Chapter 7
Fourier Transformation

Fourier transformation is a very important tool for signal analysis but also helpful to simplify the solution of differential equations or the calculation of convolution integrals. In this chapter we discuss the discrete Fourier transformation as a numerical approximation to the continuous Fourier integral. It can be realized efficiently by Goertzel's algorithm or the family of fast Fourier transformation methods. Computer experiments demonstrate trigonometric interpolation and nonlinear filtering as applications.

7.1 Fourier Integral and Fourier Series

We use the symmetric definition of the Fourier transformation:

$$\widetilde{f}(\omega) = \mathcal{F}[f](\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) \mathrm{e}^{-\mathrm{i}\omega t} dt. \tag{7.1}$$

The inverse Fourier transformation

$$f(t) = \mathcal{F}^{-1}[\widetilde{f}](t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \widetilde{f}(\omega) \mathrm{e}^{\mathrm{i}\omega t} d\omega \tag{7.2}$$

decomposes $f(t)$ into a superposition of oscillations. The Fourier transform of a convolution integral

$$g(t) = f(t) \otimes h(t) = \int_{-\infty}^{\infty} f\big(t'\big) h\big(t - t'\big) dt' \tag{7.3}$$

becomes a product of Fourier transforms:

$$\begin{aligned} \widetilde{g}(\omega) &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} dt' f\big(t'\big) \mathrm{e}^{-\mathrm{i}\omega t'} \int_{-\infty}^{\infty} h\big(t - t'\big) \mathrm{e}^{-\mathrm{i}\omega (t - t')} d\big(t - t'\big) \\ &= \sqrt{2\pi}\, \widetilde{f}(\omega) \widetilde{h}(\omega). \end{aligned} \tag{7.4}$$

P.O.J. Scherer, *Computational Physics*, Graduate Texts in Physics,
DOI 10.1007/978-3-319-00401-3_7,

A periodic function with $f(t+T) = f(t)$[1] is transformed into a Fourier series

$$f(t) = \sum_{n=-\infty}^{\infty} \mathrm{e}^{\mathrm{i}\omega_n t} \hat{f}(\omega_n) \text{ with } \omega_n = n\frac{2\pi}{T}, \quad \hat{f}(\omega_n) = \frac{1}{T}\int_0^T f(t)\mathrm{e}^{-\mathrm{i}\omega_n t}\,dt. \tag{7.5}$$

For a periodic function which in addition is real valued $f(t) = f(t)^*$ and even $f(t) = f(-t)$, the Fourier series becomes a cosine series

$$f(t) = \hat{f}(\omega_0) + 2\sum_{n=1}^{\infty} \hat{f}(\omega_n)\cos\omega_n t \tag{7.6}$$

with real valued coefficients

$$\hat{f}(\omega_n) = \frac{1}{T}\int_0^T f(t)\cos\omega_n t\,dt. \tag{7.7}$$

7.2 Discrete Fourier Transformation

We divide the time interval $0 \le t < T$ by introducing a grid of N equidistant points

$$t_n = n\Delta t = n\frac{T}{N} \quad \text{with } n = 0, 1\cdots N-1. \tag{7.8}$$

The function values (samples)

$$f_n = f(t_n) \tag{7.9}$$

are arranged as components of a vector

$$\mathbf{f} = \begin{pmatrix} f_0 \\ \vdots \\ f_{N-1} \end{pmatrix}.$$

With respect to the orthonormal basis

$$\mathbf{e}_n = \begin{pmatrix} \delta_{0,n} \\ \vdots \\ \delta_{N-1,n} \end{pmatrix}, \quad n = 0, 1\cdots N-1 \tag{7.10}$$

$\mathbf{f}$ is expressed as a linear combination

$$\mathbf{f} = \sum_{n=0}^{N-1} f_n \mathbf{e}_n. \tag{7.11}$$

[1] This could also be the periodic continuation of a function which is only defined for $0 < t < T$.

