The discrete Fourier transform

8. Approximation of the Fourier coefficients

In practical applications, we approximate the Fourier series of a function by the nth Fourier partial sum. For this, we require the Fourier coefficients $c_k[f]$ or, in other words, the amplitudes of the oscillations contained in f. Maybe, we are only interested in certain frequencies k. Nevertheless, for the discrete setting, we have to approximate the integral

$$c_k[f] := \int_0^1 f(x) \mathrm{e}^{-2\pi \mathrm{i} k x} \,\mathrm{d}x \qquad (k \in \mathbb{Z}).$$

For periodic functions, the midpoint rule applied on a uniform grid yields a quite good approximation. In so doing, we obtain

$$c_k[f] \approx \frac{1}{N} \sum_{j=0}^{N-1} f\Big(\frac{j}{N}\Big) \mathrm{e}^{-2\pi \mathrm{i} jk/N} = \frac{1}{N} \hat{f}_k$$

where

$$\hat{f}_k := \sum_{j=0}^{N-1} f\Big(\frac{j}{N}\Big) \mathrm{e}^{-2\pi \mathrm{i} jk/N} \qquad (k \in \mathbb{Z}).$$

The values $1/N\, \hat{f}_k$ thus serve as approximations of the Fourier coefficients $c_k[f]$.

Considering the periodicity of the exponential, for arbitrary $r \in \mathbb{Z}$, we notice

$$\mathrm{e}^{-2\pi \mathrm{i} j(rN+k)/N} = \mathrm{e}^{-2\pi \mathrm{i} jr}\, \mathrm{e}^{-2\pi \mathrm{i} jk/N} = \mathrm{e}^{-2\pi \mathrm{i} jk/N}$$

which leads us to the relations

$$\hat{f}_k = \hat{f}_{k+rN} \qquad \text{and} \qquad \hat{f}_{-k} = \hat{f}_{N-k}.$$

On the basis of the chosen integral approximation, the values $\hat{f}_k$ thus become N-periodic, which is quiet different to the behaviour of the decreasing Fourier

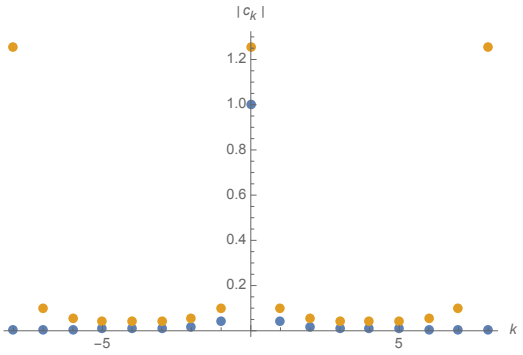


Figure 8.1.: Fourier coefficients $|c_k[f]|$ (blue) and approximations $|\frac{1}{N}\hat{f}_k|$ (orange, $N = 8$) of $f(x) = e^{x/2}$, $k = -8, \dots, 8$.

coefficients $c_k[f]$. Figure 8.1 illustrates this behaviour for the Fourier coefficients $c_k[f]$ of the function $f(x) = e^{x/2}$ and the corresponding values $\frac{1}{N}\hat{f}_k$ for $N = 8$. Since we are only interested in the approximation of the N central Fourier coefficients, here $c_{-4}[f], \dots, c_3[f]$, this systematic error is tenable.

Based on our observations, we approximate the Fourier series of f by the $N/2$th Fourier partial sum with Fourier coefficients

$$\begin{aligned} c_k[f] &\approx \frac{1}{N}\hat{f}_k, && k = 0, \dots, N/2 - 1, \\ c_{-k}[f] &\approx \frac{1}{N}\hat{f}_{N-k}, && k = 1, \dots, N/2, \end{aligned}$$

where we assume for simplicity that N is even in the moment. Note that we here use N sampling points to approximate N Fourier coefficients. Further, for computing the Fourier coefficients $c_k[f]$ for the frequencies $|k| \le N/2$, we have to sampling with the double maximal frequency or twice per oscillation – with the so called Nyquist sampling rate. To see this effect more clearly, we will derive the aliasing formula, which requires the following lemma.

Lemma 8.1. *The equispaced samples of the complex oscillator satisfy*

$$\sum_{k=0}^{N-1} \mathrm{e}^{2\pi \mathrm{i} jk/N} = \begin{cases} N & \text{if } N|j, \\ 0 & \text{else.} \end{cases}$$

(if j is a multiple of N)

Proof. 1. If j is a multiple of N, then all summands have the form $\mathrm{e}^{2\pi \mathrm{i} r} = 1$ with $r \in \mathbb{Z}$. We thus sum up exactly N ones.

