12.4 Solutions

Section 1.1

1 Setting p_{ref}=0.00002 Pa and p=100 000 Pa in the decibel expression we get

$$\begin{aligned}20\log_{10}\left(\frac{p}{p_{\text{ref}}}\right) &= 20\log_{10}\left(\frac{100000}{0.00002}\right) = 20\log_{10}\left(\frac{10^5}{2\times 10^{-5}}\right)\\ &= 20\log_{10}\left(\frac{10^{10}}{2}\right) = 20\left(10-\log_{10}2\right) \approx 194\text{db}.\end{aligned}$$

Section 1.2

1 $\sin(2\pi\nu_1 t)$ has period $1/\nu_1$, while $\sin(2\pi\nu_2 t)$ has period $1/\nu_2$. The period is not unique, however. The first one also has period n/ν_1, and the second also n/ν_2, for any n. The sum is periodic if there exist n_1, n_2 so that $n_1/\nu_1 = n_2\nu_2$. Then this value will be a common period of the two functions, which also will be a period of f. This amounts to that $\nu_1/\nu_2 = n_1/n_2$, i.e. that ν_1/ν_2 is a rational number.

Section 1.3

1 The function $f(t) = \frac{1}{\sqrt{t}} = t^{-1/2}$ can be used since it has the properties

$$\begin{aligned}\int_0^T f(t)dt &= \lim_{x\to 0+}\int_x^T t^{-1/2}dt = \lim_{x\to 0+}\left[2t^{1/2}\right]_x^T\\ &= \lim_{x\to 0+}(2T^{1/2}-2x^{1/2}) = 2T^{1/2}\\ \int_0^T f(t)^2dt &= \lim_{x\to 0+}\int_x^T t^{-1}dt = \lim_{x\to 0+}\left[\ln t\right]_x^T\\ &= \ln T - \lim_{x\to 0+}\ln x = \infty.\end{aligned}$$

Section 1.4

2 For $f(t) = t$ we get that $a_0 = \frac{1}{T}\int_0^T tdt = \frac{T}{2}$. We also get

$$\begin{aligned}a_n &= \frac{2}{T}\int_0^T t\cos(2\pi nt/T)dt\\ &= \frac{2}{T}\left(\left[\frac{T}{2\pi n}t\sin(2\pi nt/T)\right]_0^T - \frac{T}{2\pi n}\int_0^T \sin(2\pi nt/T)dt\right) = 0\\ b_n &= \frac{2}{T}\int_0^T t\sin(2\pi nt/T)dt\\ &= \frac{2}{T}\left(\left[-\frac{T}{2\pi n}t\cos(2\pi nt/T)\right]_0^T + \frac{T}{2\pi n}\int_0^T \cos(2\pi nt/T)dt\right) = -\frac{T}{\pi n}.\end{aligned}$$

The Fourier series is thus

$$\frac{T}{2} - \sum_{n\geq 1} \frac{T}{\pi n} \sin(2\pi nt/T).$$

Note that this is almost a sine series, since it has a constant term, but no other cosine terms. If we had subtracted $T/2$ we would have obtained a function which is antisymmetric, and thus a pure sine series.

For $f(t) = t^2$ we get that $a_0 = \frac{1}{T}\int_0^T t^2 dt = \frac{T^2}{3}$. We also get

$$\begin{aligned}
a_n &= \frac{2}{T}\int_0^T t^2 \cos(2\pi nt/T)dt \\
&= \frac{2}{T}\left(\left[\frac{T}{2\pi n} t^2 \sin(2\pi nt/T)\right]_0^T - \frac{T}{\pi n}\int_0^T t \sin(2\pi nt/T)dt\right) \\
&= \left(-\frac{T}{\pi n}\right)\left(-\frac{T}{\pi n}\right) = \frac{T^2}{\pi^2 n^2} \\
b_n &= \frac{2}{T}\int_0^T t^2 \sin(2\pi nt/T)dt \\
&= \frac{2}{T}\left(\left[-\frac{T}{2\pi n} t^2 \cos(2\pi nt/T)\right]_0^T + \frac{T}{\pi n}\int_0^T t \cos(2\pi nt/T)dt\right) \\
&= -\frac{T^2}{\pi n}.
\end{aligned}$$

Here we see that we could use the expressions for the Fourier coefficients of $f(t) = t$ to save some work. The Fourier series is thus

$$\frac{T^2}{3} + \sum_{n\geq 1}\left(\frac{T^2}{\pi^2 n^2}\cos(2\pi nt/T) - \frac{T^2}{\pi n}\sin(2\pi nt/T)\right).$$

For $f(t) = t^3$ we get that $a_0 = \frac{1}{T}\int_0^T t^3 dt = \frac{T^3}{4}$. We also get

$$\begin{aligned}
a_n &= \frac{2}{T}\int_0^T t^3 \cos(2\pi nt/T)dt \\
&= \frac{2}{T}\left(\left[\frac{T}{2\pi n} t^3 \sin(2\pi nt/T)\right]_0^T - \frac{3T}{2\pi n}\int_0^T t^2 \sin(2\pi nt/T)dt\right) \\
&= \left(-\frac{3T}{2\pi n}\right)\left(-\frac{T^2}{\pi n}\right) = \frac{3T^3}{2\pi^2 n^2} \\
b_n &= \frac{2}{T}\int_0^T t^3 \sin(2\pi nt/T)dt \\
&= \frac{2}{T}\left(\left[-\frac{T}{2\pi n} t^3 \cos(2\pi nt/T)\right]_0^T + \frac{3T}{2\pi n}\int_0^T t^2 \cos(2\pi nt/T)dt\right) \\
&= -\frac{T^3}{\pi n} + \frac{3T}{2\pi n}\frac{T^2}{\pi^2 n^2} = -\frac{T^3}{\pi n} + \frac{3T^3}{2\pi^3 n^3}.
\end{aligned}$$

Also here we saved some work, by reusing the expressions for the Fourier coefficients of $f(t) = t^2$. The Fourier series is thus

$$\frac{T^3}{4} + \sum_{n\geq 1} \left(\frac{3T^3}{2\pi^2 n^2} \cos(2\pi nt/T) + \left(-\frac{T^3}{\pi n} + \frac{3T^3}{2\pi^3 n^3} \right) \sin(2\pi nt/T) \right).$$

We see that all three Fourier series converge slowly. This is connected to the fact that none of the functions are continuous at the borders of the periods.

3 Let us define $a_{n,k}, b_{n,k}$ as the Fourier coefficients of t^k. When $k > 0$ and $n > 0$, integration by parts gives us the following difference equations:

$$\begin{aligned}
a_{n,k} &= \frac{2}{T}\int_0^T t^k \cos(2\pi nt/T)dt \\
&= \frac{2}{T}\left(\left[\frac{T}{2\pi n} t^k \sin(2\pi nt/T) \right]_0^T - \frac{kT}{2\pi n}\int_0^T t^{k-1} \sin(2\pi nt/T)dt \right) \\
&= -\frac{kT}{2\pi n} b_{n,k-1} \\
b_{n,k} &= \frac{2}{T}\int_0^T t^k \sin(2\pi nt/T)dt \\
&= \frac{2}{T}\left(\left[-\frac{T}{2\pi n} t^k \cos(2\pi nt/T) \right]_0^T + \frac{kT}{2\pi n}\int_0^T t^{k-1} \cos(2\pi nt/T)dt \right) \\
&= -\frac{T^k}{\pi n} + \frac{kT}{2\pi n} a_{n,k-1}.
\end{aligned}$$

When $n > 0$, these can be used to express $a_{n,k}, b_{n,k}$ in terms of $a_{n,0}, b_{n,0}$, for which we clearly have $a_{n,0} = b_{n,0} = 0$. For $n = 0$ we have that $a_{0,k} = \frac{T^k}{k+1}$ for all k. The following program computes $a_{n,k}, b_{n,k}$ recursively when $n > 0$.

```
function [ank,bnk]=findfouriercoeffs(n,k,T)
  ank=0; bnk=0;
  if k>0
    [ankprev,bnkprev]=findfouriercoeffs(n,k-1,T)
    ank=-k*T*bnkprev/(2*pi*n);
    bnk=-T^k/(pi*n) + k*T*ankprev/(2*pi*n);
  end
```

Section 1.5

1 For $n_1 \neq n_2$ we have that

$$\begin{aligned}\langle e^{2\pi i n_1 t/T}, e^{2\pi i n_2 t/T}\rangle &= \frac{1}{T}\int_0^T e^{2\pi i n_1 t/T} e^{-2\pi i n_2 t/T} dt = \frac{1}{T}\int_0^T e^{2\pi i (n_1-n_2)t/T} dt \\ &= \left[\frac{T}{2\pi i(n_1-n_2)} e^{2\pi i(n_1-n_2)t/T}\right]_0^T \\ &= \frac{T}{2\pi i(n_1-n_2)} - \frac{T}{2\pi i(n_1-n_2)} = 0.\end{aligned}$$

When $n_1 = n_2$ the integrand computes to 1, so that $\|e^{2\pi i n t/T}\| = 1$.

3

a We have that

$$\begin{aligned}\cos^n(t) &= \left(\frac{1}{2}(e^{it} + e^{-it})\right)^n \\ \sin^n(t) &= \left(\frac{1}{2i}(e^{it} - e^{-it})\right)^n\end{aligned}$$

If we multiply out here, we get a sum of terms of the form e^{ikt}, where $-n \leq k \leq n$. As long as $n \leq N$ it is clear that this is in $V_{N,2\pi}$.

b We have that

$$\begin{aligned}\cos(t) &= \frac{1}{2}(e^{it} + e^{-it}) \\ \cos^2(t) &= \frac{1}{4}(e^{it} + e^{-it})^2 = \frac{1}{4}e^{2it} + \frac{1}{2} + \frac{1}{4}e^{-2it} \\ \cos^3(t) &= \frac{1}{8}(e^{it} + e^{-it})^3 = \frac{1}{8}e^{3it} + \frac{3}{8}e^{it} + \frac{3}{8}e^{-it} + \frac{1}{8}e^{-3it}.\end{aligned}$$

Therefore, for the first function the nonzero Fourier coefficients are $y_{-1} = 1/2$, $y_1 = 1/2$, for the second function $y_{-2} = 1/4$, $y_0 = 1/2$, $y_2 = 1/4$, for the third function $y_{-3} = 1/8$, $y_{-1} = 3/8$, $y_1 = 3/8$, $y_3 = 1/8$.

c In order to find the Fourier coefficients of $\cos^n(t)$ we have to multiply out the expression $\frac{1}{2^n}(e^{it} + e^{-it})^n$. The coefficients we get after this can alos be obtained from Pascal's triangle.

4 We obtain that

$$\begin{aligned}
y_n &= \frac{1}{T}\int_0^{T/2} e^{-2\pi int/T}dt - \frac{1}{T}\int_{T/2}^{T} e^{-2\pi int/T}dt \\
&= -\frac{1}{T}\left[\frac{T}{2\pi in}e^{-2\pi int/T}\right]_0^{T/2} + \frac{1}{T}\left[\frac{T}{2\pi in}e^{-2\pi int/T}\right]_{T/2}^{T} \\
&= \frac{1}{2\pi in}\left(-e^{-\pi in}+1+1-e^{-\pi in}+\right) \\
&= \frac{1}{\pi in}\left(1-e^{-\pi in}\right) = \begin{cases} 0, & \text{if } n \text{ is even;} \\ 2/(\pi in), & \text{if } n \text{ is odd.} \end{cases}.
\end{aligned}$$

Instead using Theorem 1.26 together with the coefficients $b_n = \frac{2(1-\cos(n\pi)}{n\pi}$ we computed in Example 1.18, we obtain

$$y_n = \frac{1}{2}(a_n - ib_n) = -\frac{1}{2}i\begin{cases} 0, & \text{if } n \text{ is even;} \\ 4/(n\pi), & \text{if } n \text{ is odd.} \end{cases} = \begin{cases} 0, & \text{if } n \text{ is even;} \\ 2/(\pi in), & \text{if } n \text{ is odd.} \end{cases}$$

when $n > 0$. The case $n < 0$ follows similarly.

6 For $f(t) = t$ we get

$$\begin{aligned}
y_n &= \frac{1}{T}\int_0^T te^{-2\pi int/T}dt = \frac{1}{T}\left(\left[-\frac{T}{2\pi in}te^{-2\pi int/T}\right]_0^T + \int_0^T \frac{T}{2\pi in}e^{-2\pi int/T}dt\right) \\
&= -\frac{T}{2\pi in} = \frac{T}{2\pi n}i.
\end{aligned}$$

From Exercise 2 we had $b_n = -\frac{T}{\pi n}$, for which Theorem 1.26 gives $y_n = \frac{T}{2\pi n}i$ for $n > 0$, which coincides with the expression we obtained. The case $n < 0$ follows similarly.
For $f(t) = t^2$ we get

$$\begin{aligned}
y_n &= \frac{1}{T}\int_0^T t^2e^{-2\pi int/T}dt = \frac{1}{T}\left(\left[-\frac{T}{2\pi in}t^2e^{-2\pi int/T}\right]_0^T + 2\int_0^T \frac{T}{2\pi in}te^{-2\pi int/T}dt\right) \\
&= -\frac{T^2}{2\pi in} + \frac{T^2}{2\pi^2n^2} = \frac{T^2}{2\pi^2n^2} + \frac{T^2}{2\pi n}i.
\end{aligned}$$

From Exercise 2 we had $a_n = \frac{T^2}{\pi^2n^2}$ and $b_n = -\frac{T^2}{\pi n}$, for which Theorem 1.26 gives $y_n = \frac{1}{2}\left(\frac{T^2}{\pi^2n^2} + i\frac{T^2}{\pi n}\right)$ for $n > 0$, which also is seen to coincide with what we obtained. The case $n < 0$ follows similarly.
For $f(t) = t^3$ we get

$$\begin{aligned}
y_n &= \frac{1}{T}\int_0^T t^3e^{-2\pi int/T}dt = \frac{1}{T}\left(\left[-\frac{T}{2\pi in}t^3e^{-2\pi int/T}\right]_0^T + 3\int_0^T \frac{T}{2\pi in}t^2e^{-2\pi int/T}dt\right) \\
&= -\frac{T^3}{2\pi in} + 3\frac{T}{2\pi in}(\frac{T^2}{2\pi^2n^2} + \frac{T^2}{2\pi n}i) = 3\frac{T^3}{4\pi^2n^2} + \left(\frac{T^3}{2\pi n} - 3\frac{T^3}{4\pi^3n^3}\right)i =
\end{aligned}$$

From Exercise 2 we had $a_n = \frac{3T^3}{2\pi^2 n^2}$ and $b_n = -\frac{T^3}{\pi n} + \frac{3T^3}{2\pi^3 n^3}$ for which Theorem 1.26 gives

$$y_n = \frac{1}{2}\left(\frac{3T^3}{2\pi^2 n^2} + i\left(\frac{T^3}{\pi n} - \frac{3T^3}{2\pi^3 n^3}\right)\right) = \frac{3T^3}{4\pi^2 n^2} + \left(\frac{T^3}{2\pi n} - \frac{3T^3}{4\pi^3 n^3}\right) i$$

for $n > 0$, which also is seen to coincide with what we obtained. The case $n < 0$ follows similarly.