The discrete Fourier transformation is the transformation to an orthogonal base in frequency space

$$\mathbf{e}_{\omega_j} = \sum_{n=0}^{N-1} \mathrm{e}^{\mathrm{i}\omega_j t_n} \mathbf{e}_n = \begin{pmatrix} 1 \\ \mathrm{e}^{\mathrm{i}\frac{2\pi}{N} j} \\ \vdots \\ \mathrm{e}^{\mathrm{i}\frac{2\pi}{N} j(N-1)} \end{pmatrix} \tag{7.12}$$

with

$$\omega_j = \frac{2\pi}{T} j. \tag{7.13}$$

These vectors are orthogonal

$$\mathbf{e}_{\omega_j} \mathbf{e}^*_{\omega_{j'}} = \sum_{n=0}^{N-1} \mathrm{e}^{\mathrm{i}(j-j')\frac{2\pi}{N} n} = \begin{cases} \frac{1-\mathrm{e}^{\mathrm{i}(j-j')2\pi}}{1-\mathrm{e}^{\mathrm{i}(j-j')2\pi/N}} = 0 & \text{for } j - j' \neq 0 \\ N & \text{for } j - j' = 0 \end{cases} \tag{7.14}$$

$$\mathbf{e}_{\omega_j} \mathbf{e}^*_{\omega_{j'}} = N \delta_{j,j'}. \tag{7.15}$$

Alternatively a real valued basis can be defined:

$$\begin{aligned} &\cos\left(\frac{2\pi}{N} jn\right) && j = 0, 1 \cdots j_{\max} \\ &\sin\left(\frac{2\pi}{N} jn\right) && j = 1, 2 \cdots j_{\max} \\ &j_{\max} = \frac{N}{2} \text{ (even } N) && j_{\max} = \frac{N-1}{2} \text{ (odd } N). \end{aligned} \tag{7.16}$$

The components of $\mathbf{f}$ in frequency space are given by the scalar product

$$\widetilde{f}_{\omega_j} = \mathbf{f}\mathbf{e}_{\omega_j} = \sum_{n=0}^{N-1} f_n \mathrm{e}^{-\mathrm{i}\omega_j t_n} = \sum_{n=0}^{N-1} f_n \mathrm{e}^{-\mathrm{i}j\frac{2\pi}{T} n \frac{T}{N}} = \sum_{n=0}^{N-1} f_n \mathrm{e}^{-\mathrm{i}\frac{2\pi}{N} jn}. \tag{7.17}$$

From

$$\sum_{j=0}^{N-1} \widetilde{f}_{\omega_j} \mathrm{e}^{\mathrm{i}\omega_j t_n} = \sum_{n'} \sum_{\omega_j} f_{n'} \mathrm{e}^{-\mathrm{i}\omega_j t_{n'}} \mathrm{e}^{\mathrm{i}\omega_j t_n} = N f_n \tag{7.18}$$

we find the inverse transformation

$$f_n = \frac{1}{N} \sum_{j=0}^{N-1} \widetilde{f}_{\omega_j} \mathrm{e}^{\mathrm{i}\omega_j t_n} = \frac{1}{N} \sum_{j=0}^{N-1} \widetilde{f}_{\omega_j} \mathrm{e}^{\mathrm{i}\frac{2\pi}{N} nj}. \tag{7.19}$$

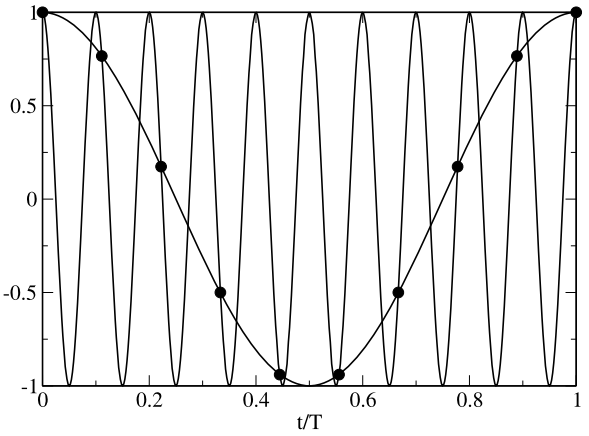

Fig. 7.1 (Equivalence of ω_1 and ω_{N-1}) The two functions $\cos\omega t$ and $\cos(N-1)\omega t$ have the same values at the sample points t_n but are very different in between

