Appendix C

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 $n_1/\nu_1 = n_2\nu_2$, i.e. so that there exists a common period between the two. This common period will also 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

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

2 . The space V_{N_1,T_1} is spanned by pure tones with frequencies $1/T_1,\dots,N_1/T_1$, while V_{N_2,T_2} is spanned by pure tones with frequencies $1/T_2,\dots,N_2/T_2$. We must have that the first set of frequencies is contained in the second. This is achieved if and only if $1/T_1 = k/T_2$ for some integer k, and also $N_1/T_1 \le N_2/T_2$. In other words, T_2/T_1 must be an integer, and $T_2/T_1 \le N_2/N_1$.

Section 1.4

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

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

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

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

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 $\|e^{2\pi i n t/T}\| = 1$.

2 . We have that

$$\begin{aligned} f(t) = \sin^2(2\pi t/T) &= \left(\frac{1}{2i}(e^{2\pi i t/T} - e^{-2\pi i t/T})\right)^2 \\ &= -\frac{1}{4}(e^{2\pi i 2t/T} - 2 + e^{-2\pi i 2t/T}) = -\frac{1}{4}e^{2\pi i 2t/T} + \frac{1}{2} - \frac{1}{4}e^{-2\pi i 2t/T}.\end{aligned}$$

This gives the Fourier series of the function (with $y_2 = y_{-2} = -1/4$, $y_0 = 1/2$). This could also have been shown by using the trigonometric identity $\sin^2 x = \frac{1}{2}(1-\cos(2x))$ first, or by computing the integral $\frac{1}{T}\int_0^T f(t)e^{-2\pi i n t/T}\,dt$ (but this is rather cumbersome).

4 .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 $V_{N,2\pi}$.

4 .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 $y_{-3} = 1/8$, $y_{-1} = 3/8$, $y_1 = 3/8$, $y_3 = 1/8$.

4 .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.

5 . 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 1.27 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.

7 . 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 1.27 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 $a_n = \frac{T^2}{\pi^2 n^2}$ and $b_n = -\frac{T^2}{\pi n}$, for which Theorem 1.27 gives $y_n = \frac{1}{2}\left(\frac{T^2}{\pi^2 n^2} + i\frac{T^2}{\pi n}\right)$ 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

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

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 1.27 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.

8 .a. If f is symmetric about 0 we have that $b_n = 0$. Theorem 1.27 then gives that $y_n = \frac{1}{2}a_n$, which is real. The same theorem gives that $y_{-n} = \frac{1}{2}a_n = y_n$.

8 .b. If f is antisymmetric about 0 we have that $a_n = 0$. Theorem 1.27 then gives that $y_n = -\frac{1}{2}b_n$, which is purely imaginary. The same theorem gives that $y_{-n} = \frac{1}{2}b_n = -y_n$.

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

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

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

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

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

This is clearly antisymmetric, but then also $\sum_{n=-N}^{N} y_n e^{2\pi i n t/T}$ 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

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

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

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

The square wave defined in this exercise can be obtained by delaying our original square wave with $-T/4$. Using Property 3 in Theorem 1.37 with $d = -T/4$ on the complex Fourier coefficients $y_n = \begin{cases} 0, & \text{if } n \text{ is even;} \\ 2/(\pi in), & \text{if } n \text{ is odd.}\end{cases}$ which we obtained for the square wave in Exercise 1.5.5, we obtain the Fourier coefficients

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

This verifies the result.

2 . Since the real Fourier series of the square wave is

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

Theorem 1.27 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

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

Using Property 4 in Theorem 1.37 we get that the $e^{-2\pi i4t/T}$ (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 $2ie^{-2\pi i4t/T}$ times the square wave has its n'th Fourier coefficient equal to $\frac{4}{\pi(n+4)}$. We thus have that the function

$$f(t)=\begin{cases}2ie^{-2\pi i4t/T} & ,0\le t<T/2\\ -2ie^{-2\pi i4t/T} & ,T/2\le t<T\end{cases}$$

has the desired Fourier series.

