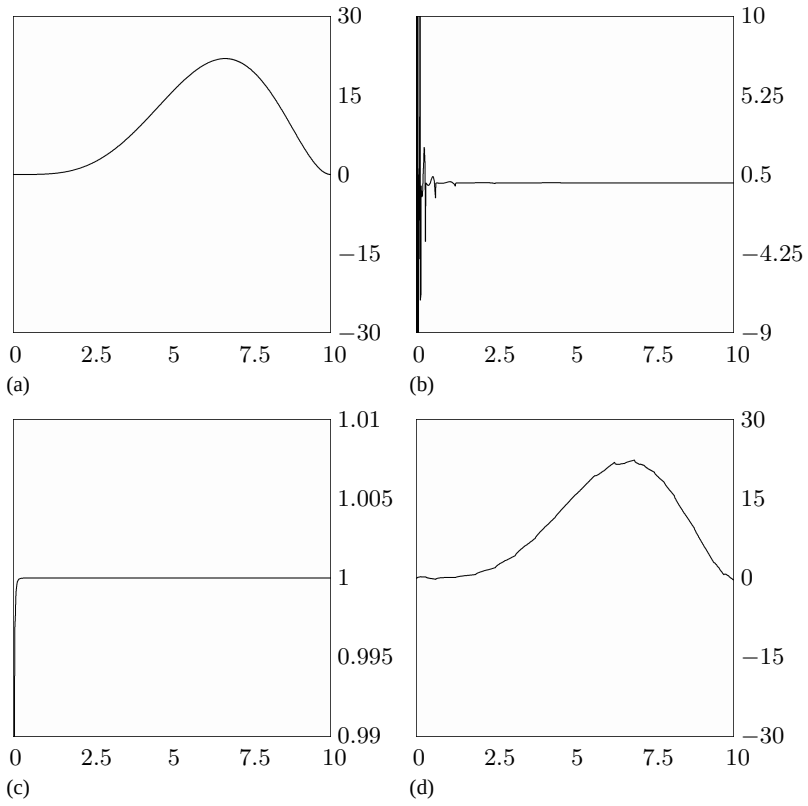


Figure 18
(a) Signal, (b) 10-level Daub4 transform, (c) energy map of Daub4 transform, 54:1 compression, 99.99% of energy.

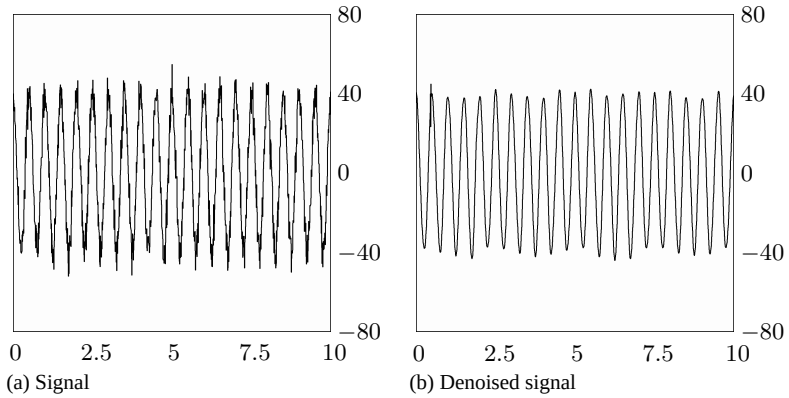


Figure 19
Solution to 3.6.1(b).

3.7.2 The maximum errors are $\mathbf{A}^1 : 1.32 \times 10^{-4}$,$\mathbf{A}^2 : 6.15 \times 10^{-4}$, $\mathbf{A}^3 : 2.31 \times 10^{-3}$,$\mathbf{A}^1 : 1.17 \times 10^{-2}$.