7

a If f is symmetric about 0 we have that $b_n = 0$. Theorem 1.26 then gives that $y_n = \frac{1}{2}a_n$, which is real. The same theorem gives that $y_{-n} = \frac{1}{2}a_n = y_n$.

b If f is antisymmetric about 0 we have that $a_n = 0$. Theorem 1.26 then gives that $y_n = -\frac{1}{2}b_n$, which is purely imaginary. The same theorem gives that $y_{-n} = \frac{1}{2}b_n = -y_n$.

c When $y_n = y_{-n}$ we can write

$$y_{-n}e^{2\pi i(-n)t/T} + y_n e^{2\pi int/T} = y_n(e^{2\pi int/T} + e^{-2\pi int/T}) = 2y_n\cos(2\pi nt/T)$$

This is clearly symmetric, but then also $\sum_{n=-N}^{N} y_n e^{2\pi int/T}$ is symmetric since it is a sum of symmetric functions.

d When $y_n = -y_{-n}$ we can write

$$y_{-n}e^{2\pi i(-n)t/T} + y_n e^{2\pi int/T} = y_n(-e^{2\pi int/T} + e^{2\pi int/T}) = 2iy_n\sin(2\pi nt/T)$$

This is clearly antisymmetric, but then also $\sum_{n=-N}^{N} y_n e^{2\pi int/T}$ is antisymmetric since it is a sum of antisymmetric functions, and since $y_0 = 0$.

Section 1.6

1 The $2n$th complex Fourier coefficient of $\breve{f}$ is

$$\begin{aligned}&\frac{1}{2T}\int_0^{2T}\breve{f}(t)e^{-2\pi i2nt/(2T)}dt\\ &= \frac{1}{2T}\int_0^T f(t)e^{-2\pi int/T}dt + \frac{1}{2T}\int_T^{2T} f(2T-t)e^{-2\pi int/T}dt.\end{aligned}$$

Substituting $u = 2T - t$ in the second integral we see that this is

$$\begin{aligned}&= \frac{1}{2T}\int_0^T f(t)e^{-2\pi int/T}dt - \frac{1}{2T}\int_T^0 f(u)e^{2\pi inu/T}du\\ &= \frac{1}{2T}\int_0^T f(t)e^{-2\pi int/T}dt + \frac{1}{2T}\int_0^T f(t)e^{2\pi int/T}dt\\ &= \frac{1}{2}y_n + \frac{1}{2}y_{-n}.\end{aligned}$$

Therefore we have $a_{2n} = y_n - y_{-n}$.

Section 1.7

1 We obtain that

$$\begin{aligned}
y_n &= \frac{1}{T}\int_{-T/4}^{T/4} e^{-2\pi i n t/T}dt - \frac{1}{T}\int_{-T/2}^{-T/4} e^{-2\pi i n t/T}dt - \frac{1}{T}\int_{T/4}^{T/2} e^{-2\pi i n t/T}dt \\
&= -\left[\frac{1}{2\pi i n}e^{-2\pi i n t/T}\right]_{-T/4}^{T/4} + \left[\frac{1}{2\pi i n}e^{-2\pi i n t/T}\right]_{-T/2}^{-T/4} + \left[\frac{1}{2\pi i n}e^{-2\pi i n t/T}\right]_{T/4}^{T/2} \\
&= \frac{1}{2\pi i n}\left(-e^{-\pi i n/2} + e^{\pi i n/2} + e^{\pi i n/2} - e^{\pi i n} + e^{-\pi i n} - e^{-\pi i n/2}\right) \\
&= \frac{1}{\pi n}\left(2\sin(\pi n/2) - \sin(\pi n)\right) = \frac{2}{\pi n}\sin(\pi n/2).
\end{aligned}$$

The square wave defined in this exercise can be obtained by delaying our original square wave with $-T/4$. Using Property 3 in Theorem 1.36 with $d = -T/4$ on the complex Fourier coefficients $y_n = \begin{cases} 0, & \text{if } n \text{ is even;} \\ 2/(\pi i n), & \text{if } n \text{ is odd.} \end{cases}$ which we obtained for the square wave in Exercise 1.5.4, we obtain the Fourier coefficients

$$\begin{aligned}
e^{2\pi i n(T/4)/T}\begin{cases} 0, & \text{if } n \text{ is even;} \\ 2/(\pi i n), & \text{if } n \text{ is odd.} \end{cases} &= \begin{cases} 0, & \text{if } n \text{ is even;} \\ \frac{2i\sin(\pi n/2)}{\pi i n}, & \text{if } n \text{ is odd.} \end{cases} \\
&= \begin{cases} 0, & \text{if } n \text{ is even;} \\ \frac{2}{\pi n}\sin(\pi n/2), & \text{if } n \text{ is odd.} \end{cases}.
\end{aligned}$$

This verifies the result.

2 Since the real Fourier series of the square wave is

$$\sum_{n\geq 1, n \text{ odd}} \frac{4}{\pi n}\sin(2\pi n t/T),$$

Theorem 1.26 gives us that the complex Fourier coefficients are $y_n = -\frac{1}{2}i\frac{4}{\pi n} = -\frac{2i}{\pi n}$, and $y_{-n} = \frac{1}{2}i\frac{4}{\pi n} = \frac{2i}{\pi n}$ for $n > 0$. This means that $y_n = -\frac{2i}{\pi n}$ for all n, so that the complex Fourier series of the square wave is

$$-\sum_{n \text{ odd}} \frac{2i}{\pi n}e^{2\pi i n t/T}.$$

Using Property 4 in Theorem 1.36 we get that the $e^{-2\pi i 4t/T}$ (i.e. set $d = -4$) times the square wave has its n'th Fourier coefficient equal to $-\frac{2i}{\pi(n+4)}$. Using linearity, this means that $2ie^{-2\pi i 4t/T}$ times the square wave has its n'th Fourier coefficient equal to $\frac{4}{\pi(n+4)}$. We thus have that the function

$$f(t) = \begin{cases} 2ie^{-2\pi i 4t/T} & , 0 \leq t < T/2 \\ -2ie^{-2\pi i 4t/T} & , T/2 \leq t < T \end{cases}$$

has the desired Fourier series.

Section 2.2

1 The code for playing the sound can look like this:

```
fs=44100;
T=1/440;
t=0:(1/fs):3;
S=2*(t>=4*T).*min((t-4*T)/(8*T),1).*sin(2*pi*440*t);
S=S/max(abs(S));
playerobj=audioplayer(S,fs);
playblocking(playerobj);
```

2 The important thing to note here is that there are two oscillations present in Figure 1.1(b): One slow oscillation with a higher amplitude, and one faster oscillation, with a lower amplitude. We see that there are 10 periods of the smaller oscillation within one period of the larger oscillation, so that we should be able to reconstruct the figure by using frequencies where one is 10 times the other, such as 440Hz and 4400Hz. Also, we see from the figure that the amplitude of the larger oscillation is close to 1, and close to 0.3 for the smaller oscillation. A good choice therefore seems to be $a = 1, b = 0.3$. The code can look this: The code can look like this:

```
fs=44100;
T=1/440;
t=0:(1/fs):3;
S=sin(2*pi*440*t)+0.3*sin(2*pi*4400*t);
S=S/max(abs(S));
playerobj=audioplayer(S,fs);
playblocking(playerobj);
```

3

a The code can look like this:

```
function playpuresound(f)
  fs=44100;
  t=0:(1/fs):3;
  sd=sin(2*pi*f*t);
  playerobj=audioplayer(sd,fs);
  playblocking(playerobj)
```

4 The code can look like this:

```
function playsquare(T)
  % Play a square wave with period T over 3 seconds
  fs=44100;
  samplesperperiod=round(fs*T);
  oneperiod=[ones(1,round(samplesperperiod/2)) ...
             -ones(1,round(samplesperperiod/2))];
  allsamples=zeros(1,floor(3/T)*length(oneperiod));
  for k=1:floor(3/T)
    allsamples(((k-1)*length(oneperiod)+1):k*length(oneperiod))=oneperiod;
  end
  playerobj=audioplayer(allsamples,fs);
  playblocking(playerobj)
```

```
function playtriangle(T)
  % Play a triangle wave with period T over 3 seconds
  fs=44100;
  samplesperperiod=round(fs*T);
  oneperiod=[linspace(-1,1,round(samplesperperiod/2)) ...
           linspace(1,-1,round(samplesperperiod/2))];
  allsamples=zeros(1,floor(3/T)*length(oneperiod));
  for k=1:floor(3/T)
    allsamples(((k-1)*length(oneperiod)+1):k*length(oneperiod))=oneperiod;
  end
  playerobj=audioplayer(allsamples,fs);
  playblocking(playerobj)
```

5

a The code can look like this:

```
function playsquaretrunk(T,N)
  fs=44100;
  t=0:(1/fs):3;
  sd=zeros(1,length(t));
  n=1;
  while n<=N
    sd = sd + (4/(n*pi))*sin(2*pi*n*t/T);
    n=n+2;
  end
  playerobj=audioplayer(sd,fs);
  playblocking(playerobj)
```

```
function playtriangletrunk(T,N)
  fs=44100;
  t=0:(1/fs):3;
  sd=zeros(1,length(t));
  n=1;
  while n<=N
    sd = sd - (8/(n^2*pi^2))*cos(2*pi*n*t/T);
    n=n+2;
  end
  playerobj=audioplayer(sd,fs);
  playblocking(playerobj)
```

6

a The code can look like this:

```
function playdifferentfs()
  [S fs]=wavread('castanets.wav');
  playerobj=audioplayer(S,fs);
  playblocking(playerobj);
  playerobj=audioplayer(S,2*fs);
  playblocking(playerobj);
  playerobj=audioplayer(S,fs/2);
  playblocking(playerobj);
```

b The code can look like this:

```
function playreverse()
  [S fs]=wavread('castanets.wav');
  sz=size(S,1);
  playerobj=audioplayer(S(sz:(-1):1,:),fs);
  playblocking(playerobj);
```

7

a The code can look like this:

```
function playnoise(c)
  [S fs]=wavread('castanets.wav');
  sz=size(S,1);
  newS=S+c*(2*rand(sz,2)-1);
  newS=newS/max(max(abs(newS)));
  playerobj=audioplayer(newS,fs);
  playblocking(playerobj);
```

Section 2.4

1 As in Example 2.19 we get

$$F_4 \begin{pmatrix} 2 \\ 3 \\ 4 \\ 5 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{pmatrix} \begin{pmatrix} 2 \\ 3 \\ 4 \\ 5 \end{pmatrix}$$
$$= \frac{1}{2} \begin{pmatrix} 2+3+4+5 \\ 2-3i-4+5i \\ 2-3+4-5 \\ 2+3i-4-5i \end{pmatrix} = \begin{pmatrix} 7 \\ -1+i \\ -1 \\ -1-i \end{pmatrix}.$$

2 For $N = 6$ the entries are on the form $\frac{1}{\sqrt{6}} e^{-2\pi ink/6} = \frac{1}{\sqrt{6}} e^{-\pi ink/3}$. This means that the entries in the Fourier matrix are the numbers $\frac{1}{\sqrt{6}} e^{-\pi i/3} = \frac{1}{\sqrt{6}}(1/2 - i\sqrt{3}/2)$, $\frac{1}{\sqrt{6}} e^{-2\pi i/3} = \frac{1}{\sqrt{6}}(-1/2 - i\sqrt{3}/2)$, and so on. The matrix is thus

$$F_6 = \frac{1}{\sqrt{6}} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1/2 - i\sqrt{3}/2 & -1/2 - i\sqrt{3}/2 & -1 & -1/2 + i\sqrt{3}/2 & 1/2 + i\sqrt{2}/2 \\ 1 & -1/2 - i\sqrt{3}/2 & -1/2 + i\sqrt{3}/2 & 1 & -1/2 - i\sqrt{3}/2 & +1/2 - i\sqrt{3}/2 \\ 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -1/2 + i\sqrt{3}/2 & -1/2 - i\sqrt{3}/2 & 1 & -1/2 + i\sqrt{3}/2 & -1/2 - i\sqrt{3}/2 \\ 1 & 1/2 + i\sqrt{2}/2 & -1/2 + i\sqrt{3}/2 & -1 & -1/2 - i\sqrt{3}/2 & 1/2 - i\sqrt{3}/2 \end{pmatrix}$$

The cases $N = 8$ and $N = 12$ follow similarly, but are even more tedious. For $N = 8$ the entries are $\frac{1}{\sqrt{8}} e^{\pi ink/4}$, which can be expressed exactly since we can express exactly any sines and cosines of a multiple of $\pi/4$. For $N = 12$ we get the base angle $\pi/6$, for which we also have exact values for sines and cosines for all multiples.

4 By Theorem 2.21 we know that $(F_N(\boldsymbol{x}))_{N-n} = \overline{(F_N(\boldsymbol{x}))_n}$ when $\boldsymbol{x}$ is a real vector. If we set $N = 8$ and $n = 2$ we get that $(F_8(\boldsymbol{x}))_6 = \overline{(F_8(\boldsymbol{x}))_2} = \overline{2-i} = 2+i$.

5 We get

$$y_n = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} c^k e^{-2\pi ink/N} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} (ce^{-2\pi in/N})^k$$
$$= \frac{1}{\sqrt{N}} \frac{1 - (ce^{-2\pi in/N})^N}{1 - ce^{-2\pi in/N}} = \frac{1}{\sqrt{N}} \frac{1 - c^N}{1 - ce^{-2\pi in/N}}.$$

7 The code can look like this

```
function x=IDFTImpl(y)
  N=length(y);
  FN=zeros(N);
```

```
for k=1:N
  FN(k,:)=exp(2*pi*1i*(k-1)*(0:(N-1))/N)/sqrt(N);
end
x=FN*y;
```

8 We have that

$$(F_N(\boldsymbol{x}))_k = (F_N(\boldsymbol{x}_1 + i\boldsymbol{x}_2))_k = (F_N(\boldsymbol{x}_1))_k + i(F_N(\boldsymbol{x}_2))_k$$
$$(F_N(\boldsymbol{x}))_{N-k} = (F_N(\boldsymbol{x}_1))_{N-k} + i(F_N(\boldsymbol{x}_2))_{N-k} = \overline{(F_N(\boldsymbol{x}_1))_k} + i\overline{(F_N(\boldsymbol{x}_2))_k},$$

where we have used Property 1 of Theorem 2.21. If we take the complex conjugate in the last equation, we are left with the two equations

$$(F_N(\boldsymbol{x}))_k = (F_N(\boldsymbol{x}_1))_k + i(F_N(\boldsymbol{x}_2))_k$$
$$\overline{(F_N(\boldsymbol{x}))_{N-k}} = (F_N(\boldsymbol{x}_1))_k - i(F_N(\boldsymbol{x}_2))_k.$$

If we add these we get

$$(F_N(\boldsymbol{x}_1))_k = \frac{1}{2}\left((F_N(\boldsymbol{x}))_k + \overline{(F_N(\boldsymbol{x}))_{N-k}}\right),$$

which is the first equation. If we instead subtract the equations we get

$$(F_N(\boldsymbol{x}_2))_k = \frac{1}{2i}\left((F_N(\boldsymbol{x}))_k - \overline{(F_N(\boldsymbol{x}))_{N-k}}\right),$$

which is the second equation

Section 2.9

1

a We get

$$F_4\boldsymbol{x}_1 = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{pmatrix}\begin{pmatrix} 1 \\ 3 \\ 5 \\ 7 \end{pmatrix} = \begin{pmatrix} 8 \\ -2+2i \\ -2 \\ -2-2i \end{pmatrix}$$
$$F_4\boldsymbol{x}_2 = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{pmatrix}\begin{pmatrix} 2 \\ 4 \\ 6 \\ 8 \end{pmatrix} = \begin{pmatrix} 10 \\ -2+2i \\ -2 \\ -2-2i \end{pmatrix}$$

b In the FFT-algorithm we split the computation of $F_4(\boldsymbol{x})$ into the computation of $F_2(\boldsymbol{x}^{(e)})$ and $F_2(\boldsymbol{x}^{(o)})$, where $x^{(e)}$ and $x^{(o)}$ are vectors of length 4 with even-indexed and odd-indexed components, respectively. In this case we have $\boldsymbol{x}^{(e)} = (1, 3, 5, 7)$ and $\boldsymbol{x}^{(o)} = (2, 4, 6, 8)$. In other words, the FFT-algorithm uses the FFT-computations we did in i a., so that we can save computation. The benefit of using the FFT-algorithm is that we save computation (the algorithm is of order $O(2N \log_2 N)$).