2. If j is not a multiple of N, then we apply the summation formula for geometric sequences and obtain

$$\sum_{k=0}^{N-1} \mathrm{e}^{2\pi \mathrm{i} jk/N} = \frac{(\mathrm{e}^{2\pi \mathrm{i} j/N})^N - 1}{\mathrm{e}^{2\pi \mathrm{i} j/N} - 1} = 0. \qquad \square$$

This can be used to prove the *aliasing formula*, which yields a relation between the Fourier coefficients $c_k[f]$ und their approximation $\hat{f}_k$.

Theorem 8.2 (Aliasing Formula). *Let $f \in C(\mathbb{T})$ be a continuous function with absolute summable Fourier coefficients, i.e. $\sum_{k\in\mathbb{Z}} |c_k[f]| < \infty$. Then the approximate Fourier coefficients satisfy*

$$\frac{1}{N}\hat{f}_k = c_k[f] + \sum_{s\in\mathbb{Z}\setminus\{0\}} c_{k+sN}[f] \qquad (k \in \mathbb{Z}).$$

Proof. Due to Theorem 6.11, the Fourier series $\mathcal{S}_\infty[f]$ converges absolutely and uniformly to f. Consequently, the Fourier series converges pointwise, and f can be written as

$$f(x) = \sum_{k\in\mathbb{Z}} c_k[f] \mathrm{e}^{2\pi \mathrm{i} kx}.$$

For the sample points $x = j/N$ with $j = 0, \ldots, N-1$, we hence obtain

$$f\left(\tfrac{j}{N}\right) = \sum_{\ell\in\mathbb{Z}} c_l[f] \mathrm{e}^{2\pi \mathrm{i} j\ell/N}.$$

Exploiting the uniform convergence, we new compute the approximate Fourier coefficients by

$$\hat{f}_k = \sum_{j=0}^{N-1} \left(\sum_{\ell=-\infty}^{\infty} c_\ell[f] \mathrm{e}^{2\pi \mathrm{i} j\ell/N} \right) \mathrm{e}^{-2\pi \mathrm{i} jk/N} = \sum_{l=-\infty}^{\infty} c_\ell[f] \sum_{j=0}^{N-1} \mathrm{e}^{2\pi \mathrm{i} j(\ell-k)/N}.$$

Since most of the summands are zero by Lemma 8.1, we finally obtain the assertion

$$\hat{f}_k = N \sum_{s=-\infty}^{\infty} c_{k+sN}[f]. \qquad \square$$

If the function f is *bandlimited* with *bandwidth* N, which means that f is a trigonometric polynomial of order $N/2$ and hence possesses the form

$$f(x) = \sum_{k=-N/2}^{N/2-1} c_k[f] \, \mathrm{e}^{2\pi \mathrm{i} kx},$$

then the aliasing effect described in Theorem 8.2 does not occur. The approximate Fourier coefficients are thus exact, i.e.

$$\tfrac{1}{N}\,\hat{f}_k = c_k[f], \quad k = -N/2, \dots, N/2 - 1.$$

Considering the definition of the approximate Fourier coefficients $\hat{f}_k$ once again, we see that these values depend linearly from the samples $f_j = f(j/N)$ with $j = 0, \dots, N-1$. Notice that this mapping is an discretized version of the finite Fourier transform.

Definition 8.3 (Discrete Fourier transform). For an arbitrary $n \in \mathbb{N}$, the linear mapping $\boldsymbol{F}_N \colon \mathbb{C}^N \to \mathbb{C}^N$ or $\hat{\cdot} \colon \mathbb{C}^N \to \mathbb{C}^N$ defined by

$$\hat{f}_k := (\boldsymbol{F}_N \boldsymbol{f})_k := \sum_{j=0}^{N-1} f_j \, \mathrm{e}^{-2\pi \mathrm{i} jk/N} \qquad (k = 0, \dots, N-1)$$

is called the *discrete Fourier transform* (*DFT*) of $\boldsymbol{f} = (f_0, \dots, f_{N-1})^{\mathrm{T}}$.

