Chapter 3

Fourier analysis for vectors

In Chapter 2 we saw how a function defined on an interval can be decomposed into a linear combination of sines and cosines, or equivalently, a linear combination of complex exponential functions. However, this kind of decomposition is not very convenient from a computational point of view. The coefficients are given by integrals that in most cases cannot be evaluated exactly, so some kind of numerical integration technique would have to be applied.

In this chapter our starting point is simply a vector of finite dimension. Our aim is then to decompose this vector in terms of linear combinations of vectors built from complex exponentials. This simply amounts to multiplying the original vector by a matrix, and there are efficient algorithms for doing this. It turns out that these algorithms can also be used for computing good approximations to the continuous Fourier series in Chapter 2.

Recall from Chapter 1 that a digital sound is simply a sequence of numbers, in other words, a vector. An algorithm for decomposing a vector into combinations of complex exponentials therefore corresponds to an algorithm for decomposing a digital sound into a combination of pure tones.

3.1 Basic ideas

We start by recalling what a digital sound is and by establishing some notation and terminology.

Fact 3.1. A digital sound is a finite sequence (or equivalently a vector) $\boldsymbol{x}$ of numbers, together with a number (usually an integer) f_s, the sample rate, which denotes the number of measurements of the sound per second. The length of the vector is usually assumed to be N, and it is indexed from 0 to $N-1$. Sample k is denoted by x_k, i.e.,

$$\boldsymbol{x} = (x_k)_{k=0}^{N-1}.$$

Note that this indexing convention for vectors is not standard in mathematics and is different from what we have used before. Note in particular that MATLAB indexes vectors from 1, so algorithms given here must be adjusted appropriately.

We also need the standard inner product and norm for complex vectors. At the outset our vectors will have real components, but we are going to perform Fourier analysis with complex exponentials which will often result in complex vectors.

Definition 3.2. For complex vectors of length N the Euclidean inner product is given by

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{k=0}^{N-1} x_k \overline{y_k}. \tag{3.1}$$

The associated norm is

$$\|\boldsymbol{x}\| = \sqrt{\sum_{k=0}^{N-1} |x_k|^2}. \tag{3.2}$$

In the previous chapter we saw that, using a Fourier series, a function with period T could be approximated by linear combinations of the functions (the pure tones) $\{e^{2\pi int/T}\}_{n=0}^{N}$. This can be generalised to vectors (digital sounds), but then the pure tones must of course also be vectors.

Definition 3.3 (Fourier analysis for vectors)**.** In Fourier analysis of vectors, a vector $\boldsymbol{x} = (x_0, \dots, x_{N-1})$ is represented as a linear combination of the N vectors

$$\boldsymbol{\phi}_n = \frac{1}{\sqrt{N}} \left(1, e^{2\pi in/N}, e^{2\pi i2n/N}, \dots, e^{2\pi ikn/N}, \dots, e^{2\pi in(N-1)/N}\right).$$

These vectors are called the normalised complex exponentials or the pure digital tones of order N. The whole collection $\mathcal{F}_N = \{\boldsymbol{\phi}_n\}_{n=0}^{N}$ is called the N-point Fourier basis.

The following lemma shows that the vectors in the Fourier basis are orthogonal, so they do indeed form a basis.

Lemma 3.4. The normalised complex exponentials $\{\boldsymbol{\phi_n}\}_{n=0}^{N-1}$ of order N form an orthonormal basis in $\mathbb{R}^N$.

Proof. Let n_1 and n_2 be two distinct integers in the range $[0, N-1]$. The inner product of $\boldsymbol{\phi}_{n_1}$ and $\boldsymbol{\phi}_{n_2}$ is then given by

$$\begin{aligned}
N\langle \boldsymbol{\phi}_{n_1}, \boldsymbol{\phi}_{n_2}\rangle &= \langle e^{2\pi i n_1 k/N}, e^{2\pi i n_2 k/N}\rangle \\
&= \sum_{k=0}^{N-1} e^{2\pi i n_1 k/N} e^{-2\pi i n_2 k/N} \\
&= \sum_{k=0}^{N-1} e^{2\pi i (n_1-n_2)k/N} \\
&= \frac{1-e^{2\pi i(n_1-n_2)}}{1-e^{2\pi i(n_1-n_2)/N}} \\
&= 0.
\end{aligned}$$

In particular, this orthogonality means that the the complex exponentials form a basis. And since we also have $\langle \boldsymbol{\phi}_n, \boldsymbol{\phi}_n\rangle = 1$ it is in fact an orthonormal basis. □

Note that the normalising factor $\frac{1}{\sqrt{N}}$ was not present for pure tones in the previous chapter. Also, the normalising factor $\frac{1}{T}$ from the last chapter is not part of the definition of the inner product in this chapter. These are small differences which have to do with slightly different notation for functions and vectors, and which will not cause confusion in what follows.

3.2 The Discrete Fourier Transform

Fourier analysis for finite vectors is focused around mapping a given vector from the standard basis to the Fourier basis, performing some operations on the Fourier representation, and then changing the result back to the standard basis. The Fourier matrix, which represents this change of basis, is therefore of crucial importance, and in this section we study some of its basic properties. We start by defining the Fourier matrix.

Definition 3.5 (Discrete Fourier Transform)**.** The change of coordinates from the standard basis of $\mathbb{R}^N$ to the Fourier basis $\mathcal{F}_N$ is called the discrete Fourier transform (or DFT). The $N\times N$ matrix F_N that represents this change of basis is called the (N-point) Fourier matrix. If $\boldsymbol{x}$ is a vector in R^N, its coordinates $\boldsymbol{y} = (y_0, y_1, \ldots, y_{N-1})$ relative to the Fourier basis are called the Fourier coefficients of $\boldsymbol{x}$, in other words $\boldsymbol{y} = F_N\boldsymbol{x}$). The DFT of $\boldsymbol{x}$ is sometimes denoted by $\hat{\boldsymbol{x}}$.

We will normally write $\boldsymbol{x}$ for the given vector in $\mathbb{R}^N$, and $\boldsymbol{y}$ for the DFT of this vector. In applied fields, the Fourier basis vectors are also called *synthesis vectors*, since they can be used used to "synthesize" the vector $\boldsymbol{x}$, with weights provided by the DFT coefficients $\boldsymbol{y} = (y_n)_{n=0}^{N-1}$. To be more precise, we have that the change of coordinates performed by the DFT can be written as

$$\boldsymbol{x} = y_0\boldsymbol{\phi}_0 + y_1\boldsymbol{\phi}_1 + \cdots + y_{N-1}\boldsymbol{\phi}_{N-1} = \begin{pmatrix}\boldsymbol{\phi}_0 & \boldsymbol{\phi}_1 & \cdots & \boldsymbol{\phi}_{N-1}\end{pmatrix}\boldsymbol{y} = F_N^{-1}\boldsymbol{y}, \quad (3.3)$$

where we have used the inverse of the defining relation $\boldsymbol{y} = F_N\boldsymbol{x}$, and that the $\boldsymbol{\phi}_n$ are the columns in F_N^{-1} (this follows from the fact that F_N^{-1} is the change of coordinates matrix from the Fourier basis to the standard basis, and the Fourier basis vectors are clearly the columns in this matrix). Equation (3.3) is also called the synthesis equation.

Let us also find the matrix F_N itself. From Lemma 3.4 we know that the columns of F_N^{-1} are orthonormal. If the matrix was real, it would have been called orthogonal, and the inverse matrix could be obtained by transposing. F_N^{-1} is complex however, and it is easy to see that the conjugation present in the definition of the inner product (3.1) translates into that the inverse of a complex matrix with orthonormal columns is given by the matrix where the entries are both transposed and conjugated. Let us denote the conjugated transpose of T by T^H, and say that a complex matrix is unitary when $T^{-1} = T^H$. From our discussion it is clear that F_N^{-1} is a unitary matrix, i.e. its inverse, F_N, is its conjugate transpose. Moreover since F_N^{-1} is symmetric, its inverse is in fact just its conjugate,

$$F_N = \overline{F_N^{-1}}.$$

Theorem 3.6. The Fourier matrix F_N is the unitary $N \times N$-matrix with entries given by

$$(F_N)_{nk} = \frac{1}{\sqrt{N}} e^{-2\pi ink/N},$$

for $0 \leq n, k \leq N-1$.

Note that in the signal processing literature, it is not common to include the normalizing factor $1/\sqrt{N}$ in the definition of the DFT. From our more mathematical point of view this is useful since it makes the Fourier matrix unitary.

In practical applications of Fourier analysis one typically applies the DFT, performs some operations on the coefficients, and then maps the result back using the inverse Fourier matrix. This inverse transformation is so common that it deserves a name of its own.

Definition 3.7 (IDFT)**.** If $\boldsymbol{y} \in \mathbb{R}^N$ the vector $\boldsymbol{x} = (F_N)^H\boldsymbol{y}$ is referred to as the inverse discrete Fourier transform or (IDFT) of $\boldsymbol{y}$.

That $\boldsymbol{y}$ is the DFT of $\boldsymbol{x}$ and $\boldsymbol{x}$ is the IDFT of $\boldsymbol{y}$ can also be expressed in component form

$$x_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} y_n e^{2\pi i n k/N}, \tag{3.4}$$

$$y_n = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} x_k e^{-2\pi i n k/N}. \tag{3.5}$$

In applied fields such as signal processing, it is more common to state the DFT and IDFT in these component forms, rather than in the matrix forms $\boldsymbol{x} = (F_N)^H \boldsymbol{y}$ and $\boldsymbol{y} = F_N \boldsymbol{y}$.

Let us use now see how these formulas work out in practice by considering some examples.

Example 3.8 (DFT on a square wave)**.** Let us attempt to apply the DFT to a signal $\boldsymbol{x}$ which is 1 on indices close to 0, and 0 elsewhere. Assume that

$$x_{-L} = \ldots = x_{-1} = x_0 = x_1 = \ldots = x_L = 1,$$

while all other values are 0. This is similar to a square wave, with some modifications: First of all we assume symmetry around 0, while the square wave of Example 1.11 assumes antisymmetry around 0. Secondly the values of the square wave are now 0 and 1, contrary to -1 and 1 before. Finally, we have a different proportion of where the two values are assumed. Nevertheless, we will also refer to the current digital sound as a square wave.

Since indices with the DFT are between 0 an $N-1$, and since $\boldsymbol{x}$ is assumed to have period N, the indices $[-L, L]$ where our signal is 1 translates to the indices $[0, L]$ and $[N-L, N-1]$ (i.e., it is 1 on the first and last parts of the vector). Elsewhere our signal is zero. Since $\sum_{k=N-L}^{N-1} e^{-2\pi i n k/N} = \sum_{k=-L}^{-1} e^{-2\pi i n k/N}$ (since $e^{-2\pi i n k/N}$ is periodic with period N), the DFT of $\boldsymbol{x}$ is

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

This computation does in fact also give us the IDFT of the same vector, since the IDFT just requires a change of sign in all the exponents. From this example we see that, in order to represent $\boldsymbol{x}$ in terms of frequency components, all components are actually needed. The situation would have been easier if only a few frequencies were needed.

Example 3.9. In most cases it is difficult to compute a DFT by hand, due to the entries $e^{-2\pi ink/N}$ in the matrices, which typically can not be represented exactly. The DFT is therefore usually calculated on a computer only. However, in the case $N = 4$ the calculations are quite simple. In this case the Fourier matrix takes the form

$$F_4 = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{pmatrix}.$$

We now can compute the DFT of a vector like $(1,2,3,4)^T$ simply as

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

Example 3.10 (Direct implementation of the DFT)**.** MATLAB supports complex arithmetic, so the DFT can be implemented very simply and directly by the code

```
function y=DFTImpl(x)
  N=length(x);
  FN=zeros(N);
  for n=1:N
    FN(n,:)=exp(-2*pi*1i*(n-1)*(0:(N-1))/N)/sqrt(N);
  end
  y=FN*x;
```

Note that n has been replaced by $n-1$ in this code since n runs from 1 to N (array indices must start at 1 in MATLAB).

A direct implementation of the IDFT, which we could call `IDFTImpl` can be done similarly. Multiplying a full $N \times N$ matrix by a vector requires roughly N^2 arithmetic operations. The DFT algorithm above will therefore take a long time when N becomes moderately large, particularly in MATLAB. It turns out that if N is a power of 2, there is a much more efficient algorithm for computing the DFT which we will study in a later chapter. MATLAB also has a built-in implementation of the DFT which uses such an efficient algorithm.

The DFT has properties which are very similar to those of Fourier series, as they were listed in Theorem 2.18. The following theorem sums this up:

Theorem 3.11 (DFT properties)**.** Let $\boldsymbol{x}$ be a real vector of length N. The DFT has the following properties:

1. $(\widehat{\boldsymbol{x}})_{N-n} = \overline{(\widehat{\boldsymbol{x}})_n}$ for $0 \leq n \leq N-1$.
2. If $\boldsymbol{z}$ is the vector with the components of $\boldsymbol{x}$ reversed so that $z_k = x_{N-k}$ for $0 \leq k \leq N-1$, then $\widehat{\boldsymbol{z}} = \overline{\widehat{\boldsymbol{x}}}$. In particular,
   (a) if $x_k = x_{N-k}$ for all n (so $\boldsymbol{x}$ is symmetric), then $\widehat{\boldsymbol{x}}$ is a real vector.
   (b) if $x_k = -x_{N-k}$ for all k (so $\boldsymbol{x}$ is antisymmetric), then $\widehat{\boldsymbol{x}}$ is a purely imaginary vector.
3. If d is an integer and $\boldsymbol{z}$ is the vector with components $z_k = x_{k-d}$ (the vector $\boldsymbol{x}$ with its elements delayed by d), then $(\widehat{\boldsymbol{z}})_n = e^{-2\pi i dn/N} (\widehat{\boldsymbol{x}})_n$.
4. If d is an integer and $\boldsymbol{z}$ is the vector with components $z_k = e^{2\pi i dk/N} x_k$, then $(\widehat{\boldsymbol{z}})_n = (\widehat{\boldsymbol{x}})_{n-d}$.
5. Let d be a multiple of $1/2$. Then the following are equivalent:
   (a) $x_{d+k} = x_{d-k}$ for all k so that $d+k$ and $d-k$ are integers (in other words $\boldsymbol{x}$ is symmetric about d).
   (b) The argument of $(\widehat{\boldsymbol{x}})_n$ is $-2\pi dn/N$ for all n.

Proof. The methods used in the proof are very similar to those used in the proof of Theorem 2.18. From the definition of the DFT we have

$$\begin{aligned}(\widehat{\boldsymbol{x}})_{N-n} &= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{-2\pi i k(N-n)/N} x_k = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i kn/N} x_k \\ &= \overline{\frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{-2\pi i kn/N} x_k} = \overline{(\widehat{\boldsymbol{x}})_n}\end{aligned}$$

which proves property 1. To prove property 2, we write

$$\begin{aligned}(\widehat{\boldsymbol{z}})_n &= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} z_k e^{-2\pi i kn/N} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} x_{N-k} e^{-2\pi i kn/N} \\ &= \frac{1}{\sqrt{N}} \sum_{u=1}^{N} x_u e^{-2\pi i (N-u)n/N} = \frac{1}{\sqrt{N}} \sum_{u=0}^{N-1} x_u e^{2\pi i un/N} \\ &= \overline{\frac{1}{\sqrt{N}} \sum_{u=0}^{N-1} x_u e^{-2\pi i un/N}} = \overline{(\widehat{\boldsymbol{x}})_n}.\end{aligned}$$

If $\boldsymbol{x}$ is symmetric it follows that $\boldsymbol{z} = \boldsymbol{x}$, so that $(\widehat{\boldsymbol{x}})_n = \overline{(\widehat{\boldsymbol{x}})_n}$. Therefore $\boldsymbol{x}$ must be real. The case of antisymmetry follows similarly.