2

a By inserting $N = 2^r$ and $x_r = M^{2^r}$ in $M_N = 2M_{N/2} + 2N$ we get first $x_r = 2x_{r-1} + 2 \cdot 2^r$. Inserting $r+1$ for r we get $x_{r+1} - 2x_r = 4 \cdot 2^r$.

b The homogeneous equation $x_{r+1} - 2x_r = 0$ has the general solution $(x_h)_r = C2^r$. For a particular solution to the equation $x_{r+1} - 2x_r = 4 \cdot 2^r$, we should try $(x_p)_r = Ar2^r$ (since 2 is a root in the homogeneous equation), and we get that $A = 2$, so that $(x_p)_r = 2r2^r$, and the general solution to the difference equation is $x_r = 2r2^r + C2^r$.

c We get that

$$M_N = M_{2^r} = 2r2^r + C2^r = 2N \log_2 N + CN = O(2N \log_2 N),$$

since the first terms dominates in this expression, in particular, it does not matter what C is (although we can find C from $x_0 = 0$, since a DFT for $N = 1$ requires no multiplications).

3 When we compute $e^{-2\pi i n/N}$, we do some multiplications in the exponent. These are not counted because the multilication do not depend on $\boldsymbol{x}$, and may therefore be precomputed. We also have a multiplication with $\frac{1}{\sqrt{2}}$. These are typically not counted because one often defines a DFT so that this multiplication is absorbed in the definition.

5

a From the formula we see that the first third of the Fourier cofficients can be written

$$y_n = \frac{1}{\sqrt{3}} \left(F_{N/3}\boldsymbol{x}_1 + D_{N/3}F_{N/3}\boldsymbol{x}_2 + D_{N/3}^2 F_{N/3}\boldsymbol{x}_3 \right).$$

where $D_{N/3}$ is defined in the same way as $D_{N/2}$, but as a $(N/3) \times (N/3)$-matrix, and where $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3$ denotes the splitting of $\boldsymbol{x}$ into vectors for the corresponding indices.

b The second third of the Fourier cofficients can be written

$$\begin{aligned}
y_{N/3+n} &= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} x_k e^{-2\pi i(N/3+n)k/N} \\
&= \frac{1}{\sqrt{N}} \sum_{k=0}^{N/3-1} x_{3k} e^{-2\pi i(N/3+n)3k/N} + \frac{1}{\sqrt{N}} \sum_{k=0}^{N/3-1} x_{3k+1} e^{-2\pi i(N/3+n)(3k+1)/N} \\
&+ \frac{1}{\sqrt{N}} \sum_{k=0}^{N/3-1} x_{3k+2} e^{-2\pi i(N/3+n)(3k+2)/N} \\
&= \frac{1}{\sqrt{N}} \sum_{k=0}^{N/3-1} x_{3k} e^{-2\pi i n3k/N} + \frac{1}{\sqrt{N}} e^{-2\pi i(N/3+n)/N} \sum_{k=0}^{N/3-1} x_{3k+1} e^{-2\pi i n3k/N} \\
&+ \frac{1}{\sqrt{N}} e^{-2\pi i(N/3+n)2/N} \sum_{k=0}^{N/3-1} x_{3k+2} e^{-2\pi i n3k/N} \\
&= \frac{1}{\sqrt{3}} \left(F_{N/3}\boldsymbol{x}_1 + e^{-2\pi i/3} D_{N/3} F_{N/3} \boldsymbol{x}_2 + e^{-2\pi i 2/3} D_{N/3}^2 F_{N/3} \boldsymbol{x}_3 \right).
\end{aligned}$$

The third part of the Fourier coefficients can be written

$$\begin{aligned}
y_{2N/3+n} &= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} x_k e^{-2\pi i(2N/3+n)k/N} \\
&= \frac{1}{\sqrt{N}} \sum_{k=0}^{N/3-1} x_{3k} e^{-2\pi i(2N/3+n)3k/N} + \frac{1}{\sqrt{N}} \sum_{k=0}^{N/3-1} x_{3k+1} e^{-2\pi i(2N/3+n)(3k+1)/N} \\
&+ \frac{1}{\sqrt{N}} \sum_{k=0}^{N/3-1} x_{3k+2} e^{-2\pi i(2N/3+n)(3k+2)/N} \\
&= \frac{1}{\sqrt{N}} \sum_{k=0}^{N/3-1} x_{3k} e^{-2\pi i n3k/N} + \frac{1}{\sqrt{N}} e^{-2\pi i(2N/3+n)/N} \sum_{k=0}^{N/3-1} x_{3k+1} e^{-2\pi i n3k/N} \\
&+ \frac{1}{\sqrt{N}} e^{-2\pi i(2N/3+n)2/N} \sum_{k=0}^{N/3-1} x_{3k+2} e^{-2\pi i n3k/N} \\
&= \frac{1}{\sqrt{3}} \left(F_{N/3}\boldsymbol{x}_1 + e^{-2\pi i 2/3} D_{N/3} F_{N/3} \boldsymbol{x}_2 + e^{-2\pi i 4/3} D_{N/3}^2 F_{N/3} \boldsymbol{x}_3 \right) \\
&= \frac{1}{\sqrt{3}} \left(F_{N/3}\boldsymbol{x}_1 + e^{-2\pi i 2/3} D_{N/3} F_{N/3} \boldsymbol{x}_2 + D_{N/3}^2 F_{N/3} \boldsymbol{x}_3 \right)
\end{aligned}$$

We get a similar factorization as in Theorem 2.34, but with the block matrix replaced by

$$\frac{1}{\sqrt{3}} \left(\begin{array}{c|c|c} F_{N/3} & D_{N/3}F_{N/3} & D_{N/3}^2 F_{N/3} \\ \hline F_{N/3} & e^{-2\pi i/3} D_{N/3} F_{N/3} & e^{-2\pi i 2/3} D_{N/3}^2 F_{N/3} \\ \hline F_{N/3} & e^{-2\pi i 2/3} D_{N/3} F_{N/3} & D_{N/3}^2 F_{N/3} \end{array} \right).$$

c We see that $M_N = 3M_{N/3} + 2N$ when we count complex multiplications, so that $M_N = 3M_{N/3} + 8N$ when we count real multiplications. We get a difference equation of the form $x_{r+1} = 3x_r + 24 \cdot 3^r$. A particular solution to this is $(x_p)_r = 8r3^r$. Solving as above we get $M_N = O(8N \log_3 N)$. $\log_3 N$ can be written on the form $c \log_2 N$ for a constant c, this is on the form $O(c \log_2 N)$ for some c.

d It is clear that this procedure can be developed also for numbers divisible by 5, 7, and so on (the number of blocks in the block matrix increase, though). In particular, we can develop a procedure for any factorization into prime numbers.

Section 3.1

1 Here we have that $t_{-1} = 1/4$, $t_0 = 1/4$, $t_1 = 1/4$, and $t_2 = 1/4$. We now get that $s_0 = t_0 = 1/4$, $s_1 = t_1 = 1/4$, and $s_2 = t_2 = 1/4$ (first formula), and $s_{N-1} = s_7 = t_{-1} = 1/4$ (second formula). This means that the matrix of S is

$$S = \frac{1}{4}\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

Section 3.2

3

a The eigenvalues of S are $1, 5, 9$, and are found by computing a DFT of the first column (and multiplying by $\sqrt{N} = 2$). The eigenvectors are the Fourier basis vectors. 1 has multiplicity 2.

c Matlab uses some numeric algorithm to find the eigenvectors. However, eigenvectors may not be unique, so you have no control on which eigenvectors Matlab actually selects. In particular, here the eigenspace for $\lambda = 1$ has dimension 2, so that any linear combination of the two eigenvectors from this eigenspace also is an eigenvector. Here it seems that Matlab has chosen a linear combination which is different from a Fourier basis vector.

4

a If we write $S_1 = F_N^H D_1 F_N$ and $S_2 = F_N^H D_2 F_N$ we get

$$S_1 + S_2 = F_N^H (D_1 + D_2) F_N S_1 S_2 = F_N^H D_1 F_N F_N^H D_2 F_N = F_N^H D_1 D_2 F_N$$

This means that the eigenvalues of $S_1 + S_2$ are the sum of the eigenvalues of S_1 and S_2. The actual eigenvalues which are added are dictated by the index of the frequency response, i.e. $\lambda_{S_1+S_2,n} = \lambda_{S_1,n} + \lambda_{S_2,n}$.

b As above we have that $S_1S_2 = F_N^H D_1 F_N F_N^H D_2 F_N = F_N^H D_1 D_2 F_N$, and the same reasoning gives that the eigenvalues of S_1S_2 are the product of the eigenvalues of S_1 and S_2. The actual eigenvalues which are multiplied are dictated by the index of the frequency response, i.e. $\lambda_{S_1S_2,n} = \lambda_{S_1,n}\lambda_{S_2,n}$.

c In general there is no reason to believe that there is a formula for the eigenvalues for the sum or product of two matrices, based on eigenvalues of the individual matrices. However, the same type of argument as for filters can be used in all cases where the eigenvectors are equal.

5 The matrix for the operation which keeps every second component is

$$\begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 0 \end{pmatrix},$$

where 1 and 0 are repeated in alternating order along the main diagonal. Since the matrix is not constant on the main diagonal, it is not a circulant Toeplitz matrix, and hence not a filter.

Section 3.3

1 The frequency response is

$$\lambda_S(\omega) = \frac{1}{4}(e^{i\omega} + 1 + e^{-i\omega} + e^{-2i\omega}) = \frac{e^{i\omega}(1 - e^{-4i\omega})}{4(1 - e^{-i\omega})} = \frac{1}{4}e^{-i\omega/2}\frac{\sin(2\omega)}{\sin(\omega/2)}.$$

Section 3.4

1 We have that $\lambda_S(\omega) = \frac{1}{2}(1 + \cos\omega)$. This clearly has the maximum point $(0, 1)$, and the minimum point $(\pi, 0)$.

2 We have that $|\lambda_T(\omega)| = \frac{1}{2}(1 - \cos\omega)$. This clearly has the maximum point $(\pi, 1)$, and the minimum point $(0, 0)$. The connection between the frequency responses is that $\lambda_T(\omega) = \lambda_S(\omega + \pi)$.

3 Here we have that $s_0 = t_0 = 3$, $s_1 = t_1 = 4$, $s_2 = t_2 = 5$, and $s_3 = t_3 = 6$ (first formula), and $s_{N-2} = t_{-2} = 1$, $s_{N-1} = t_{-1} = 2$ (second formula). This means that the matrix of S is

$$S = \begin{pmatrix} 3 & 2 & 1 & 0 & 0 & 6 & 5 & 4 \\ 4 & 3 & 2 & 1 & 0 & 0 & 6 & 5 \\ 5 & 4 & 3 & 2 & 1 & 0 & 0 & 6 \\ 6 & 5 & 4 & 3 & 2 & 1 & 0 & 0 \\ 0 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ 0 & 0 & 6 & 5 & 4 & 3 & 2 & 1 \\ 1 & 0 & 0 & 6 & 5 & 4 & 3 & 2 \\ 2 & 1 & 0 & 0 & 6 & 5 & 4 & 3 \end{pmatrix}$$

The frequency response is

$$\lambda_S(\omega) = e^{2i\omega} + 2e^{i\omega} + 3 + 4e^{-i\omega} + 5e^{-2i\omega} + 6e^{-3i\omega}.$$

4 The filter coefficients are $t_0 = s_0 = 1/5$, $t_1 = s_1 = 1/5$ (first formula), and $t_{-1} = s_{N-1} = 1/5$, $t_{-2} = s_{N-2} = 1/5$, $t_{-3} = s_{N-3} = 1/5$ (second formula). All other t_k are zero. This means that the filter can be written as $\frac{1}{5}\{1, 1, 1, \underline{1}, 1\}$, using our compact notation.

5 The frequency response is

$$\sum_{s=0}^{k} c^s e^{-is\omega} = \frac{1 - c^{k+1}e^{-i(k+1)\omega}}{1 - ce^{-i\omega}}.$$

It is straightforward to compute the limit as $\omega \to 0$ as $c^k(k+1)$. This means that as we increase k or c, this limit also increases. The value of k also dictates oscillations in the frequency response, since the numerator oscillates fastest. When $c = 1$, k dictates how often the frequency response hits 0.

Section 3.5

2

a The code can look like this:

```
function playwithecho(c,d)
  [S fs]=wavread('castanets.wav');
  N=size(S,1);
  S((d+1):N,:)=S((d+1):N,:)+c*S(1:(N-d),:); % Add echo
  S(:,1)=S(:,1)/max(max(abs(S(:,1)))); % Scale so that sound values are
  S(:,2)=S(:,2)/max(max(abs(S(:,2)))); % within [-1,1].
  playerobj=audioplayer(S,fs);
  playblocking(playerobj);
```

3 The sum of two digital filters is again a digital filter, and the first column in the sum can be obtained by summing the first columns in the two matrices. This means that the filter coefficients in $\frac{1}{2}(S_1+S_2)$ can be obtained by summing the filter coefficients of S_1 and S_2, and we obtain

$$\frac{1}{2}\left(\{\underline{1}, 0, \ldots, 0, c\} + \{\underline{1}, 0, \ldots, 0, -c\}\right) = \{\underline{1}\}.$$

This means that $\frac{1}{2}(S_1 + S_2) = I$, since I is the unique filter with $\boldsymbol{e}_0$ as first column. The interpretation in terms of echos is that the echo from S_2 cancels that from S_1.

4

a The code can look like this:

```
function reducebass(k)
  c=[1/2 1/2];
  for z=1:(2*k-1)
    c=conv(c,[1/2 1/2]);
  end
  c=(-1).^(0:(2*k)).*c;
  [S fs]=wavread('castanets.wav');
  N=size(S,1);

  y=zeros(N,2);
  y(1:k,:)=S(1:k,:);
  for t=(k+1):(N-k)
    for j=1:(2*k+1)
      y(t,:)=y(t,:)+c(j)*S(t+k+1-j,:);
    end
  end
  y((N-k+1):N,:)=S((N-k+1):N,:);
  y=y/max(max(abs(y)));

  playerobj=audioplayer(y,fs);
  playblocking(playerobj);
```

```
function reducetreble(k)
  c=[1/2 1/2];
  for z=1:(2*k-1)
    c=conv(c,[1/2 1/2]);
  end
  [S fs]=wavread('castanets.wav');
  N=size(S,1);

  y=zeros(N,2);
  y(1:k,:)=S(1:k,:);
  for t=(k+1):(N-k)
    for j=1:(2*k+1)
      y(t,:)=y(t,:)+c(j)*S(t+k+1-j,:);
    end
  end
  y((N-k+1):N,:)=S((N-k+1):N,:);

  playerobj=audioplayer(y,fs);
  playblocking(playerobj);
```

9

a In the code a filter is run on the sound samples from the file `castanets.wav`. Finally the new sound is played. In the first two lines after the `for`-loop, the first and the last sound samples in teh filteres sound are computed, under the assumption that the sound has been extended to a periodic sound with period `N`. After this, the sound is normalized so that the sound samples lie in the range between -1 and 1. In this case the filter is a lowpass-filter (as we show in b.), so that we can expect that that the treble in the sound is reduced.

b Compact filter notation for the filter which is run is $\{2, \underline{4}, 2\}$. A 5×5 circulant Toeplitz matrix becomes

$$\begin{pmatrix} 4 & 2 & 0 & 0 & 2 \\ 2 & 4 & 2 & 0 & 0 \\ 0 & 2 & 4 & 2 & 0 \\ 0 & 0 & 2 & 4 & 2 \\ 2 & 0 & 0 & 2 & 4 \end{pmatrix}.$$

The frequency response is $\lambda_S(\omega) = 2e^{i\omega} + 4 + 2e^{-i\omega} = 4 + 4\cos\omega$. It is clear that this gives a lowpass filter.

c The frequency response for the new filter is

$$-2e^{i\omega} + 4 - 2e^{-i\omega} = 4 - 4\cos\omega = 4 + 4\cos(\omega + \pi) = \lambda_S(\omega + \pi),$$

where S is the filter from the first part of the exercise. The new filter therefore becomes a highpass filter, since to add π to ω corresponds to swapping the frequencies 0 and π. We could also here refered to Observation 3.39, where we stated that adding an alternating sign in the filter coefficients turns a lowpass filter into a highpass filter and vice versa.