Section 2.2

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

```
fs=44100;
t1=0:(1/fs):(4/440); % 1/fs is the distance between the samples
% You can also write t1=linspace(0,4/440,fs*(4/440));
t2=(4/440):(1/fs):(12/440);
t3=(12/440):(1/fs):(20/440);

f1=0*t1; % The first part of f
f2=2*((440*t2-4)/8).*sin(2*pi*440*t2); % The second part of f
f3=2*sin(2*pi*440*t3); % The third part of f
x=[f1 f2 f3];

x=x/max(abs(x));
playerobj=audioplayer(x,fs);
playblocking(playerobj);
```

Note that the sound has duration less than 0.05s, so you should only hear a very short beep. You also need to scale the values to be within -1 and 1, since some of the listed values are outside this range.

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;
x=sin(2*pi*440*t)+0.3*sin(2*pi*4400*t);
x=x/max(abs(x));
```

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

3 .a. The code can look like this:

```
function playpuresound(f)
  fs=2.5*f;
  t=0:(1/fs):3;
  x=sin(2*pi*f*t);
  playerobj=audioplayer(x,fs);
  playblocking(playerobj)
```

4 . The code can look like this:

```
function playsquare(T)
  % Play a square wave with period T over 3 seconds
  fs=40000;
  numsec=3;
  samplesperperiod=round(fs*T);
  oneperiod=[ones(1,round(samplesperperiod/2)) ...
             -ones(1,round(samplesperperiod/2))];
  numperiods=floor(numsec/T);
  x=zeros(1,numperiods*length(oneperiod));
  for k=1:numperiods
    x((((k-1)*length(oneperiod)+1):k*length(oneperiod))=oneperiod;
  end
  % It is simpler to replace for-loop with Matlabs repmat command as follows
  % x=repmat(oneperiod,1,numperiods);
  playerobj=audioplayer(x,fs);
  playblocking(playerobj)
```

```
function playtriangle(T)
  % Play a triangle wave with period T over 3 seconds
  fs=40000;
  numsec=3;
  samplesperperiod=round(fs*T);
  oneperiod=[linspace(-1,1,round(samplesperperiod/2)) ...
           linspace(1,-1,round(samplesperperiod/2))];
  numperiods=floor(numsec/T);
  x=zeros(1,numperiods*length(oneperiod));
  for k=1:numperiods
    x((((k-1)*length(oneperiod)+1):k*length(oneperiod))=oneperiod;
  end
  % It is simpler to replace for-loop with Matlabs repmat command as follows
  % x=repmat(oneperiod,1,numperiods);
  playerobj=audioplayer(x,fs);
  playblocking(playerobj)
```

5 .a. The code can look like this:

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

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

6 .a. The code can look like this:

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

6 .b. The code can look like this:

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

7 .a. The code can look like this:

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

3 . $\boldsymbol{z}$ is the vector $\boldsymbol{x}$ delayed with $d = 5$ samples, and then Property 3 of Theorem 2.21 gives us that $(F_N\boldsymbol{z})_n = e^{-2\pi i5k/N}(F_N\boldsymbol{x})_n$. In particular $|(F_N\boldsymbol{z})_n| = |(F_N\boldsymbol{x})_n| = 2$, since $|e^{-2\pi i5k/N}| = 1$.

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 . The idea is to express $\boldsymbol{x}$ as a linear combination of the Fourier basis vectors ϕ_n, and use that $F_N\phi_n = \boldsymbol{e}_n$. We have that

$$\begin{aligned}\cos^2(2\pi k/N) &= \left(\frac{1}{2}\left(e^{2\pi ik/N} + e^{-2\pi ikn/N}\right)\right)^2\\ &= \frac{1}{4}e^{2\pi i2k/N} + \frac{1}{2} + \frac{1}{4}e^{-2\pi i2k/N} = \frac{1}{4}e^{2\pi i2k/N} + \frac{1}{2} + \frac{1}{4}e^{2\pi i(N-2)k/N}\\ &= \sqrt{N}\left(\frac{1}{4}\phi_2 + \frac{1}{2}\phi_0 + \frac{1}{4}\phi_{N-2}\right).\end{aligned}$$

We here used the periodicity of $e^{2\pi ikn/N}$, i.e. that $e^{-2\pi i2k/N} = e^{2\pi i(N-2)k/N}$. Since F_N is linear and $F_N(\phi_n) = \boldsymbol{e}_n$, we have that

$$F_N(\boldsymbol{x}) = \sqrt{N}\left(\frac{1}{4}\boldsymbol{e}_2 + \frac{1}{2}\boldsymbol{e}_0 + \frac{1}{4}\boldsymbol{e}_{N-2}\right) = \sqrt{N}(1/2, 0, 1/4, 0, \dots, 0, 1/4, 0).$$

6 . We get

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

8 . 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;
```