7.2.1 Trigonometric Interpolation

The last equation can be interpreted as an interpolation of the function $f(t)$ at the sampling points t_n by a linear combination of trigonometric functions

$$f(t) = \frac{1}{N}\sum_{j=0}^{N-1} \widetilde{f}_{\omega_j} \left(\mathrm{e}^{\mathrm{i}\frac{2\pi}{T}t}\right)^j \tag{7.20}$$

which is a polynomial of

$$q = \mathrm{e}^{\mathrm{i}\frac{2\pi}{T}t}. \tag{7.21}$$

Since

$$\mathrm{e}^{-\mathrm{i}\omega_j t_n} = \mathrm{e}^{-\mathrm{i}\frac{2\pi}{N}jn} = \mathrm{e}^{\mathrm{i}\frac{2\pi}{N}(N-j)n} = \mathrm{e}^{\mathrm{i}\omega_{N-j}t_n} \tag{7.22}$$

the frequencies ω_j and ω_{N-j} are equivalent (Fig. 7.1)

$$\widetilde{f}_{\omega_{N-j}} = \sum_{n=0}^{N-1} f_n \mathrm{e}^{-\mathrm{i}\frac{2\pi}{N}(N-j)n} = \sum_{n=0}^{N-1} f_n \mathrm{e}^{\mathrm{i}\frac{2\pi}{N}jn} = \widetilde{f}_{\omega_{-j}}. \tag{7.23}$$

If we use trigonometric interpolation to approximate $f(t)$ between the grid points, the two frequencies are no longer equivalent and we have to restrict the frequency range to avoid unphysical high frequency components (Fig. 7.2):

$$\begin{aligned} -\frac{2\pi}{T}\frac{N-1}{2} \le \omega_j \le \frac{2\pi}{T}\frac{N-1}{2} \quad & N \text{ odd} \\ -\frac{2\pi}{T}\frac{N}{2} \le \omega_j \le \frac{2\pi}{T}\left(\frac{N}{2}-1\right) \quad & N \text{ even}. \end{aligned} \tag{7.24}$$

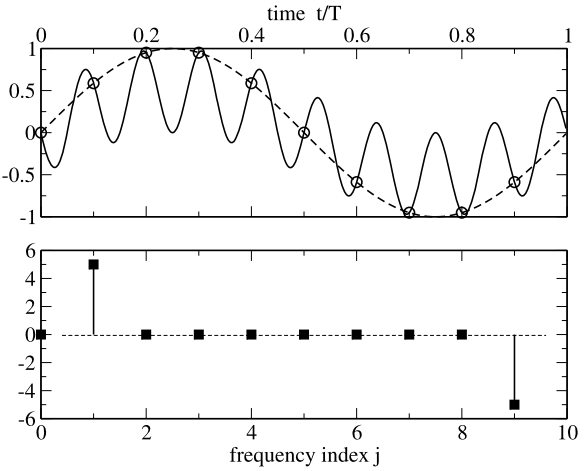

Fig. 7.2 (Trigonometric interpolation) For trigonometric interpolation the high frequencies have to be replaced by the corresponding negative frequencies to provide meaningful results between the sampling points. The *circles* show sampling points which are fitted using only positive frequencies (*full curve*) or replacing the unphysical high frequency by its negative counterpart (*broken curve*). The *squares* show the calculated Fourier spectrum

The interpolating function (N even) is

$$f(t) = \frac{1}{N} \sum_{j=-\frac{N}{2}}^{\frac{N}{2}-1} \widetilde{f}_{\omega_j} \mathrm{e}^{\mathrm{i}\omega_j t} \quad \text{even } N \tag{7.25}$$

$$f(t) = \frac{1}{N} \sum_{j=-\frac{N-1}{2}}^{\frac{N-1}{2}} \widetilde{f}_{\omega_j} \mathrm{e}^{\mathrm{i}\omega_j t} \quad \text{odd } N. \tag{7.26}$$

The maximum frequency is

$$\omega_{\max} = \frac{2\pi}{T} \frac{N}{2} \tag{7.27}$$

and hence

$$f_{\max} = \frac{1}{2\pi} \omega_{\max} = \frac{N}{2T} = \frac{f_s}{2}. \tag{7.28}$$

This is known as the sampling theorem which states that the sampling frequency f_s must be larger than twice the maximum frequency present in the signal.