Section 3.6

1

a The matrix for time reversal is the matrix

$$\begin{pmatrix} 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \end{pmatrix}$$

This is not a circulant Toeplitz matrix, since all diagonals assume the values 0 and 1, so that they are not constant on each diagonal. Time reversal is thus not a digital filter.

b Let S denote time reversal. Clearly $S\boldsymbol{e}_1 = \boldsymbol{e}_{N-2}$. If S was time-invariant we would have that $S\boldsymbol{e}_0 = \boldsymbol{e}_{N-3}$, where we have delayed the input and output. But this clearly is not the case, since by definition $S\boldsymbol{e}_0 = \boldsymbol{e}_{N-1}$.

Section 3.7

Section 3.8

Section 4.1

2 The code can look like this:

```
function y=filterS(t,x)
  N=length(x);
  y=zeros(length(x),1);
  E=length(t)-1;

  n=0;
  while n<E
      y(n+1)= t(1)*x(n+1);
      for k=1:n
          y(n+1) = y(n+1) + t(k+1)*(x(n+k+1)+x(n-k+1));
      end
      for k=(n+1):E
          y(n+1) = y(n+1) + t(k+1)*(x(n+k+1)+x(n-k+N+1));
      end
      n=n+1;
  end
  while n<(N-E)
      y(n+1)= t(1)*x(n+1);
      for k=1:E
          y(n+1) = y(n+1)+ t(k+1)*(x(n+k+1)+x(n-k+1));
      end
      n=n+1;
  end
  while n<N
    y(n+1) = t(1)*x(n+1);
    for k=1:(N-1-n)
        y(n+1) = y(n+1) + t(k+1)*(x(n+k+1)+x(n-k+1));
    end
    for k=(N-1-n+1):E
        y(n+1) = y(n+1) + t(k+1)*(x(n+k-N+1)+x(n-k+1));
    end
    n=n+1;
  end
```

4 We have that

$$\begin{aligned} H\cos\left(2\pi\frac{n(k+1/2)\pi}{2N}\right) &= \frac{1}{2}He^{-2\pi in(k+1/2)/(2N)} + \frac{1}{2}He^{2\pi in(k+1/2)/(2N)} \\ &= \frac{1}{2}e^{-\pi in/(2N)}He^{-\pi ink/N} + \frac{1}{2}e^{\pi in/(2N)}He^{\pi ink/N}. \end{aligned}$$

Her $He^{-\pi ink/N}$ is the filter where the frequency response of H is delayed by $\pi n/N$, while the delay goes the opposite way for $He^{\pi ink/N}$. This shows that H_n has the stated center frequencies.

Section 4.2

Section 4.3

2 We first see that $d_{0,3} = \sqrt{\frac{1}{3}}$ and $d_{k,3} = \sqrt{\frac{2}{3}}$ for $k = 1, 2$. We also have that

$$\cos\left(2\pi\frac{n}{2N}\left(k+\frac{1}{2}\right)\right) = \cos\left(\pi\frac{n}{3}\left(k+\frac{1}{2}\right)\right),$$

so that the DCT matrix can be written as

$$\begin{aligned} D_3 &= \begin{pmatrix} \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} \\ \sqrt{\frac{2}{3}}\cos\left(\frac{\pi}{3}\frac{1}{2}\right) & \sqrt{\frac{2}{3}}\cos\left(\frac{\pi}{3}\frac{3}{2}\right) & \sqrt{\frac{2}{3}}\cos\left(\frac{\pi}{3}\frac{5}{2}\right) \\ \sqrt{\frac{2}{3}}\cos\left(\frac{2\pi}{3}\frac{1}{2}\right) & \sqrt{\frac{2}{3}}\cos\left(\frac{2\pi}{3}\frac{3}{2}\right) & \sqrt{\frac{2}{3}}\cos\left(\frac{2\pi}{3}\frac{5}{2}\right) \end{pmatrix} \\ &= \begin{pmatrix} \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} \\ \sqrt{\frac{2}{3}}\cos(\pi/6) & \sqrt{\frac{2}{3}}\cos(\pi/2) & \sqrt{\frac{2}{3}}\cos(5\pi/6) \\ \sqrt{\frac{2}{3}}\cos(\pi/3) & \sqrt{\frac{2}{3}}\cos(\pi) & \sqrt{\frac{2}{3}}\cos(5\pi/3) \end{pmatrix} \\ &= \begin{pmatrix} \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} \\ \sqrt{\frac{2}{3}}(\sqrt{3}/2 + i/2) & 0 & \sqrt{\frac{2}{3}}(-\sqrt{3}/2 + i/2) \\ \sqrt{\frac{2}{3}}(1/2 + \sqrt{3}i/2) & -\sqrt{\frac{2}{3}} & \sqrt{\frac{2}{3}}(1/2 - \sqrt{3}i/2) \end{pmatrix} \end{aligned}$$

Section 4.4

Section 5.2

1 We have that $f(t) = \sum_{n=0}^{N-1} c_n\phi_{0,n}$, where c_n are the coordinates of f in the basis $\{\phi_{0,0}, \phi_{0,1}, \ldots, \phi_{0,N-1}\}$. We now get that

$$f(k) = \sum_{n=0}^{N-1} c_n\phi_{0,n}(k) = c_k,$$

since $\phi_{0,n}(k) = 0$ when $n \neq k$. This shows that $(f(0), f(1), \ldots . f(N-1))$ are the coordinates of f.

2

a From lemma 5.9 it follows that

$$\begin{aligned} \text{proj}_{V_0}(\phi_{1,2n}) &= \phi_{0,n}/\sqrt{2} \\ \text{proj}_{V_0}(\phi_{1,2n+1}) &= \phi_{0,n}/\sqrt{2} \end{aligned}$$

This means that

$$\begin{aligned}
[\mathrm{proj}_{V_0}(\phi_{1,2n})]_{\boldsymbol{\phi}_0} &= \boldsymbol{e}_n/\sqrt{2} \\
[\mathrm{proj}_{V_0}(\phi_{1,2n+1})]_{\boldsymbol{\phi}_0} &= \boldsymbol{e}_n/\sqrt{2}.
\end{aligned}$$

These are the columns in the matrix for proj_{V_0} relative to the bases $\boldsymbol{\phi}_1$ and $\boldsymbol{\phi}_0$. This matrix is thus

$$\frac{1}{\sqrt{2}}\begin{pmatrix}
1 & 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & \cdots & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 1
\end{pmatrix}.$$

b From lemma 5.12 it follows that

$$\begin{aligned}
\mathrm{proj}_{W_0}(\phi_{1,2n}) &= \psi_{0,n}/\sqrt{2} \\
\mathrm{proj}_{W_0}(\phi_{1,2n+1}) &= -\psi_{0,n}/\sqrt{2}
\end{aligned}$$

This means that

$$\begin{aligned}
[\mathrm{proj}_{W_0}(\phi_{1,2n})]_{\boldsymbol{\psi}_0} &= \boldsymbol{e}_n/\sqrt{2} \\
[\mathrm{proj}_{W_0}(\phi_{1,2n+1})]_{\boldsymbol{\psi}_0} &= -\boldsymbol{e}_n/\sqrt{2}.
\end{aligned}$$

These are the columns in the matrix for proj_{W_0} relative to the bases $\boldsymbol{\phi}_1$ and $\boldsymbol{\psi}_0$. This matrix is thus

$$\frac{1}{\sqrt{2}}\begin{pmatrix}
1 & -1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\
0 & 0 & 1 & -1 & 0 & \cdots & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & 0 & 0 & \cdots & 0 & 1 & -1
\end{pmatrix}.$$

3

a Since $\phi \in V_0$ we must have that $\mathrm{proj}_{V_0}(\phi) = \phi$. Since ψ is in the orthogonal complement of V_0 in V_1 we must have that $\mathrm{proj}_{V_0}(\psi) = 0$.

b The first columns in the matrix of proj_{V_0} relative to $(\boldsymbol{\phi}_0, \boldsymbol{\psi}_0)$ are

$$\begin{aligned}
[\mathrm{proj}_{V_0}(\phi_{0,0})]_{(\boldsymbol{\phi}_0,\boldsymbol{\psi}_0)} &= [\phi_{0,0}]_{(\boldsymbol{\phi}_0,\boldsymbol{\psi}_0)} = \boldsymbol{e}_0 \\
[\mathrm{proj}_{V_0}(\phi_{0,1})]_{(\boldsymbol{\phi}_0,\boldsymbol{\psi}_0)} &= [\phi_{0,1}]_{(\boldsymbol{\phi}_0,\boldsymbol{\psi}_0)} = \boldsymbol{e}_1 \\
&\vdots
\end{aligned}$$

The last columns in the matrix of proj_{V_0} relative to $(\boldsymbol{\phi}_0, \boldsymbol{\psi}_0)$ are

$$\begin{aligned}
[\mathrm{proj}_{V_0}(\psi_{0,0})]_{(\boldsymbol{\phi}_0,\boldsymbol{\psi}_0)} &= [\mathbf{0}]_{(\boldsymbol{\phi}_0,\boldsymbol{\psi}_0)} = \mathbf{0} \\
[\mathrm{proj}_{V_0}(\psi_{0,1})]_{(\boldsymbol{\phi}_0,\boldsymbol{\psi}_0)} &= [\mathbf{0}]_{(\boldsymbol{\phi}_0,\boldsymbol{\psi}_0)} = \mathbf{0} \\
&\vdots
\end{aligned}$$

It follows that the matrix of proj_{V_0} relative to $(\boldsymbol{\phi}_0, \boldsymbol{\psi}_0)$ is given by the diagonal matrix where the first half of the entries on the diagonal are 1, the second half 0.

c Follows in the same way as (b).

4 We have that

$$\text{proj}_{V_0}(f) = \sum_{n=0}^{N-1} \left(\int_0^N f(t)\phi_{0,n}(t)dt \right) \phi_{0,n} = \sum_{n=0}^{N-1} \left(\int_n^{n+1} f(t)dt \right) \phi_{0,n},$$

where we have used the orthogonal decomposition formula. Note also that, if $f(t) \in V_1$, and $f_{n,1}$ is the value f attains on $[n, n+1/2)$, and $f_{n,2}$ is the value f attains on $[n+1/2, n+1)$, we have that

$$\begin{aligned}
\text{proj}_{V_0}(f) &= \sum_{n=0}^{N-1} \left(\int_n^{n+1} f(t)dt \right) \phi_{0,n}(t) \\
&= \sum_{n=0}^{N-1} \left(\frac{1}{2} f_{n,1} + \frac{1}{2} f_{n,2} \right) \phi_{0,n}(t) = \sum_{n=0}^{N-1} \frac{f_{n,1} + f_{n,2}}{2} \phi_{0,n}(t),
\end{aligned}$$

which is the function which is $(f_{n,1} + f_{n,2})/2$ on $[n, n+1)$. This proves the first part of Proposition 5.13.

5 We have that

$$\begin{aligned}
\|f - \text{proj}_{V_0}(f)\|^2 &= \langle f - \text{proj}_{V_0}(f), f - \text{proj}_{V_0}(f) \rangle \\
&= \langle f, f \rangle - 2\langle f, \text{proj}_{V_0}(f) \rangle + \langle \text{proj}_{V_0}(f), \text{proj}_{V_0}(f) \rangle
\end{aligned}$$

Now, note that

$$\langle \text{proj}_{V_0}(f), \text{proj}_{V_0}(f) \rangle = \sum_{n=0}^{N-1} \left(\int_n^{n+1} f(t)dt \right)^2$$

from what we just showed in Exercise 4 (use that the $\phi_{0,n}$ are orthonormal). This means that the above can be written

$$\begin{aligned}
&= \langle f, f \rangle - 2 \sum_{n=0}^{N-1} \int_0^N \left(\int_n^{n+1} f(s)ds \right) \phi_{0,n}(t) f(t)dt + \sum_{n=0}^{N-1} \left(\int_n^{n+1} f(t)dt \right)^2 \\
&= \langle f, f \rangle - 2 \sum_{n=0}^{N-1} \int_n^{n+1} \left(\int_n^{n+1} f(s)ds \right) f(t)dt + \sum_{n=0}^{N-1} \left(\int_n^{n+1} f(t)dt \right)^2 \\
&= \langle f, f \rangle - 2 \sum_{n=0}^{N-1} \left(\int_n^{n+1} f(t)dt \right)^2 + \sum_{n=0}^{N-1} \left(\int_n^{n+1} f(t)dt \right)^2 \\
&= \langle f, f \rangle - \sum_{n=0}^{N-1} \left(\int_n^{n+1} f(t)dt \right)^2 .
\end{aligned}$$

6 The orthogonal decomposition theorem gives that

$$\begin{aligned}
\mathrm{proj}_{W_0}(f) &= \sum_{n=0}^{N-1} \langle f, \psi_{0,n}\rangle \psi_{0,n}(t) = \sum_{n=0}^{N-1} \left(\int_0^N f(t)\psi_{0,n}(t)dt \right) \psi_{0,n}(t) \\
&= \sum_{n=0}^{N-1} \left(\int_n^{n+1} f(t)\psi_{0,n}(t)dt \right) \psi_{0,n}(t) \\
&= \sum_{n=0}^{N-1} \left(\int_n^{n+1/2} f(t)dt - \int_{n+1/2}^{n+1} f(t)dt \right) \psi_{0,n}(t),
\end{aligned}$$

where we used that $\psi_{0,n}$ is nonzero only on $[n, n+1)$, and is 1 on $[n, n+1/2)$, and -1 on $[n+1/2, n+1)$. Note also that, if $f(t) \in V_1$, and $f_{n,1}$ is the value f attains on $[n, n+1/2)$, and $f_{n,2}$ is the value f attains on $[n+1/2, n+1)$, we have that

$$\begin{aligned}
\mathrm{proj}_{W_0}(f) &= \sum_{n=0}^{N-1} \left(\int_n^{n+1/2} f(t)dt - \int_{n+1/2}^{n+1} f(t)dt \right) \psi_{0,n}(t) \\
&= \sum_{n=0}^{N-1} \left(\frac{1}{2} f_{n,1} - \frac{1}{2} f_{n,2} \right) \psi_{0,n}(t) = \sum_{n=0}^{N-1} \frac{f_{n,1} - f_{n,2}}{2} \psi_{0,n}(t),
\end{aligned}$$

which is the function which is $(f_{n,1} - f_{n,2})/2$ on $[n, n+1/2)$, and $-(f_{n,1} - f_{n,2})/2$ on $[n+1/2, n+1)$. This proves the second part of Proposition 5.13.