3.7.5 The minimum number of terms are listed in the following table.

Levels:	1	2	3	4	5	6
Terms:	335	206	206	233	246	220

3.7.6 To obtain the result we compute

```
20u^2(1-u)^4 cos(12pi u) \u = x
```

over the interval $[0, 1]$ using $16\,384$ points. Then we perform a 1-level `DD 5/3 (2,2)` transform. For this transform we change the X-interval to $[0, .5]$ and then plot the function

```
20u^2(1-u)^4 cos(12pi u) \u = 2x
```

(note the change from `\u = x` to `\u = 2x`) with the option `Auto-fit` *not* selected (so that the X-interval displayed remains as $[0, 0.5]$). Finding the Sup-norm difference between these two graphs yields a maximum error of 5.92×10^{-7}.

3.8.2 The maximum errors are $\mathbf{A}^1 : 2.74 \times 10^{-6}$,$\mathbf{A}^2 : 5.36 \times 10^{-5}$, $\mathbf{A}^3 : 7.91 \times 10^{-4}$,$\mathbf{A}^1 : 9.63 \times 10^{-3}$.

3.8.5 The minimum number of terms are listed in the following table.

Levels:	1	2	3	4	5	6
Terms:	333	168	85	64	67	65

3.8.6 To obtain the result we compute

```
20u^2(1-u)^4 cos(12pi u) \u = x
```

over the interval $[0, 1]$ using $16\,384$ points. Then we perform a 1-level `Daub 9/7` transform. For this transform we change the X-interval to $[0, .5]$ and then plot the function

```
sqr(2)20u^2(1-u)^4 cos(12pi u) \u = 2x
```

(note the change from `\u = x` to `\u = 2x` and the factor `sqr(2)`) with the option `Auto-fit` *not* selected (so that the X-interval displayed remains as $[0, 0.5]$). Finding the Sup-norm difference between these two graphs yields a maximum error of 7.08×10^{-7}.

Chapter 4

4.1.1(a) The 1-level Haar transform array is

$$\begin{pmatrix} -4 & 1 & 0 & -1 \\ 5 & 9 & -3 & 1 \\ 2 & 3 & -2 & 1 \\ 9 & 15 & 1 & -1 \end{pmatrix}.$$

4.1.2 (a) The 2-level Haar transform array is

$$\begin{pmatrix} -4 & 1 & 0 & -1 \\ 5 & 9 & -3 & 1 \\ 9.5 & -2.5 & -2 & 1 \\ 14.5 & -3.5 & 1 & -1 \end{pmatrix}.$$

4.1.4 $\mathbf{W}_1^1 \otimes \mathbf{V}_1^1 = \begin{pmatrix} 1/2 & -1/2 \\ 1/2 & -1/2 \end{pmatrix}$.

4.2.1 The PSNR values for the reconstructions of the `Airfield.bmp` image are

Method/C.R.	*8:1*	*16:1*	*32:1*
JPEG	29.3	26.6	23.5
J2K	31.0	27.9	25.2
ASWDR	31.4	28.4	25.7

4.3.2 The PSNRs for the zooms of the compressions are

Method/C.R.	*8:1*	*16:1*	*32:1*
JPEG	30.54	25.48	19.86
J2K	33.12	28.20	24.42
ASWDR	32.46	27.99	24.54

4.4.1 The quantized transform at threshold 2 is shown in Figure 20(a) and for half-threshold 1 in (b).

4	6	8	6	2	4	2	6
−2	−2	−4	−2	0	2	−2	−4
−4	−4	−6	−4	2	−2	−6	−10
2	4	6	4	2	2	0	−2
6	4	6	8	10	−4	4	6
8	8	6	4	8	−6	6	8
14	28	−10	8	8	−6	4	8
22	24	−8	10	8	−4	8	8

(a) Threshold 2

5	7	9	7	3	5	3	7
−3	−3	−5	−3	1	3	−3	−5
−5	−5	−7	−5	3	−3	−7	−11
3	5	7	5	3	3	1	−3
7	5	7	9	11	−5	5	7
9	9	7	5	9	−7	7	9
15	29	−11	9	9	−7	5	9
23	25	−9	11	9	−5	9	9

(b) Round to half-threshold 1

Figure 20
Quantized transforms for Exercise 4.4.1.