9 .a. We have that

$$\begin{aligned}
y_n &= \frac{1}{\sqrt{N}}\left(\sum_{k=0}^{N/2-1} x_k e^{-2\pi i k n/N} + \sum_{k=N/2}^{N-1} x_k e^{-2\pi i k n/N}\right) \\
&= \frac{1}{\sqrt{N}}\left(\sum_{k=0}^{N/2-1} x_k e^{-2\pi i k n/N} + \sum_{k=0}^{N/2-1} x_k e^{-2\pi i (k+N/2) n/N}\right) \\
&= \frac{1}{\sqrt{N}}\sum_{k=0}^{N/2-1} x_k (e^{-2\pi i k n/N} + (-1)^n e^{-2\pi i k n/N}) \\
&= (1+(-1)^n)\frac{1}{\sqrt{N}}\sum_{k=0}^{N/2-1} x_k e^{-2\pi i k n/N}
\end{aligned}$$

If n is odd, we see that $y_n = 0$.

9 .b. The proof is the same as in a., except for a sign change.

9 .c. Clearly the set of vectors which satisfies $x_{k+N/2} = \pm x_k$ is a vector space V of dimension $N/2$. The set of vectors where every second component is zero is also a vector space of dimension $N/2$, let us denote this by W. We have shown that $F_N(V) \subset W$, but since F_N is unitary, $F_N(V)$ also has dimension $N/2$, so that $F_N(V) = W$. This shows that when every second y_n is 0, we must have that $x_{k+N/2} = \pm x_k$, and the proof is done.

9 . In the proofs above, compute the IDFT instead.

10 . We have that

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

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

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

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 2.6

1 . On the first line a sound file is read. On the next line we keep only the first sound canal, and restrict to the first 2^{17} sound samples. We then perform a DFT, zero out the frequencies which correspond to DFT-indices between 2^{15} and $2^{17} - 2^{15} - 1$, run an IDFT, and play the new sound. On the third line from bottom we scale the sound samples so that these lie between -1 and 1, so that they lie in the range Matlab demands for sound samples.

2 . As we have seen, DFT index n corresponds to frequency $\nu = nf_s/N$. Above $N = 2^{17}$, so that we get the connection $\nu = nf_s/N = n \times 44100/2^{17}$. We zeroed the DFT indices above $n = 2^{15}$, so that frequencies above $\nu = 2^{15} \times 44100/2^{17} = 11025Hz$ are affected.