To prove property 3 we observe that

$$\begin{aligned}(\widehat{\boldsymbol{z}})_n &= \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} x_{k-d}e^{-2\pi ikn/N} = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} x_k e^{-2\pi i(k+d)n/N} \\ &= e^{-2\pi idn/N}\frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} x_k e^{-2\pi ikn/N} = e^{-2\pi idn/N}\left(\widehat{\boldsymbol{x}}\right)_n.\end{aligned}$$

For the proof of property 4 we note that the DFT of $\boldsymbol{z}$ is

$$(\widehat{\boldsymbol{z}})_n = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} e^{2\pi idk/N}x_n e^{-2\pi ikn/N} = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} x_n e^{-2\pi i(n-d)k/N} = (\widehat{\boldsymbol{x}})_{n-d}.$$

Finally, to prove property 5, we note that if d is an integer, the vector $\boldsymbol{z}$ where $\boldsymbol{x}$ is delayed by $-d$ samples satisfies the relation $(\widehat{\boldsymbol{z}})_n = e^{2\pi idn/N}\left(\widehat{\boldsymbol{x}}\right)_n$ because of property 3. Since $\boldsymbol{z}$ satisfies $z_n = z_{N-n}$, we have by property 2 that $(\widehat{\boldsymbol{z}})_n$ is real, and it follows that the argument of $(\widehat{\boldsymbol{x}})_n$ is $-2\pi dn/N$. It is straightforward to convince oneself that property 5 also holds when d is not an integer also (i.e., a multiple of 1/2). $\square$

For real sequences, Property 1 says that we need to store only about one half of the DFT coefficients, since the remaining coefficients can be obtained by conjugation. In particular, when N is even, we only need to store $y_0, y_1, \ldots, y_{N/2}$, since the other coefficients can be obtained by conjugating these.

3.2.1 Connection between the DFT and Fourier series

So far we have focused on the DFT as a tool to rewrite a vector in terms of digital, pure tones. In practice, the given vector $\boldsymbol{x}$ will often be sampled from some real data given by a function $f(t)$. We may then talk about the frequency content of the vector $\boldsymbol{x}$ and the frequency content of f and ask ourselves how these are related. More precisely, what is the relationship between the Fourier coefficients of f and the DFT of $\boldsymbol{x}$?

In order to study this, assume for simplicity that f is a sum of finitely many frequencies. This means that there exists an M so that f is equal to its Fourier approximation f_M,

$$f(t) = f_M(t) = \sum_{n=-M}^{M} z_n e^{2\pi int/T}, \tag{3.6}$$

where z_n is given by

$$z_n = \frac{1}{T}\int_0^T f(t)e^{-2\pi int/T}\,dt.$$

We recall that in order to represent the frequency n/T fully, we need the corresponding exponentials with both positive and negative arguments, i.e., both $e^{2\pi int/T}$ and $e^{-2\pi int/T}$.

Fact 3.12. Suppose f is given by its Fourier series (3.6). Then the total frequency content for the frequency n/T is given by the two coefficients z_n and z_{-n}.

Suppose that the vector $\boldsymbol{x}$ contains values sampled uniformly from f at N points,

$$x_k = f(kT/N), \quad \text{for } k = 0, 1, \ldots, N-1. \tag{3.7}$$

The vector $\boldsymbol{x}$ can be expressed in terms of its DFT $\boldsymbol{y}$ as

$$x_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} y_n e^{2\pi i n k/N}. \tag{3.8}$$

If we evaluate f at the sample points we have

$$f(kT/N) = \sum_{n=-M}^{M} z_n e^{2\pi i n k/N}, \tag{3.9}$$

and a comparison now gives

$$\sum_{n=-M}^{M} z_n e^{2\pi i n k/N} = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} y_n e^{2\pi i n k/N} \quad \text{for } k = 0, 1, \ldots, N-1.$$

Exploiting the fact that both $\boldsymbol{y}$ and the complex exponentials are periodic with period N, and assuming that we take N samples with N odd, we can rewrite this as

$$\sum_{n=-M}^{M} z_n e^{2\pi i n k/N} = \frac{1}{\sqrt{N}} \sum_{n=-(N-1)/2}^{(N-1)/2} y_n e^{2\pi i n k/N}.$$

This is a matrix relation on the form $G\boldsymbol{z} = H\boldsymbol{y}/\sqrt{N}$, where

1. G is the $N \times (2M+1)$-matrix with entries $\frac{1}{\sqrt{N}} e^{2\pi i n k/N}$,
2. H is the $N \times N$-matrix with entries $\frac{1}{\sqrt{N}} e^{2\pi i n k/N}$.

In Exercise 6 you will be asked to show that $G^H G = I_{2M+1}$, and that $G^H H = \begin{pmatrix} I & \mathbf{0} \end{pmatrix}$, when $N \geq 2M+1$. Thus, if we choose the number of sample points N so that $N \geq 2M+1$, multiplying with G^H on both sides in $G\boldsymbol{z} = H\boldsymbol{y}/\sqrt{N}$ gives us that

$$\boldsymbol{z} = \begin{pmatrix} I & \mathbf{0} \end{pmatrix} \left(\frac{1}{\sqrt{N}} \boldsymbol{y} \right),$$

i.e. $\boldsymbol{z}$ consists of the first $2M+1$ elements in $\boldsymbol{y}/\sqrt{N}$. Setting $N = 2M+1$ we can summarize this.

Proposition 3.13 (Relation between Fourier coefficients and DFT coefficients)**.** Let f be a Fourier series

$$f(t) = \sum_{n=-M}^{M} z_n e^{2\pi i n t/T},$$

on the interval $[0, T]$ and let $N = 2M + 1$ be an odd integer. Suppose that $\boldsymbol{x}$ is sampled from f by

$$x_k = f(kT/N), \quad \text{for } k = 0, 1, \ldots, N-1.$$

and let $\boldsymbol{y}$ be the DFT of $\boldsymbol{x}$. Then $\boldsymbol{z} = \boldsymbol{y}/\sqrt{N}$, and the total contribution to f from frequency n/T, where n is an integer in the range $0 \le n \le M$, is given by y_n and y_{N-n}.

We also need a remark on what we should interpret as high and low frequency contributions, when we have applied a DFT. The low "frequency contribution" in f is the contribution from

$$e^{-2\pi i L t/T}, \ldots, e^{-2\pi i t/T}, 1, e^{2\pi i t/T}, \ldots, e^{2\pi i L t/T}$$

in f, i.e. $\sum_{n=-L}^{L} z_n e^{2\pi i n t/T}$. This means that low frequencies correspond to indices n so that $-L \le n \le L$. However, since DFT coefficients have indices between 0 and $N-1$, low frequencies correspond to indices n in $[0, L] \cup [N - L, N - 1]$. If we make the same argument for high frequencies, we see that they correspond to DFT indices near $N/2$:

Observation 3.14 (DFT indices for high and low frequencies)**.** When $\boldsymbol{y}$ is the DFT of $\boldsymbol{x}$, the low frequencies in $\boldsymbol{x}$ correspond to the indices in $\boldsymbol{y}$ near 0 and N. The high frequencies in $\boldsymbol{x}$ correspond to the indices in $\boldsymbol{y}$ near $N/2$.

We will use this observation in the following example, when we use the DFT to distinguish between high and low frequencies in a sound.

Example 3.15 (Using the DFT to adjust frequencies in sound)**.** Since the DFT coefficients represent the contribution in a sound at given frequencies, we can listen to the different frequencies of a sound by adjusting the DFT coefficients. Let us first see how we can listen to the lower frequencies only. As explained, these correspond to DFT-indices n in $[0, L] \cup [N - L, N - 1]$. In MATLAB these have indices from 1 to $L{+}1$, and from $N{-}L{+}1$ to N. The remaining frequencies, i.e. the higher frequencies which we want to eliminate, thus have MATLAB-indices between $L + 2$ and $N - L$. We can now perform a DFT, eliminate high frequencies by setting the corresponding frequencies to zero, and perform an inverse DFT to recover the sound signal with these frequencies eliminated. With the help of the DFT implementation from Example 3.10, all this can be achieved with the following code:

```
y=DFTImpl(x);
y((L+2):(N-L))=zeros(N-(2*L+1),1);
newx=IDFTImpl(y);
```

To test this in practice, we also need to obtain the actual sound samples. If we use our sample file `castanets.wav`, you will see that the code runs very slowly. In fact it seems to never complete. The reason is that `DFTImpl` attempts to construct a matrix F_N with as many rows and columns as there are sound samples in the file, and there are just too many samples, so that F_N grows too big, and matrix multiplication with it gets too time-consuming. We will shortly see much better strategies for applying the DFT to a sound file, but for now we will simply attempt instead to split the sound file into smaller blocks, each of size $N = 32$, and perform the code above on each block. It turns out that this is less time-consuming, since big matrices are avoided. You will be spared the details for actually splitting the sound file into blocks: you can find the function `playDFTlower(L)` which performs this splitting, sets the relevant frequency components to 0, and plays the resulting sound samples. If you try this for $L = 7$ (i.e. we keep only 15 of the DFT coefficients) the result sounds like this. You can hear the disturbance in the sound, but we have not lost that much even if more than half the DFT coefficients are dropped. If we instead try $L = 3$ the result will sound like this. The quality is much poorer now. However we can still recognize the song, and this suggests that most of the frequency information is contained in the lower frequencies.

Similarly we can listen to high frequencies by including only DFT coefficients with index close to $\frac{N}{2}$. The function `playDFThigher(L)` sets all DFT coefficients to zero, except for those with indices $\frac{N}{2} - L, \ldots, \frac{N}{2}, \ldots, \frac{N}{2} + L$. Let us verify that there is less information in the higher frequencies by trying the same values for L as above for this function. For $L = 7$ (i.e. we keep only the middle 15 DFT coefficients) the result sounds like this, for $L = 3$ the result sounds like this. Both sounds are quite unrecognizable, confirming that most information is contained in the lower frequencies.

Note that there may be a problem in the previous example: for each block we compute the frequency representation of the values in that block. But the frequency representation may be different when we take all the samples into consideration. In other words, when we split into blocks, we can't expect that we exactly eliminate all the frequencies in question. This is a common problem in signal processing theory, that one in practice needs to restrict to smaller segments of samples, but that this restriction may have undesired effects in terms of the frequencies in the output.

3.2.2 Interpolation with the DFT

There are two other interesting facets to Theorem 3.13, besides connecting the DFT and the Fourier series: The first has to do with interpolation: The theorem enables us to find (unique) trigonometric functions which interpolate (pass through) a set of data points. We have in elementary calculus courses seen how to determine a polynomial of degree $N-1$ that interpolates a set of N data points — such polynomials are called interpolating polynomials. The following result tells how we can find an interpolating trigonometric function using the DFT.

Corollary 3.16 (Interpolation with the Fourier basis)**.** Let f be a function defined on the interval $[0, T]$, and let $\boldsymbol{x}$ be the sampled sequence given by

$$x_k = f(kT/N) \quad \text{for } k = 0, 1, \ldots, N-1.$$

There is exactly one linear combination $g(t)$ on the form

$$g(t) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} y_n e^{2\pi i n t/T}$$

which satisfies the conditions

$$g(kT/N) = f(kT/N), \quad k = 0, 1, \ldots, N-1$$

and its coefficients are determined by the DFT $\boldsymbol{y} = \hat{\boldsymbol{x}}$ of $\boldsymbol{x}$.

The proof for this follows by inserting $t = 0$, $t = T/N$, $t = 2T/N$, ..., $t = (N-1)T/N$ in the equation $f(t) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} y_n e^{2\pi i n t/T}$ to arrive at the equations

$$f(kT/N) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} y_n e^{2\pi i n k/N} \qquad 0 \le k \le N-1.$$

This gives us an equation system for finding the y_n with the invertible Fourier matrix as coefficient matrix, and the result follows.

3.2.3 Sampling and reconstruction with the DFT

The second interesting facet to Theorem 3.13 has to do with when reconstruction of a function from its sample values is possible. An example of sampling a function is illustrated in Figure 3.1. From Figure 3.1(b) it is clear that some information is lost when we discard everything but the sample values. There may however be an exception to this, if we assume that the function satisfies some property. Assume that f is equal to a finite Fourier series. This means that f can be written on the form (3.6), so that the highest frequency in the signal is bounded by M/T. Such functions also have their own name:

Definition 3.17 (Band-limited functions)**.** A function f is said to be *band-limited* if there exists a number ν so that f does not contain frequencies higher than ν.

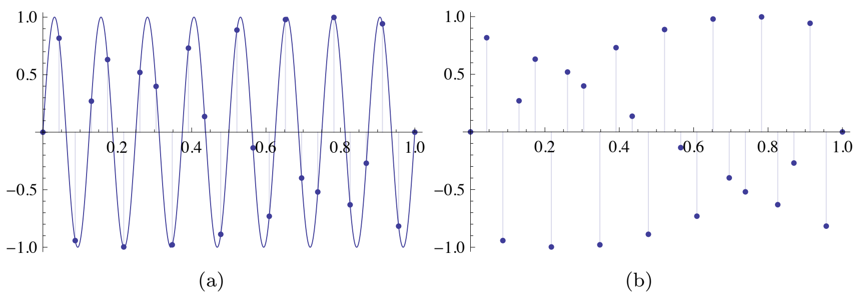


Figure 3.1: An example of sampling. Figure (a) shows how the samples are picked from underlying continuous time function. Figure (b) shows what the samples look like on their own.

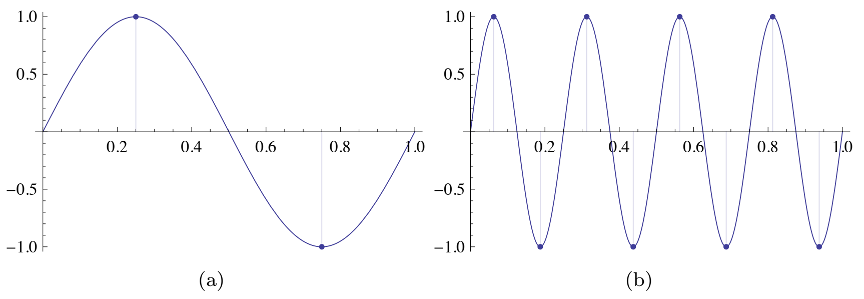


Figure 3.2: Sampling the function $\sin 2\pi t$ with two points, and the function $\sin 2\pi 4t$ with eight points.

Our analysis prior to Theorem 3.13 states that all periodic, band-limited functions can be reconstructed exactly from their samples, using the DFT, as long as the number of samples is $N \geq 2M+1$, taken uniformly over a period. Moreover, the DFT is central in the reconstruction formula. We say that we reconstruct f from its samples. Dividing by T we get $\frac{N}{T} \geq \frac{2M+1}{T}$, which states that the sampling frequency ($f_s = N/T$ is the number of samples per second) should be bigger than two times the highest frequency (M/T). In Figure 3.2 we try to get some intuition on this by considering some pure tones. In Figure (a) we consider one period of $\sin 2\pi t$, and see that we need at least two sample points in $[0,1]$, since one point would clearly be too little. This translates directly into having at least eight sample points in Figure (b) where the function is $\sin 2\pi 4t$, which has four periods in the interval $[0,1]$.