Since the discrete Fourier transform is a linear mapping between to finit-dimensional vector spaces, it can be represented as matrix multiplication. For the DFT, we have

$$\hat{\boldsymbol{f}} = \boldsymbol{F}_N \boldsymbol{f},$$

where $\boldsymbol{F}_N$ denotes the *Nth Fourier matrix* defined by

$$\boldsymbol{F}_N := \left(\mathrm{e}^{-2\pi \mathrm{i} jk/N}\right)_{j,k=0}^{N-1}.$$

Example 8.4 (Fourier matrix). Due to the N-periodicity of $\mathrm{e}^{-2\pi \mathrm{i} k/N}$ with respect to $k \in \mathbb{Z}$, each Fourier matrix $\boldsymbol{F}_N$ consists of N different values only. For instance, the second and fourth Fourier matrix are

$$\boldsymbol{F}_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad \text{and} \qquad \boldsymbol{F}_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -\mathrm{i} & -1 & \mathrm{i} \\ 1 & -1 & 1 & -1 \\ 1 & \mathrm{i} & -1 & -\mathrm{i} \end{pmatrix}.$$

Figure 8.2 on the next page displays the real and imaginary part of the sixteenth Fourier matrix $\boldsymbol{F}_{16}$ and a plot of the first row of both. Please note that the subscription for the Fourier matrix starts with zero; so the first row is the row beneath the zeroth row. As indicated by Euler's formula, the real and imaginary part of the first row are discretized versions of the sine and cosine function. ○

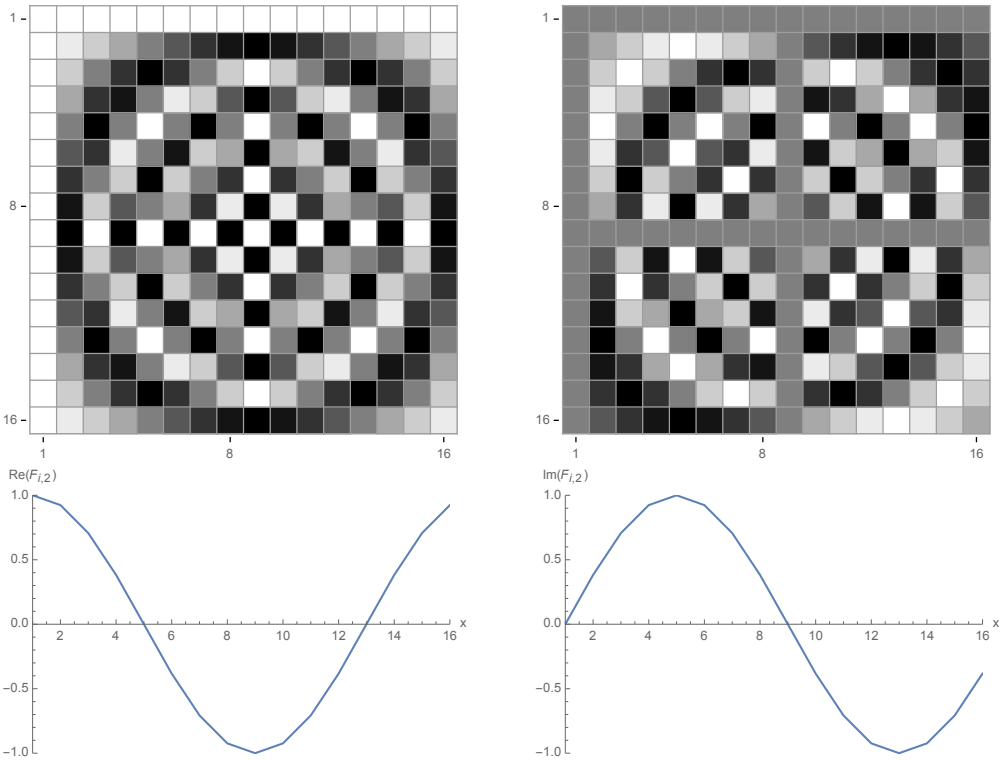


Figure 8.2.: Real and imaginary part of the Fourier matrix F_{16} (top left and right, resp.) and the corresponding first row (bottom).

The Fourier matrix is nearly unitary, which makes the inversion of the discrete Fourier transform especially simple.

Theorem 8.5 (Inversion). *The discrete Fourier transform is bijective with inverse*

$$F_N^{-1} = \frac{1}{N}\bar{F}_N.$$

Proof. We simply show that $F_N\bar{F}_N$ is a multiple of the identity matrix. Using Lemma 8.1, we here obtain

$$(F_N\bar{F}_N)_{s,t} = \sum_{k=0}^{N-1} e^{-2\pi isk/N} e^{2\pi ikt/N} = \sum_{k=0}^{N-1} e^{2\pi ik(t-s)/N} = \begin{cases} N & \text{if } t = s, \\ 0 & \text{otherwise.} \end{cases}$$

□

The inverse Fourier matrix F_N^{-1} corresponds to the inverse discrete Fourier transform.