4.4.5 + + + + 1 1 1 0 1 +

4.5.5 The images are shown in Figure 21.

4.5.9 The images are shown in Figure 22. If the entire image is transmitted losslessly, then $187,044$ bytes are needed. If only the image in (d) is sent (with just ROI lossless), then only $10,264$ bytes are needed, a 94.5% savings.

4.6.3 The RD-curve is shown in Figure 23.

4.7.5 The denoisings are shown in Figure 24. We leave the subjective interpretation of them to the reader.

4.8.2 The edge enhancement is shown in Figure 25.

(a) Original image (b) 0.25 bpp (32:1 compression)

(c) 0.5 bpp (16:1 compression) (d) 1.0 bpp (8:1 compression)

Figure 21
Illustration of Progressive Reconstruction. Each image in (b) to (d) was computed from a single compressed file (saved at 1.0 **bpp). First, (b) is reconstructed, then (c), then (d).**

4.8.6 The relative 2-norm differences for these images (compared to a 32:1 decompression of `Lena.pgm`) are

Image	*Full*	*Second*	*Third*	*Fourth*	*Fifth*
Barb	0.499	0.467	0.455	0.433	0.379
Zelda	0.548	0.543	0.537	0.524	0.490
Lena	0.040	0.015	0.009	0.005	0.002
Barb	0.609	0.604	0.596	0.579	0.537

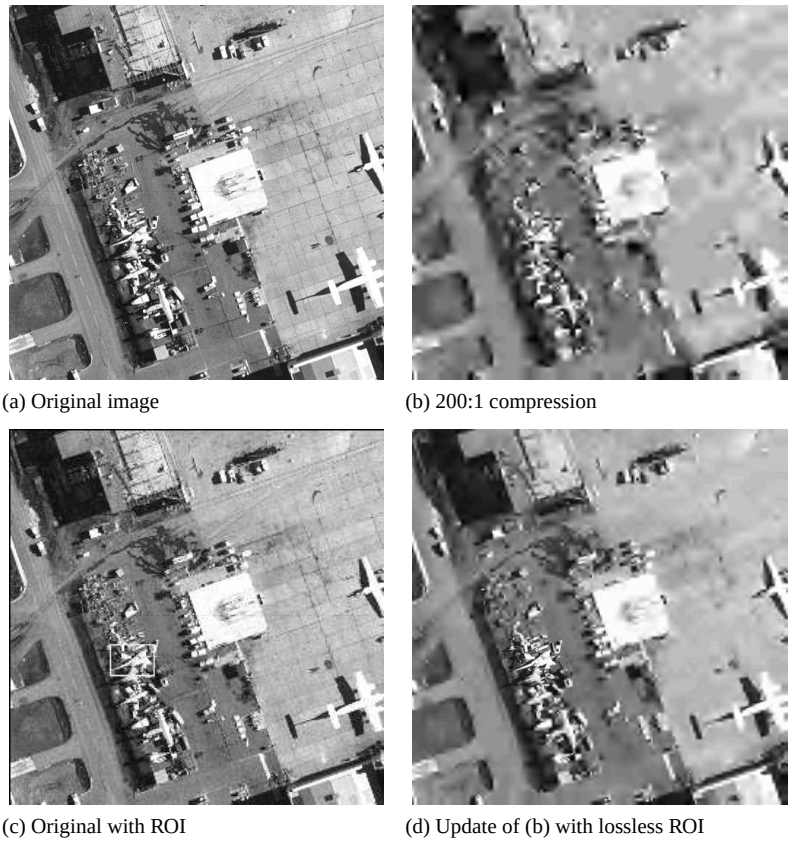
Chapter 5

5.1.1 The graphs are shown in Figure 26.

5.1.3 The spectra are shown in Figure 27.

5.2.1 The DFT of $\alpha\mathbf{f} + \beta\mathbf{g}$ is

$$\sum_{m=1}^{N}(\alpha f_m + \beta g_m)e^{-i2\pi(m-1)(n-1)/N} = \alpha\sum_{m=1}^{N} f_m e^{-i2\pi(m-1)(n-1)/N} + \beta\sum_{m=1}^{N} g_m e^{-i2\pi(m-1)(n_1)/N}$$
$$= \alpha(\mathcal{F}\mathbf{f})_n + \beta(\mathcal{F}\mathbf{g})_n.$$