Let us restate the reconstruction of f without the DFT. The reconstruction formula was

$$f(t) = \frac{1}{\sqrt{N}} \sum_{n=-M}^{M} y_n e^{2\pi i n t/T}.$$

If we here substitute $\boldsymbol{y} = F_N \boldsymbol{x}$ we get that this equals

$$\begin{aligned}
&\frac{1}{N} \sum_{n=-M}^{M} \sum_{k=0}^{N-1} x_k e^{-2\pi i n k/N} e^{2\pi i n t/T} \\
&= \sum_{k=0}^{N-1} \frac{1}{N} \left(\sum_{n=-M}^{M} x_k e^{2\pi i n (t/T - k/N)} \right) \\
&= \sum_{k=0}^{N-1} \frac{1}{N} e^{-2\pi i M (t/T - k/N)} \frac{1 - e^{2\pi i (2M+1)(t/T-k/N)}}{1 - e^{2\pi i (t/T - k/N)}} x_k \\
&= \sum_{k=0}^{N-1} \frac{1}{N} \frac{\sin(\pi(t - kT_s)/T_s)}{\sin(\pi(t - kT_s)/T)} f(kT_s),
\end{aligned}$$

where we have substituted $N = T/T_s$ (deduced from $T = NT_s$ with T_s being the sampling period). Let us summarize our findings as follows:

Theorem 3.18 (Sampling theorem and the ideal interpolation formula for periodic functions)**.** Let f be a periodic function with period T, and assume that f has no frequencies higher than νHz. Then f can be reconstructed exactly from its samples $f(0), \ldots, f((N-1)T_s)$ (where T_s is the sampling period and $N = \frac{T}{T_s}$ is the number of samples per period) when the sampling rate $F_s = \frac{1}{T_s}$ is bigger than 2ν. Moreover, the reconstruction can be performed through the formula

$$f(t) = \sum_{k=0}^{N-1} f(kT_s) \frac{1}{N} \frac{\sin(\pi(t - kT_s)/T_s)}{\sin(\pi(t - kT_s)/T)}. \tag{3.10}$$

Formula (3.10) is also called the ideal interpolation formula for periodic functions. Such formulas, where one reconstructs a function based on a weighted sum of the sample values, are more generally called *interpolation formulas*. We will return to other interpolation formulas later, which have different properties.

Note that f itself may not be equal to a finite Fourier series, and reconstruction is in general not possible then. Interpolation as performed in Section 3.2.2 is still possible, however, but the $g(t)$ we obtain from Corollary 3.16 may be different from $f(t)$.

Exercises for Section 3.2

Ex. 1 — Compute the 4 point DFT of the vector $(2, 3, 4, 5)^T$.

Ex. 2 — As in Example 3.9, state the exact cartesian form of the Fourier matrix for the cases $N = 6$, $N = 8$, and $N = 12$.

Ex. 3 — Let $\boldsymbol{x}$ be the vector with entries $x_k = c^k$. Show that the DFT of $\boldsymbol{x}$ is given by the vector with components

$$y_n = \frac{1}{\sqrt{N}} \frac{1 - c^N}{1 - ce^{-2\pi i n/N}}$$

for $n = 0, \ldots, N - 1$.

Ex. 4 — If $\boldsymbol{x}$ is complex, Write the DFT in terms of the DFT on real sequences. Hint: Split into real and imaginary parts, and use linearity of the DFT.

Ex. 5 — As in Example 3.10, write a function

```
function x=IDFTImpl(y)
```

which computes the IDFT.

Ex. 6 — Let G be the $N \times (2M + 1)$-matrix with entries $\frac{1}{\sqrt{N}} e^{2\pi i n k/N}$, and H the $N \times N$-matrix with entries $\frac{1}{\sqrt{N}} e^{2\pi i n k/N}$. Show that $G^H G = I_{2M+1}$ and that $G^H H = \begin{pmatrix} I & \mathbf{0} \end{pmatrix}$ when $N \geq 2M + 1$. Write also down an expression for $G^H G$ when $N < 2M+1$, to show that it is in general different from the identity matrix.

3.3 Operations on vectors: filters

In Chapter 1 we defined some operations on digital sounds, which we loosely referred to as filters. One example was the averaging filter

$$z_n = \frac{1}{4}(x_{n-1} + 2x_n + x_{n+1}), \quad \text{for } n = 0, 1, \ldots, N - 1 \tag{3.11}$$

of Example 1.25 where $\boldsymbol{x}$ denotes the input vector and $\boldsymbol{z}$ the output vector. Before we state the formal definition of filters, let us consider Equation (3.11) in some more detail to get more intuition about filters.

As before we assume that the input vector is periodic with period N, so that $x_{n+N} = x_n$. Our first observation is that the output vector $\boldsymbol{z}$ is also periodic with period N since

$$z_{n+N} = \frac{1}{4}(x_{n+N-1} + 2x_{n+N} + x_{n+N+1}) = \frac{1}{4}(x_{n-1} + 2x_n + x_{n+1}) = z_n.$$

The filter is also clearly a linear transformation and may therefore be represented by an $N \times N$ matrix S that maps the vector $\boldsymbol{x} = (x_0, x_1, \ldots, x_{N-1})$ to the vector $\boldsymbol{z} = (z_0, z_1, \ldots, z_{N-1})$, i.e., we have $\boldsymbol{z} = S\boldsymbol{x}$. To find S we note that for $1 \leq n \leq N-2$ it is clear from Equation (3.11) that row n has the value $1/4$ in column $n-1$, the value $1/2$ in column n, and the value $1/4$ in column $n+1$. For row 0 we must be a bit more careful, since the index -1 is outside the legal range of the indices. This is where the periodicity helps us out so that

$$z_0 = \frac{1}{4}(x_{-1} + 2x_0 + x_1) = \frac{1}{4}(x_{N-1} + 2x_0 + x_1) = \frac{1}{4}(2x_0 + x_1 + x_{N-1}).$$

From this we see that row 0 has the value $1/4$ in columns 1 and $N-1$, and the value $1/2$ in column 0. In exactly the same way we can show that row $N-1$ has the entry $1/4$ in columns 0 and $N-2$, and the entry $1/2$ in column $N-1$. In summary, the matrix of the averaging filter is given by

$$S = \frac{1}{4}\begin{pmatrix} 2 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 1 \\ 1 & 2 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 2 & 1 \\ 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & 2 \end{pmatrix}. \tag{3.12}$$

A matrix on this form is called a Toeplitz matrix. Such matrices are very popular in the literature and have many applications. The general definition may seem complicated, but is in fact quite straightforward:

Definition 3.19 (Toeplitz matrices)**.** An $N \times N$-matrix S is called a Toeplitz matrix if its elements are constant along each diagonal. More formally, $S_{k,l} = S_{k+s,l+s}$ for all nonnegative integers k, l, and s such that both $k+s$ and $l+s$ lie in the interval $[0, N-1]$. A Toeplitz matrix is said to be circulant if in addition

$$S_{(k+s) \bmod N,(l+s) \bmod N} = S_{k,l}$$

for all integers k, l in the interval $[0, N-1]$, and all s (Here mod denotes the remainder modulo N).

As the definition says, a Toeplitz matrix is constant along each diagonal, while the additional property of being circulant means that each row and column of the matrix 'wraps over' at the edges. It is quite easy to check that the matrix S given by Equation (3.12) satisfies Definition 3.19 and is a circulant Toeplitz matrix. A Toeplitz matrix is uniquely identified by the values on its nonzero diagonals, and a circulant Toeplitz matrix is uniquely identified by the $N/2$ diagonals above or on the main diagonal, and the $N/2$ diagonals below the main diagonal. We will encounter Toeplitz matrices also in other contexts in these notes.

In Chapter 1, the operations we loosely referred to as filters, such as formula (3.11), could all be written on the form

$$z_n = \sum_k t_k x_{n-k}. \tag{3.13}$$

Many other operations are also defined in this way. The values t_k will be called *filter coefficients*. The range of k is not specified, but is typically an interval around 0, since z_n usually is calculated by combining x_ks with indices close to n. Both positive and negative indices are allowed. As an example, for formula (3.11) k ranges over $-1, 0$, and 1, and we have that $t_{-1} = t_1 = 1/4$, and $t_0 = 1/2$. By following the same argument as above, the following is clear:

Proposition 3.20. Any operation defined by Equation (3.13) is a linear transformation which transforms a vector of period N to another of period N. It may therefore be represented by an $N \times N$ matrix S that maps the vector $\boldsymbol{x} = (x_0, x_1, \ldots, x_{N-1})$ to the vector $\boldsymbol{z} = (z_0, z_1, \ldots, z_{N-1})$, i.e., we have $\boldsymbol{z} = S\boldsymbol{x}$. Moreover, the matrix S is a circulant Toeplitz matrix, and the first column $\boldsymbol{s}$ of this matrix is given by

$$s_k = \begin{cases} t_k, & \text{if } 0 \leq k < N/2; \\ t_{k-N} & \text{if } N/2 \leq k \leq N-1. \end{cases} \tag{3.14}$$

In other words, the first column of S can be obtained by placing the coefficients in (3.13) with positive indices at the beginning of $\boldsymbol{s}$, and the coefficients with negative indices at the end of $\boldsymbol{s}$.

This proposition will be useful for us, since it explains how to pass from the form (3.13), which is most common in practice, to the matrix form S.

Example 3.21. Let us apply Proposition 3.20 on the operation defined by formula (3.11):

1. for $k = 0$ Equation 3.14 gives $s_0 = t_0 = 1/2$.
2. For $k = 1$ Equation 3.14 gives $s_1 = t_1 = 1/4$.
3. For $k = N-1$ Equation 3.14 gives $s_{N-1} = t_{-1} = 1/4$.

For all k different from 0, 1, and $N-1$, we have that $s_k = 0$. Clearly this gives the matrix in Equation (3.12).

Proposition 3.20 is also useful when we have a circulant Toeplitz matrix S, and we want to find filter coefficients t_k so that $\boldsymbol{z} = S\boldsymbol{x}$ can be written as in Equation (3.13):

Example 3.22. Consider the matrix

$$S = \begin{pmatrix} 2 & 1 & 0 & 3 \\ 3 & 2 & 1 & 0 \\ 0 & 3 & 2 & 1 \\ 1 & 0 & 3 & 2 \end{pmatrix}.$$

This is a circulant Toeplitz matrix with $N = 4$, and we see that $s_0 = 2$, $s_1 = 3$, $s_2 = 0$, and $s_3 = 1$. The first equation in (3.14) gives that $t_0 = s_0 = 2$, and $t_1 = s_1 == 3$. The second equation in (3.14) gives that $t_{-2} = s_2 = 0$, and $t_{-1} = s_3 = 1$. By including only the t_k which are nonzero, the operation can be written as

$$z_n = t_{-1}x_{n-(-1)} + t_0 x_n + t_1 x_{n-1} + t_2 x_{n-2} = x_{n+1} + 2x_0 + 3x_{n-1}.$$

3.3.1 Formal definition of filters and frequency response

Let us now define filters formally, and establish their relationship to Toeplitz matrices. We have seen that a sound can be decomposed into different frequency components, and we would like to define filters as operations which adjust these frequency components in a predictable way. One such example is provided in Example 3.15, where we simply set some of the frequency components to 0. The natural starting point is to require for a filter that the output of a pure tone is a pure tone with the same frequency.

Definition 3.23 (Digital filters and frequency response)**.** A linear transformation $S : \mathbb{R}^N \mapsto \mathbb{R}^N$ is a said to be a digital filter, or simply a filter, if it maps any Fourier vector in $\mathbb{R}^N$ to a multiple of itself. In other words, for any integer n in the range $0 \leq n \leq N-1$ there exists a value $\lambda_{S,n}$ so that

$$S(\phi_n) = \lambda_{S,n}\phi_n, \tag{3.15}$$

i.e., the N Fourier vectors are the eigenvectors of S. The vector of (eigen)values $\boldsymbol{\lambda}_S = (\lambda_{S,n})_{n=0}^{N-1}$ is often referred to as the frequency response of S.

We will identify the linear transformation S with its matrix relative to the standard basis. Since the Fourier basis vectors are orthogonal vectors, S is clearly orthogonally diagonalizable. Since also the Fourier basis vectors are the columns in $(F_N)^H$, we have that

$$S = F_N^H D F_N \tag{3.16}$$

whenever S is a digital filter, where D has the frequency response (i.e. the eigenvalues) on the diagonal[1]. In particular, if S_1 and S_2 are digital filters, we

[1]Recall that the orthogonal diagonalization of S takes the form $S = PDP^T$, where P contains as columns an orthonormal set of eigenvectors, and D is diagonal with the eigenvectors listed on the diagonal (see Section 7.1 in [7]).

can write $S_1 = F_N^H D_1 F_N$ and $S_2 = F_N^H D_2 F_N$, so that

$$S_1 S_2 = F_N^H D_1 F_N F_N^H D_2 F_N = F_N^H D_1 D_2 F_N.$$

Since $D_1 D_2 = D_2 D_1$ for any diagonal matrices, we get the following corollary:

Corollary 3.24. All digital filters commute, i.e. if S_1 and S_2 are digital filters, $S_1 S_2 = S_2 S_1$.

There are several equivalent characterizations of a digital filter. The first one was stated above in terms of the definition through eigenvectors and eigenvalues. The next characterization helps us prove that the operations from Chapter 1 actually are filters.

Theorem 3.25. A linear transformation S is a digital filter if and only if it is a circulant Toeplitz matrix.

Proof. That S is a filter is equivalent to the fact that $S = (F_N)^H D F_N$ for some diagonal matrix D. We observe that the entry at position (k, l) in S is given by

$$S_{k,l} = \frac{1}{N}\sum_{n=0}^{N-1} e^{2\pi ikn/N}\lambda_{S,n}e^{-2\pi inl/N} = \frac{1}{N}\sum_{n=0}^{N-1} e^{2\pi(k-l)n/N}\lambda_{S,n}.$$

Another entry on the same diagonal (shifted s rows and s columns) is

$$\begin{aligned} S_{(k+s) \bmod N,(l+s) \bmod N} &= \frac{1}{N}\sum_{n=0}^{N-1} e^{2\pi i((k+s) \bmod N-(l+s) \bmod N)n/N}\lambda_{S,n} \\ &= \frac{1}{N}\sum_{n=0}^{N-1} e^{2\pi i(k-l)n/N}\lambda_{S,n} = S_{k,l}, \end{aligned}$$

which proves that S is a circulant Toeplitz matrix. □

In particular, operations defined by (3.13) are digital filters, when restricted to vectors with period N. The following results enables us to compute the eigenvalues/frequency response easily, so that we do not need to form the characteristic polynomial and find its roots:

Theorem 3.26. Any digital filter is uniquely characterized by the values in the first column of its matrix. Moreover, if $\boldsymbol{s}$ is the first column in S, the frequency response of S is given by

$$\boldsymbol{\lambda}_S = \sqrt{N} F_N \boldsymbol{s}. \tag{3.17}$$

Conversely, if we know the frequency response $\boldsymbol{\lambda}_S$, the first column $\boldsymbol{s}$ of S is given by

$$\boldsymbol{s} = \frac{1}{\sqrt{N}}(F_N)^H \boldsymbol{\lambda}_S. \tag{3.18}$$

Proof. If we replace S by $(F_N)^H D F_N$ we find that

$$F_N \boldsymbol{s} = F_N S \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = F_N F_N^H D F_N \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = D F_N \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \frac{1}{\sqrt{N}} D \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix},$$

where we have used the fact that the first column in F_N has all entries equal to $1/\sqrt{N}$. But the the diagonal matrix D has all the eigenvalues of S on its diagonal, and hence the last expression is the vector of eigenvalues $\boldsymbol{\lambda}_S$, which proves (3.17). Equation (3.18) follows directly by applying the inverse DFT to (3.17). □

Since the first column $\boldsymbol{s}$ characterizes the filter S uniquely, one often refers to S by the vector $\boldsymbol{s}$. The first column $\boldsymbol{s}$ is also called the *impulse response*. This name stems from the fact that we can write $\boldsymbol{s} = S\boldsymbol{e}_0$, i.e., the vector $\boldsymbol{s}$ is the output (often called response) to the vector $\boldsymbol{e}_0$ (often called an impulse).

Example 3.27. The identity matrix is a digital filter since $I = (F_N)^H I F_N$. Since $\boldsymbol{e}_0 = S\boldsymbol{e}_0$, it has impulse response $\boldsymbol{s} = \boldsymbol{e}_0$. Its frequency response has 1 in all components and therefore preserves all frequencies, as expected.