7.2.2 Real Valued Functions

For a real valued function

$$f_n = f_n^* \tag{7.29}$$

and hence

$$\widetilde{f}^*_{\omega_j} = \left(\sum_{n=0}^{N-1} f_n \mathrm{e}^{-\mathrm{i}\omega_j t_n}\right)^* = \sum_{n=0}^{N-1} f_n \mathrm{e}^{\mathrm{i}\omega_j t_n} = \widetilde{f}_{\omega_{-j}}. \tag{7.30}$$

Here it is sufficient to calculate the sums for $j = 0 \cdots N/2$. If the function is real valued and also even

$$f_{-n} = f_n \tag{7.31}$$

$$\widetilde{f}_{\omega_j} = \sum_{n=0}^{N-1} f_{-n} \mathrm{e}^{-\mathrm{i}\omega_j t_n} = \sum_{n=0}^{N-1} f_n \mathrm{e}^{-\mathrm{i}(-\omega_j) t_n} = \widetilde{f}_{\omega_{-j}} \tag{7.32}$$

and the Fourier sum (7.19) turns into a cosine sum

$$f_n = \frac{1}{2M-1}\tilde{f}_{\omega_0} + \frac{2}{2M-1}\sum_{j=1}^{M-1} \widetilde{f}_{\omega_j} \cos\left(\frac{2\pi}{2M-1} jn\right) \quad \text{odd } N = 2M-1 \tag{7.33}$$

$$f_n = \frac{1}{2M}\tilde{f}_{\omega_0} + \frac{1}{M}\sum_{j=1}^{M-1} \widetilde{f}_{\omega_j} \cos\left(\frac{\pi}{M} jn\right) + \frac{1}{2M}\tilde{f}_{\omega_M} \cos(n\pi) \quad \text{even } N = 2M \tag{7.34}$$

which correspond to two out of eight different versions [268] of the discrete cosine transformation [2, 211].

Equation (7.34) can be used to define the interpolating function

$$f(t) = \frac{1}{2M}\tilde{f}_{\omega_0} + \frac{1}{M}\sum_{j=1}^{M-1} \widetilde{f}_{\omega_j} \cos(\omega_j t) + \frac{1}{2M}\tilde{f}_{\omega_M} \cos\left(\frac{2\pi M}{T} t\right). \tag{7.35}$$

The real valued Fourier coefficients are given by

$$\widetilde{f}_{\omega_j} = f_0 + 2\sum_{n=1}^{M-1} f_n \cos(\omega_j t_n) \quad \text{odd } N = 2M-1 \tag{7.36}$$

$$\widetilde{f}_{\omega_j} = f_0 + 2\sum_{n=1}^{M-1} f_n \cos(\omega_j t_n) + f_M \cos(j\pi) \quad \text{even } N = 2M. \tag{7.37}$$

7.2.3 Approximate Continuous Fourier Transformation

We continue the function $f(t)$ periodically by setting

$$f_N = f_0 \tag{7.38}$$

and write

$$\widetilde{f}_{\omega_j} = \sum_{n=0}^{N-1} f_n \mathrm{e}^{-\mathrm{i}\omega_j n} = \frac{1}{2} f_0 + \mathrm{e}^{-\mathrm{i}\omega_j} f_1 + \cdots + \mathrm{e}^{-\mathrm{i}\omega_j (N-1)} f_{N-1} + \frac{1}{2} f_N. \tag{7.39}$$