(a) Original image (b) 200:1 compression

(c) Original with ROI (d) Update of (b) with lossless ROI

Figure 22
Illustration of ROI Property

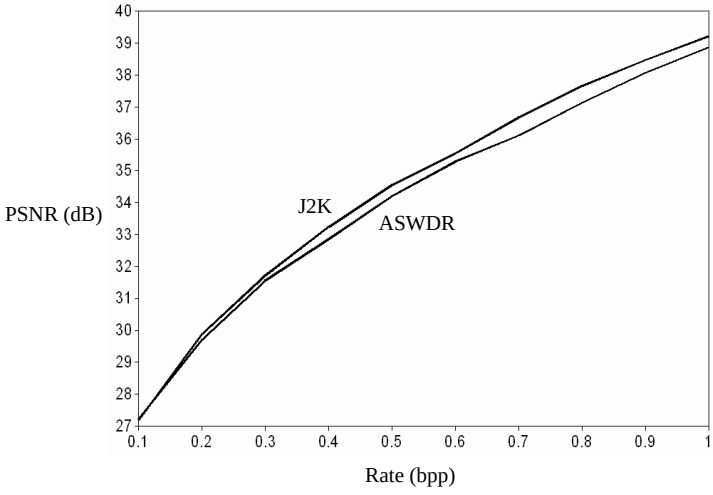


Figure 23
Rate-Distortion curves for the `Boat.bmp` image.

(a) Original image (b) Wiener2 denoising

(c) TAWS denoising (d) TAWS-SPIN denoising

Figure 24
Zooms of denoisings of `Elaine.bmp` image.

5.2.2 We have

$$\begin{aligned}(\mathcal{F}\mathbf{f})_{n+N} &= \sum_{m=1}^{N} f_m e^{-i2\pi(m-1)(n-1+N)/N} \\ &= \sum_{m=1}^{N} f_m e^{-i2\pi(m-1)(n-1)/N} e^{-i2\pi(m-1)} \\ &= \sum_{m=1}^{N} f_m e^{-i2\pi(m-1)(n-1)/N} \cdot 1 \\ &= (\mathcal{F}\mathbf{f})_n.\end{aligned}$$

(a) Original (b) Edge enhancement

Figure 25
Edge enhancement for Exercise 4.8.2.

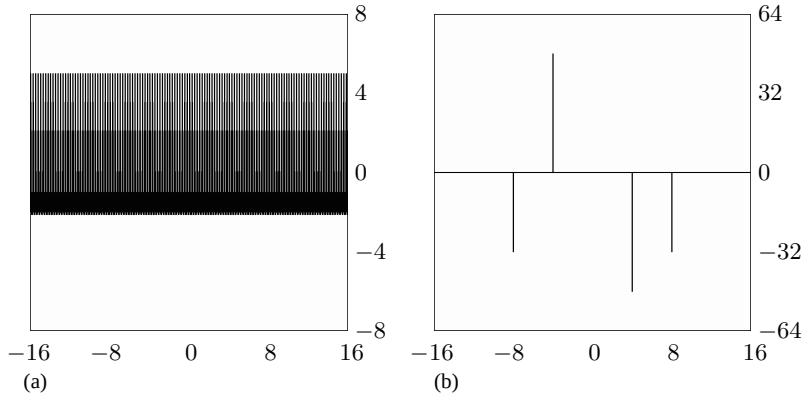


Figure 26
(a) Graph of function. (b) Frequency content.