Section 5.3

1 Since $\phi_{m,n} \in V_m$ we must have that $T(\phi_{m,n}) = \phi_{m,n}$. Since $\psi_{m,n}$ is in the orthogonal complement of V_m in V_{m+1} we must have that $T(\psi_{m,n}) = 0$. The first half of the columns in the matrix of proj_{V_m} relative to $(\boldsymbol{\phi}_m, \boldsymbol{\psi}_m)$ are

$$\begin{aligned}
[\mathrm{proj}_{V_m}(\phi_{m,0})]_{(\boldsymbol{\phi}_m,\boldsymbol{\psi}_m)} &= [\phi_{m,0}]_{(\boldsymbol{\phi}_m,\boldsymbol{\psi}_m)} = \boldsymbol{e}_0 \\
[\mathrm{proj}_{V_m}(\phi_{m,1})]_{(\boldsymbol{\phi}_m,\boldsymbol{\psi}_m)} &= [\phi_{m,1}]_{(\boldsymbol{\phi}_m,\boldsymbol{\psi}_m)} = \boldsymbol{e}_1 \\
&\vdots
\end{aligned}$$

The second half of the columns are

$$\begin{aligned}
[T(\psi_{m,0})]_{(\boldsymbol{\phi}_m,\boldsymbol{\psi}_m)} &= [\mathbf{0}]_{(\boldsymbol{\phi}_m,\boldsymbol{\psi}_m)} = \mathbf{0} \\
[T(\psi_{m,1})]_{(\boldsymbol{\phi}_m,\boldsymbol{\psi}_m)} &= [\mathbf{0}]_{(\boldsymbol{\phi}_m,\boldsymbol{\psi}_m)} = \mathbf{0} \\
&\vdots
\end{aligned}$$

Thus, as before, the matrix of proj_{V_m} relative to $(\boldsymbol{\phi}_m, \boldsymbol{\psi}_m)$ is given by the diagonal matrix where the first half of the diagonal consists of 1's, and the second half consists of 0's. (c) follows in the same way.

2 If $f \in V_m$ we can write $f(t) = \sum_{n=0}^{2^m N-1} c_{m,n}\phi_{m,n}(t)$. We now get

$$\begin{aligned} g(t) = f(2t) &= \sum_{n=0}^{2^m N-1} c_{m,n}\phi_{m,n}(2t) = \sum_{n=0}^{2^m N-1} c_{m,n}2^{m/2}\phi(2^m 2t - n) \\ &= \sum_{n=0}^{2^m N-1} c_{m,n}2^{-1/2}2^{(m+1)/2}\phi(2^{m+1}t - n) = \sum_{n=0}^{2^m N-1} c_{m,n}2^{-1/2}\phi_{m+1,n}(t). \end{aligned}$$

This shows that $g \in V_{m+1}$. To prove the other way, assume that $g(t) = f(2t) \in V_{m+1}$. This means that we can write $g(t) = \sum_{n=0}^{2^{m+1}N-1} c_{m+1,n}\phi_{m+1,n}(t)$. We now have

$$\begin{aligned} f(t) = g(t/2) &= \sum_{n=0}^{2^{m+1}N-1} c_{m+1,n}\phi_{m+1,n}(t/2) = \sum_{n=0}^{2^{m+1}N-1} c_{m+1,n}2^{(m+1)/2}\phi(2^m t - n) \\ &= \sum_{n=0}^{2^m N-1} c_{m+1,n}2^{(m+1)/2}\phi(2^m t - n) + \sum_{n=2^m N}^{2^{m+1}N-1} c_{m+1,n}2^{(m+1)/2}\phi(2^m t - n) \\ &= \sum_{n=0}^{2^m N-1} c_{m+1,n}2^{(m+1)/2}\phi(2^m t - n) + \sum_{n=0}^{2^m N-1} c_{m+1,n+2^m N}2^{(m+1)/2}\phi(2^m t - n - 2^m N) \\ &= \sum_{n=0}^{2^m N-1} c_{m+1,n}2^{(m+1)/2}\phi(2^m t - n) + \sum_{n=0}^{2^m N-1} c_{m+1,n+2^m N}2^{(m+1)/2}\phi(2^m t - n) \\ &= \sum_{n=0}^{2^m N-1} (c_{m+1,n} + c_{m+1,n+2^m N})2^{1/2}2^{m/2}\phi(2^m t - n) \\ &= \sum_{n=0}^{2^m N-1} (c_{m+1,n} + c_{m+1,n+2^m N})2^{1/2}\phi_{m,n}(t) \in V_m \end{aligned}$$

The thing which made this a bit difficult was that the range of the n-indices here was outside $[0, 2^m N - 1]$ (which describe the legal indices in the basis V_m), so that we had to use the periodicty of ϕ.

3 By definition, $[T_1]_{\mathcal{B}_1} \oplus [T_2]_{\mathcal{B}_2} \oplus \cdots \oplus [T_n]_{\mathcal{B}_n}$ is a block matrix where the blocks on the diagonal are the matrices $[T_1]_{\mathcal{B}_1}$, $[T_2]_{\mathcal{B}_2}$, and so on. If $\boldsymbol{b}_i$ are the basis vectors in $\mathcal{B}_i$, the columns in $[T_i]_{\mathcal{B}_i}$ are $[T(\boldsymbol{b}_j)]_{\mathcal{B}_i}$. This means that $[T_1]_{\mathcal{B}_1} \oplus [T_2]_{\mathcal{B}_2} \oplus \cdots \oplus [T_n]_{\mathcal{B}_n}$ has $[T(\boldsymbol{b}_j)]_{\mathcal{B}_i}$ in the j'th block, and $\mathbf{0}$ elsewhere. This means that we can write it as

$$\mathbf{0} \oplus \cdots \mathbf{0} \oplus [T(\boldsymbol{b}_j)]_{\mathcal{B}_i} \oplus \mathbf{0} \cdots \mathbf{0}.$$

On the other hand, $[T_1 \oplus T_2 \oplus \ldots \oplus T_n]_{\mathcal{B}_1 \oplus \mathcal{B}_2 \oplus \ldots \oplus \mathcal{B}_n}$ is a matrix of the same size, and the corresponding column to that of the above is

$$\begin{aligned} &[(T_1 \oplus T_2 \oplus \ldots \oplus T_n)(\mathbf{0} \oplus \cdots \mathbf{0} \oplus \boldsymbol{b}_j \oplus \mathbf{0} \cdots \mathbf{0})]_{\mathcal{B}_1 \oplus \mathcal{B}_2 \oplus \ldots \oplus \mathcal{B}_n} \\ &= [\mathbf{0} \oplus \cdots \mathbf{0} \oplus T(\boldsymbol{b}_j) \oplus \mathbf{0} \cdots \mathbf{0}]_{\mathcal{B}_1 \oplus \mathcal{B}_2 \oplus \ldots \oplus \mathcal{B}_n} \\ &= \mathbf{0} \oplus \cdots \mathbf{0} \oplus [T(\boldsymbol{b}_j)]_{\mathcal{B}_i} \oplus \mathbf{0} \cdots \mathbf{0}. \end{aligned}$$

Here $\boldsymbol{b}_j$ occurs as the i'th summand. This is clearly the same as what we computed for the right hand side above.

4 Assume that λ is an eigenvalue common to both T_1 and T_2. Then there exists a vector $\boldsymbol{v}_1$ so that $T_1\boldsymbol{v}_1 = \lambda\boldsymbol{v}_1$, and a vector $\boldsymbol{v}_2$ so that $T_2\boldsymbol{v}_2 = \lambda\boldsymbol{v}_2$. We now have that

$$\begin{aligned}(T_1 \oplus T_2)(\boldsymbol{v}_1 \oplus \boldsymbol{v}_2) &= \begin{pmatrix} T_1 & 0 \\ 0 & T_2 \end{pmatrix} \begin{pmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \end{pmatrix} \\ &= \begin{pmatrix} T_1\boldsymbol{v}_1 \\ T_2\boldsymbol{v}_2 \end{pmatrix} = \begin{pmatrix} \lambda\boldsymbol{v}_1 \\ \lambda\boldsymbol{v}_2 \end{pmatrix} \\ &= \lambda \begin{pmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \end{pmatrix} = \lambda(\boldsymbol{v}_1 \oplus \boldsymbol{v}_2).\end{aligned}$$

This shows that λ is an eigenvalue for λ also, and that $\boldsymbol{v}_1 \oplus \boldsymbol{v}_2$ is a corresponding eigenvector.

5 We have that

$$\begin{aligned}(A \oplus B)(A^{-1} \oplus B^{-1}) &= \begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix} \begin{pmatrix} A^{-1} & 0 \\ 0 & B^{-1} \end{pmatrix} \\ &= \begin{pmatrix} AA^{-1} & 0 \\ 0 & BB^{-1} \end{pmatrix} = \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix} = I\end{aligned}$$

where we have multiplied as block matrices. This proves that $A \oplus B$ is invertible, and states what the inverse is.

6 We have that

$$(A \oplus B)(C \oplus D) = \begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix} \begin{pmatrix} C & 0 \\ 0 & D \end{pmatrix} = \begin{pmatrix} AC & 0 \\ 0 & BD \end{pmatrix} = (AC) \oplus (BD)$$

where we again have multiplied as block matrices.

8 The following code achieves this:

```
[S,fs]=wavread('castanets.wav');
newx=DWTHaarImpl(S(1:2^17,1),2);
plot(0:(2^17-1),newx(1:2^17,1))
axis([0 2^17 -1 1]);
```

The values from V_0 corresponds to the first 1/4 values in the plot, the values from W_0 corresponds to the next 1/4 values in the plot, while the values from W_1 correspond to the last 1/2 of the values in the plot.

9

a The following code achieves the task

```
function playDWTlower(m)
  [S fs]=wavread('castanets');
  newx=DWTHaarImpl(S(1:2^17,1),m);
  len=length(newx);
  newx((len/2^m+1):length(newx))=zeros(length(newx)-len/2^m,1);
  newx=IDWTHaarImpl(newx,m);
  playerobj=audioplayer(newx,fs);
  playblocking(playerobj);
```

b For $m = 2$ we clearly hear a degradation in the sound. For $m = 4$ and above most of the sound is unrecognizable.

c There is no reason to believe that sound samples returned by the function lie in $[-1, 1]$. you can check this by printing the maximum value in the returned array on screen inside this method.

11 The following code can be used

```
function playDWTlowerdifference(m)
  [S fs]=wavread('castanets');
  newx=DWTHaarImpl(S(1:2^17,1),m);
  len=length(newx);
  newx(1:(len/2^m))=zeros(len/2^m,1);
  newx=IDWTHaarImpl(newx,m);
  playerobj=audioplayer(newx,fs);
  playblocking(playerobj);
```

12 Note first that, similarly to the computation in Exercise 5.6, we have that

$$\int_0^N f(t)\psi_{m,n}(t)dt = 2^{m/2}\left(\int_{n2^{-m}}^{(n+1/2)2^{-m}} f(t)dt - \int_{(n+1/2)2^{-m}}^{(n+1)2^{-m}} f(t)dt\right).$$

With $f(t) = 1 - 2|1/2 - t/N|$ we have two possibilities: when $n < N2^{m-1}$ we have that $[n2^{-m}, (n+1)2^{-m}) \subset [0, N/2]$, so that $f(t) = 2t/N$, and we get

$$\begin{aligned}
w_{m,n} &= 2^{m/2}\left(\int_{n2^{-m}}^{(n+1/2)2^{-m}} 2t/N dt - \int_{(n+1/2)2^{-m}}^{(n+1)2^{-m}} 2t/N dt\right)\\
&= 2^{m/2}[t^2/N]_{n2^{-m}}^{(n+1/2)2^{-m}} - 2^{m/2}[t^2/N]_{(n+1/2)2^{-m}}^{(n+1)2^{-m}}\\
&= \frac{2^{-3m/2}}{N}\left(2(n+1/2)^2 - n^2 - (n+1)^2\right) = -\frac{2^{-3m/2-1}}{N}.
\end{aligned}$$

When $n \geq N2^{m-1}$ we have that $f(t) = 2-2t/N$, and using that $\int_0^N \psi_{m,n}(t)dt = 0$ we must get that $w_{m,n} = \frac{2^{-3m/2-1}}{N}$.

For $f(t) = 1/2 + \cos(2\pi t/N)/2$, note first that this has the same coefficients as $\cos(2\pi t/N)/2$, since $\int_0^N \psi_{m,n}(t)dt = 0$. We now get

$$\begin{aligned}
w_{m,n} &= 2^{m/2}\left(\int_{n2^{-m}}^{(n+1/2)2^{-m}} \cos(2\pi t/N)/2dt - \int_{(n+1/2)2^{-m}}^{(n+1)2^{-m}} \cos(2\pi t/N)/2dt\right) \\
&= 2^{m/2}[N\sin(2\pi t/N)/(4\pi)]_{n2^{-m}}^{(n+1/2)2^{-m}} - 2^{m/2}[N\sin(2\pi t/N)/(4\pi)]_{(n+1/2)2^{-m}}^{(n+1)2^{-m}} \\
&= \frac{2^{m/2-2}N}{\pi}\left(2\sin(2\pi(n+1/2)2^{-m}/N) - \sin(2\pi n2^{-m}/N) - \sin(2\pi(n+1)2^{-m}/N)\right).
\end{aligned}$$

There seems to be no more possibilities for simplification here.

13 We get

$$\begin{aligned}
w_{m,n} &= 2^{m/2}\left(\int_{n2^{-m}}^{(n+1/2)2^{-m}} (t/N)^k dt - \int_{(n+1/2)2^{-m}}^{(n+1)2^{-m}} (t/N)^k dt\right) \\
&= 2^{m/2}[t^{k+1}/((k+1)N^k)]_{n2^{-m}}^{(n+1/2)2^{-m}} - 2^{m/2}[t^{k+1}/((k+1)N^k)]_{(n+1/2)2^{-m}}^{(n+1)2^{-m}} \\
&= \frac{2^{-m(k+1/2)}}{(k+1)N^k}\left(2(n+1/2)^{k+1} - n^{k+1} - (n+1)^{k+1}\right).
\end{aligned}$$

The leading term n^{k+1} will here cancel, but the others will not, so there is no room for further simplification here.

14 When we run the 10-level DWT we make a change of coordinates from V_{10} to $V_0 \oplus W_0 \oplus \cdots \oplus W_9$. This means that we must look at the function which has $\boldsymbol{x}$ as coordinates in V_{10}. It is clear that this is the function defined on $[0,1)$ which is $2^{10/2} = 2^5$ on $[0,1/2)]$, and 0 elsewhere. It is clear that this is the function $2^5 2^{-1/2}\phi_{1,0} = 2^5 2^{-1/2} 2^{-1/2}(\phi + \psi) = 2^4(\phi+\psi)$. This has coordinates $(2^4, 2^4, 0, 0, \ldots, 0)$ in $V_0 \oplus W_0 \oplus \cdots \oplus W_9$ (V_0 and W_0 are here one-dimensional).

Section 6.1

1 Let us write $f(t) = \sum_{n=0}^{N-1} c_n\phi_{0,n}(t)$. If k is an integer we have that

$$f(k) = \sum_{n=0}^{N-1} c_n\phi_{0,n}(k) = \sum_{n=0}^{N-1} c_n\phi(k-n).$$

Clearly the only integer for which $\phi(s) \neq 0$ is $s = 0$ (since $\phi(0) = 1$), so that the only n which contributes in the sum is $n = k$. This means that $f(k) = c_k$, so that $[f]_{\boldsymbol{\phi}_0} = (f(0), f(1), \ldots, f(N-1))$.