Comparing with the trapezoidal rule (4.13) for the integral

$$\int_0^T \mathrm{e}^{-\mathrm{i}\omega_j t} f(t) dt \approx \frac{T}{N} \left(\frac{1}{2} \mathrm{e}^{-\mathrm{i}\omega_j 0} f(0) + \mathrm{e}^{-\mathrm{i}\omega_j \frac{T}{N}} f\left(\frac{T}{N}\right) + \cdots \right.$$
$$\left. + \mathrm{e}^{-\mathrm{i}\omega_j \frac{T}{N}(N-1)} f\left(\frac{T}{N}(N-1)\right) + \frac{1}{2} f(T) \right) \tag{7.40}$$

we find

$$\hat{f}(\omega_j) = \frac{1}{T} \int_0^T \mathrm{e}^{-\mathrm{i}\omega_j t} f(t) dt \approx \frac{1}{N} \widetilde{f}_{\omega_j} \tag{7.41}$$

which shows that the discrete Fourier transformation is an approximation to the Fourier series of a periodic function with period T which coincides with $f(t)$ in the interval $0 < t < T$. The range of the integral can be formally extended to $\pm\infty$ by introducing a windowing function

$$W(t) = \begin{cases} 1 & \text{for } 0 < t < T \\ 0 & \text{else.} \end{cases} \tag{7.42}$$

The discrete Fourier transformation approximates the continuous Fourier transformation but windowing leads to a broadening of the spectrum. For practical purposes smoother windowing functions are used like a triangular window or one of the following [119]:

$$W(t_n) = \mathrm{e}^{-\frac{1}{2}\left(\frac{n-(N-1)/2}{\sigma(N-1)/2}\right)^2} \quad \sigma \leq 0.5 \quad \text{Gaussian window}$$
$$W(t_n) = 0.53836 - 0.46164 \cos\left(\frac{2\pi n}{N-1}\right) \quad \text{Hamming window}$$
$$W(t_n) = 0.5\left(1 - \cos\left(\frac{2\pi n}{N-1}\right)\right) \quad \text{Hann(ing) window.}$$

7.3 Fourier Transform Algorithms

Straightforward evaluation of the sum

$$\widetilde{f}_{\omega_j} = \sum_{n=0}^{N-1} \cos\left(\frac{2\pi}{N} jn\right) f_n + \mathrm{i} \sin\left(\frac{2\pi}{N} jn\right) f_n \tag{7.43}$$

needs $O(N^2)$ additions, multiplications and trigonometric functions.

7.3.1 Goertzel's Algorithm

Goertzel's method [103] is very useful if not the whole Fourier spectrum is needed but only some of the Fourier components, for instance to demodulate a frequency shift key signal or the dial tones which are used in telephony.

The Fourier transform can be written as

$$\sum_{n=0}^{N-1} f_n \mathrm{e}^{-\mathrm{i}\frac{2\pi}{N} jn} = f_0 + \mathrm{e}^{-\frac{2\pi\mathrm{i}}{N} j} \left(f_1 + \mathrm{e}^{-\frac{2\pi\mathrm{i}}{N} j} f_2 \cdots \left(f_{N-2} + \mathrm{e}^{-\frac{2\pi\mathrm{i}}{N} j} f_{N-1}\right) \cdots\right) \tag{7.44}$$

which can be evaluated recursively

$$\begin{aligned} y_{N-1} &= f_{N-1} \\ y_n &= f_n + \mathrm{e}^{-\frac{2\pi\mathrm{i}}{N} j} y_{n+1} \quad n = N-2 \cdots 0 \end{aligned} \tag{7.45}$$

to give the result

$$\hat{f}_{\omega_j} = y_0. \tag{7.46}$$

Equation (7.45) is a simple discrete filter function. Its transmission function is obtained by application of the z-transform [144]

$$u(z) = \sum_{n=0}^{\infty} u_n z^{-n} \tag{7.47}$$

(the discrete version of the Laplace transform) which yields

$$y(z) = \frac{f(z)}{1 - z\mathrm{e}^{-\frac{2\pi\mathrm{i}}{N} j}}. \tag{7.48}$$

One disadvantage of this method is that it uses complex numbers. This can be avoided by the following more complicated recursion

$$\begin{aligned} u_{N+1} &= u_N = 0 \\ u_n &= f_n + 2u_{n+1} \cos\frac{2\pi}{N} k - u_{n+2} \quad \text{for } n = N-1 \cdots 0 \end{aligned} \tag{7.49}$$

with the transmission function

$$\frac{u(z)}{f(z)} = \frac{1}{1 - z(\mathrm{e}^{\frac{2\pi \mathrm{i}}{N} j} + \mathrm{e}^{-\frac{2\pi \mathrm{i}}{N} j}) + z^2} = \frac{1}{(1 - z\mathrm{e}^{-\frac{2\pi \mathrm{i}}{N} j})(1 - z\mathrm{e}^{\frac{2\pi \mathrm{i}}{N} j})}. \tag{7.50}$$