5.2.3 Fix a value of m from 1 to N. We then have

$$\begin{aligned}\frac{1}{N}\sum_{n=-N/2}^{N/2-1}(\mathcal{F}\mathbf{f})_n e^{+i2\pi(n-1)(m-1)/N} &= \sum_{n=-N/2}^{N/2-1}\frac{1}{N}\sum_{\ell=1}^{N} f_\ell e^{-i2\pi(\ell-1)(n-1)/N}\, e^{+i2\pi(n-1)(m-1)/N} \\ &= \sum_{\ell=1}^{N} f_\ell \left(\frac{1}{N}\sum_{n=-N/2}^{N/2-1} e^{-i2\pi(\ell-1)(n-1)/N}\, e^{+i2\pi(n-1)(m-1)/N}\right) \\ &= \sum_{\ell=1}^{N} f_\ell \left(\frac{1}{N}\sum_{n=-N/2}^{N/2-1} e^{i2\pi(m-\ell)(n-1)/N}\right).\end{aligned}$$

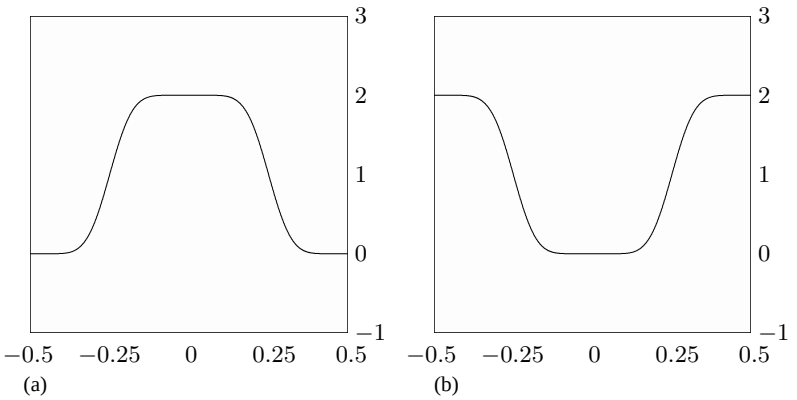


Figure 27
(a) Spectrum of Coif12 scaling signal $\mathbf{V}_{20}^1$. (b) Spectrum of Coif12 wavelet $\mathbf{W}_{20}^1$.

We now simplify the innermost sum in the last line above. If $\ell = m$, then we have

$$\sum_{n=-N/2}^{N/2-1} e^{i2\pi(m-\ell)(n-1)/N} = \sum_{n=-N/2}^{N/2-1} e^0 = N.$$

If $\ell \neq m$, then we have (using a finite geometric series sum):

$$\begin{aligned}
\sum_{n=-N/2}^{N/2-1} e^{i2\pi(m-\ell)(n-1)/N} &= \sum_{n=-N/2}^{N/2-1} (e^{i2\pi(m-\ell)/N})^{n-1} \\
&= \frac{e^{-i\pi(m-\ell)}e^{-i2\pi(m-\ell)/N} - e^{i\pi(m-\ell)}e^{-i2\pi(m-\ell)/N}}{1 - e^{i2\pi(m-\ell)/N}} \\
&= \frac{(e^{-i\pi(m-\ell)} - e^{i\pi(m-\ell)})e^{-i2\pi(m-\ell)/N}}{1 - e^{i2\pi(k-\ell)/N}} \\
&= 0
\end{aligned}$$

since $e^{\pm i2\pi(m-\ell)}$ have the same value (of either -1 or $+1$). Therefore, we have

$$\frac{1}{N} \sum_{n=-N/2}^{N/2-1} (\mathcal{F}\mathbf{f})_n e^{+i2\pi(m-1)(n-1)/N} = f_m \cdot 1$$

and inversion is proved.

5.2.5 The z-transform is $1 + z + z^2 + z^3$ and its roots are $-1, \pm i$.

5.3.1 The portions of the DFT that correspond to $\mathbf{A}^3$, $\mathbf{D}^3$, $\mathbf{D}^2$, and $\mathbf{D}^1$ are shown in Figure 28.

5.5.3 Before doing normalized correlation, both the text image and the single letter image are *inverted* in the sense that black pixels are converted to white pixels and vice versa. As to the relation to vision, we know that there are neurons that respond to darkness in the foreground versus brightness in the background, just as there are neurons that respond to whiteness in the foreground versus darkness in the background. These complementary responses of neurons is loosely analogous to how we can choose to invert the black and white relationships for our normalized correlations.