Equations (3.16), (3.17), and (3.18) are important relations between the matrix- and frequency representations of a filter. We see that the DFT is a crucial ingredient in these relations. A consequence is that, once you recognize a matrix as circulant Toeplitz, you do not need to make the tedious calculation of eigenvectors and eigenvalues which you are used to. Let us illustrate this with an example.

Example 3.28. Let us compute the eigenvalues and eigenvectors of the simple matrix

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

It is straightforward to compute the eigenvalues and eigenvectors of this matrix the way you learnt in your first course in linear algebra. However, this matrix is also a circulant Toeplitz matrix, so that we can also use the results in this section to compute the eigenvalues and eigenvectors. Since here $N = 2$, we have that $e^{2\pi ink/N} = e^{\pi ink} = (-1)^{nk}$. This means that the Fourier basis vectors are $(1,1)/\sqrt{2}$ and $(1,-1)/\sqrt{2}$, which also are the eigenvectors of S. The eigenvalues are the frequency response of S, which can be obtained as

$$\sqrt{N} F_N \boldsymbol{s} = \sqrt{2}\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 4 \\ 1 \end{pmatrix} = \begin{pmatrix} 5 \\ 3 \end{pmatrix}$$

The eigenvalues are thus 3 and 5. You could have obtained the same result with Matlab. Note that Matlab may not return the eigenvectors exactly as the Fourier basis vectors, since the eigenvectors are not unique (the multiple of an eigenvector is still an eigenvector). In this case Matlab may for instance switch the signs of the eigenvectors. We have no control over what Matlab actually chooses to do, since it uses some underlying numerical algorithm for computing eigenvectors which we can't influence.

In signal processing, the frequency content of a vector (i.e., its DFT) is also referred to as its spectrum. This may be somewhat confusing from a linear algebra perspective, because in this context the term spectrum is used to denote the eigenvalues of a matrix. But because of Theorem 3.26 this is not so confusing after all if we interpret the spectrum of a vector (in signal processing terms) as the spectrum of the corresponding digital filter (in linear algebra terms).

3.3.2 Some properties of the frequency response

Equation (3.17) states that the frequency response can be written as

$$\lambda_{S,n} = \sum_{k=0}^{N-1} s_k e^{-2\pi i n k/N}, \quad \text{for } n = 0, 1, \ldots, N-1, \tag{3.19}$$

where s_k are the components of the impulse response $\boldsymbol{s}$.

Example 3.29. When only few of the coefficients s_k are nonzero, it is possible to obtain nice expressions for the frequency response. To see this, let us compute the frequency response of the filter defined from formula (3.11). We saw that the first column of the corresponding Toeplitz matrix satisfied $s_0 = 1/2$, and $s_{N-1} = s_1 = 1/4$. The frequency response is thus

$$\begin{aligned}\lambda_{S,n} &= \frac{1}{2}e^0 + \frac{1}{4}e^{-2\pi i n/N} + \frac{1}{4}e^{-2\pi i n(N-1)/N} \\ &= \frac{1}{2}e^0 + \frac{1}{4}e^{-2\pi i n/N} + \frac{1}{4}e^{2\pi i n/N} = \frac{1}{2} + \frac{1}{2}\cos(2\pi n/N).\end{aligned}$$

If we make the substitution $\omega = 2\pi n/N$ in the formula for $\lambda_{S,n}$, we may interpret the frequency response as the values on a continuous function on $[0, 2\pi)$.

Theorem 3.30. The function $\lambda_S(\omega)$ defined on $[0, 2\pi)$ by

$$\lambda_S(\omega) = \sum_k t_k e^{-ik\omega}, \tag{3.20}$$

where t_k are the filter coefficients of S, satisfies

$$\lambda_{S,n} = \lambda_S(2\pi n/N) \text{ for } n = 0, 1, \ldots, N-1$$

for any N. In other words, regardless of N, the frequency reponse lies on the curve λ_S.

Proof. For any N we have that

$$\begin{aligned}
\lambda_{S,n} &= \sum_{k=0}^{N-1} s_k e^{-2\pi ink/N} = \sum_{0\le k<N/2} s_k e^{-2\pi ink/N} + \sum_{N/2\le k\le N-1} s_k e^{-2\pi ink/N} \\
&= \sum_{0\le k<N/2} t_k e^{-2\pi ink/N} + \sum_{N/2\le k\le N-1} t_{k-N} e^{-2\pi ink/N} \\
&= \sum_{0\le k<N/2} t_k e^{-2\pi ink/N} + \sum_{-N/2\le k\le -1} t_k e^{-2\pi in(k+N)/N} \\
&= \sum_{0\le k<N/2} t_k e^{-2\pi ink/N} + \sum_{-N/2\le k\le -1} t_k e^{-2\pi ink/N} \\
&= \sum_{-N/2\le k<N/2} t_k e^{-2\pi ink/N} = \lambda_S(\omega).
\end{aligned}$$

where we have used Equation (3.14). □

Both $\lambda_S(\omega)$ and $\lambda_{S,n}$ will be referred to as frequency responses in the following. When there is a need to distinguish the two we will call $\lambda_{S,n}$ the *vector frequency response*, and $\lambda_S(\omega))$ the *continuous frequency response*. ω is also called *angular frequency*.

The difference in the definition of the continuous- and the vector frequency response lies in that one uses the filter coefficients t_k, while the other uses the impulse response s_k. While these contain the same values, they are stored differently. Had we used the impulse response to define the continuous frequency response, we would have needed to compute $\sum_{k=0}^{N-1} s_k e^{-\pi i\omega}$, which does not converge when $N \to \infty$ (although it gives the right values at all points $\omega = 2\pi n/N$ for all N)! The filter coefficcients avoid this convergence problem, however, since we assume that only t_k with $|k|$ small are nonzero. In other words, filter coefficients are used in the definition of the continuous frequency response so that we can find a continuous curve where we can find the vector frequency response values for all N.

The frequency response contains the important characteristics of a filter, since it says how it behaves for the different frequencies. When analyzing a filter, we therefore often plot the frequency response. Often we plot only the absolute value (or the magnitude) of the frequency response, since this is what explains how each frequency is amplified or attenuated. Since λ_S is clearly periodic with period 2π, we may restrict angular frequency to the interval $[0, 2\pi)$. The conclusion in Observation 3.14 was that the low frequencies in a vector correspond to DFT indices close to 0 and $N-1$, and high frequencies correspond to DFT indices close to $N/2$. This observation is easily translated to a statement about angular frequencies:

Observation 3.31. When plotting the frequency response on $[0, 2\pi)$, angular frequencies near 0 and 2π correspond to low frequencies, angular frequencies near π correspond to high frequencies

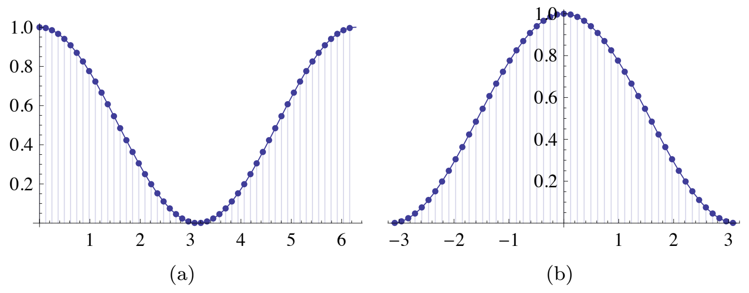


Figure 3.3: The (absolute value of the) frequency response of the smoothing filter in Example 1.25 which we discussed at the beginning of this section.

λ_S may also be viewed as a function defined on the interval $[-\pi, \pi)$. Plotting on $[-\pi, \pi]$ is often done in practice, since it makes clearer what corresponds to lower frequencies, and what corresponds to higher frequencies:

Observation 3.32. When plotting the frequency response on $[-\pi, \pi)$, angular frequencies near 0 correspond to low frequencies, angular frequencies near $\pm\pi$ correspond to high frequencies.

Example 3.33. In Example 3.29 we computed the vector frequency response of the filter defined in formula (3.11). The filter coefficients are here $t_{-1} = 1/4$, $t_0 = 1/2$, and $t_1 = 1/4$. The continuous frequency response is thus

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

Clearly this matches the computation from Example 3.29. Figure 3.3 shows plots of this frequency response, plotted on the intervals $[0, 2\pi)$ and $[-\pi, \pi)$. Both the continuous frequency response and the vector frequency response for $N = 51$ are shown. Figure (b) shows clearly how the high frequencies are softened by the filter.

Since the frequency response is essentially a DFT, it inherits several properties from Theorem 3.11.

Theorem 3.34. The frequency response has the properties:

1. The continuous frequency response satisfies $\lambda_S(-\omega) = \overline{\lambda_S(\omega)}$.
2. If S is a digital filter, S^T is also a digital filter. Morever, if the frequency response of S is $\lambda_S(\omega)$, then the frequency response of S^T is $\overline{\lambda_S(\omega)}$.

3. If S is symmetric, λ_S is real. Also, if S is antisymmetric (the element on the opposite side of the diagonal is the same, but with opposite sign), λ_S is purely imaginary.

4. If S_1 and S_2 are digital filters, then S_1S_2 also is a digital filter, and $\lambda_{S_1S_2}(\omega) = \lambda_{S_1}(\omega)\lambda_{S_2}(\omega)$.

Proof. Property 1. and 3. follow directly from Theorem 3.11. Transposing a matrix corresponds to reversing the first colum of the matrix and thus also the filter coefficients. Due to this Property 2. also follows from Theorem 3.11. The last property follows in the same was as we showed that filters commute:

$$S_1S_2 = (F_N)^H D_1 F_N (F_N)^H D_2 F_N = (F_N)^H D_1 D_2 F_N.$$

The frequency response of S_1S_2 is thus obtained by multiplying the frequency responses of S_1 and S_2. □

In particular the frequency response may not be real, although this was the case in the first example of this section. Theorem 3.34 applies both for the vector- and continuous frequency response. Also, clearly $S_1 + S_2$ is a filter when S_1 and S_2 are. The set of all filters is thus a vector space, which also is closed under multiplication. Such a space is called an *algebra*. Since all filters commute, this algebra is also called a *commutative algebra*.

Example 3.35. Assume that the filters S_1 and S_2 have the frequency responses $\lambda_{S_1}(\omega) = \cos(2\omega)$, $\lambda_{S_2}(\omega) = 1+3\cos\omega$. Let us see how we can use Theorem 3.34 to compute the filter coefficients and the matrix of the filter $S = S_1S_2$. We first notice that, since both frequency responses are real, all S_1, S_2, and $S = S_1S_2$ are symmetric. We rewrite the frequency responses as

$$\begin{aligned}
\lambda_{S_1}(\omega) &= \frac{1}{2}(e^{2i\omega} + e^{-2i\omega}) = \frac{1}{2}e^{2i\omega} + \frac{1}{2}e^{-2i\omega} \\
\lambda_{S_2}(\omega) &= 1 + \frac{3}{2}(e^{i\omega} + e^{-i\omega}) = \frac{3}{2}e^{i\omega} + 1 + \frac{3}{2}e^{-i\omega}.
\end{aligned}$$

We now get that

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

From this expression we see that the filter coefficients of S are $t_{\pm 1} = 3/4$, $t_{\pm 2} = 1/2$, $t_{\pm 3} = 3/4$. All other filter coefficients are 0. Using Theorem 3.20, we get that $s_1 = 3/4$, $s_2 = 1/2$, and $s_3 = 3/4$, while $s_{N-1} = 3/4$, $s_{N-2} = 1/2$, and $s_{N-3} = 3/4$ (all other s_k are 0). This gives us the matrix representation of S.

3.3.3 Assembling the filter matrix and compact notation

Let us return to how we first defined a filter in Equation (3.13):

$$z_n = \sum_k t_k x_{n-k}.$$

As mentioned, the range of k may not be specified. In some applications in signal processing there may in fact be infinitely many nonzero t_k. However, when $\boldsymbol{x}$ is assumed to have period N, we may as well assume that the k's range over an interval of length N (else many of the t_k's can be added together to simplify the formula). Also, any such interval can be chosen. It is common to choose the interval so that it is centered around 0 as much as possible. For this, we can choose for instance $[\lfloor N/2 \rfloor - N + 1, \lfloor N/2 \rfloor]$. With this choice we can write Equation (3.13) as

$$z_n = \sum_{k=\lfloor N/2 \rfloor - N+1}^{\lfloor N/2 \rfloor} t_k x_{n-k}. \tag{3.21}$$

The index range in this sum is typically even smaller, since often much less than N of the t_k are nonzero (For Equation (3.11), there were only three nonzero t_k). In such cases one often uses a more compact notation for the filter:

Definition 3.36 (Compact notation for filters)**.** Let $k_{\text{min}} \leq 0$, $k_{\text{max}} \geq 0$ be the smallest and biggest index of a filter coefficient in Equation (3.21) so that $t_k \neq 0$ (if no such values exist, let $k_{\text{min}} = 0$, or $k_{\text{max}} = 0$), i.e.

$$z_n = \sum_{k=k_{min}}^{k_{max}} t_k x_{n-k}. \tag{3.22}$$

We will use the following compact notation for S:

$$S = \{t_{k_{min}}, \ldots, t_{-1}, \underline{t_0}, t_1, \ldots, t_{k_{max}}\}.$$

In other words, the entry with index 0 has been underlined, and only the nonzero t_k's are listed. By the length of S, denoted $l(S)$, we mean the number $k_{max} - k_{min}$.

One seldom writes out the matrix of a filter, but rather uses this compact notation. Note that the length of S can also be written as the number of nonzero filter coefficients minus 1. $l(S)$ thus follows the same convention as the degree of a polynomial: It is 0 if the polynomial is constant (i.e. one nonzero filter coefficient).

Example 3.37. Using the compact notation for a filter, we would write $S = \{1/4, \underline{1/2}, 1/4\}$ for the filter given by formula (3.11)). For the filter

$$z_n = x_{n+1} + 2x_0 + 3x_{n-1}$$

from Example 3.22, we would write $S = \{1, \underline{2}, 3\}$.

Equation (3.13) is also called the convolution of the two vectors $\boldsymbol{t}$ and $\boldsymbol{x}$. Convolution is usually defined without the assumption that the vectors are periodic, and without any assumption on their lengths (i.e. they may be sequences of inifinite length):

Definition 3.38 (Convolution of vectors). By the *convolution* of two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ we mean the vector $\boldsymbol{x} * \boldsymbol{y}$ defined by

$$(\boldsymbol{x} * \boldsymbol{y})_n = \sum_k x_k y_{n-k}. \tag{3.23}$$

In other words, applying a filter S corresponds to convolving the filter coefficients of S with the input. If both $\boldsymbol{x}$ and $\boldsymbol{y}$ have infinitely many nonzero entries, the sum is an infinite one, which may diverge. For the filters we look at, we always have a finite number of nonzero entries t_k, so we never have this convergence problem since the sum is a finite one. MATLAB has the built-in function `conv` for convolving two vectors of finite length. This function does not indicate which indices the elements of the returned vector belongs to, however. Exercise 11 explains how one may keep track of these indices.

Since the number of nonzero filter coefficients is typically much less than N (the period of the input vector), the matrix S have many entries which are zero. Multiplication with such matrices requires less additions and multiplications than for other matrices: If S has k nonzero filter coefficients, S has Nk nonzero entries, so that kN multiplications and $(k-1)N$ additions are needed to compute $S\boldsymbol{x}$. This is much less than the N^2 multiplications and $(N-1)N$ additions needed in the general case. Perhaps more important is that we need not form the entire matrix, we can perform the matrix multiplication directly in a loop. Exercise 10 investigates this further. For large N we risk running into out of memory situations if we had to form the entire matrix.

3.3.4 Some examples of filters

We have now established the basic theory of filters, so it is time to study some specific examples. Many of the filters below were introduced in Section 1.4.