Definition 8.6 (Inverse discrete Fourier transform). For an arbitrary $n \in \mathbb{N}$, the linear mapping $F_N^{-1} \colon \mathbb{C}^N \to \mathbb{C}^N$ or $\check{\cdot} \colon \mathbb{C}^N \to \mathbb{C}^N$ defined by

$$f_j := (F_N^{-1} \hat{f})_j := \frac{1}{N} \sum_{k=0}^{N-1} \hat{f}_k \, \mathrm{e}^{2\pi \mathrm{i} k j / N} \qquad (j = 0, \dots, N-1)$$

is called the *inverse discrete Fourier transform* (*IDFT*) of $\hat{f} = (\hat{f}_0, \dots, \hat{f}_{N-1})^{\mathrm{T}}$.

Due to the symmetry of the Fourier matrix, the inversion theorem for the discrete Fourier transform gives us immediately a discretized version of Parseval's identity for the Euclidean inner product and norm

$$\langle f, g \rangle := \sum_{k=0}^{N-1} f_k \, \bar{g}_k, \qquad \| f \|_2 := \left(\sum_{k=0}^{N-1} |f_k|^2 \right)^{\frac{1}{2}} \qquad (f, g \in \mathbb{C}^N).$$

Corollary 8.7 (Parseval's identity). *For every $f \in \mathbb{C}^N$ and $g \in \mathbb{C}^N$, we have*

$$\langle f, g \rangle = \tfrac{1}{N} \langle F_N f, F_N g \rangle \qquad \text{and} \qquad \| f \|_2 = \tfrac{1}{\sqrt{N}} \| F_N f \|_2.$$

If we apply the discrete Fourier transform twice, we obtain the cyclic reflection of the orignal signal, where we reflect around the index zero. For the cyclic operations, we take the indices modulo N, which is indicated by

$$(k)_N := k \bmod N.$$

Definition 8.8 (Cyclic reflection). For every vector $f \in \mathbb{C}^N$, the *cyclic reflection* R_N is defined by

$$R_N f := \left(f_{(-k)_N} \right)_{k=0}^{N-1}.$$

Remark 8.9 (Cyclic reflection). The cyclic reflection corresponds to the *reversion* or *flip matrix*

$$R_N := \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 1 \\ \vdots & \vdots & \iddots & \vdots \\ 0 & 1 & \cdots & 0 \end{pmatrix} = \left(\delta_{(j+k)_N} \right)_{j,k=0}^{N-1},$$

where δ_n denotes the *Kronecker symbol*

$$\delta_n := \begin{cases} 1 & \text{if } n = 0, \\ 0 & \text{otherwise.} \end{cases}$$

The cyclic reflection matrix is orthogonal, i.e. $\boldsymbol{R}_N^2 = \boldsymbol{I}_N$, where $\boldsymbol{I}_N$ is the $(N \times N)$-dimensional identity matrix. If we apply $\boldsymbol{R}_N$ to a vector $\boldsymbol{f} \in \mathbb{C}^N$, the vector is flipped as

$$\boldsymbol{R}_N \boldsymbol{f} = (f_0, f_{N-1}, f_{N-2}, \ldots, f_1)^{\mathrm{T}}. \qquad \circ$$

Lemma 8.10 (Squared Fourier matrix). *The Nth Fourier matrix satisfies*

$$\boldsymbol{F}_N^2 = N \boldsymbol{R}_N.$$

Proof. Using Lemma 8.1, we can compute the components of the squared Fourier matrix by

$$(\boldsymbol{F}_N \boldsymbol{F}_N)_{s,t} = \sum_{k=0}^{N-1} \mathrm{e}^{-2\pi \mathrm{i} s k/N} \, \mathrm{e}^{-2\pi \mathrm{i} k t/N} = \sum_{k=0}^{N-1} \mathrm{e}^{-2\pi \mathrm{i}(s+t)/N} = N \, \delta_{(s+t)_N}, \qquad \square$$

Corollary 8.11 (Inversion). *The Fourier matrix fulfils*

$$\bar{\boldsymbol{F}}_N = \boldsymbol{R}_N \boldsymbol{F}_N = \boldsymbol{F}_N \boldsymbol{R}_N \qquad \text{and} \qquad \boldsymbol{F}_N^{-1} = \tfrac{1}{N} \boldsymbol{R}_N \boldsymbol{F}_N.$$

As a consequence, we can compute the inverse discrete Fourier transform numerically by applying an algorithm for the discrete Fourier transform and reverting the outcome cyclically. Since the cyclic reflection matrix is orthogonal, we can simply compute the eigenvalues of the Fourier matrix.