Section 2.8

1 . The code can look as follows

```
[S,fs] = wavread('../castanets.wav');
mono = S(:,1);

kvals=4:12;
slowtime=zeros(1,length(kvals));
fasttime=slowtime; fastesttime=slowtime;
N = 2.^(kvals);
for k=kvals
    x = mono(1:2^k);

    start=toc;
    y = DFTImpl(x);
    slowtime(k-3) = toc-start;

    start=toc;
    y = FFTImpl(x);
    fasttime(k-3) = toc-start;

    start=toc;
    y = fft(x);
    fastesttime(k-3) = toc-start;
end

% a
plot(kvals, slowtime, 'ro-')
hold on
plot(kvals,fasttime, 'bo-')
plot(kvals,fastesttime, 'go-')
grid on
```

```
title('time usage of the DFT methods')
legend('slow DFT', 'FFT algorithm', 'Matlab FFT algorithm')
xlabel('log_2 N')
ylabel('time used [s]')

% b
figure(2)
loglog(N, slowtime, 'ro-')
hold on
loglog(N,fasttime, 'bo-')
axis equal
legend('slow DFT', 'FFT algorithm')
```

1 .b. The two different curves you see should have a derivative approximately equal to one and two, respectively.

1 .d. There are several reasons for this. Perhaps most important is that Matlab functions copy the parameters each time a function is called. When the vectors are large, this leads to extensive copying, also since the recursion depth is big. Another issue has to do with that Matlab code itself runs slowly when compared to native code. Also, Matlab's built-in `fft` has been subject to much more optimization than we have covered here.

2 . We get

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

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

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 first made, so that we can save computation. The benefit of using the FFT-algorithm is that we save computations, so that we end up with $O(5N\log_2 N)$ real arithmetic operations.

4 .a. We have that $\boldsymbol{x}^{(p)}$ is a constant vector of length N_2 for $0 \le p < N_1$. But then the DFT of all the $\boldsymbol{x}^{(p)}$ has zero outside entry zero. Multiplying with $e^{-2\pi ikn/N}$ does not affect this. The last $N_2 - 1$ rows are thus zero before the final DFT is applied, so that these rows are zero also after this final DFT. After assembling the polyphase components again we have that yrN_2 are the only nonzero DFT-coefficients.

5 . When we compute $e^{-2\pi i n/N}$, we do some multiplications/divisions in the exponent. These are not counted because they do not depend on $\boldsymbol{x}$, and may therefore be precomputed.

8 . The algorithm for the nonecursive FFT can look as follows

```
function y=FFTImplnonrec(x)
  y= bitreverse(x);
  N=length(y);
  Ns=2.^(0:(log2(N)-1));
  for nextN=Ns
    D=exp(-2*pi*1i*(0:(nextN-1))'/(2*nextN));
    k=1;
    while k<=N
      int1=k:(k+nextN-1); int2=int1+nextN;
      y(int2)=D.*y(int2);
      y([int1 int2])=[y(int1)+y(int2); y(int1)-y(int2)];
      k=k+2*nextN;
    end
  end
  y=y/sqrt(N);
```

If you add the nonrecursive algorithm to the code from Exercise 1, you will see that the non-recursive algorithm performs much better. There may be several reasons for this. First of all, recursive function calls which are omitted. Secondly, the values in the matrices $D_{N/2}$ are constructed once and for all with the non-recursive algorithm.

9 .e. The code for the split-radix algorithm can look as follows

```
function y=FFTImplsplitradix(x)
  N = length(x);
  if N == 1
    y = x(1);
  elseif N==2
    y=[x(1)+x(2);x(1)-x(2)];
  else
    ye  = FFTImplsplitradix(x(1:2:(N-1)));
    yo1 = FFTImplsplitradix(x(2:4:(N-2)));
    yo2 = FFTImplsplitradix(x(4:4:N));
    G=exp(-2*pi*1j*(0:(N/4-1))'/N);
    H=G.*exp(-2*pi*1j*(0:(N/4-1))'/(N/2));
    yo1=G.*yo1;
    yo2=H.*yo2;
    yo=[yo1+yo2; 1i*(yo2-yo1)];
    y = [ye + yo; ye - yo];
  end
```

If you add the split-radix FFT algorithm also to the code from Exercise 1, you will see that it performs better than the FFT algorithm, but worse than the non-recursive algorithm. That it performs better than the FFT algorithm is as expected, since it has a reduced number of arithmetic operations, and also a smaller number of recursive calls. It is not surprising that the non-recursive function performs better, since only that function omits recursive calls, and computes the values in the diagonal matrices once and for all.