Example 3.39 (Time delay filters). The simplest possible type of Toeplitz matrix is one where there is only one nonzero diagonal. Let us define the Toeplitz matrix E_d as the one which has first column equal to $\boldsymbol{e}_d$. In the notation above, and when $d > 0$, this filter can also be written as $S = \{\underline{0}, \ldots, 1\}$ where the 1 occurs at position d. We observe that

$$(E_d\boldsymbol{x})_n = \sum_{k=0}^{N-1} (E_d)_{n,k}\, x_k = \sum_{k=0}^{N-1} (E_d)_{(n-k) \bmod N, 0}\, x_k = x_{(n-d) \bmod N},$$

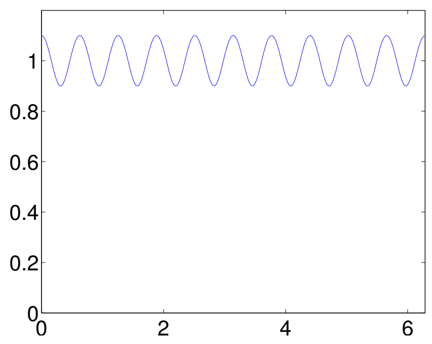


Figure 3.4: The frequency response of a filter which adds an echo with damping factor $c = 0.1$ and delay $d = 10$.

since only when $(n - k) \bmod N = d$ do we have a contribution in the sum. It is thus clear that multiplication with E_d delays a vector by d samples, in a circular way. For this reason E_d is also called a *time delay filter*. The frequency response of the time delay filter is clearly the function $\lambda_S(\omega) = e^{-id\omega}$, which has magnitude 1. This filter therefore does not change the magnitude of the different frequencies.

Example 3.40 (Adding echo)**.** In Example 1.23 we encountered a filter which could be used for adding echo to sound. Using our compact filter notation this can be written as

$$S = \{\underline{1}, 0, \ldots, 0, c\},$$

where the damping factor c appears after the delay d. The frequency response of this is $\lambda_S(\omega) = 1 + ce^{-id\omega}$. This frequency response is not real, which means that the filter is not symmetric. In Figure 3.4 we have plotted the magnitude of this frequency response with $c = 0.1$ and $d = 10$. We see that the response varies between 0.9 and 1.1, so that adding exho changes frequencies according to the damping factor c. The deviation from 1 is controlled by the damping factor c. Also, we see that the oscillation in the frequency response, as visible in the plot, is controlled by the delay d.

Previously we have claimed that some operations, such as averaging the samples, can be used for adjusting the bass and the treble of a sound. Theorem 3.25 supports this, since the averaging operations we have defined correspond to circulant Toeplitz matrices, which are filters which adjust the frequencies as dictated by the frequency response. Below we will analyze the frequency response of the corresponsing filters, to verify that it works as we have claimed for the frequencies corresponding to bass and treble in sound.

Example 3.41 (Reducing the treble)**.** In Example 1.25 we encountered the moving average filter

$$S = \left\{ \frac{1}{3}, \underline{\frac{1}{3}}, \frac{1}{3} \right\}.$$

This could be used for reducing the treble in a sound. If we set $N = 4$, the corresponding circulant Toeplitz matrix for the filter is

$$S = \frac{1}{3} \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

The frequency response is $\lambda_S(\omega) = (e^{i\omega} + 1 + e^{-i\omega})/3 = (1 + 2\cos(\omega))/3$. More generally, if the filter is $\boldsymbol{s} = (1, \cdots, \underline{1}, \cdots, 1)/(2L+1)$, where there is symmetry around 0, we recognize this as $\boldsymbol{x}/(2L+1)$, where $\boldsymbol{x}$ is a vector of ones and zeros, as defined in Example 3.8. From that example we recall that

$$\boldsymbol{x} = \frac{1}{\sqrt{N}} \frac{\sin(\pi n(2L+1)/N)}{\sin(\pi n/N)},$$

so that the frequency response of S is

$$\lambda_{S,n} = \frac{1}{2L+1} \frac{\sin(\pi n(2L+1)/N)}{\sin(\pi n/N)},$$

and

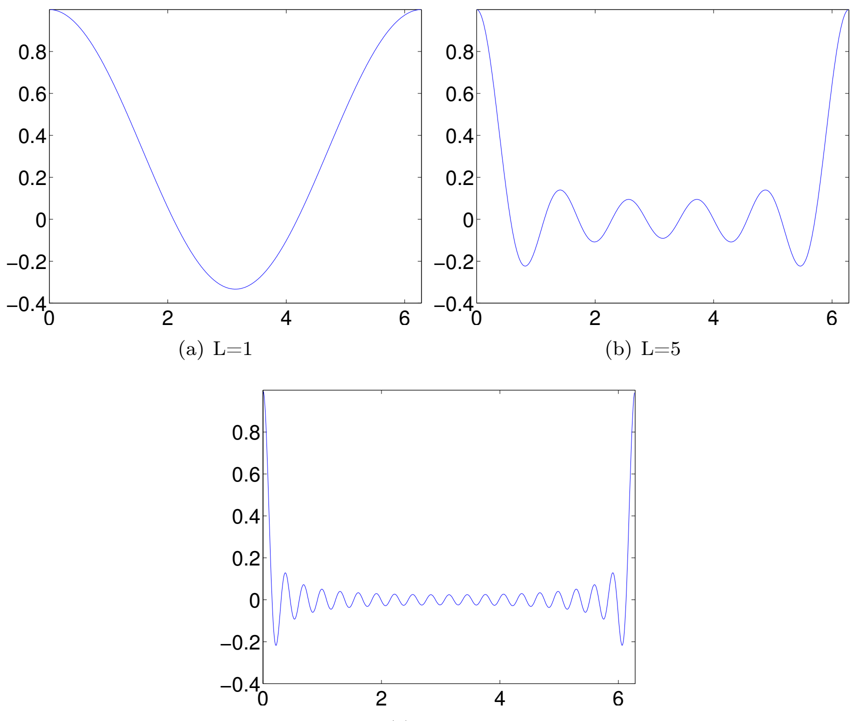
$$\lambda_S(\omega) = \frac{1}{2L+1} \frac{\sin((2L+1)\omega/2)}{\sin(\omega/2)}.$$

We clearly have

$$0 \leq \frac{1}{2L+1} \frac{\sin((2L+1)\omega/2)}{\sin(\omega/2)} \leq 1,$$

so this frequency response approaches 1 as $\omega \to 0^+$. The frequency response thus peaks at 0, and it is clear that this peak gets narrower and narrower as L increases, i.e. we use more and more samples in the averaging process. This appeals to our intuition that this kind of filters smooths the sound by keeping only lower frequencies. In Figure 3.5 we have plotted the frequency response for moving average filters with $L = 1$, $L = 5$, and $L = 20$. We see, unfortunately, that the frequency response is far from a filter which keeps some frequencies unaltered, while annihilating others (this is a desirable property which is refered to as being a *bandpass filter*): Although the filter distinguishes between high and low frequencies, it slightly changes the small frequencies. Moreover, the higher frequencies are not annihilated, even when we increase L to high values.

In the previous example we mentioned a filter which kept some frequencies unaltered, and annihilated others. This is a desirable property for filters, so let us give names to such filters:



(a) L=1 (b) L=5

(c) L=20

Figure 3.5: The frequency response of moving average filters of different length.

Definition 3.42. A filter S is called

1. a *lowpass filter* if $\lambda_S(\omega) \approx 1$ when ω is close to 0, and $\lambda_S(\omega) \approx 0$ when ω is close to π (i.e. S keeps low frequencies and annhilates high frequencies),
2. a *highpass filter* if $\lambda_S(\omega) \approx 1$ when ω is close to π, and $\lambda_S(\omega) \approx 0$ when ω is close to 0 (i.e. S keeps high frequencies and annhilates low frequencies),
3. a *bandpass filter* if $\lambda_S(\omega) \approx 1$ within some interval $[a, b] \subset [0, 2\pi]$, and $\lambda_S(\omega) \approx 0$ outside this interval.

This definition should be considered rather vague when it comes to what we mean by "ω close to $0, \pi$", and "$\lambda_S(\omega) \approx 0, 1$": in practice, when we talk about lowpass and highpass filters, it may be that the frequency responses are still quite far from what is commonly refered to as *ideal lowpass or highpass filters*, where the frequency response only assumes the values 0 and 1 near 0 and π. The next example considers an ideal lowpass filter.

Example 3.43 (Ideal lowpass filters)**.** By definition, the ideal lowpass filter keeps frequencies near 0, and removes frequencies near π. In Chapter 1 we mentioned that we were not able to find the filter coefficients for such a filter. We now have the theory in place in order to achieve this: In Example 3.15 we implemented the ideal lowpass filter with the help of the DFT. Mathematically, the code was equivalent to computing $(F_N)^H D F_N$ where D is the diagonal matrix with the entries $0, \ldots, L$ and $N - L, \ldots, N - 1$ being 1, the rest being 0. Clearly this is a digital filter, with frequency response as stated. If the filter should keep the angular frequencies $|\omega| \le \omega_c$ only, where ω_c describes the highest frequency we should keep, we should choose L so that $\omega_c = 2\pi L/N$. In Example 3.8 we computed the DFT of this vector, and it followed from Theorem 3.11 that the IDFT of this vector equals its DFT. This means that we can find the filter coefficients by using Equation (3.18): Since the IDFT was $\frac{1}{\sqrt{N}} \frac{\sin(\pi k(2L+1)/N)}{\sin(\pi k/N)}$, the filter coefficients are

$$\frac{1}{\sqrt{N}} \frac{1}{\sqrt{N}} \frac{\sin(\pi k(2L+1)/N)}{\sin(\pi k/N)} = \frac{1}{N} \frac{\sin(\pi k(2L+1)/N)}{\sin(\pi k/N)}.$$

This means that the filter coefficients lie as N points uniformly spaced on the curve $\frac{1}{N} \frac{\sin(\omega(2L+1)/2)}{\sin(\omega/2)}$ between 0 and π. This curve has been encountered many other places in these notes. The filter which keeps only the frequency $\omega_c = 0$ has all filter coefficients being $\frac{1}{N}$ (set $L = 1$), and when we include all frequencies (set $L = N$) we get the filter where $x_0 = 1$ and all other filter coefficients are 0. When we are between these two cases, we get a filter where s_0 is the biggest coefficient, while the others decrease towards 0 along the curve we have computed. The bigger L and N are, the quicker they decrease to zero. All filter coefficients are typically nonzero for this filter, since this curve is zero only at certain points. This is unfortunate, since it means that the filter is time-consuming to compute.

The two previous examples show an important duality between vectors which are 1 on some elements and 0 on others (also called window vectors), and the vector $\frac{1}{N}\frac{\sin(\pi k(2L+1)/N)}{\sin(\pi k/N)}$ (also called a sinc): filters of the one type correspond to frequency responses of the other type, and vice versa. The examples also show that, in some cases only the filter coefficients are known, while in other cases only the frequency response is known. In any case we can deduce the one from the other, and both cases are important.

Filters are much more efficient when there are few nonzero filter coefficients. In this respect the second example displays a problem: in order to create filters with particularly nice properties (such as being an ideal lowpass filter), one may need to sacrifice computational complexity by increasing the number of nonzero filter coefficients. The trade-off between computational complexity and desirable filter properties is a very important issue in *filter design theory.*

Example 3.44. In order to decrease the computational complexity for the ideal lowpass filter in Example 3.43, one can for instance include only the first filter coefficients, i.e. $\{\frac{1}{N}\frac{\sin(\pi k(2L+1)/N)}{\sin(\pi k/N)}\}_{k=-N_0}^{N_0}$, ignoring the last ones. Hopefully this gives us a filter where the frequency reponse is not that different from the ideal lowpass filter. In Figure 3.6 we show the corresponding frequency responses. In the figure we have set $N = 128$, $L = 32$, so that the filter removes all frequencies $\omega > \pi/2$. N_0 has been chosen so that the given percentage of all coefficients are included. Clearly the figure shows that we should be careful when we omit filter coefficients: if we drop too many, the frequency response is far away from that of an ideal bandpass filter.

Example 3.45 (Reducing the treble II)**.** Let S be the moving average filter of two elements, i.e.

$$(S\boldsymbol{x})_n = \frac{1}{2}(x_{n-1} + x_n).$$

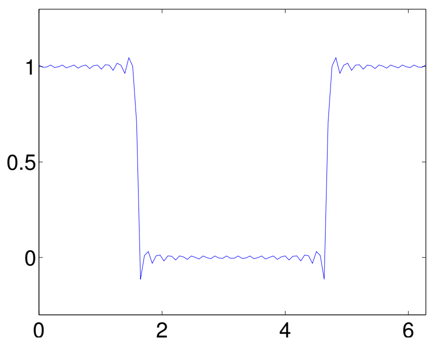
In Example 3.41 we had an odd number of filter coefficients. Here we have only two. We see that the frequency response in this case is

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

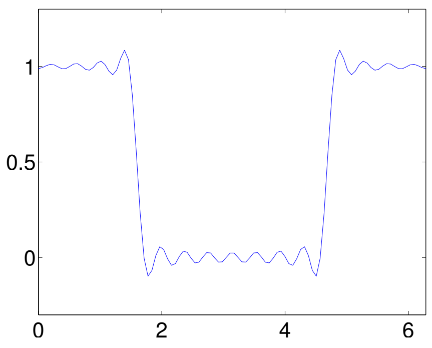
The frequency response is complex now, since the filter is not symmetric in this case. Let us now apply this filter k times, and denote by S_k the resulting filter. Theorem 3.34 gives us that the frequency response of S_k is

$$\lambda_{S^k}(\omega) = \frac{1}{2^k}(1 + e^{-i\omega})^k = e^{-ik\omega/2}\cos^k(\omega/2),$$

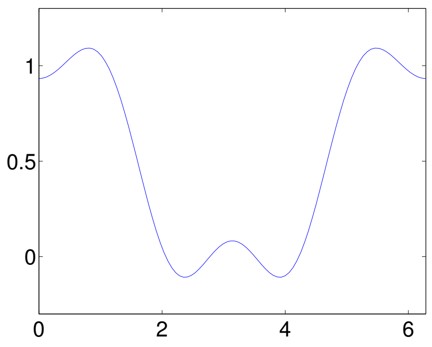
which is a polynomial in $e^{-i\omega}$ with the coefficients taken from Pascal's triangle. At least, this partially explains how filters with coefficients taken from Pascal's triangle appear, as in Example 1.25. These filters are more desirable than the



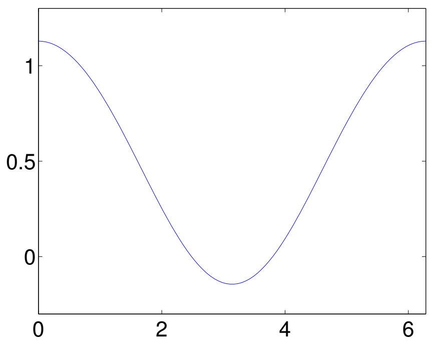
(a) all $N = 128$ filter coefficients



(b) 1/4 of all filter coefficients



(c) 1/16 of all filter coefficients



(d) 1/32 of all filter coefficients

Figure 3.6: The frequency response which results by omitting the last filter coefficients for the ideal lowpass filter.

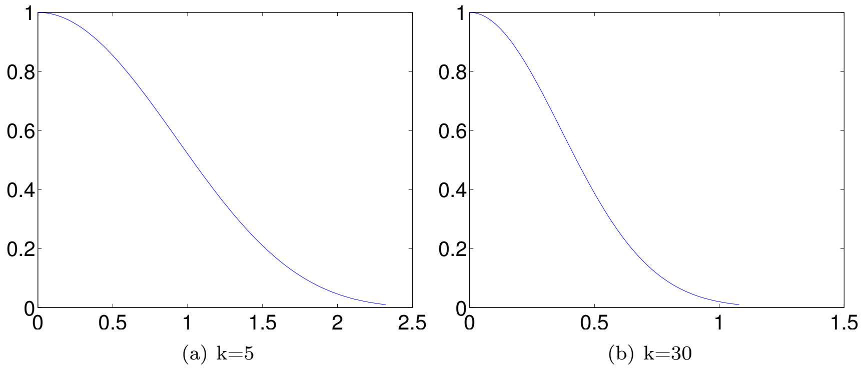


Figure 3.7: The frequency response of filters corresponding to a moving average filter convolved with itself k times.

moving average filters, and are used for smoothing abrupt changes in images and in sound. The reason is that, since we take a k'th power with k large, λ_{S^k} is more square-like near 0, i.e. it becomes more and more like a bandpass filter near 0. In Figure 3.7 we have plotted the magnitude of the frequence response when $k = 5$, and when $k = 30$. This behaviour near 0 is not so easy to see from the figure. Note that we have zoomed in on the frequency response to the area where it actually decreases to 0.