Corollary 8.12 (Eigenvalues). *The eigenvalues of the Nth Fourier matrix are $\pm\sqrt{N}$ and $\pm \mathrm{i}\sqrt{N}$ with appropriate multiplicities.*

Proof. If λ is an eigenvalue corresponding to the eigenvector $\boldsymbol{f}$, then we obtain

$$\boldsymbol{F}_N^4 \boldsymbol{f} = N^2 \boldsymbol{f} = \lambda^4 \boldsymbol{f}.$$

The eigenvalues are thus solutions of $\lambda^4 - N^2 = 0$. $\square$

The multiplicities of the eigenvalues in dependance of the length N of the Fourier transform $\boldsymbol{F}_N$ are shown in Table 8.1 on the following page. Further, the discrete Fourier transform inherit the shift and modulation properties of the finite Fourier transform.

	eigenvalues			
N	$\sqrt{N}$	$-\sqrt{N}$	$\mathrm{i}\sqrt{N}$	$-\mathrm{i}\sqrt{N}$
$4n$	$n+1$	n	$n-1$	n
$4n+1$	$n+1$	n	n	n
$4n+2$	$n+1$	$n+1$	$n-1$	n
$4n+3$	$n+1$	n	n	$n+1$

Table 8.1.: Multiplicities of the eigenvalues of the Fourier matrix F_N.

Definition 8.13 (Cyclic transition). For every vector $f \in \mathbb{C}^N$, the *cyclic forward transition* T_N is defined by

$$T_N f := \left(f_{(k-1)_N}\right)_{k=0}^{N-1}.$$

Remark 8.14 (Cyclic transition). The cyclic forward transition corresponds to the *cyclic shift matrix*

$$T_N := \begin{pmatrix} 0 & \cdots & 0 & 1 \\ 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & 0 \end{pmatrix} = \left(\delta_{(j-k+1)_N}\right)_{j,k=0}^{N-1}.$$

The multiplication with a vector $f \in \mathbb{C}^N$ yields

$$T_N f = (f_{N-1}, f_0, \ldots, f_{N-2})^{\mathrm{T}}.$$

The *cyclic backward transition* corresponds to $T_N^{-1} = T_N^{\mathrm{T}}$. ○

Definition 8.15 (Cyclic modulation). For every vector $f \in \mathbb{C}^N$, the *cyclic modulation* M_N is defined by

$$M_N f := \left(\mathrm{e}^{-2\pi \mathrm{i} k/N} f_k\right)_{k=0}^{N-1}.$$

Remark 8.16 (Cyclic modulation). The cyclic modulation corresponds to the multiplication with the diagonal matrix $\operatorname{diag}(\mathrm{e}^{-2\pi \mathrm{i} k/N})_{k=0}^{N-1}$. ○

Similarly to the finite Fourier transform, shifts are transformed to modulations and the other way round.

Proposition 8.17 (Transition/modulation). *For $f \in \mathbb{C}^N$, the discrete Fourier transform satisfies*

$$F_N T_N f = M_N F_N f \qquad \text{and} \qquad F_N M_N f = T_N^{-1} F_N f.$$

Proof. For the cyclic transition, we have

$$(T_N f)_k^\wedge = \sum_{j=0}^{N-1} f_{(j-1)_N} \, \mathrm{e}^{-2\pi \mathrm{i} jk/N} = \sum_{j=0}^{N-1} f_j \, \mathrm{e}^{-2\pi \mathrm{i} k/N} \, \mathrm{e}^{-2\pi \mathrm{i} jk/N} = \mathrm{e}^{-2\pi \mathrm{i} k/N} \, \hat{f}_k.$$

The discrete Fourier transform of the cyclic modulation is given by

$$(M_N f)_k^\wedge = \sum_{j=0}^{N-1} \mathrm{e}^{-2\pi \mathrm{i} j/N} \, f_j \, \mathrm{e}^{-2\pi \mathrm{i} jk/N} = \sum_{j=0}^{N-1} f_j \, \mathrm{e}^{-2\pi \mathrm{i} j(k+1)/N} = \hat{f}_{(k+1)_N}. \qquad \square$$

Since the discrete Fourier transform already shares many properties with the finite Fourier transform, we may also think about a Fourier convolution theorem for vectors. In so doing, we define the cyclic convolution of two vectors. Similarly to the discrete Fourier transform, the cyclic convolution for vectors can be deduced by applying the midpoint rule.