A second filter removes one factor in the denominator

$$\frac{y(z)}{u(z)} = \left(1 - z\mathrm{e}^{\frac{2\pi \mathrm{i}}{N} j}\right) \tag{7.51}$$

which in the time domain corresponds to the simple expression

$$y_n = u_n - \mathrm{e}^{\frac{2\pi \mathrm{i}}{N} j} u_{n+1}.$$

The overall filter function finally again is (7.48).

$$\frac{y(z)}{f(z)} = \frac{1}{1 - z\mathrm{e}^{-\frac{2\pi \mathrm{i}}{N} j}}. \tag{7.52}$$

Hence the Fourier component of $\mathbf{f}$ is given by

$$\hat{f}_{\omega_j} = y_0 = u_0 - \mathrm{e}^{\frac{2\pi \mathrm{i}}{N} j} u_1. \tag{7.53}$$

The order of the iteration (7.44) can be reversed by writing

$$\hat{f}_{\omega_j} = f_0 \cdots \mathrm{e}^{\frac{2\pi \mathrm{i}}{N}(N-1)} f_{N-1} = \mathrm{e}^{-\frac{2\pi \mathrm{i}}{N} j(N-1)} \left(f_0 \mathrm{e}^{\frac{2\pi \mathrm{i}}{N} j(N-1)} \cdots f_{N-1}\right) \tag{7.54}$$

which is very useful for real time filter applications.

7.3.2 *Fast Fourier Transformation*

If the number of samples is $N = 2^p$, the Fourier transformation can be performed very efficiently by this method.[2] The phase factor

$$\mathrm{e}^{-\mathrm{i}\frac{2\pi}{N} jm} = W_N^{jm} \tag{7.55}$$

can take only N different values. The number of trigonometric functions can be reduced by reordering the sum. Starting from a sum with N samples

$$F_N(f_0 \cdots f_{N-1}) = \sum_{n=0}^{N-1} f_n W_N^{jn} \tag{7.56}$$

[2] There exist several Fast Fourier Transformation algorithms [74, 187]. We consider only the simplest one here [62].

we separate even and odd powers of the unit root

$$\begin{aligned} F_N(f_0 \cdots f_{N-1}) &= \sum_{m=0}^{\frac{N}{2}-1} f_{2m} W_N^{j2m} + \sum_{m=0}^{\frac{N}{2}-1} f_{2m+1} W_N^{j(2m+1)} \\ &= \sum_{m=0}^{\frac{N}{2}-1} f_{2m} \mathrm{e}^{-\mathrm{i}\frac{2\pi}{N/2}jm} + W_N^j \sum_{m=0}^{\frac{N}{2}-1} f_{2m+1} \mathrm{e}^{-\mathrm{i}\frac{2\pi}{N/2}jm} \\ &= F_{N/2}(f_0, f_2 \cdots f_{N-2}) + W_N^j F_{N/2}(f_1, f_3 \cdots f_{N-1}). \end{aligned} \tag{7.57}$$

This division is repeated until only sums with one summand remain

$$F_1(f_n) = f_n. \tag{7.58}$$

For example, consider the case $N = 8$:

$$\begin{gathered} F_8(f_0 \cdots f_7) = F_4(f_0 f_2 f_4 f_6) + W_8^j F_4(f_1 f_3 f_5 f_7) \\ --- \\ F_4(f_0 f_2 f_4 f_6) = F_2(f_0 f_4) + W_4^j F_2(f_2 f_6) \\ F_4(f_1 f_3 f_5 f_7) = F_2(f_1 f_5) + W_4^j F_2(f_3 f_7) \\ --- \\ F_2(f_0 f_4) = f_0 + W_2^j f_4 \\ F_2(f_2 f_6) = f_2 + W_2^j f_6 \\ F_2(f_1 f_5) = f_1 + W_2^j f_5 \\ F_2(f_3 f_7) = f_3 + W_2^j f_7. \end{gathered}$$