In Example 1.27 we claimed that we could obtain a bass-reducing filter by using alternating signs on the filter coefficients in a treble-reducing filter. Let us explain why this is the case. Let S be a filter with filter coefficients s_k, and let us consider the filter T with filter coefficient $(-1)^k s_k$. The frequency response of T is

$$\begin{aligned}\lambda_T(\omega) &= \sum_k (-1)^k s_k e^{-i\omega k} = \sum_k (e^{-i\pi})^k s_k e^{-i\omega k}\\ &= \sum_k e^{-i\pi k} s_k e^{-i\omega k} = \sum_k s_k e^{-i(\omega+\pi)k} = \lambda_S(\omega+\pi).\end{aligned}$$

where we have set $-1 = e^{-i\pi}$ (note that this is nothing but Property 4. in Theorem 3.11, with $d = N/2$). Now, for a lowpass filter S, $\lambda_S(\omega)$ has values near 1 when ω is close to 0 (the low frequencies), and values near 0 when ω is close to π (the high frequencies). For a highpass filter T, $\lambda_T(\omega)$ has values near 0 when ω is close to 0 (the low frequencies), and values near 1 when ω is close to π (the high frequencies). When T is obtained by adding an alternating sign to the filter coefficicents of S, The relation $\lambda_T(\omega) = \lambda_S(\omega + \pi)$ thus says that T is a highpass filter when S is a lowpass filter, and vice versa:

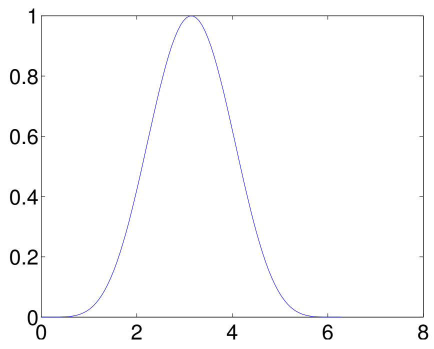


Figure 3.8: The frequency response of the bass reducing filter, which corresponds to row 5 of Pascal's triangle.

Observation 3.46. Assume that T is obtained by adding an alternating sign to the filter coefficicents of S. If S is a lowpass filter, then T is a highpass filter. If S is a highpass filter, then T is a lowpass filter.

The following example explains why this is the case.

Example 3.47 (Reducing the bass)**.** In Example 1.27 we constructed filters where the rows in Pascal's triangle appeared, but with alternating sign. The frequency response of this when using row 5 of Pascal's triangle is shown in Figure 3.8. It is just the frequency response of the corresponding treble-reducing filter shifted with π. The alternating sign can also be achieved if we write the frequency response $\frac{1}{2^k}(1+e^{-i\omega})^k$ from Example 3.45 as $\frac{1}{2^k}(1-e^{-i\omega})^k$, which corresponds to applying the filter $S(\boldsymbol{x}) = \frac{1}{2}(-x_{n-1}+x_n)$ k times.

3.3.5 Time-invariance of filters

The third characterization of digital filters we will prove is stated in terms of the following concept:

Definition 3.48 (Time-invariance)**.** A linear transformation from $\mathbb{R}^N$ to $\mathbb{R}^N$ is said to be time-invariant if, for any d, the output of the *delayed* input vector $\boldsymbol{z}$ defined by $z_n = x_{(n-d) \bmod N}$ is the delayed output vector $\boldsymbol{w}$ defined by $w_n = y_{(n-d) \bmod N}$.

We have the following result:

Theorem 3.49. A linear transformation S is a digital filter if and only if it is time-invariant.

Proof. Let $\boldsymbol{y} = S\boldsymbol{x}$, and $\boldsymbol{z}, \boldsymbol{w}$ as defined above. We have that

$$\begin{aligned} w_n = (S\boldsymbol{x})_{n-d} &= \sum_{k=0}^{N-1} S_{n-d,k}x_k \\ &= \sum_{k=0}^{N-1} S_{n,k+d}x_k = \sum_{k=0}^{N-1} S_{n,k}x_{k-d} \\ &= \sum_{k=0}^{N-1} S_{n,k}z_k = (S\boldsymbol{z})_n \end{aligned}$$

This proves that $S\boldsymbol{z} = \boldsymbol{w}$, so that S is time-invariant. □

By Example 3.39, delaying a vector with d elements corresponds to multiplication with the filter E_d. That S is time-invariant could thus also have been defined by demanding that $SE_d = E_dS$ for any d. That all filters are time invariant follows also immediately from the fact that all filters commute.

Due to Theorem 3.49, digital filters are also called LTI filters (LTI stands for Linear, Time-Invariant). By combining the definition of a digital filter with theorems 3.26, and 3.49, we get the following:

Theorem 3.50 (Characterizations of digital filters)**.** The following are equivalent characterizations of a digital filter:

1. $S = (F_N)^H DF_N$ for a diagonal matrix D, i.e. the Fourier basis is a basis of eigenvectors for S.
2. S is a circulant Toeplitz matrix.
3. S is linear and time-invariant.

3.3.6 Linear phase filters

Some filters are particularly important for applications:

Definition 3.51 (Linear phase)**.** We say that a digital filter S has linear phase if there exists some d so that $S_{d+n,0} = S_{d-n,0}$ for all n.

From Theorem 3.11 4. it follows that the argument of the frequency response at n for S is $-2\pi dn/N$. Moreover, the frequency response is real if $d = 0$, and this also corresponds to that the matrix is symmetric. One reason that linear phase filters are important for applications is that they can be more efficiently implemented than general filters. As an example, if S is symmetric around 0, we can write

$$\begin{aligned}(S\boldsymbol{x})_n &= \sum_{k=0}^{N-1} s_k x_{n-k} = \sum_{k=0}^{N/2-1} s_k x_{n-k} + \sum_{k=N/2}^{N-1} s_k x_{n-k} \\ &= \sum_{k=0}^{N/2-1} s_k x_{n-k} + \sum_{k=0}^{N/2-1} s_{k+N/2} x_{n-k-N/2} \\ &= \sum_{k=0}^{N/2-1} s_k x_{n-k} + \sum_{k=0}^{N/2-1} s_{N/2-k} x_{n-k-N/2} \\ &= \sum_{k=0}^{N/2-1} s_k x_{n-k} + \sum_{k=0}^{N/2-1} s_k x_{n+k} = \sum_{k=0}^{N/2-1} s_k (x_{n-k} + x_{n+k})\end{aligned}$$

If we compare the first and last expressions here, we need the same number of summations, but the number of multiplications needed in the latter expression has been halved. The same point can also be made about the factorization into a composition of many moving average filters of length 2 in Example 3.45. This also corresponds to a linear phase filter. Each application of a moving average filter of length 2 does not really require any multiplications, since multiplication with $\frac{1}{2}$ really corresponds to a bitshift. Therefore, the factorization of Example 3.45 removes the need for doing any multiplications at all, while keeping the number of additions the same. There is a huge computational saving in this. We will see another desirable property of linear phase filters in the next section, and we will also return to these filters later.

3.3.7 Perfect reconstruction systems

The following is easily proved, and left as exercises:

Theorem 3.52. The following hold:

1. The set of (circulant) Toeplitz matrices form a vector space.
2. If G_1 and G_2 are (circulant) Toeplitz matrices, then G_1G_2 is also a (circulant) Toeplitz matrix, and $l(G_1G_2) = l(G_1) + l(G_2)$.
3. $l(G) = 0$ if and only if G has only one nonzero diagonal.

An immediate corollary of this is in terms of what is called *perfect reconstruction systems*:

Definition 3.53 (Perfect reconstruction system)**.** By a perfect reconstruction system we mean a pair of $N \times N$-matrices (G_1, G_2) so that $G_2G_1 = I$. For a vector $\boldsymbol{x}$ we refer to $\boldsymbol{z} = G_1\boldsymbol{x}$ as the transformed vector. For a vector $\overline{\boldsymbol{z}}$ we refer to $\overline{\boldsymbol{x}} = G_2\overline{\boldsymbol{z}}$ as the reconstructed vector.

The terms perfect reconstruction, transformation, and reconstruction come from signal processing, where one thinks of G_1 as a transform, and G_2 as another transform which reconstructs the input to the first transform from its output. In practice, we are interested in finding perfect reconstruction systems where the transformed $G_1\boldsymbol{x}$ is so that it is more suitable for further processing, such as compression, or playback in an audio system. One example is the DFT: We have already proved that $(F_N, (F_N)^H)$ is a perfect reconstruction system for ant N. One problem with this system is that the Fourier matrix is not sparse. Although efficient algorithms exist for the DFT, one may find systems which are quicker to compute in the transform and reconstruction steps. We are therefore in practice interested in establishing perfect reconstruction systems, where the involved matrices have particular forms. Digital filters is one such form, since these are quick to compute when there are few nonzero filter coefficients. Unfortunately, related to this we have the following corollary to Theorem 3.52:

Corollary 3.54. let G_1 and G_2 be circulant Toeplitz matrices so that (G_1, G_2) is a perfect reconstruction system. Then there exist a scalar α and an integer d so that $G_1 = \alpha E_d$ and $G_2 = \alpha^{-1}E_{-d}$, i.e. both matrices have only one nonzero diagonal, with the values being inverse of oneanother, and the diagonals being symmetric about the main diagonal.

In short, this states that there do not exist perfect reconstruction systems involving nontrivial digital filters. This sounds very bad, since filters, as we will see, represent some of the nicest operations which can be implemented. Note that, however, it may still be possible to construct such (G_1, G_2) so that G_1G_2 is "close to" I. Such systems can be called "recontruction systems", and may be very important in settings where some loss in the transformation process is acceptable. We will not consider such systems here.

In a search for other perfect reconstruction systems than those given by the DFT (and DCT in the next section), we thus have to look for other matrices than those given by digital filters. In Section 6.1 we will see that it is possible to find such systems for the matrices we define in the next chapter, which are a natural generalization of digital filters.

Exercises for Section 3.3

Ex. 1 — Compute and plot the frequency response of the filter $S = \{1/4, \underline{1/2}, 1/4\}$. Where does the frequency response achieve its maximum and minimum value, and what are these values?

Ex. 2 — Plot the frequency response of the filter $T = \{1/4, \underline{-1/2}, 1/4\}$. Where does the frequency response achieve its maximum and minimum value, and what are these values? Can you write down a connection between this frequency response and that from Exercise 1?

Ex. 3 — Consider the two filters $S_1 = \{\underline{1}, 0, \ldots, 0, c\}$ and $S_2 = \{\underline{1}, 0, \ldots, 0, -c\}$. Both of these can be interpreted as filters which add an echo. Show that $\frac{1}{2}(S_1 + S_2) = I$. What is the interpretation of this relation in terms of echos?

Ex. 4 — In Example 1.19 we looked at time reversal as an operation on digital sound. In $\mathbb{R}^N$ this can be defined as the linear mapping which sends the vector $\boldsymbol{e}_k$ to $\boldsymbol{e}_{N-1-k}$ for all $0 \leq k \leq N-1$.

a. Write down the matrix for the time reversal linear mapping, and explain from this why time reversal is not a digital filter.
b. Prove directly that time reversal is not a time-invariant operation.

Ex. 5 — Consider the linear mapping S which keeps every second component in $\mathbb{R}^N$, i.e. $S(\boldsymbol{e}_{2k}) = \boldsymbol{e}_{2k}$, and $S(\boldsymbol{e}_{2k-1}) = \boldsymbol{0}$. Is S a digital filter?

Ex. 6 — A filter S_1 has the frequency response $\frac{1}{2}(1+\cos\omega)$, and another filter has the frequency response $\frac{1}{2}(1+\cos(2\omega))$.

a. Is S_1S_2 a lowpass filter, or a highpass filter?
b. What does the filter S_1S_2 do with angular frequencies close to $\omega = \pi/2$.
c. Find the filter coefficients of S_1S_2.
   Hint: Use Theorem 3.34 to compute the frequency response of S_1S_2 first.
d. Write down the matrix of the filter S_1S_2 for $N = 8$.

Ex. 7 — Let E_{d_1} and E_{d_2} be two time delay filters. Show that $E_{d_1}E_{d_2} = E_{d_1+d_2}$ (i.e. that the composition of two time delays again is a time delay) in two different ways

a. Give a direct argument which uses no computations.
b. By using Property 3 in Theorem 3.11, i.e. by using a property for the Discrete Fourier Transform.

Ex. 8 — Let S be a digital filter. Show that S is symmetric if and only if the frequency response satisfies $s_k = s_{N-k}$ for all k.

Ex. 9 — Consider again Example 3.43. Find an expression for a filter so that only frequencies so that $|\omega - \pi| < \omega_c$ are kept, i.e. the filter should only keep angular frequencies close to π (i.e. here we construct a highpass filter).

Ex. 10 — Assume that S is a circulant Toeplitz matrix so that only

$$S_{0,0}, \ldots, S_{0,F} \text{ and } S_{0,N-E}, \ldots, S_{0,N-1}$$

are nonzero on the first row, where E, F are given numbers. When implementing this filter on a computer we need to make sure that the vector indices lie in $[0, N-1]$. Show that $y_n = (S\boldsymbol{x})_n$ can be split into the following different formulas, depending on n, to achieve this:

a. $0 \leq n < E$:

$$y_n = \sum_{k=0}^{n-1} S_{0,N+k-n}x_k + \sum_{k=n}^{F+n} S_{0,k-n}x_k + \sum_{k=N-1-E+n}^{N-1} S_{0,k-n}x_k. \quad (3.24)$$

b. $E \leq n < N-F$:

$$y_n = \sum_{k=n-E}^{n+F} S_{0,k-n}x_k. \quad (3.25)$$

c. $N-F \leq n < N$:

$$y_n = \sum_{k=0}^{n-(N-F)} S_{0,k-n}x_k + \sum_{k=n-E}^{n-1} S_{0,N+k-n}x_k + \sum_{k=n}^{N-1} S_{0,k-n}x_k. \quad (3.26)$$

These three cases give us the full implementation of the filter. This implementation is often more useful than writing down the entire matrix S, since we save computation when many of the matrix entries are zero.

Ex. 11 — In this exercise we will find out how to keep to track of the length and the start and end indices when we convolve two sequences

a. Let $\boldsymbol{g}$ and $\boldsymbol{h}$ be two sequences with finitely many nonzero elements. Show that $\boldsymbol{g}*\boldsymbol{h}$ also has finitely many nonzero elements, and show that $l(\boldsymbol{g}*\boldsymbol{h}) = l(\boldsymbol{g}) + l(\boldsymbol{h})$.

b. Find expressions for the values k_{min}, k_{max} for the filter $\boldsymbol{g}*\boldsymbol{h}$, in terms of those for the filters $\boldsymbol{g}$ and $\boldsymbol{h}$.

Ex. 12 — Write a function

```
function [gconvh gconvhmin]=filterimpl(g,gmin,h,hmin)
```

which performs the convolution of two sequences, but also keeps track of the index of the smallest nonzero coefficient in the sequences.