5.6.3 To see the solution, you should consult the article by Strichartz (reference [21] for this chapter) where the Daub6 case is discussed in detail.

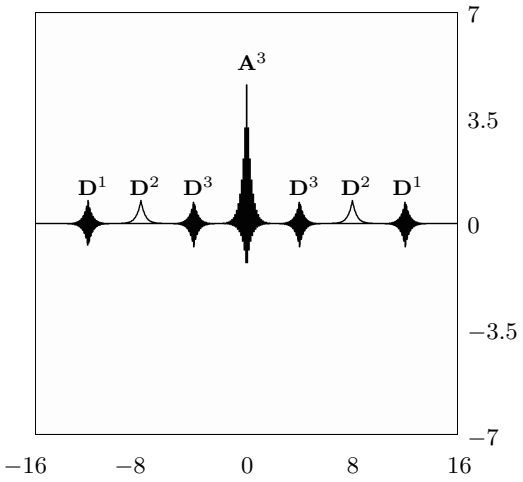
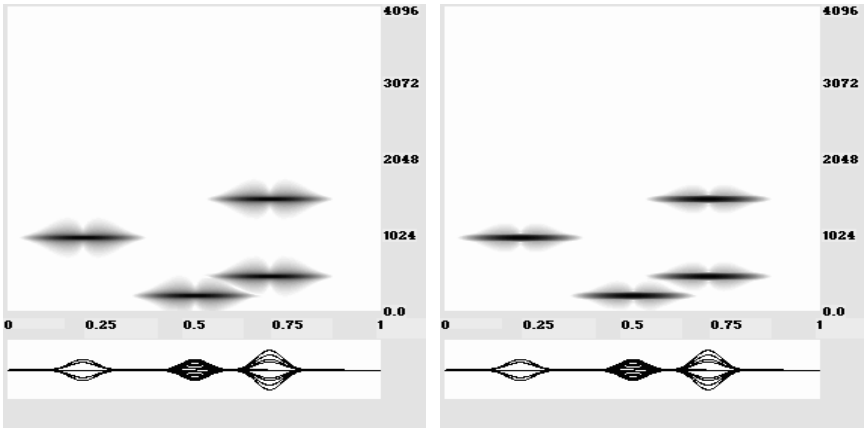


Figure 28
DFT with parts labelled (e.g. $\mathbf{A}^3$ labels the DFT portion that corresponds to $\mathbf{A}^3$).

5.7.1 The spectrograms are shown in Figure 29. They are quite similar; however, the Hanning windowed spectrogram shows a bit more "smearing" (slightly more extensive light gray patches above and below the darker frequency bands). Engineers refer to this smearing as "leakage." Because they show less leakage, we generally use Blackman windows for our spectrograms.



(a) (b)

Figure 29
(a) Hanning windowed spectrogram. (b) Blackman windowed spectrogram.

5.8.2[c] The spectrogram is shown in Figure 30. The two diagonal segments—forming a top of a triangle shaped object enclosed in the rectangle at bottom left center—is repeated with different sizes and positions in the time-frequency plane, and is also inverted in a region at its right edge and again inverted (over a slightly longer time-scale) at the end of the spectrogram. All of these structures are even clearer in the color version produced by AUDACITY, and reveal their full effect when the spectrogram is traced as the clip is played. This analysis provides further confirmation of our Multiresolution Principle.