2 We have that

$$\begin{aligned}\langle \phi_{0,n}, \phi_{0,n}\rangle &= \int_{n-1}^{n+1} (1-|t-n|)^2 dt \\ &= \int_{n-1}^{n+1} \left(1-2|t-n|+(t-n)^2\right) dt \\ &= 2-2+\left[\frac{1}{3}(t-n)^3\right]_{n-1}^{n+1} = \frac{2}{3}.\end{aligned}$$

We also have

$$\begin{aligned}\langle \phi_{0,n}, \phi_{0,n+1}\rangle &= \int_{n}^{n+1} (1-(t-n))(1+(t-n-1))dt = \int_0^1 (1-u)(1+u-1)du \\ &= \int_0^1 (t-t^2)dt = \frac{1}{2}-\frac{1}{3} = \frac{1}{6}.\end{aligned}$$

Finally, the supports of $\phi_{0,n}$ and $\phi_{0,n\pm k}$ are disjoint for $k>1$, so that we must have $\langle \phi_{0,n}, \phi_{0,n\pm k}\rangle = 0$ in that case.

3 We have that

$$\chi_{[-1/2,1/2)} * \chi_{[-1/2,1/2)}(x) = \int_{-\infty}^{\infty} \chi_{[-1/2,1/2)}(t)\chi_{[-1/2,1/2)}(x-t)dt.$$

The integrand here is 1 when $-1/2 < t < 1/2$ and $-1/2 < x-t < 1/2$, or in other words when $\max(-1/2, -1/2+x) < t < \min(1/2, 1/2+x)$ (else it is 0). When $x>0$ this happens when $-1/2+x < t < 1/2$, and when $x<0$ this happens when $-1/2 < t < 1/2+x$. This means that

$$\chi_{[-1/2,1/2)} * \chi_{[-1/2,1/2)}(x) = \begin{cases} \int_{-1/2+x}^{1/2} dt = 1-x & ,x>0 \\ \int_{-1/2}^{1/2+x} dt = 1+x & ,x<0. \end{cases}$$

But this is by definition ϕ.

Section 6.3

1

a The function $\hat{\psi}$ is a sum of the functions $\psi = \phi_{1,1}$, ϕ, and $\phi_{0,1}$ (i.e. we have set $n=0$ in Equation (6.14)). All these are continuous and piecewise linear, and we can write

$$\phi_{1,1}(t) = \begin{cases} 2t & 0 \le t < 1/2 \\ 2 - 2t & 1/2 \le t < 1 \\ 0 & \text{elsewhere} \end{cases}$$

$$\phi(t)(t) = \begin{cases} 1 + t & -1 \le t < 0 \\ 1 - t & 0 \le t < 1 \\ 0 & \text{elsewhere} \end{cases}$$

$$\phi_{0,1}(t) = \begin{cases} t & 0 \le t < 1 \\ 2 - t & 1 \le t < 2 \\ 0 & \text{elsewhere} \end{cases}.$$

It follows that $\hat{\psi}(t) = \phi_{1,1}(t) - \alpha\phi(t) - \beta\phi_{1,1}$ is piecewise linear, and linear on the segments $[-1, 0]$, $[0, 1/2]$, $[1/2, 1]$, $[1, 2]$.

On the segment $[-1, 0]$ only the function ϕ is seen to be nonzero, and since $\phi(t) = 1 + t$ here, we have that $\hat{\psi}(t) = -\alpha(1 + t) = -\alpha - \alpha t$ here.

On the segment $[0, 1/2]$ all three functions are nonzero, and

$$\begin{aligned} \phi_{1,1}(t) &= 2t \\ \phi(t)(t) &= 1 - t \\ \phi_{0,1}(t) &= t \end{aligned}$$

on this interval. This means that $\hat{\psi}(t) = 2t - \alpha(1 - t) - \beta t = (2 + \alpha - \beta)t - \alpha$ on $[0, 1/2]$.

On the segment $[0, 1/2]$ all three functions are nonzero, and

$$\begin{aligned} \phi_{1,1}(t) &= 2 - 2t \\ \phi(t)(t) &= 1 - t \\ \phi_{0,1}(t) &= t \end{aligned}$$

on this interval. This means that $\hat{\psi}(t) = 2-2t-\alpha(1-t)-\beta t = (\alpha-\beta-2)t-\alpha+2$ on $[1/2, 1]$.

On the segment $[1, 2]$ only the function $\phi_{0,1}$ is seen to be nonzero, and since $\phi_{0,1}(t) = 2 - t$ here, we have that $\hat{\psi}(t) = -\beta(2 - t) = \beta t - 2\beta$ here. For all other values of t, $\hat{\psi}$ is zero. This proves the formulas for $\hat{\psi}$ on the different intervals.

b We can write

$$\begin{aligned}\int_0^N \hat{\psi}(t)dt &= \int_{-1}^2 \hat{\psi}(t)dt = \int_{-1}^0 \hat{\psi}(t)dt + \int_0^{1/2} \hat{\psi}(t)dt + \int_{1/2}^1 \hat{\psi}(t)dt + \int_1^2 \hat{\psi}(t)dt \\ &= \int_{-1}^0 (-\alpha - \alpha t)dt + \int_0^{1/2} (2+\alpha-\beta)t - \alpha)dt \\ &+ \int_{1/2}^1 ((\alpha-\beta-2)t - \alpha + 2)dt + \int_1^2 (\beta t - 2\beta)dt \\ &= \left[-\alpha t - \frac{1}{2}\alpha t^2\right]_{-1}^0 + \left[\frac{1}{2}(2+\alpha-\beta)t^2 - \alpha t\right]_0^{1/2} \\ &+ \left[\frac{1}{2}(\alpha-\beta-2)t^2 + (2-\alpha)t\right]_{1/2}^1 + \left[\frac{1}{2}\beta t^2 - 2\beta t\right]_1^2 \\ &= -\alpha + \frac{1}{2}\alpha + \frac{1}{8}(2+\alpha-\beta) - \frac{1}{2}\alpha + \frac{3}{8}(\alpha-\beta-2) + \frac{1}{2}(2-\alpha) + \frac{3}{2}\beta - 2\beta \\ &= \frac{1}{2} - \alpha - \beta,\end{aligned}$$

$\int_0^N t\hat{\psi}(t)dt$ is computed similarly, so that we in the end arrive at $\frac{1}{4} - \beta$.

c The equation system

$$\begin{aligned}\frac{1}{2} - \alpha - \beta &= 0 \\ \frac{1}{4} - \beta &= 0\end{aligned}$$

has the unique solution $\alpha = \beta = \frac{1}{4}$, which we already have found.

2

a In order for ψ to have vanishing moments we must have that $\int \hat{\psi}(t)dt = \int t\hat{\psi}(t)dt = 0$ Substituting $\hat{\psi} = \psi - \alpha\phi_{0,0} - \beta\phi_{0,1}$ we see that, for $k = 0, 1$,

$$\int t^k \left(\alpha\phi_{0,0} + \beta\phi_{0,1}\right) dt = \int t^k \psi(t)dt.$$

The left hand side can here be written

$$\begin{aligned}\int t^k \left(\alpha\phi_{0,0} + \beta\phi_{0,1}\right) dt &= \alpha \int t^k \phi_{0,0} dt + \beta \int t^k \phi_{0,1}(t)dt \\ &= \alpha \int_{-1}^1 t^k (1 - |t|)dt + \beta \int_0^2 t^k (1 - |t-1|)dt = \alpha a_k + \beta b_k.\end{aligned}$$

The right hand side is

$$\int t^k \psi(t)dt = \int t^k \phi_{1,1}(t)dt = \int_0^1 (1 - 2|t - 1/2|)dt = e_k.$$

The following program sets up the corresponding equation systems, and solves it by finding α, β.

```
A=zeros(2);
b=zeros(2,1);
for k=0:1
    A(k+1,:) = [quad(@(t)t.^k.*(1-abs(t)),-1,1)...
                quad(@(t)t.^k.*(1-abs(t-1)),0,2)];
    b(k+1)=quad(@(t)t.^k.*(1-2*abs(t-1/2)),0,1);
end
A\b
```

b Similarly to a., Equation (6.20) gives that

$$\int t^k \left(\alpha\phi_{0,0} + \beta\phi_{0,1} + \gamma\phi_{0,-1} + \delta\phi_{0,2}\right) dt = \int t^k \psi(t) dt.$$

The correspodning equation system is deduced exactly as in a. The following program sets up the corresponding equation systems, and solves it by finding $\alpha, \beta, \gamma, \delta$.

```
A=zeros(4);
b=zeros(4,1);
for k=0:3
    A(k+1,:) = [quad(@(t)t.^k.*(1-abs(t)),-1,1)...
                quad(@(t)t.^k.*(1-abs(t-1)),0,2)...
                quad(@(t)t.^k.*(1-abs(t+1)),-2,0)...
                quad(@(t)t.^k.*(1-abs(t-2)),1,3)];
    b(k+1)=quad(@(t)t.^k.*(1-2*abs(t-1/2)),0,1);
end
coeffs=A\b
```

c The function $\hat{\psi}$ now is supported on $[-2,3]$, and can be plotted as follows:

```
t=linspace(-2,3,100);
plot(t, (t>=0).*(t<=1).*(1-2*abs(t-0.5)) ...
        -coeffs(1)*(t>=-1).*(t<=1).*(1-abs(t))...
        -coeffs(2)*(t>=0).*(t<=2).*(1-abs(t-1))...
        -coeffs(3)*(t>=-2).*(t<=0).*(1-abs(t+1))...
        -coeffs(4)*(t>=1).*(t<=3).*(1-abs(t-2)))
```

e If we define

$$\hat{\psi} = \psi_{0,0} - \sum_{k=0}^{K} \left(\alpha_k \phi_{0,-k} - \beta_k \phi_{0,k+1}\right),$$

we have $2k$ unknowns. These can be determined if we require $2k$ vanishing moments.

Section 7.1

2 You can set for instance $H_0 = \{1/4, \underline{1/2}, 1/4\}$, and $H_1 = \{\underline{1}\}$ (when you write down the corresponding matrix you will see that $A_{0,1} = 1/2$, $A_{1,0} = 0$, so that the matrix is not symmetric)

3 The Haar wavelet is a counterexample!

Section 7.3

1 As before we obtain that

$$\left\{1, \left\{\cos\left(2\pi\frac{n}{2N-2}k\right)\right\}_{n=1}^{N-2}, (-1)^k\right\}$$

is an orthogonal basis of eigenvectors, which also can be written

$$\left\{\cos\left(2\pi\frac{n}{2N-2}k\right)\right\}_{n=0}^{N-1}.$$

Section 7.4

1

a We have that $H_0 = \frac{1}{5}\{1, 1, \underline{1}, 1, 1\}$, and $H_1 = \frac{1}{3}\{-1, \underline{1}, -1\}$. The frequency responses are

$$\begin{aligned}
\lambda_{H_0}(\omega) &= \frac{1}{5}e^{2i\omega} + \frac{1}{5}e^{i\omega} + \frac{1}{5} + \frac{1}{5}e^{-i\omega}\frac{1}{5}e^{-2i\omega} \\
&= \frac{2}{5}\cos(2\omega) + \frac{2}{5}\cos\omega + \frac{1}{5} \\
\lambda_{H_1}(\omega) &= -\frac{1}{3}e^{i\omega} + \frac{1}{3} - \frac{1}{3}e^{-i\omega} = -\frac{2}{3}\cos\omega + \frac{1}{3}.
\end{aligned}$$

Both filters are symmetric, and we have that `h0`$=(1/5, 1/5, 1/5, 1/5, 1/5)$, and `h1`$=(-1/3, 1/3, -1/3)$.

b We have that $G_0 = \{1/4, \underline{1/2}, 1/4\}$, and $G_1 = \{1/16, -1/4, \underline{3/8}, -1/4, 1/16\}$. The frequency responses are

$$\begin{aligned}
\lambda_{G_0}(\omega) &= \frac{1}{4}e^{i\omega} + \frac{1}{2} + \frac{1}{4}e^{-i\omega} \\
&= \frac{1}{2}\cos(\omega) + \frac{1}{2} \\
\lambda_{G_1}(\omega) &= \frac{1}{16}e^{2i\omega} - \frac{1}{4}e^{i\omega} + \frac{3}{8} - \frac{1}{4}e^{-i\omega}\frac{1}{16}e^{-2i\omega} \\
&= \frac{1}{8}\cos(2\omega) - \frac{1}{2}\cos\omega + \frac{3}{8}.
\end{aligned}$$

Both filters are symmetric, and we have that `g0`$=(1/4, 1/2, 1/4)$, and `g1`$=(1/16, -1/4, 3/8, -1/4, 1/16)$.

2

a We have that $H_0 = \{1/16, 1/4, \underline{3/8}, 1/4, 1/16\}$, and $H_1 = \{-1/4, \underline{1/2}, -1/4\}$. The frequency responses are

$$\begin{aligned}
\lambda_{H_0}(\omega) &= \frac{1}{16}e^{2i\omega} + \frac{1}{4}e^{i\omega} + \frac{3}{8} + \frac{1}{4}e^{-i\omega}\frac{1}{16}e^{-2i\omega} \\
&= \frac{1}{8}\cos(2\omega) + \frac{1}{2}\cos\omega + \frac{3}{8} \\
\lambda_{H_1}(\omega) &= -\frac{1}{4}e^{i\omega} + \frac{1}{2} - \frac{1}{4}e^{-i\omega} \\
&= -\frac{1}{2}\cos(\omega) + \frac{1}{2}.
\end{aligned}$$

The two first rows in $P_{\mathcal{C}_1 \leftarrow \phi_1}$ are

$$\begin{pmatrix} 3/8 & 1/4 & 1/16 & 0 & \cdots & 1/16 & 1/4 \\ -1/4 & 1/2 & -1/4 & 0 & \cdots & 0 & 0 \end{pmatrix}$$

The remaining rows are obtained by translating these in alternating order.

b We have that $G_0 = \frac{1}{3}\{1, \underline{1}, 1\}$, and $G_1 = \frac{1}{5}\{1, -1, \underline{1}, -1, 1\}$. The frequency responses are

$$\begin{aligned}
\lambda_{G_0}(\omega) &= \frac{1}{3}e^{i\omega} + \frac{1}{3} + \frac{1}{3}e^{-i\omega} = \frac{2}{3}\cos\omega + \frac{1}{3} \\
\lambda_{G_1}(\omega) &= \frac{1}{5}e^{2i\omega} - \frac{1}{5}e^{i\omega} + \frac{1}{5} - \frac{1}{5}e^{-i\omega}\frac{1}{5}e^{-2i\omega} \\
&= \frac{2}{5}\cos(2\omega) - \frac{2}{5}\cos\omega + \frac{1}{5}
\end{aligned}$$

The two first columns in $P_{\phi_1 \leftarrow \mathcal{C}_1}$ are

$$\begin{pmatrix} 1/3 & -1/5 \\ 1/3 & 1/5 \\ 0 & -1/5 \\ 0 & 1/5 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ 1/3 & 1/5 \end{pmatrix}$$

The remaining columns are obtained by translating these in alternating order.