Ex. 13 — Consider the matrix

$$S = \begin{pmatrix} 4 & 1 & 3 & 1 \\ 1 & 4 & 1 & 3 \\ 3 & 1 & 4 & 1 \\ 1 & 4 & 1 & 3 \end{pmatrix}.$$

a. Compute the eigenvalues and eigenvectors of S using the results of this section. You should only need to perform one DFT in order to achieve this.
b. Verify the result from a. by computing the eigenvectors and eigenvalues the way you taught in your first course in linear algebra. This should be a much more tedious task.
c. Use Matlab to compute the eigenvectors and eigenvalues of S also. For some reason some of the eigenvectors seem to be different from the Fourier basis vectors, which you would expect from the theory in this section. Try to find an explanation for this.

Ex. 14 — Define the filter S by $S = \{1, 2, \underline{3}, 4, 5, 6\}$. Write down the matrix for S when $N = 8$. Plot (the magnitude of))$\lambda_S(\omega)$, and indicate the values $\lambda_{S,n}$ for $N = 8$ in this plot.

Ex. 15 — Assume that the filter S is defined by the formula

$$z_n = \frac{1}{4}x_{n+1} + \frac{1}{4}x_n + \frac{1}{4}x_{n-1} + \frac{1}{4}x_{n-2}.$$

Write down the matrix for S when $N = 8$. Compute and plot (the magnitude of) $\lambda_S(\omega)$.

Ex. 16 — Given the circulant Toeplitz matrix

$$S = \frac{1}{5}\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 1 & 1 & \cdots & 0 \\ 0 & 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 1 & 0 & 0 & \cdots & 1 \\ 1 & 1 & 0 & \cdots & 1 \\ 1 & 1 & 1 & \cdots & 1 \end{pmatrix}$$

Write down the filter coefficients of this matrix, and use the compact notation $\{t_{k_{min}}, \ldots, t_{-1}, \underline{t_0}, t_1, \ldots, t_{k_{max}}\}$. Compute and plot (the magnitude) of $\lambda_S(\omega)$.

Ex. 17 — Assume that $S = \{\underline{1}, c, c^2, \ldots, c^k\}$. Compute and plot $\lambda_S(\omega)$ when $k = 4$ and $k = 8$. How does the choice of k influence the frequency response? How does the choice of c influence the frequency response?

Ex. 18 — Assume that S_1 and S_2 are two circulant Toeplitz matrices.

a. How can you express the eigenvalues of $S_1 + S_2$ in terms of the eigenvalues of S_1 and S_2?
b. How can you express the eigenvalues of $S_1 S_2$ in terms of the eigenvalues of S_1 and S_2?
c. If A and B are general matrices, can you find a formula which expresses the eigenvalues of $A + B$ and AB in terms of those of A and B? If not, can you find a counterexample to what you found in a. and b.?

Ex. 19 — In this exercise we will investigate how we can combine lowpass and highpass filters to produce other filters

a. Assume that S_1 and $S2$ are lowpass filters. What kind of filter is $S_1 S_2$? What if both S_1 and S_2 are highpass filters?
b. Assume that one of S_1, S_2 is a highpass filter, ans that the other is a lowpass filter. What kind of filter $S_1 S_2$ in this case?

3.4 Symmetric digital filters and the DCT

We have mentioned that most periodic functions can be approximated well by Fourier series on the form $\sum_n y_n e^{2\pi i n t/T}$. The convergence speed of this Fourier series may be slow however, and we mentioned that it depends on the regularity of the function. In particular, the convergence speed is higher when the function is continuous. For $f \in C[0, T]$ where $f(0) \neq f(T)$, we do not get a continuous function if we repeat the function in periods. However, we demonstrated that we could create a symmetric extension g, defined on $[0, 2T]$, so that $g(0) = g(2T)$. The Fourier series of g actually took the form of a cosine-series, and we saw that this series converged faster to g, than the Fourier series of f did to f.

In this section we will specialize this argument to vectors: by defining the symmetric extension of a vector, we will attempt to find a better approximation than we could with the Fourier basis vectors. This approach is useful for digital filters also, if the filters preserve these symmetric extensions. Let us summarize with the following idea:

Idea 3.55 (Increasing the convergence speed of the DFT)**.** Assume that we have a function f, and that we take the samples $x_k = f(kT/N)$. From $\boldsymbol{x} \in \mathbb{R}^N$ we would like to create an extension $\breve{\boldsymbol{x}}$ so that the first and last values are equal. For such an extension, the Fourier basis vectors can give a very good approximation to f.

As our candidate for the extension of $\boldsymbol{x}$, we will consider the following:

Definition 3.56 (Symmetric extension of a vector)**.** By the symmetric extension of $\boldsymbol{x} \in \mathbb{R}^N$, we mean $\breve{\boldsymbol{x}} \in \mathbb{R}^{2N}$ defined by

$$\breve{\boldsymbol{x}}_k = \begin{cases} x_k & 0 \le k < N \\ x_{2N-1-k} & N \le k < 2N-1 \end{cases} \tag{3.27}$$

We say that a vector in $\mathbb{R}^{2N}$ is symmetric if it can be written as the symmetric extension of a vector in $\mathbb{R}^N$.

The symmetric extension $\breve{\boldsymbol{x}}$ thus has the original vector $\boldsymbol{x}$ as its first half, and a copy of $\boldsymbol{x}$ in reverse order as its second half. Clearly, the first and last values of $\breve{\boldsymbol{x}}$ are equal. In other words, a vector in $\mathbb{R}^{2N}$ is a symmetric extension of a vector in $\mathbb{R}^N$ if and only if it is symmetric about $N - \frac{1}{2}$. Clearly also the set of symmetric extensions is a vector space. Our idea in terms of filters is the following:

Idea 3.57 (Increasing the precision of a digital filter)**.** If a filter maps a symmetric extension of one vector to a symmetric extension of another vector then it is a good approximation of an analog version in terms of Fourier series.

We will therefore be interested in finding filters which preserves symmetric extensions. We will show the following, which characterize such filters:

Theorem 3.58 (Characterization of filters which preserve symmetric extensions)**.** A digital filter S of size $2N$ preverves symmetric extensions if and only if S is symmetric. Moreover, S is uniquely characterized by its restriction to $\mathbb{R}^N$, denoted by S_r, which is given by $S_1 + (S_2)^f$, where $S = \begin{pmatrix} S_1 & S_2 \\ S_3 & S_4 \end{pmatrix}$, and where $(S_2)^f$ is the matrix S_2 with the columns reversed. Moreover, if we define

$$d_{n,N} = \begin{cases} \sqrt{\frac{1}{N}} & , n = 0 \\ \sqrt{\frac{2}{N}} & , 1 \le n < N \end{cases}$$

and $\boldsymbol{d}_n = d_{n,N} \cos\left(2\pi \frac{n}{2N}\left(k + \frac{1}{2}\right)\right)$ for $0 \le n \le N-1$, then $\{\boldsymbol{d}_0, \boldsymbol{d}_1, \ldots, \boldsymbol{d}_{N-1}\}$ is an orthonormal basis of eigenvectors for S_r.

Proof. Let $\boldsymbol{z} = \hat{\breve{\boldsymbol{x}}}$. Since $\breve{\boldsymbol{x}}$ is symmetric about $N - \frac{1}{2}$, by Theorem 3.11 it follows that the argument of z_n is $-2\pi(N - \frac{1}{2})n/(2N)$. Since z_{2N-n} is the conjugate of z_n by the same theorem, it follows that $z_{2N-n} = e^{4\pi i(N-\frac{1}{2})n/(2N)}z_n = e^{-2\pi in/(2N)}z_n$. It follows that a vector in $\mathbb{R}^{2N}$ is a symmetric extension if and only if its DFT is in the span of the vectors

$$\{\boldsymbol{e}_0, \{\boldsymbol{e}_n + e^{-2\pi in/(2N)}\boldsymbol{e}_{2N-n}\}_{n=1}^{N-1}\}.$$

These vectors are clearly orthogonal. Their span can also be written as the span of the vectors

$$\left\{\boldsymbol{e}_0, \left\{\frac{1}{\sqrt{2}}\left(e^{\pi in/(2N)}\boldsymbol{e}_n + e^{-\pi in/(2N)}\boldsymbol{e}_{2N-n}\right)\right\}_{n=1}^{N-1}\right\}, \tag{3.28}$$

where the last vectors have been first multiplied with $e^{\pi in/(2N)}$, and then normalized so that they have norm 1. Equation (3.28) now gives us an orthononormal basis for the DFT's of all symmetric extensions. Let us map these vectors back with the IDFT. We get first that

$$(F_{2N})^H(\boldsymbol{e}_0) = \left(\frac{1}{\sqrt{2N}}, \frac{1}{\sqrt{2N}}, \ldots, \frac{1}{\sqrt{2N}}\right) = \frac{1}{\sqrt{2N}}\cos\left(2\pi\frac{0}{2N}\left(k + \frac{1}{2}\right)\right).$$

We also get that

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

Since F_{2N} is unitary, and thus preserves the scalar product, this means that

$$\left\{\frac{1}{\sqrt{2N}}\cos\left(2\pi\frac{0}{2N}\left(k + \frac{1}{2}\right)\right), \left\{\frac{1}{\sqrt{N}}\cos\left(2\pi\frac{n}{2N}\left(k + \frac{1}{2}\right)\right)\right\}_{n=1}^{N-1}\right\} \tag{3.29}$$

is an orthonormal basis for the set of symmetric extensions in $\mathbb{R}^{2N}$. We have that

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

where we have used that $e^{2\pi ink/(2N)}$ is an eigenvector of S with eigenvalue $\lambda_{S,n}$, and $e^{-2\pi ink/(2N)}=e^{2\pi i(2N-n)k/(2N)}$ is an eigenvector of S with eigenvalue $\lambda_{S,2N-n}$. If S preserves symmetric extensions, we see that we must have that $\lambda_{S,n}=\lambda_{S,2N-n}$, and also that the vectors listed in Equation (3.29) must be eigenvectors for S. This is reflected in that the entries in D in the diagonalization $S=(F_{2N})^H DF_{2N}$ are symmetric about the midpoint on the diagonal. From Exercise 3.3.8 we know that this occurs if and only if S is symmetric, which proves the first part of the theorem.

Since S preserves symmetric extensions it is clearly characterized by its restriction to the first N elements. With $S=\begin{pmatrix} S_1 & S_2\\ S_3 & S_4\end{pmatrix}$, we compute this restriction as

$$\begin{pmatrix} y_0\\ \vdots \\ y_{N-1}\end{pmatrix}=\begin{pmatrix} S_1 & S_2\end{pmatrix}\begin{pmatrix} x_0\\ \vdots\\ x_{N-1}\\ \hline x_N\\ \vdots\\ x_{2N-1}\end{pmatrix}=\begin{pmatrix} S_1+(S_2)^f\end{pmatrix}\begin{pmatrix} x_0\\ \vdots\\ x_{N-1}\end{pmatrix}.$$

S_2 contains the circulant part of the matrix, and forming $(S_2)^f$ means that the circulant parts switch corners. This shows that S is uniquely characterized by the matrix S_r as defined in the text of the theorem. Finally, since (3.29) are eigenvectors of S, the vectors in $\mathbb{R}^N$ restricted to their first N elements are eigenvectors for S_r. Since the scalar product of two symmetric extensions is the double of the scalar product of the first half of the vectors, we have that these vectors must also be orthogonal, and that

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

is an orthonormal basis of eigenvectors for S_r. We see that we now can define $d_{n,N}$ and the vectors $\boldsymbol{d}_n$ as in the text of the theorem, and this completes the proof. $\square$

From the proof we clearly see the analogy between symmetric functions and vectors: while the first can be written as a cosine-series, the second can be written as a sum of cosine-vectors:

Corollary 3.59.

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

form an orthonormal basis for the set of all symmetric vectors in $\mathbb{R}^{2N}$.

Note also that S_r is not a circulant matrix. Therefore, its eigenvectors are not pure tones. An example should clarify this:

Example 3.60. Consider the averaging filter $\boldsymbol{g} = \{\frac{1}{4}, \underline{\frac{1}{2}}, \frac{1}{4}\}$. Let us write down the matrix S_r for the case when $N = 4$. 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

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

The orthogonal basis we have found is given its own name:

Definition 3.61 (DCT basis)**.** We denote by $\mathcal{D}_N$ the orthogonal basis $\{\boldsymbol{d}_0, \boldsymbol{d}_1, \ldots, \boldsymbol{d}_{N-1}\}$. We also call $\mathcal{D}_N$ the N-point *DCT* basis.

Using the DCT basis instead of the Fourier basis we can make the following definitions, which parallel those for the DFT:

Definition 3.62 (Discrete Cosine Transform)**.** The change of coordinates from the standard basis of $\mathbb{R}^N$ to the DCT basis $\mathcal{D}_N$ is called the *discrete cosine transform* (or DCT). The $N \times N$ matrix D_N that represents this change of basis is called the (N-point) DCT matrix. If $\boldsymbol{x}$ is a vector in R^N, its coordinates $\boldsymbol{y} = (y_0, y_1, \ldots, y_{N-1})$ relative to the DCT basis are called the DCT coefficients of $\boldsymbol{x}$ (in other words, $\boldsymbol{y} = D_N \boldsymbol{x}$).

As with the Fourier basis vectors, the DCT basis vectors are called synthesis vectors, since we can write

$$\boldsymbol{x} = y_0 \boldsymbol{d}_0 + y_1 \boldsymbol{d}_1 + \cdots + y_{N-1} \boldsymbol{d}_{N-1} \tag{3.30}$$

in the same way as for the DFT. Following the same reasoning as for the DFT, D_N^{-1} is the matrix where the $\boldsymbol{d}_n$ are columns. But since these vectors are real and orthonormal, D_N must be the matrix where the $\boldsymbol{d}_n$ are rows. Moreover, since Theorem 3.58 also states that the same vectors are eigenvectors for filters which preserve symmetric extensions, we can state the following:

Theorem 3.63. D_N is the orthogonal matrix where the rows are $\boldsymbol{d}_n$. Moreover, for any digital filter S which preserves symmetric extensions, $(D_N)^T$ diagonalizes S_r, i.e. $S_r = D_N^T D D_N$ where D is a diagonal matrix.

Let us also make the following definition:

Definition 3.64 (IDCT)**.** We will call $\boldsymbol{x} = (D_N)^T \boldsymbol{y}$ the inverse DCT or (IDCT) of $\boldsymbol{x}$.

Example 3.65. As with Example 3.9, exact expressions for the DCT can be written down just for a few specific cases. It turns out that the case $N = 4$ as considered in Example 3.9 does not give the same type of nice, exact values, so let us instead consider the case $N = 2$. We have that

$$D_4 = \begin{pmatrix} \frac{1}{\sqrt{2}} \cos(0) & \frac{1}{\sqrt{2}} \cos(0) \\ \cos\left(\frac{\pi}{2}\left(0 + \frac{1}{2}\right)\right) & \cos\left(\frac{\pi}{2}\left(1 + \frac{1}{2}\right)\right) \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

The DCT of the same vector as in Example 3.9 can now be computed as:

$$D_2 \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} \frac{3}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix}.$$

Example 3.66. A direct implementation of the DCT could be made as follows:

```
function y=DCTImpl(x)
  N=length(x);
  DN=zeros(N);
  DN(1,:)=ones(1,N)/sqrt(N);
```

```
for n=1:N
  DN(n,:)=cos(2*pi*((n-1)/(2*N))*((0:N-1)+1/2))*sqrt(2/N);
end
y=DN*x;
```

In the next chapter we will see that one also can make a much more efficient implementation of the DCT than this.

With the DCT one constructs a vector twice as long. One might think due to this that one actually use matrices twice the size. This is, however, avoided since D_N has the same dimensions as F_N, and we will see shortly that the same algorithms as for the DFT also can be used for the DCT. By construction we also see that it is easy to express the N-point DCT in terms of the $2N$-point DFT. Let us write down this connection:

1. Write $e_0 = \sqrt{2}$ and $e_n = e^{\pi i n/(2N)}$ for $n \neq 0$, and define E as the diagonal matrix with the values $e_0, e_1, \ldots$ on the diagonal.
2. Let B be the $2N \times N$-matrix which is nonzero only when $i = j$ or $i + j = 2N - 1$, and 1 in all these places.
3. Let also A be the $N \times 2N$-matrix with 1 on the diagonal.