Definition 8.18 (Cyclic convolution). The *cyclic convolution* of two vectors $f \in \mathbb{C}^N$ and $g \in \mathbb{C}^N$ is given by the vector $h := f * g \in \mathbb{C}^N$ having the entries

$$h_k = \sum_{j=0}^{N-1} f_j \, g_{(k-j)_N} = \sum_{j=0}^{N-1} f_{(k-j)_N} \, g_j.$$

If we fix the vector g, then the cyclic convolution becomes a linear mapping for the vector f. Consequently, the cyclic convolution can be represented by a matrix multiplication. More precisely, we have

$$h = C(g) \, f$$

with the *circulant matrix*

$$C(g) := \begin{pmatrix} g_0 & g_{N-1} & \cdots & g_1 \\ g_1 & g_0 & \cdots & g_2 \\ \vdots & \vdots & \ddots & \vdots \\ g_{N-1} & g_{N-2} & \cdots & g_0 \end{pmatrix}.$$

The circulant matrix $C(g)$ has g as first column. The remaining columns are obtained by a circular shift of the previous column.

In Theorem 5.5, we have established the relation between the cyclic convolution of two periodic functions and the pointwise multiplication of their Fourier coefficients. The following lemma introduces the same relation for discrete vectors. The proof and the opposite direction is as an exercise.

Theorem 8.19 (Fourier convolution). *For $f, g \in \mathbb{C}^N$, the discrete Fourier transform fulfils*

$$F_N(f * g) = (F_N f) \circ (F_N g).$$

The element by element multiplication of two vectors is here denoted by $\circ$, which is also known as *Hadamard product*. As a consequence of the discrete Fourier convolution theorem, each circulant matrix can be easily diagonalized using Fourier matrices.

Corollary 8.20 (Diagonalization). *Every circulant matrix $C(g)$ with $g \in \mathbb{C}^N$ can be diagonalized as*

$$C(g) = F_N^{-1}(\operatorname{diag}(F_N g))\, F_N.$$

Proof. Simply multiply both sides of the assertion with any $f \in \mathbb{C}^N$. Due to Theorem 8.19, both sides then coincide. Since f is arbitrary, the matrices on both sides have to conincide too. □

In applications, the vector g is often called a *filter*. The idea behind this is simply that the convolution with g can be used to filter or block certain frequencies of a given f. Usually, one distinguish the following kinds of filters:

- *Low-pass filters* lower the high frequency parts of f and leave the low frequency parts nearly the same.
- *High-pass filters* lowers the low frequency parts of f and leave the high frequency parts nearly the same.
- *Band-pass filters* lower a certain frequency band of f and leave the other frequency parts nearly the same.

9. The fast Fourier transform

Computing the discrete Fourier transform $\hat{\boldsymbol{f}} = \boldsymbol{F}_N \boldsymbol{f}$ by a matrix multiplication requires exactly N^2 (complex-valued) multiplications and $N\,(N-1)$ (complex-valued) additions. In total, this means that we need $\mathcal{O}(N^2)$ arithmetic operation. For practical applications, this is unfortunately too time consuming (even nowadays), because doubling the length of the input vector results in a quadruple computation time.

An important step towards applications has been the development of a fast numerical algorithm to compute the discrete Fourier transform. The so-called *fast Fourier transform (FFT)* proposed by Cooley and Tukey in 1965 only requires $\mathcal{O}(N \log N)$ arithmetic operations. Originally, the fast Fourier transform has been developed for discrete Fourier transforms of length N being a power of 2 and has afterwards been generalized to arbitrary sizes of N.

The basic concepts of the fast Fourier transform have been well known before Cooley and Tukey. For instance, Carl Friedrich Gauß has already used this concepts while examining the path of the planetoid 'Pallas' in 1805. Also Carl David Tolmé Runge employed similar ideas in 1903:

> Once the [FFT] method was established it became clear that it had a long and interesting prehistory going back as far as Gauß. But until the advent of computing machines it was a solution looking for a problem.
>
> Source: T. W. Körner, *Fourier Analysis* (1988)

Currently the fastest FFT software is the publicly available FFTW (fastest Fourier transform in the West) on `http://www.fftw.org`. This packages performs a preprocessing to determine the fastest variant for the signal input length at hand. In this lecture, we only provide one of the simplest FFT algorithms – the Sande–Tukey algorithm.

The radix-2-FFT This fast Fourier transform works only for signals or vectors of length $N := 2^n$ with $n \in \mathbb{N}$ and is based on the divide-and-conquer strategy, which is named after the divide-et-impera strategy of ancient Rome. Assume that we want to compute the discrete Fourier transform of length N that is abbreviated by DFT(N) in the following. In so doing, we have

$$\hat{f}_k = \sum_{j=0}^{N-1} f_j \, \mathrm{e}^{-2\pi \mathrm{i} jk/N} = \sum_{j=0}^{N-1} f_j \, w_N^{jk} \qquad (k = 0, \ldots, N-1),$$

where $w_N := \mathrm{e}^{-2\pi\mathrm{i}/N}$. Now, we divide the sum in the middle into

$$\hat{f}_k = \sum_{j=0}^{N/2-1} f_j \, w_N^{jk} + \sum_{j=0}^{N/2-1} f_{N/2+j} \, w_N^{(N/2+j)k}, \quad k = 0, \ldots, N-1 \tag{9.1}$$

and look at the even and odd indices k of $\hat{f}_k$ separately.