5.9.2 A musical analysis of the `warbler.wav` recording can be found in subsection **2.3 Analysis III: A Warbler's**

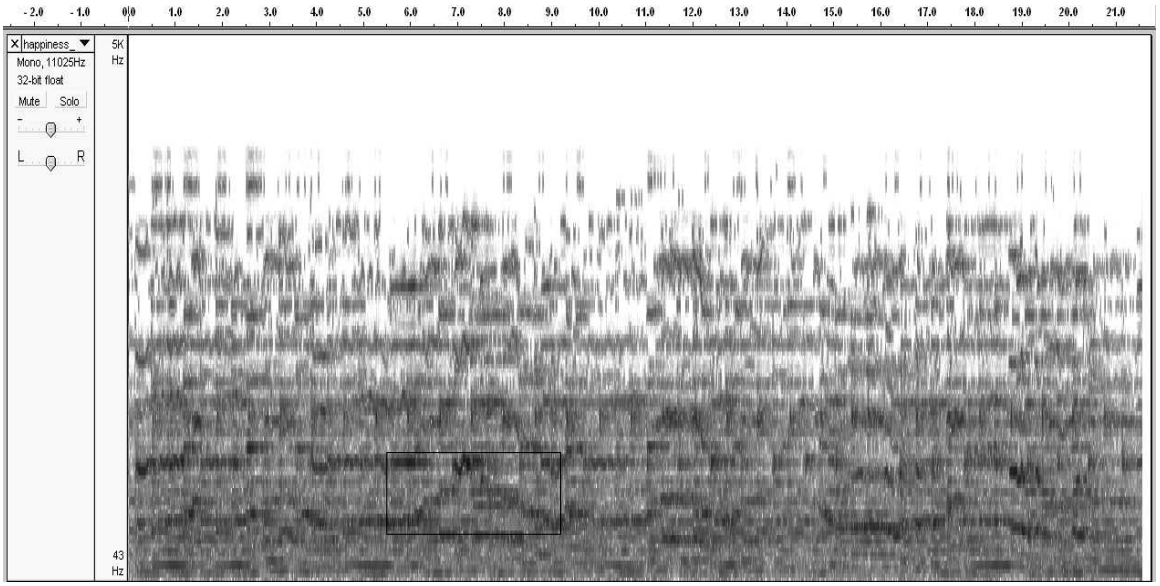


Figure 30
AUDACITY computed spectrogram of `Happiness_clip.wav`.

Song of the article *Music: a time-frequency approach* available at

`http://www.uwec.edu/walkerjs/media/TFAM.pdf` (1)

5.9.4 One solution to selectively amplifying the harp glissando can be found in subsection **3.3 Synthesis II: Altering figure-ground** of the article *Music: a time-frequency approach* available at the webpage listed in (1). More details on how FAWAV can be used for this example can be found in the document listed above in the statement of problem 5.8.2.

Chapter 6

6.1.1 $(10, 10\,|\,-4, -4\,|\,0, -2\,|\,-2, 0)$

6.1.3 Wavelet packet series: 183. Wavelet series: 228.

6.2.1 Using the *Threshold* choice for a wavelet and wavelet packet series, with threshold setting `1/2^7:1/2^5` in both cases, we obtained the following results:

Transform	*Sig. values*	*Bpp*	*RMS Error*
wavelet	1219	0.26	189.5
wavelet packet	1185	0.25	182.2

There are some high-frequency artifacts audible in the compressions, these are very high compressions (≈64:1), a lower compression ratio (using lower threshold settings) would undoubtedly sound better.

6.2.4 The RMS error between the original and the noisy signal is 1003.6. We then performed a 5-level Coif30 transform and applied the thresholding `g1(x)(abs(g1(x))>4000)` where we obtained the threshold of `4000` by visual inspection of the transform. After inverse transforming the thresholded transform, we obtained a denoised signal with RMS error of 969.1, but there was much less audible noise. Performing the same thresholding on a 5-level Coif30

wavelet packet transform, we obtained a denoised signal with RMS error of 811.6, which is smaller, and this denoising sounded better. The wavelet packet denoising was better because there is a lot of energy within the fluctuations of the osprey's call. By performing wavelet transforms on those fluctuations we amplify the fluctuations, while at the same time leaving the noise variance unchanged. That allows for a more effective thresholding operation.

6.2.6 The analog of Table 6.2 for the `Boat.bmp` image is the following:

Transform	*Bpp*	*Sig. values*	*PSNR*
wavelet	0.5	24034	34.12
wavelet packet	0.5	22229	33.17
wavelet	0.25	11557	30.59
wavelet packet	0.25	10646	29.94