We can now write

$$D_N = E^{-1} A F_{2N} B. \tag{3.31}$$

A here extracts the first rows of the matrix, E^{-1} eliminates the complex coefficients, while B adds columns symmetrically. This factorization enables us to use the efficient FFT-implementation, since the matrices A, B, E all are sparse. We will, however, find an even more efficient implementation of the DCT, which will avoid computing a DFT of twice the size as here.

Similarly to Theorem 3.16 for the DFT, one can think of the DCT as a least squares approximation and the unique representation of a function having the same sample values, but this time in terms of sinusoids instead of complex exponentials:

Theorem 3.67 (Interpolation with the DCT basis)**.** Let f be a function defined on the interval $[0, T]$, and let $\boldsymbol{x}$ be the sampled vector given by

$$x_k = f((2k+1)T/(2N)) \quad \text{for } k = 0, 1, \ldots, N-1.$$

There is exactly one linear combination $g(t)$ on the form

$$\sum_{n=0}^{N-1} y_n d_{n,N} \cos(2\pi(n/2)t/T)$$

which satisfies the conditions

$$g((2k+1)T/(2N)) = f((2k+1)T/(2N)), \quad k = 0, 1, \ldots, N-1,$$

and its coefficients are determined by $\boldsymbol{y} = D_N \boldsymbol{x}$.

The proof for this follows by inserting $t = (2k+1)T/(2N)$ in the equation $g(t) = \sum_{n=0}^{N-1} y_n d_{n,N} \cos(2\pi(n/2)t/T)$ to arrive at the equations

$$f(kT/N) = \sum_{n=0}^{N-1} y_n d_{n,N} \cos\left(2\pi \frac{n}{2N}\left(k+\frac{1}{2}\right)\right) \qquad 0 \le k \le N-1.$$

This gives us an equation system for finding the y_n with the invertible DCT matrix as coefficient matrix, and the result follows.

Note the subtle change in the sample points of these cosine functions, from kT/N for the DFT, to $(2k+1)T/(2N)$ for the DCT. The sample points for the DCT are thus the midpoints on the intervals in a uniform partition of $[0,T]$ into N intervals, while they for the DFT are the start points on the intervals. Also, the frequencies are divided by 2. In Figure 3.9 we have plotted the sinusoids of Theorem 3.67 for $T = 1$, as well as the sample points used in that theorem. The sample points in (a) correspond to the first column in the DCT matrix, the sample points in (b) to the second column of the DCT matrix, and so on (up to normalization with $d_{n,N}$). As n increases, the functions oscillate more and more. As an example, y_5 says how much content of maximum oscillation there is. In other words, the DCT of an audio signal shows the proportion of the different frequencies in the signal, and the two formulas $\boldsymbol{y} = D_N\boldsymbol{x}$ and $\boldsymbol{x} = (D_N)^T\boldsymbol{y}$ allow us to switch back and forth between the time domain representation and the frequency domain representation of the sound. In other words, once we have computed $\boldsymbol{y} = D_N\boldsymbol{x}$, we can analyse the frequency content of $\boldsymbol{x}$. If we want to reduce the bass we can decrease the $\boldsymbol{y}$-values with small indices and if we want to increase the treble we can increase the $\boldsymbol{y}$-values with large indices.

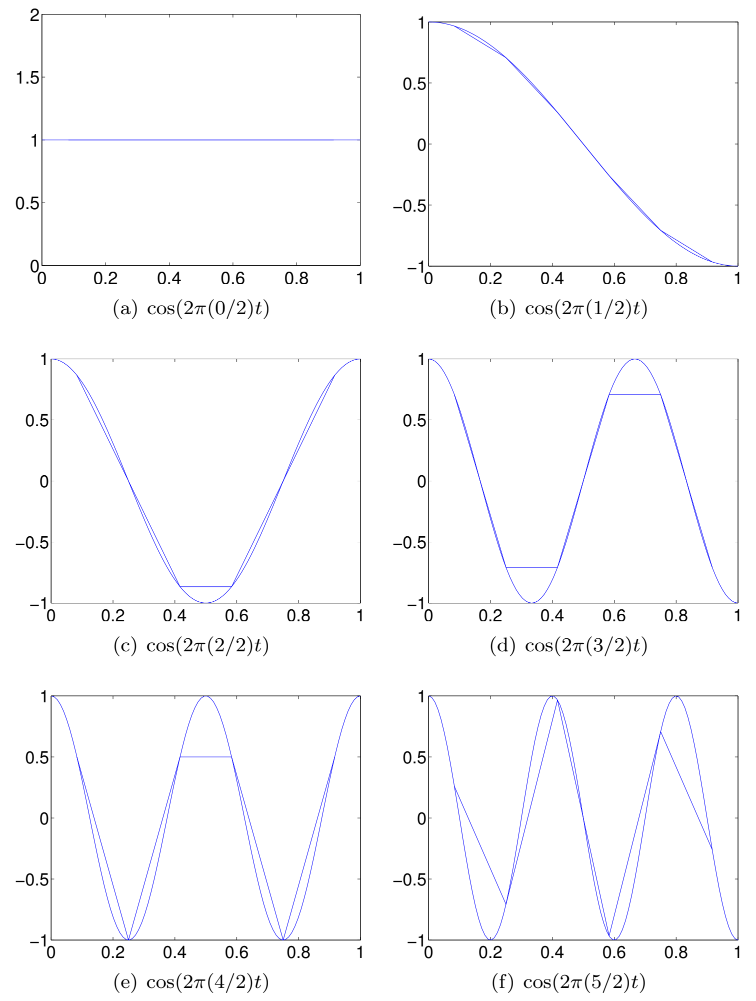
3.4.1 Other types of symmetric extensions

Note that our definition of symmetric extension duplicates the values x_0 and x_{N-1}: both are repeated when creating the symmetric extension. This is in fact unnecessary when we are creating a longer vector which has equal first and last values, and is primarily motivated from existing efficient implementations for the DCT when all vector lengths are powers of 2. When an efficient DCT implementation is not important, we can change the definition of the symmetric extension as follows (it is this type of symmetric extension we will use later):

Definition 3.68 (Symmetric extension of a vector)**.** By the symmetric extension of $\boldsymbol{x} \in \mathbb{R}^N$, we mean $\breve{\boldsymbol{x}} \in \mathbb{R}^{2N-2}$ defined by

$$\breve{\boldsymbol{x}}_k = \begin{cases} x_k & 0 \le k < N \\ x_{2N-2-k} & N \le k < 2N-3 \end{cases} \qquad (3.32)$$

In other words, a vector in $\mathbb{R}^{2N}$ is a symmetric extension of a vector in $\mathbb{R}^N$ if and only if it is symmetric about $N-1$. Theorem 3.58 now instead takes the following form:



(a) $\cos(2\pi(0/2)t)$ (b) $\cos(2\pi(1/2)t)$

(c) $\cos(2\pi(2/2)t)$ (d) $\cos(2\pi(3/2)t)$

(e) $\cos(2\pi(4/2)t)$ (f) $\cos(2\pi(5/2)t)$

Figure 3.9: The 6 different sinusoids used in DCT for $N = 6$, i.e. $\cos(2\pi(n/2)t)$, $0 \leq n < 6$. The plots also show piecewise linear functions between the the sample points $\frac{2k+1}{2N}$ $0 \leq k < 6$, since only the values at these points are used in Theorem 3.67.

Theorem 3.69 (Characterization of filters which preserve symmetric extensions)**.** A real, circulant $(2N-2)\times(2N-2)$-Toeplitz matrix preverves symmetric extensions if and only if it is symmetric. For such S, S is uniquely characterized by its restriction to $\mathbb{R}^N$, denoted by S_r, which is given by $T_1 + \begin{pmatrix} 0 & (T_2)^f & 0 \end{pmatrix}$, where $T = \begin{pmatrix} T_1 & T_2 \\ T_3 & T_4 \end{pmatrix}$, where T_1 is $N\times N$, T_2 is $N\times(N-2)$. Moreover, an orthogonal basis of eigenvectors for S_r are $\{\cos\left(2\pi\frac{n}{2N}n\right)\}_{n=0}^{N-1}$.

Proof. Let $\boldsymbol{z} = \hat{\breve{\boldsymbol{x}}}$. Since $\breve{\boldsymbol{x}}$ is symmetric about 0, by Theorem 3.11 it follows that $\boldsymbol{z}_n = \boldsymbol{z}_{2(N-1)-n)}$, so that the DFT of a symmetrix extension (as now defined) is in the span of the vectors

$$\{\boldsymbol{e}_0, \{\boldsymbol{e}_n + \boldsymbol{e}_{2(N-1)-n}\}_{n=1}^{N-1}\}.$$

It follows as before that

$$\cos\left(2\pi\frac{n}{2(N-1)}n\right)$$

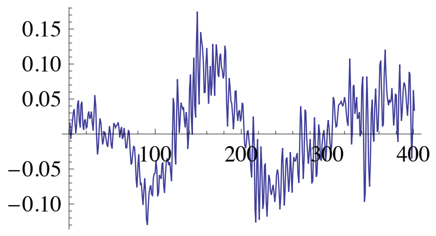
is a basis of eigenvectors. The same type of symmetry about the midpoint on the diagonal follows as before, which as before is equivalent to symmetry of the matrix. □

It is not customary to write down an orthonormal basis for the eigenvectors in this case, since we don't have the same type of efficient DCT implementation due to the lact of powers of 2.

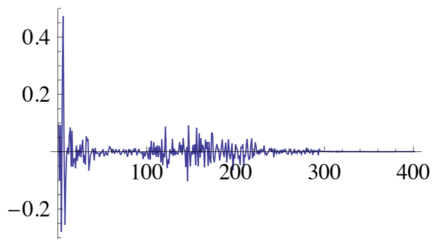
3.4.2 Use of DCT in lossy compression of sound

The DCT is particularly popular for processing the sound before compression. MP3 is based on applying a variant of the DCT (called the Modified Discrete Cosine Transform, MDCT) to groups of 576 (in special circumstances 192) samples. One does not actually apply the DCT directly. Rather one applies a much more complex transformation, which can be implemented in parts by using DCT in a clever way.

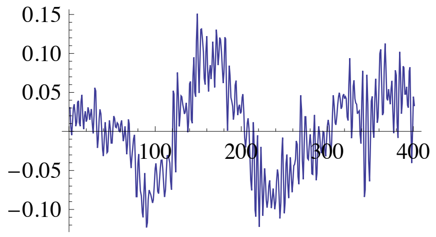
We mentioned previously that we could achieve compression by setting the Fourier coefficients which are small to zero. Translated to the DCT, we should set the DCT coefficients which are small to zero, and we apply the inverse DCT in order to reconstruct the signal in order to play it again. Let us test compression based on this idea. The plots in figure 3.10 illustrate the principle. A signal is shown in (a) and its DCT in (b). In (d) all values of the DCT with absolute value smaller than 0.02 have been set to zero. The signal can then be reconstructed with the inverse DCT of theorem **??**; the result of this is shown in (c). The two signals in (a) and (c) visually look almost the same even though the signal in (c) can be represented with less than 25 % of the information present in (a).



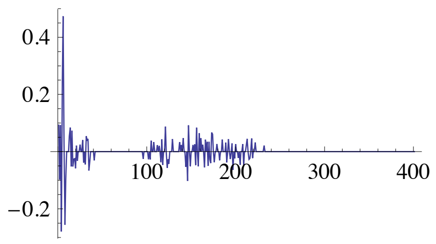
(a)



(b)



(c)



(d)

Figure 3.10: The signal in (a) are the sound samples from a small part of a song. The plot in (b) shows the DCT of the signal. In (d), all values of the DCT that are smaller than 0.02 in absolute value have been set to 0, a total of 309 values. In (c) the signal has been reconstructed from these perturbed values of the DCT. The values have been connected by straight lines to make it easier to interpret the plots.

We test this compression strategy on a data set that consists of 300 001 points. We compute the DCT and set all values smaller than a suitable tolerance to 0. With a tolerance of 0.04, a total of 142 541 values are set to zero. When we then reconstruct the sound with the inverse DCT, we obtain a signal that differs at most 0.019 from the original signal. To verify that the new file is not too different from the old file, we can take the read sound samples from `castanets.wav`, run the following function for different `eps`

```
function A=skipsmallvals(eps,A)
  B=dct(A);
  B=(B>=eps).*B;
  A=invdct(B);
```

and play the new samples. Finally we can store the signal by storing a `gzip`'ed version of the DCT-values (as 32-bit floating-point numbers) of the perturbed signal. This gives a file with 622 551 bytes, which is 88 % of the `gzip`'ed version of the original data.

The choice of the DCT in the MP3 standard has much to do with that the DCT, just as the DFT, has a very efficient implementation, as we will see next.

Exercises for Section 3.4

Ex. 1 — In Section 3.4.2 we implemented the function `skipsmallvals`, which ran a DCT on the entire vector. Explain why there are less computation involved in splitting the vector into many parts and performing a DCT for each part. Change the code accordingly.

Ex. 2 — As in Example 3.65, state the exact cartesian form of the DCT matrix for the case $N = 3$.

Ex. 3 — Assume that S is a symmetric digital filter with support $[-E, E]$. Let us, as in Exercise 3.3.10, see how we can make sure that the indices keep inside $[0, N-1]$. Show that $z_n = (T\boldsymbol{x})_n$ in this case can be split into the following different formulas, depending on n:

a. $0 \leq n < E$:

$$z_n = T_{0,0}x_n + \sum_{k=1}^{n} T_{0,k}(x_{n+k}+x_{n-k}) + \sum_{k=n+1}^{E} T_{0,k}(x_{n+k}+x_{n-k+N}). \quad (3.33)$$

b. $E \leq n < N-E$:

$$z_n = T_{0,0}x_n + \sum_{k=1}^{E} T_{0,k}(x_{n+k} + x_{n-k}). \quad (3.34)$$

c. $N - F \leq n < N$:

$$z_n = T_{0,0}x_n + \sum_{k=1}^{N-1-n} T_{0,k}(x_{n+k} + x_{n-k}) + \sum_{k=N-1-n+1}^{E} T_{0,k}(x_{n+k-N} + x_{n-k}). \tag{3.35}$$

Ex. 4 — Assume that $\{T_{0,-E}, \ldots, T_{0,0}, \ldots, T_{0,E}\}$ are the coefficicients of a symmetric, digital filter S, and let $\boldsymbol{t} = \{T_{0,1}, \ldots, T_{0,E}\}$. Write a function

```
function z=filterT(t,x)
```

which takes the vector $\boldsymbol{t}$ as input, and returns $\boldsymbol{z} = T\boldsymbol{x}$ using the formulas deduced in Exercise 3.

Ex. 5 — Repeat Exercise 1.4.9 by reimplementing the functions `reducetreble` and `reducesbass` using the function `filterT` from the previous exercise. The resulting sound files should sound the same, since the only difference is that we have modified the way we handle the beginning and end portion of the sound samples.

Ex. 6 — Using Python, define a class `Transform` with methods `transformImpl` and `ItransformImpl`. Define two subclasses of `Transform`, `DCTTransform`, `FFTTransform`), which implements these two functions by calling Python counterparts of `FFTImpl`, `IFFTImpl`, `DCTImpl`, and `IDCTImpl`.

3.5 Summary

We defined the Discrete Fourier transform, which could be thought of as the Fourier series of a vector. We exploited properties of the DFT, which corresponded nicely to the corresponding properties for Fourier series. We defined digital filters, which turned out to be linear transformations diagonalized by the DFT. Also we showed that the techniques from the last section we used to speed up the convergence of the Fourier series, could also be used for the DFT. In this way we arrived at the definition of the DCT.