THE EVEN INDICES First we consider the even indices, which means that we assume $k = 2\ell$. Plugging the representation of the index into (9.1), we obtain

$$\hat{f}_{2\ell} = \sum_{j=0}^{N/2-1} f_j \, w_N^{2j\ell} + \sum_{j=0}^{N/2-1} f_{N/2+j} \, w_N^{(N/2+j)2\ell} \qquad (\ell = 0, \ldots, N/2-1)$$

Exploiting the identity $w_N^{2jl} = w_{N/2}^{jl}$ as well as the identity

$$w_N^{(N/2+j)2\ell} = \mathrm{e}^{-2\pi\mathrm{i}(N/2+j)2l/N} = \mathrm{e}^{-2\pi\mathrm{i}\ell} \, \mathrm{e}^{-2\pi\mathrm{i}j\ell/(N/2)} = w_{N/2}^{jl},$$

we combine the left-hand and right-hand sum in order to get

$$\hat{f}_{2\ell} = \sum_{j=0}^{N/2-1} f_j \, w_{N/2}^{j\ell} + \sum_{j=0}^{N/2-1} f_{N/2+j} \, w_{N/2}^{j\ell} = \sum_{j=0}^{N/2-1} (f_j + f_{N/2+j}) w_{N/2}^{j\ell}$$

for every $\ell = 0, \ldots, N/2-1$. Adding the upper and lower half of the input signal by $f_j + f_{N/2+j}$ for $j = 0, \ldots, N/2-1$, we thus have to compute a DFT$(N/2)$ – a discrete FOURIER transform of half length.

THE ODD INDICES Proceeding likewise for the odd indices $k = 2\ell + 1$, we obtain the calculation rule

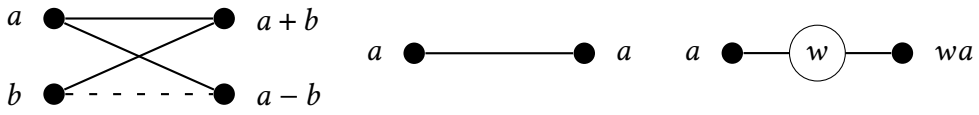
$$\hat{f}_{2\ell+1} = \sum_{j=0}^{N/2-1} f_j \, w_N^{j(2\ell+1)} + \sum_{j=0}^{N/2-1} f_{N/2+j} \, w_N^{(N/2+j)(2l+1)} \qquad (\ell = 0, \ldots, N/2-1).$$

from (9.1). With the identity

$$w_N^{(N/2+j)(2\ell+1)} = w_N^{N/2\,(2\ell+1)} \, w_N^{j(2\ell+1)} = \mathrm{e}^{-2\pi\mathrm{i}(N/2)(2l+1)/N} \, w_N^j \, w_{N/2}^{j\ell} = -w_N^j \, w_{N/2}^{j\ell},$$

we rewrite the DFT(N) as

$$\hat{f}_{2\ell+1} = \sum_{j=0}^{N/2-1} f_j \, w_{N/2}^{j\ell} w_N^j - \sum_{j=0}^{N/2-1} f_{N/2+j} \, w_{N/2}^{j\ell} \, w_N^j = \sum_{j=0}^{N/2-1} (f_j - f_{N/2+j}) \, w_N^j \, w_{N/2}^{j\ell}$$



(a) Addition/Subtraction **(b)** Identity **(c)** Multiplication

Figure 9.1.: Basic operations for the FFT flow chart.

for every $\ell = 0, \dots, N/2 - 1$. Thus, we have to compute $N/2$ differences $(f_j - f_{N/2+j})$ for $j = 0, \dots, N/2-1)$ and $N/2$ multiplications with roots of unity called the *twiddle factors* w_N^j and afterwards again a DFT$(N/2)$.