3 The following code can be used:

```
function playDWTfilterslower(m,h0,h1,g0,g1)
  [S fs]=wavread('castanets');
  newx=DWTImpl(h0,h1,S(1:2^17,1),m);
  len=length(newx);
  newx((len/2^m+1):length(newx))=zeros(length(newx)-len/2^m,1);
  newx=IDWTImpl(g0,g1,newx,m);
  playerobj=audioplayer(newx,fs);
  playblocking(playerobj);
```

4

a The following code can be used:

```
function playDWTfilterslowerdifference(m,h0,h1,g0,g1)
  [S fs]=wavread('castanets');
  newx=DWTImpl(h0,h1,S(1:2^17,1),m);
  len=length(newx);
  newx(1:(len/2^m))=zeros(len/2^m,1);
  newx=IDWTImpl(g0,g1,newx,m);
  playerobj=audioplayer(newx,fs);
  playblocking(playerobj);
```

b After the replacements in the function `playDWTall`, we get code which looks like this

```
function playDWTalldifference(m)
disp('Haar wavelet');
playDWTlowerdifference(m);
disp('Wavelet for piecewise linear functions');
playDWTfilterslowerdifference(m,sqrt(2)*[1],...
    sqrt(2)*[-1/2 1 -1/2],...
    [1/2 1 1/2]/sqrt(2),...
    [1]/sqrt(2));
disp('Wavelet for piecewise linear functions, alternative  version');
playDWTfilterslowerdifference(m,...
   sqrt(2)*[-1/8 1/4 3/4 1/4 -1/8],...
   sqrt(2)*[-1/2 1 -1/2],...
   [1/2 1 1/2]/sqrt(2),...
   [-1/8 -1/4 3/4 -1/4 -1/8]/sqrt(2));
```

5

a The code which can be used looks like this:

```
newx=IDWTImpl([1/2 1 1/2]/sqrt(2),[1]/sqrt(2),...
              [-coeffs(1); -coeffs(2); -coeffs(4); 0; 0; 0; 0; ...
               -coeffs(3); 1; 0; 0; 0; 0; 0; 0; 0],1);
```

b The code which can be used looks like this:

```
g1=[newx((length(newx)-2):length(newx))' newx(1:6)'] % compact filter notation
g0=[1/2 1 1/2]/sqrt(2);
omega=linspace(0,2*pi,100);
plot(omega,g1(5)+g1(6)*2*cos(omega)+g1(7)*2*cos(2*omega)...
           +g1(8)*2*cos(3*omega)+g1(9)*2*cos(4*omega))
```

c The code which can be used looks like this:

```
alpha=1/(g0(2)*g1(5)+2*g0(3)*g1(6));
h0=alpha*(-1).^(1:(length(g1))).*g1;
h1=alpha*(-1).^(0:(length(g0)-1)).*g0;
```

d The code which can be used looks like this:

```
for m=1:4
    m
    playDWTfilterslower(m,h0,h1,g0,g1);
    playDWTfilterslowerdifference(m,h0,h1,g0,g1);
end
```

7

a `c` and `w` represent the coordinates in the wavelet bases ϕ_0 and ψ_0. The code runs a Haar wavelet transform. The sound is normalized so that the sound samples lie in the range between -1 and 1, and the resulting sound is played. The sound is split into two parts, and `c` represents a low-resolution version of the sound (with half the number of samples), so that we first will hear the sound played at double pace. After this we will hear the detail `w` in the sound, also played at double pace. We should also be able to recognize the sound from this detail.

b This corresponds to reconstructing a low-resolution approximation of the sound.

Section 8.3

Section 8.5

Section 8.6

Section 9.2

2 The code can look like this:

```
function xnew=DWTImpl53(x,m)
lambda1=-1;
lambda2=0.125;
alpha=0.5;
beta=2;

for mres=1:m
  len=length(x)/2^(mres-1);
  x(1:2:(len-1))=x(1:2:(len-1))/alpha;
  x(2:2:len)=x(2:2:len)/beta;
  x(1:len)=liftingstepapplyB(-lambda2,x(1:len));
  x(1:len)=liftingstepapplyA(-lambda1,x(1:len));

  % Reorganize the coefficients
  l=x(1:2:(len-1));
  h=x(2:2:len);
  x(1:len)=[l h];
end
xnew=x;
```

```
function x=IDWTImpl53(xnew,m)
  lambda1=-1;
  lambda2=0.125;
  alpha=0.5;
  beta=2;

  for mres=m:(-1):1
    len=length(xnew)/2^(mres-1);

    % Reorganize the coefficients first
    l=xnew(1:(len/2));
    h=xnew((len/2+1):len);
    xnew(1:2:(len-1))=l;
    xnew(2:2:len)=h;

    xnew(1:len)=liftingstepapplyA(lambda1,xnew(1:len));
    xnew(1:len)=liftingstepapplyB(lambda2,xnew(1:len));
    xnew(1:2:(len-1))=xnew(1:2:(len-1))*alpha;
    xnew(2:2:len)=xnew(2:2:len)*beta;
  end
  x=xnew;
```

3 The code can look like this:

```
function xnew=DWTImpl97(x,m)
```

```
lambda1=-0.586134342059950;
lambda2=-0.668067171029734;
lambda3=0.070018009414994;
lambda4=1.200171016244178;
alpha=0.869864451624777;
beta=1.149604398860250;
for mres=1:m
  len=length(x)/2^(mres-1);
  x(1:2:(len-1))=x(1:2:(len-1))/alpha;
  x(2:2:len)=x(2:2:len)/beta;
  x(1:len)=liftingstepapplyB(-lambda4,x(1:len));
  x(1:len)=liftingstepapplyA(-lambda3,x(1:len));
  x(1:len)=liftingstepapplyB(-lambda2,x(1:len));
  x(1:len)=liftingstepapplyA(-lambda1,x(1:len));

  % Reorganize the coefficients
  l=x(1:2:(len-1));
  h=x(2:2:len);
  x(1:len)=[l h];
end
xnew=x;
```

```
function x=IDWTImpl97(xnew,m)
  lambda1=-0.586134342059950;
  lambda2=-0.668067171029734;
  lambda3=0.070018009414994;
  lambda4=1.200171016244178;
  alpha=0.869864451624777;
  beta=1.149604398860250;
  for mres=m:(-1):1
    len=length(xnew)/2^(mres-1);

    % Reorganize the coefficients first
    l=xnew(1:(len/2));
    h=xnew((len/2+1):len);
    xnew(1:2:(len-1))=l;
    xnew(2:2:len)=h;

    xnew(1:len)=liftingstepapplyA(lambda1,xnew(1:len));
    xnew(1:len)=liftingstepapplyB(lambda2,xnew(1:len));
    xnew(1:len)=liftingstepapplyA(lambda3,xnew(1:len));
    xnew(1:len)=liftingstepapplyB(lambda4,xnew(1:len));
    xnew(1:2:(len-1))=xnew(1:2:(len-1))*alpha;
    xnew(2:2:len)=xnew(2:2:len)*beta;
```

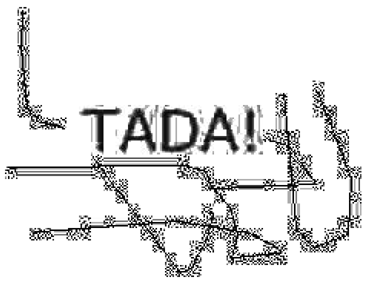

Figure 12.21: Secret message revealed!

```
  end
  x=xnew;
```

Section 10.2

1 The following code can be used:

```
255*((img(:,:,1)+img(:,:,2)+img(:,:,3))/3>=128
```

2 The following code can be used:

```
imwrite(uint8(contrastadjust(img,0.01)),'contrast4.png','png');
```

3

a The code could look as follows:

```
function newimg=contrastadjust0(img,n)
  newimg = img/255; % Maps the pixel values to [0,1]
  newimg = atan(n*(newimg-1/2))/(2*atan(n/2)) + 1/2;
  newimg = newimg*255; % Maps the values back to [0,255]
```

b The following code can be used:

```
imwrite(uint8(contrastadjust0(img,10)),'contrast3.png','png');
```

4

b The secret message is revealed in Figure 12.21.

5 The following code can be used:

```
excerpt=255*mapto01(sqrt(img(:,:,1).^2+img(:,:,2).^2+img(:,:,3).^2));
excerpt=excerpt(170:340,170:340);
imwrite(uint8(excerpt),'ggl2part.png','png');
imwrite(uint8(smooth(excerpt,[1 2 1; 2 4 2; 1 2 1]/16)),'smooth1.png','png');
newmolecule=conv([1 3 3 1],[1 3 3 1]);
newmolecule=newmolecule'*newmolecule/4096;
imwrite(uint8(smooth(excerpt,newmolecule)),'smooth2.png','png');
```

6 The following code can be used:

```
excerpt=255*mapto01(sqrt(img(:,:,1).^2+img(:,:,2).^2+img(:,:,3).^2));
excerpt=excerpt(170:340,170:340);

res=smooth(excerpt,[0 0 0; -1 0 1; 0 0 0]/2);
imwrite(uint8(contrastadjust0(255*mapto01(res),50)),'px31.png','png');

res1=res;
res2=smooth(excerpt,[0 -1 0; 0 0 0; 0 1 0]/2);
grad=sqrt(res1.^2+res2.^2);
imwrite(uint8(grad),'grad1.png','png');
imwrite(uint8(255*mapto01(grad)),'grad2.png','png');
imwrite(uint8(contrastadjust0(255*mapto01(grad),50)),'grad.png','png');

imwrite(uint8(contrastadjust(255*mapto01(res1),0.01)),'px3.png','png');
imwrite(uint8(contrastadjust(255*mapto01(res2),0.01)),'py.png','png');

resxx=smooth(res1,[0 0 0; -1 0 1; 0 0 0]/2);
resxy=smooth(res1,[0 -1 0; 0 0 0; 0 1 0]/2);
resyy=smooth(res2,[0 -1 0; 0 0 0; 0 1 0]/2);
imwrite(uint8(contrastadjust0(255*mapto01(resxx),100)),'dxx.png','png');
imwrite(uint8(contrastadjust0(255*mapto01(resxy),100)),'dxy.png','png');
imwrite(uint8(contrastadjust0(255*mapto01(resyy),100)),'dyy.png','png');
```

Section 11.1

2 We have that the computational molecule of $I \otimes T$ is

$$\begin{aligned}\text{rev}(\underline{1}) \otimes \text{rev}(1/2, \underline{0}, -1/2) &= (\underline{1}) \otimes (-1/2, \underline{0}, 1/2)\\ &= \begin{pmatrix} 0 \\ \underline{1} \\ 0 \end{pmatrix} \begin{pmatrix} -1/2 & \underline{0} & 1/2 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}\end{aligned}$$

This is seen to coincide with the molecule from Equation 10.7 in Example 10.19.

3 We have that the computational molecule of $T \otimes T$ is

$$\begin{aligned}
&\text{rev}(1/2, \underline{0}, -1/2) \otimes \text{rev}(1/2, \underline{0}, -1/2) \\
&= (-1/2, \underline{0}, 1/2) \otimes (-1/2, \underline{0}, 1/2) \\
&= \begin{pmatrix} -1/2 \\ \underline{0} \\ 1/2 \end{pmatrix} \begin{pmatrix} -1/2 & \underline{0} & 1/2 \end{pmatrix} = \frac{1}{4} \begin{pmatrix} 1 & 0 & -1 \\ 0 & \underline{0} & 0 \\ -1 & 0 & 1 \end{pmatrix}
\end{aligned}$$

This is seen to coincide with the molecule from Equation 10.10 in Example 10.19.

6 We have that

$$\begin{aligned}
F(\alpha \boldsymbol{x}_1 + \beta \boldsymbol{x}_2, \boldsymbol{y}) &= (\alpha \boldsymbol{x}_1 + \beta \boldsymbol{x}_2) \otimes \boldsymbol{y} = (\alpha \boldsymbol{x}_1 + \beta \boldsymbol{x}_2)\boldsymbol{y}^T \\
&= \alpha \boldsymbol{x}_1 \boldsymbol{y}^T + \beta \boldsymbol{x}_2 \boldsymbol{y}^T = \alpha(\boldsymbol{x}_1 \otimes \boldsymbol{y}) + \beta(\boldsymbol{x} \otimes \boldsymbol{y}) \\
&= \alpha F(\boldsymbol{x}_1, \boldsymbol{y}) + \beta F(\boldsymbol{x}_1, \boldsymbol{y}).
\end{aligned}$$

The second statement follows similarly.

7

a Multiplicaton with the matrix

$$T = \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 & 0 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 1 & 0 & 0 & \cdots & 0 & 0 & 0 \end{pmatrix}$$

reverses the elements in a vector. This means that

$$((T \otimes I)(\boldsymbol{x} \otimes \boldsymbol{y}))_{i,j} = ((T\boldsymbol{x}) \otimes \boldsymbol{y})_{i,j} = (T\boldsymbol{x})_i y_j = x_{M-1-i} y_j = (\boldsymbol{x} \otimes \boldsymbol{y})_{M-1-i,j}.$$

This means that also $((T \otimes I)X)_{i,j} = X_{M-1-i,j}$ for all X, so that $T \otimes I$ reverses rows, and thus is a solution to a..

b Similarly one shows that $I \otimes T$ reverses columns, and is thus a solution to b..

c It turns out that it is impossible to find S and T so that transposing a matrix X corresponds to computing $(S \otimes T)X$. To see why, S and T would need to fulfill

$$(S \otimes T)(\boldsymbol{e}_i \otimes \boldsymbol{e}_j) = (S\boldsymbol{e}_i) \otimes (T\boldsymbol{e}_j) = \boldsymbol{e}_j \otimes \boldsymbol{e}_i,$$

since $\boldsymbol{e}_j \otimes \boldsymbol{e}_i$ is the transpose of $\boldsymbol{e}_i \otimes \boldsymbol{e}_j$. This would require that $S\boldsymbol{e}_i = \boldsymbol{e}_j$ for all i, j, which is impossible.

9

a The computational molecule of $T \otimes T$ is

$$\text{rev}(1,\underline{2},1) \otimes \text{rev}(1,\underline{2},1) = (1,\underline{2},1) \otimes (1,\underline{2},1) = \begin{pmatrix} 1 \\ \underline{2} \\ 1 \end{pmatrix} \begin{pmatrix} 1 & \underline{2} & 1 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 1 \\ 2 & \underline{4} & 2 \\ 1 & 2 & 1 \end{pmatrix}.$$

b We get that

$$\begin{aligned} A &= \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \begin{pmatrix} 3 & 2 & 1 \end{pmatrix} + \begin{pmatrix} 2 \\ 2 \\ 2 \end{pmatrix} \begin{pmatrix} 1 & 4 & 2 \end{pmatrix} \\ &= \begin{pmatrix} 3 & 2 & 1 \\ 6 & 4 & 2 \\ 9 & 6 & 3 \end{pmatrix} + \begin{pmatrix} 2 & 8 & 4 \\ 2 & 8 & 4 \\ 2 & 8 & 4 \end{pmatrix} = \begin{pmatrix} 5 & 10 & 5 \\ 8 & 12 & 6 \\ 11 & 14 & 7 \end{pmatrix}. \end{aligned}$$

c We need to compute $(T \otimes T)A = TAT^T$, which corresponds to first applying T to every column in the image, and then applying T to every row in the resulting image. If we apply T to every column in the image we first get the matrix $TA = \begin{pmatrix} 29 & 46 & 23 \\ 32 & 48 & 24 \\ 35 & 50 & 25 \end{pmatrix}$. If we apply the filter to the rows here we get $TAT^T = \begin{pmatrix} 127 & 144 & 121 \\ 136 & 152 & 128 \\ 145 & 160 & 135 \end{pmatrix}$. Since the filter which is applied is a lowpass filter, the new image should look a bit more smooth than the original image.