Expansion gives

$$\begin{aligned} F_8 = {} & f_0 + W_2^j f_4 + W_4^j f_2 + W_4^j W_2^j f_6 \\ & + W_8^j f_1 + W_8^j W_2^j f_5 + W_8^j W_4^j f_3 + W_8^j W_4^j W_2^j f_7. \end{aligned} \tag{7.59}$$

Generally a summand of the Fourier sum can be written using the binary representation of n

$$n = \sum l_i \quad l_i = 1, 2, 4, 8 \cdots \tag{7.60}$$

in the following way:

$$f_n \mathrm{e}^{-\mathrm{i}\frac{2\pi}{N}jn} = f_n \mathrm{e}^{-\mathrm{i}\frac{2\pi}{N}(l_1+l_2+\cdots)j} = f_n W_{N/l_1}^j W_{N/l_2}^j \cdots. \tag{7.61}$$

The function values are reordered according to the following algorithm

(i) count from 0 to $N-1$ using binary numbers $m = 000, 001, 010, \cdots$
(ii) bit reversal gives the binary numbers $n = 000, 100, 010, \cdots$
(iii) store f_n at the position m. This will be denoted as $s_m = f_n$

As an example for $N = 8$ the function values are in the order

$$\begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{pmatrix} = \begin{pmatrix} f_0 \\ f_4 \\ f_2 \\ f_6 \\ f_1 \\ f_5 \\ f_3 \\ f_7 \end{pmatrix}. \tag{7.62}$$

Now calculate sums with two summands. Since W_2^j can take only two different values

$$W_2^j = \begin{cases} 1 & \text{for } j = 0, 2, 4, 6 \\ -1 & \text{for } j = 1, 3, 5, 7 \end{cases} \tag{7.63}$$

a total of 8 sums have to be calculated which can be stored again in the same workspace:

$$\begin{pmatrix} f_0 + f_4 \\ f_0 - f_4 \\ f_2 + f_6 \\ f_2 - f_6 \\ f_1 + f_5 \\ f_1 - f_5 \\ f_3 + f_7 \\ f_3 - f_7 \end{pmatrix} = \begin{pmatrix} s_0 + W_2^0 s_1 \\ s_0 + W_2^1 s_1 \\ s_2 + W_2^2 s_3 \\ s_2 + W_2^3 s_3 \\ s_4 + W_2^4 s_5 \\ s_4 + W_2^5 s_5 \\ s_6 + W_2^6 s_7 \\ s_6 + W_2^7 s_7 \end{pmatrix}. \tag{7.64}$$

Next calculate sums with four summands. W_4^j can take one of four values

$$W_4^j = \begin{cases} 1 & \text{for } j = 0, 4 \\ -1 & \text{for } j = 2, 6 \\ W_4 & \text{for } j = 1, 5 \\ -W_4 & \text{for } j = 3, 7. \end{cases} \tag{7.65}$$

The following combinations are needed:

$$\begin{pmatrix} f_0+f_4+(f_2+f_6) \\ f_0+f_4-(f_2+f_6) \\ (f_0-f_4)+W_4(f_2-f_6) \\ (f_0-f_4)-W_4(f_2-f_6) \\ f_1+f_5+(f_3+f_7) \\ f_1+f_5-(f_3+f_7) \\ (f_1-f_5)\pm W_4(f_3-f_7) \\ (f_1-f_5)\pm W_4(f_3-f_7) \end{pmatrix} = \begin{pmatrix} s_0+W_4^0 s_2 \\ s_1+W_4^1 s_3 \\ s_0+W_4^2 s_2 \\ s_1+W_4^3 s_3 \\ s_4+W_4^4 s_6 \\ s_5+W_4^5 s_7 \\ s_4+W_4^6 s_6 \\ s_5+W_4^7 s_7 \end{pmatrix}. \tag{7.66}$$