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.

3 .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

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

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. $\lambda_{S_1+S_2,n} = \lambda_{S_1,n} + \lambda_{S_2,n}$.

4 .b. As above we have that $S_1 S_2 = F_N^H D_1 F_N F_N^H D_2 F_N = F_N^H D_1 D_2 F_N$, and the same reasoning gives that the eigenvalues of $S_1 S_2$ 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. $\lambda_{S_1 S_2,n} = \lambda_{S_1,n} \lambda_{S_2,n}$.

4 .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

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

2 .a. The filter coefficients are $t_0 = t_3 = 1/3$, $t_1 = t_2 = 1$. We have that

$$\begin{aligned}\lambda_S(\omega) &= \sum_k t_k e^{-ik\omega} = \frac{1}{3}(1 + 3e^{-i\omega} + 3e^{-2i\omega} + e^{-3i\omega}) \\ &= \frac{2}{3}e^{-3i\omega/2}\frac{1}{2}(e^{3i\omega/2} + 3e^{i\omega/2} + 3e^{-i\omega/2} + e^{-3i\omega/2}) \\ &= \frac{2}{3}e^{-3i\omega/2}(\cos(3\omega/2) + 3\cos(\omega/2)).\end{aligned}$$

From this expression it is easy to plot the frequency response, but since this is complex, we have to plot the magnitude, i.e. $|\lambda_S(\omega)| = \frac{2}{3}|\cos(3\omega/2) + 3\cos(\omega/2)|$. We also see that $\lambda_S(0) = \frac{2}{3}$, and that $\lambda_S(\pi) = 0$, so that the filter is a lowpass filter.

2 .b. If we use the connection between the vector frequency response and the continuous frequency response we get

$$\lambda_{S,2} = \lambda_S(2\pi 2/N) = \frac{2}{3}e^{-6\pi i/N}(\cos(6\pi/N) + 3\cos(2\pi/N)).$$

Alternatively you can here compute that the first column in the circulant Toeplitz matrix for S is given by $s_0 = t_1$, $s_2 = t_2$, $s_3 = t_3$, and $s_4 = t_4$, and insert this in the definition of the vector frequency response, $\lambda_{S,2} = \sum_{k=0}^{N-1} s_k e^{-2\pi i 2k/N}$. We know that $e^{2\pi i 2k/N}$ is an eigenvector for S since S is a filter, and that $\lambda_{S,2}$ is the corresponding eigenvalue. We therefore get that

$$S\boldsymbol{x} = \lambda_{S,2}\boldsymbol{x} = \frac{2}{3}e^{-6\pi i/N}(\cos(6\pi/N) + 3\cos(2\pi/N))\boldsymbol{x}.$$

3 .a. Since clearly $t_{-2} = t_2 = 1/16$, $t_{-1} = t_1 = 1/4$, and $t_0 = 6/16$, the first column $\boldsymbol{s}$ in the circulant Toeplitz matrix is given by $s_0 = t_0 = 6/16$, $s_1 = t_1 = 4/16$, $s_2 = t_2 = 1/16$, $s_{N-2} = t_{-2} = 1/16$, $s_{N-1} = t_{-1} = 4/16$. An 8×8 circulant Toeplitz matrix which corresponds to applying S_1 to a periodic signal of length $N = 8$ is therefore

$$\frac{1}{16}\begin{pmatrix} 6 & 4 & 1 & 0 & 0 & 0 & 1 & 4 \\ 4 & 6 & 4 & 1 & 0 & 0 & 0 & 1 \\ 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 \\ 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 \\ 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 \\ 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 \\ 1 & 0 & 0 & 0 & 1 & 4 & 6 & 4 \\ 4 & 1 & 0 & 0 & 0 & 1 & 4 & 6 \end{pmatrix}.$$