Section 11.2

1 The following code can be used:

```
function newX=FFT2Impl(X)
  for k=1:2
      for s=1:size(X,2)
          X(:,s)=FFTImpl(X(:,s));
      end
      X=X';
  end
  newX=X;
```

```
function newX=IFFT2Impl(X)
  for k=1:2
      for s=1:size(X,2)
          X(:,s)=IFFTImpl(X(:,s));
      end
      X=X';
  end
  newX=X;
```

```
function newX=DCT2Impl(X)
  for k=1:2
      for s=1:size(X,2)
          X(:,s)=DCTImpl(X(:,s));
      end
      X=X';
  end
  newX=X;
```

```
function newX=IDCT2Impl(X)
  for k=1:2
      for s=1:size(X,2)
          X(:,s)=IDCTImpl(X(:,s));
      end
      X=X';
  end
  newX=X;
```

2 The following code can be used:

```
function newX=transform2jpeg(X)
  numblocksx=size(X,1)/8;
  numblocksy=size(X,2)/8;
  for bx=0:(numblocksx-1)
    for by=0:(numblocksy-1)
      X((1+8*bx):8*(bx+1),(1+8*by):8*(by+1))=...
          DCT2Impl(X((1+8*bx):8*(bx+1),(1+8*by):8*(by+1)));
    end
  end
  newX=X;
```

```
function X=transform2invjpeg(newX)
  numblocksx=size(newX,1)/8;
  numblocksy=size(newX,2)/8;
  for bx=0:(numblocksx-1)
    for by=0:(numblocksy-1)
      newX((1+8*bx):8*(bx+1),(1+8*by):8*(by+1))=...
           IDCT2Impl(newX((1+8*bx):8*(bx+1),(1+8*by):8*(by+1)));
    end
  end
  X=newX;
```

3 In the first part of the code one makes a change of coordinates with the DFT. More precisely, this is a change of coordinates on a tensor product, as we have defined it. In the last part the change of coordinates is performed the opposite way. Both these change of coordinates is performed is performed the way we have described them, first on the rows in the matrix, then on the columns. The parameter `threshold` is used to neglect DFT-coefficients which are below a certain value. We have seen that this can give various visual artefacts in the image, even though the main contents of the image still may be visible. If we increase `threshold`, these artefacts will be more dominating since we then neglect many DFT-coefficients.

Section 12.1

2 The following code can be used:

```
function Xnew=DWT2HaarImpl(X,m)
  for mres=1:m
    l1=size(X,1)/2^(mres-1);
    l2=size(X,2)/2^(mres-1);
    for s=1:l2
      X(1:l1,s)=DWTHaarImpl(X(1:l1,s),1);
    end
    X=X';

    for s=1:l1
      X(1:l2,s)=DWTHaarImpl(X(1:l2,s),1);
    end
    X=X';
  end
  Xnew=X;
```

```
function X=IDWT2HaarImpl(Xnew,m)
  for mres=m:(-1):1
    l1=size(Xnew,1)/2^(mres-1);
    l2=size(Xnew,2)/2^(mres-1);
    for s=1:l2
      Xnew(1:l1,s)=IDWTHaarImpl(Xnew(1:l1,s),1);
    end
    Xnew=Xnew';

    for s=1:l1
      Xnew(1:l2,s)=IDWTHaarImpl(Xnew(1:l2,s),1);
    end
    Xnew=Xnew';
  end
  X=Xnew;
```

3

a The following code achieves the task:

```
function showDWTlowerHaar(m)
  img=double(imread('lena.png','png'));
  for k=1:3
      img(:,:,k)=DWT2HaarImpl(img(:,:,k),m);
  end
  [l1,l2,l3]=size(img);
  tokeep=img(1:(l1/(2^m)),1:(l2/(2^m)),:);
  img=zeros(size(img));
  img(1:(l1/(2^m)),1:(l2/(2^m)),:)=tokeep;
  for k=1:3
      img(:,:,k)=IDWT2HaarImpl(img(:,:,k),m);
  end
  imshow(uint8(255*mapto01(img)));
```

c There is no reason to believe that image samples returned by the function lie in $[0, 255]$. You can check this by printing the maximum value in the returned array on screen inside this method.

4 The following code can be used

```
function showDWTlowerdifferenceHaar(m)
  img=double(imread('lena.png','png'));
  for k=1:3
      img(:,:,k)=DWT2HaarImpl(img(:,:,k),m);
  end
  [l1,l2,l3]=size(img);
  img(1:(l1/2^m),1:(l2/2^m),:)=zeros(l1/2^m,l1/2^m,3);
  for k=1:3
      img(:,:,k)=IDWT2HaarImpl(img(:,:,k),m);
  end
  imshow(uint8(255*mapto01(img)));
```

5

a The following code can be used:

```
function Xnew=DWT2Impl53(X,m)
  for mres=1:m
    l1=size(X,1)/2^(mres-1);
    l2=size(X,2)/2^(mres-1);
    for s=1:l2
      X(1:l1,s)=DWTImpl53(X(1:l1,s),1);
    end
    X=X';
    for s=1:l1
      X(1:l2,s)=DWTImpl53(X(1:l2,s),1);
    end
    X=X';
  end
  Xnew=X;
```

```
function X=IDWT2Impl53(Xnew,m)
  for mres=m:(-1):1
    l1=size(Xnew,1)/2^(mres-1);
    l2=size(Xnew,2)/2^(mres-1);
    for s=1:l2
      Xnew(1:l1,s)=IDWTImpl53(Xnew(1:l1,s),1);
    end
    Xnew=Xnew';

    for s=1:l1
      Xnew(1:l2,s)=IDWTImpl53(Xnew(1:l2,s),1);
    end
    Xnew=Xnew';
  end
  X=Xnew;
```

b The following code can be used:

```
function Xnew=DWT2Impl97(X,m)
  for mres=1:m
    l1=size(X,1)/2^(mres-1);
    l2=size(X,2)/2^(mres-1);
    for s=1:l2
      X(1:l1,s)=DWTImpl97(X(1:l1,s),1);
    end
    X=X';

    for s=1:l1
      X(1:l2,s)=DWTImpl97(X(1:l2,s),1);
    end
    X=X';
  end
  Xnew=X;
```

```
function X=IDWT2Impl97(Xnew,m)
  for mres=m:(-1):1
    l1=size(Xnew,1)/2^(mres-1);
    l2=size(Xnew,2)/2^(mres-1);
    for s=1:l2
      Xnew(1:l1,s)=IDWTImpl97(Xnew(1:l1,s),1);
    end
    Xnew=Xnew';

    for s=1:l1
      Xnew(1:l2,s)=IDWTImpl97(Xnew(1:l2,s),1);
    end
    Xnew=Xnew';
  end
  X=Xnew;
```

6 The following code can be used:

```
function showDWTlower53(m)
  img=double(imread('lena.png','png'));
  for k=1:3
      img(:,:,k)=DWT2Impl53(img(:,:,k),m);
  end
  [l1,l2,l3]=size(img);
  tokeep=img(1:(l1/(2^m)),1:(l2/(2^m)),:);
  img=zeros(size(img));
  img(1:(l1/(2^m)),1:(l2/(2^m)),:)=tokeep;
  for k=1:3
      img(:,:,k)=IDWT2Impl53(img(:,:,k),m);
  end
  imshow(uint8(255*mapto01(img)));
```

```
function showDWTlower97(m)
  img=double(imread('lena.png','png'));
  for k=1:3
      img(:,:,k)=DWT2Impl97(img(:,:,k),m);
  end
  [l1,l2,l3]=size(img);
  tokeep=img(1:(l1/(2^m)),1:(l2/(2^m)),:);
  img=zeros(size(img));
  img(1:(l1/(2^m)),1:(l2/(2^m)),:)=tokeep;
  for k=1:3
      img(:,:,k)=IDWT2Impl97(img(:,:,k),m);
  end
  imshow(uint8(255*mapto01(img)));
```

7

a The following code can be used:

```
function showDWTlowerdifference53(m)
  img=double(imread('lena.png','png'));
  for k=1:3
      img(:,:,k)=DWT2Impl53(img(:,:,k),m);
  end
  [l1,l2,l3]=size(img);
  img(1:(l1/2^m),1:(l2/2^m),:)=zeros(l1/2^m,l1/2^m,3);
  for k=1:3
      img(:,:,k)=IDWT2Impl53(img(:,:,k),m);
  end
  imshow(uint8(255*mapto01(img)));
```

```
function showDWTlowerdifference97(m)
  img=double(imread('lena.png','png'));
  for k=1:3
      img(:,:,k)=DWT2Impl97(img(:,:,k),m);
  end
  [l1,l2,l3]=size(img);
  img(1:(l1/2^m),1:(l2/2^m),:)=zeros(l1/2^m,l1/2^m,3);
  for k=1:3
      img(:,:,k)=IDWT2Impl97(img(:,:,k),m);
  end
  imshow(uint8(255*mapto01(img)));
```

b After the replacements in the function `showDWTall`, we get code which looks like this

```
function showDWTalldifference(m)
  disp('Haar wavelet');
  showDWTlowerdifferenceHaar(m);
  disp('5/3 wavelet');
  showDWTlowerdifference53(m);
  disp('9/7 wavelet');
  showDWTlowerdifference97(m);
```

Section 12.2

Nomenclature

$\breve{\boldsymbol{x}}$	Symmetric extension of a vector
$\hat{\boldsymbol{x}}$	DFT of the vector $\boldsymbol{x}$
λ_S	Continuous requency response of a filter
$\lambda_{S,n}$	Vector frequency response of a filter
ω	Angular frequency
$\oplus$	Direct sum
$\otimes$	Tensor product
$\boldsymbol{x}^{(e)}$	Vector of even samples
$\boldsymbol{x}^{(o)}$	Vector of odd samples
$\boldsymbol{\phi}_m$	Wavelet basis, before transform
E_d	Filter which delays with d samples
F_N	$N \times N$-DCT matrix
F_N	$N \times N$-Fourier matrix
$O(f(\boldsymbol{x}))$	Order of a function
$O(N)$	Order of an algorithm
S^f	Matrix with the columns reversed
$V_{N,T}$	N'th order Fourier space
$W_m^{(0,1)}$	Resolution m Complementary wavelet space, LH
$W_m^{(1,0)}$	Resolution m Complementary wavelet space, HL
$W_m^{(1,1)}$	Resolution m Complementary wavelet space, HH
$\mathcal{C}_m$	Wavelet basis, after transform, reordered

$\mathcal{D}_N = \{\boldsymbol{d}_0, \boldsymbol{d}_1, \cdots, \boldsymbol{d}_{N-1}\}$ N-point DCT basis for $\mathbb{R}^N$

$\mathcal{D}_{N,T}$ Order N Fourier basis for $V_{N,T}$

$\mathcal{E}_N = \{\boldsymbol{e}_0, \boldsymbol{e}_1, \cdots, \boldsymbol{e}_{N-1}\}$ Standard basis for $\mathbb{R}^N$

$\mathcal{F}_{N,T}$ Order N complex Fourier basis for $V_{N,T}$

Mathematics index

AD conversion, 47
algebra, 87
analysis, 20
  equations, 20
angular frequency, 94

bandpass filter, 107
bit rate, 47
block, 285
block diagonal matrices, 204
block matrix, 75

change of coordinates
  in tensor product, 297
Complex Fourier coefficients, 29
continuous sound, 8
convolution
  sequences, 101
  vectors, 100
coordinate matrix, 296

DCT basis, 137
DCT coefficients, 137
DCT matrix, 137
detail space, 163
DFT coefficients, 56
digital
  sound, 8, 46
digital filter, 86
Direct sum
  canonical basis, 170
  linear transformations, 180
direct sum
  vector spaces, 163
Dirichlet conditions, 21
Discrete Cosine transform, 137
Discrete Wavelet Transform, 171

elementary lifting matrix
  even type, 249
  odd type, 249
error-resilient, 285

FFT factorization, 76
filter coefficients, 84
Fourier analysis, 17
Fourier coefficients, 19
Fourier domain, 20
Fourier matrix, 56
Fourier series, 18
Fourier space, 18
Fourier transform
  discrete, 56
frequency domain, 20
frequency response
  continuous, 94
  vector, 86

Haar wavelet, 172
highpass filter, 107

ideal highpass filter, 107
ideal lowpass filter, 107
IDFT, 58
impulse response, 90
interpolating polynomial, 69
interpolation formula, 72
  ideal
    periodic functions, 72
Inverse Discrete Wavelet Transform, 171

least square error, 18
length, 100
lifting factorization, 251
linear phase
  digital filters, 132
lowpass filter, 107
LTI filters, 119

mother wavelets, 166
MRA-matrix, 205
multiresolution analysis, 177
multiresolution model, 155

Order N complex Fourier basis for $V_{N,T}$, 28
Order N Fourier basis for $V_{N,T}$, 20

Order of a function, 78
Order of an algorithm, 78

Parallel computing
    with the DCT, 147
    with the DWT, 285
    with the FFT, 79
    with the revised FFT, 151
perfect reconstruction system, 211
pure digital tones, 55

samples, 47
sampling, 47
    frequency, 47
    period, 47
    rate, 47
scaling function, 159, 177
separable extension, 303
sound channel, 49
square wave, 15
subband
    HH, 306
    HL, 306
    LH, 306
    LL, 307
symmetric extension, 65
synthesis, 20
    equation, 20
    vectors, 57

tensor product, 287
    of function spaces, 303
    of functions, 303
    of matrices, 289
    of vectors, 289
tile, 285
time domain, 20
time-invariant, 119
Toeplitz matrix, 83
    circulant, 83
triangle wave, 15

Wavelet basis, after transform, reordered, 204
Wavelet basis, before transform, 204
wavelet coefficients, 171

Index for MATLAB commands

audioplayer, 49

conv, 101, 116

dct, 141
double, 268

fft, 77

idct, 141
ifft, 77
imageview, 269
imread, 268
imwrite, 268

play, 48
playblocking, 48

rand, 52

uint8, 268

wavread, 49
wavwrite, 49

Bibliography

[1] *ISO/IEC FDIS 15444-1. JPEG2000 Part 1 final draft international standard*, 2000.

[2] C. M. Brislawn. Fingerprints go digital. *Notices of the AMS*, 42(11):1278–1283, 1995.

[3] B. A. Cipra. The best of the 20th century: Editors name top 10 algorithms. *SIAM News*, 33(4). http://www.uta.edu/faculty/rcli/TopTen/topten.pdf.

[4] J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex fourier series. *Math. Comput.*, 19:297Ð301, 1965.

[5] FBI. WSQ gray-scale fingerprint image compression specification. Technical report, IAFIS-IC, 1993.

[6] M. W. Frazier. *An Introduction to Wavelets Through Linear Algebra.* Springer, 1999.

[7] ISI/IEC. Information technology - coding of moving pictures and associated audio for digital storage media at up to about 1.5 mbit/s. Technical report, ISO/IEC, 1993.

[8] D. C. Lay. *linear algebra and its applications (4th edition).* Addison Wesley, 2011.

[9] P. Noll. MPEG digital audio coding. *IEEE Signal processing magazine*, pages 59–81, September 1997.

[10] D. Pan. A tutorial on MPEG/audio compression. *IEEE Multimedia*, pages 60–74, Summer 1995.

[11] W. B. Pennebaker and J. L. Mitchell. *JPEG Still Image Data Compression Standard.* Van Nostrand Reihnold, 1993.

[12] J. G. Proakis and D. G. Manolakis. *Digital Signal Processing. Principles, algorithms, and applications. Fourth edition.* Pearson, 2007.

[13] J. H. Rothweiler. Polyphase quadrature filters - a new subband coding technique. *ICASSP 83, Boston*, pages 1280–1283, 1983.

[14] D. S. Taubman and M. W. Marcellin. *JPEG2000. Image compression. Fundamentals, standards and practice.* Kluwer Academic Publishers, 2002.