The next step gives the sums with eight summands. With

$$W_8^j = \begin{cases} 1 & j=0 \\ W_8 & j=1 \\ W_8^2 & j=2 \\ W_8^3 & j=3 \\ -1 & j=4 \\ -W_8 & j=5 \\ -W_8^2 & j=6 \\ -W_8^3 & j=7 \end{cases} \tag{7.67}$$

we calculate

$$\begin{pmatrix} f_0+f_4+(f_2+f_6)+(f_1+f_5+(f_3+f_7)) \\ f_0+f_4-(f_2+f_6)+W_8(f_1+f_5-(f_3+f_7)) \\ (f_0-f_4)+W_4(f_2-f_6)+W_8^2(f_1-f_5)\pm W_4(f_3-f_7) \\ (f_0-f_4)-W_4(f_2-f_6)+W_8^3((f_1-f_5)\pm W_4(f_3-f_7)) \\ f_0+f_4+(f_2+f_6)-(f_1+f_5+(f_3+f_7)) \\ f_0+f_4-(f_2+f_6)-W_8(f_1+f_5-(f_3+f_7)) \\ (f_0-f_4)+W_4(f_2-f_6)-W_8^2((f_1-f_5)\pm W_4(f_3-f_7)) \\ (f_0-f_4)-W_4(f_2-f_6)-W_8^3((f_1-f_5)\pm W_4(f_3-f_7)) \end{pmatrix} = \begin{pmatrix} s_0+W_8^0 s_4 \\ s_1+W_8^1 s_5 \\ s_2+W_8^2 s_6 \\ s_3+W_8^3 s_7 \\ s_0+W_8^4 s_4 \\ s_1+W_8^5 s_5 \\ s_2+W_8^6 s_6 \\ s_3+W_8^7 s_7 \end{pmatrix} \tag{7.68}$$

which is the final result.

The following shows a simple Fast Fourier Transformation algorithm. The number of trigonometric function evaluations can be reduced but this reduces the readability. At the beginning $Data[k]$ are the input data in bit reversed order.

```
size := 2
first := 0
While first < Number_of_Samples do begin
```

```
    for n := 0 to size/2 − 1 do begin
      j := first + n
      k := j + size/2 − 1
      T := exp(−2∗Pi∗i∗n/Number_of_Samples)∗Data[k]
      Data[j] := Data[j] + T
      Data[k] := Data[k] − T
    end;
    first := first∗2
    size := size∗2
  end;
```

7.4 Problems

Problem 7.1 (Discrete Fourier transformation) In this computer experiment for a given set of input samples

$$f_n = f\left(n\frac{T}{N}\right) \quad n = 0 \cdots N-1 \tag{7.69}$$

- the Fourier coefficients

$$\widetilde{f}_{\omega_j} = \sum_{n=0}^{N-1} f_n e^{-i\omega_j t_n} \quad \omega_j = \frac{2\pi}{T} j, \quad j = 0 \cdots N-1 \tag{7.70}$$

  are calculated with Goertzel's method (7.3.1).
- The results from the inverse transformation

$$f_n = \frac{1}{N} \sum_{j=0}^{N-1} \widetilde{f}_{\omega_j} e^{i\frac{2\pi}{N}nj} \tag{7.71}$$

  are compared with the original function values $f(t_n)$.
- The Fourier sum is used for trigonometric interpolation with only positive frequencies

$$f(t) = \frac{1}{N} \sum_{j=0}^{N-1} \widetilde{f}_{\omega_j} \left(e^{i\frac{2\pi}{T}t}\right)^j. \tag{7.72}$$

- Finally the unphysical high frequencies are replaced by negative frequencies (7.24). The results can be studied for several kinds of input data.

Problem 7.2 (Noise filter) This computer experiment demonstrates a nonlinear filter.

First a noisy input signal is generated.

The signal can be chosen as

- monochromatic $\sin(\omega t)$
- the sum of two monochromatic signals $a_1 \sin \omega_1 t + a_2 \sin \omega_2 t$
- a rectangular signal with many harmonic frequencies $\text{sign}(\sin \omega t)$

Different kinds of white noise can be added

- dichotomous ± 1
- constant probability density in the range $[-1, 1]$
- Gaussian probability density

The amplitudes of signal and noise can be varied. All Fourier components are removed which are below a threshold value and the filtered signal is calculated by inverse Fourier transformation.