3 .b. The frequency response is

$$\lambda_{S_1}(\omega) = \frac{1}{16}(e^{2i\omega} + 4e^{i\omega} + 6 + 4e^{-i\omega} + e^{-2i\omega}) = \left(\frac{1}{2}(e^{i\omega/2} + e^{-i\omega/2})\right)^4 = \cos^4(\omega/2),$$

where we recognized $(1,4,6,4,1)$ as a row in Pascal's triangle, so that we could write the expression as a power. From this expression it is easy to plot the frequency response, and it is clear that the filter is a lowpass filter, since $\lambda_{S_1}(0) = 1$, $\lambda_{S_1}(\pi) = 0$.

3 .c. We have that

$$\lambda_{S_2}(\omega) = (e^{i\omega} + 2 + e^{-i\omega})/4 = \left(\frac{1}{2}(e^{i\omega/2} + e^{-i\omega/2})\right)^2 = \cos^2(\omega/2).$$

We then get that

$$\begin{aligned}\lambda_{S_1S_2}(\omega) &= \lambda_{S_1}(\omega)\lambda_{S_2}(\omega) = \cos^4(\omega/2)\cos^2(\omega/2) = \cos^6(\omega/2) \\ &= \left(\frac{1}{2}(e^{i\omega/2} + e^{-i\omega/2})\right)^6 \\ &= \frac{1}{64}(e^{3i\omega} + 6e^{2i\omega} + 15e^{i\omega} + 20 + 15e^{-i\omega} + 6e^{-2i\omega} + e^{-3i\omega}),\end{aligned}$$

where we have used that, since we have a sixth power, the values can be obtained from fra a row in Pascal's triangle also here. It is now clear that

$$S_1S_2 = \frac{1}{64}\{1,6,15,\underline{20},15,6,1\}.$$

You could also have argumented here by taking the convolution of $\frac{1}{16}(1,4,6,4,1)$ with $\frac{1}{4}(1,2,1)$.

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 . We have that

$$\begin{aligned}\lambda_{S_2}(\omega) &= \sum_k \cos(2\pi kn/N) t_k e^{-ik\omega} = \frac{1}{2}\sum_k (e^{2\pi ikn/N} + e^{-2\pi ikn/N}) t_k e^{-ik\omega} \\ &= \frac{1}{2}\left(\sum_k t_k e^{-ik(\omega-2\pi n/N)} + \sum_k t_k e^{-ik(\omega+2\pi n/N)}\right) \\ &= \frac{1}{2}(\lambda_{S_1}(\omega-2\pi n/N) + \lambda_{S_1}(\omega+2\pi n/N)).\end{aligned}$$

1 0.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 the filtered 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.

1 0.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.

1 0.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.

1 .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 . First we obtain the matrix S as

$$\left(\begin{array}{cccc|cccc} \frac{1}{2} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{2} \end{array}\right)$$

where we have drawn the boundaries between the blocks S_1, S_2, S_3, S_4. From this we see that

$$S_1 = \begin{pmatrix} \frac{1}{2} & \frac{1}{4} & 0 & 0 \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & 0 \\ 0 & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ 0 & 0 & \frac{1}{4} & \frac{1}{2} \end{pmatrix} \quad S_2 = \begin{pmatrix} 0 & 0 & 0 & \frac{1}{4} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{4} & 0 & 0 & 0 \end{pmatrix} \quad (S_2)^f = \begin{pmatrix} \frac{1}{4} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{4} \end{pmatrix}.$$

From this we get

$$S_r = S_1 + (S_2)^f = \begin{pmatrix} \frac{3}{4} & \frac{1}{4} & 0 & 0 \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & 0 \\ 0 & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ 0 & 0 & \frac{1}{4} & \frac{3}{4} \end{pmatrix}.$$