Summarizing both cases, we have to apply N additions and $N/2$ multiplications in order to divide the DFT(N) into two DFT$(N/2)$. To compute these, we apply the same procedure for both DFT$(N/2)$s and end up with four DFT$(N/4)$s and so on. For $N = 2^n$, this leads us to

$$\mathrm{DFT}(N) \longrightarrow 2\,\mathrm{DFT}(N/2) \longrightarrow 4\,\mathrm{DFT}(N/4) \longrightarrow \cdots \longrightarrow N\,\mathrm{DFT}(1).$$

In every division step ($\longrightarrow$), we have to add to upper half $\boldsymbol{f}_{\mathrm{up}}$ of the input vector with the lower half $\boldsymbol{f}_{\mathrm{low}}$ and to substract the lower half multiplied with the twiddle factors $\boldsymbol{w} = (w_N^0, \dots, w_N^{N/2-1})^{\mathrm{T}}$ from the upper half. Schematically, for the first division, this looks like

$$\begin{pmatrix} \hat{\boldsymbol{f}}_{\mathrm{even}} \\ \hat{\boldsymbol{f}}_{\mathrm{odd}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{F}_{N/2}(\boldsymbol{f}_{\mathrm{up}} + \boldsymbol{f}_{\mathrm{low}}) \\ \boldsymbol{F}_{N/2}(\boldsymbol{w} \circ (\boldsymbol{f}_{\mathrm{up}} - \boldsymbol{f}_{\mathrm{low}})). \end{pmatrix}$$

Then, the procedure is repeated for $\hat{\boldsymbol{f}}_{\mathrm{even}}$ and $\hat{\boldsymbol{f}}_{\mathrm{odd}}$. After $n = \log_2 N$ divisions, we finally have to compute N DFT(1)s, which is trivial since the DFT(1) is just the identity or the multiplication with one. In total, this approach requires nN additions, where we count subtractions as additions, and $nN/2$ multiplications with the twiddle factors; so we require $\mathcal{O}(Nn) = \mathcal{O}(N \log N)$ arithmetic operations.

For the sake of clarity, we illustrate the Sande–Tukey algorithm using a flow diagram. The basic operations like addition, subtraction, and multiplication are shown in Figure 9.1, where on the left-hand side is the input, and on the right-hand side the output. The complete flow diagram of the DFT(8) is shown in Figure 9.2 on the following page.

If we implement the Sande–Tukey algorithm as presented in the flow chart, we can perform every step in place, which means that the output of every basic step require as much memory as the input, which is not needed afterwards. Since we split the output vector in each division into the vector with even and the vector with odd indices, we compute a permuted version of the discrete Fourier

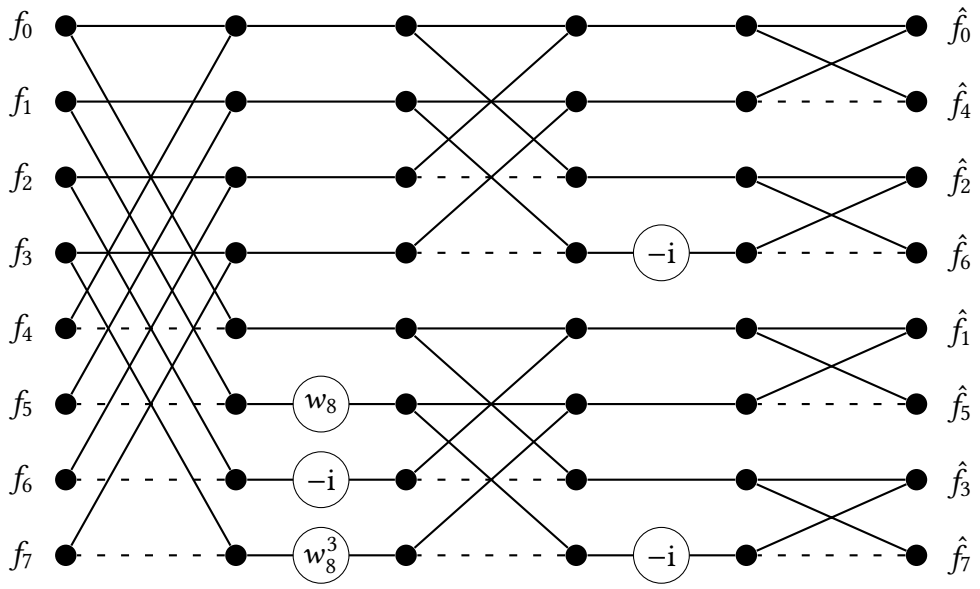


Figure 9.2.: Flow chart of the FFT of length $N = 8$

transform. Writing the indices as binary numbers, we sort the elements of $\hat{\boldsymbol{f}}$ in the first step with respect to the first binary place, in the second step with respect to the second binary place and so on. In this manner, we obtain the result in bit-reversal order. More precisely, at position

$$r = 2^{n-1}r_{n-1} + \cdots + 2r_1 + r_0 = (r_{n-1}, \ldots, r_0)_2$$

with $r_i \in \{0, 1\}$ of the result vector, we have the value $\hat{f}_k$ with index

$$k = (r_0, \ldots, r_{n-1})_2 = 2^{n-1}r_0 + \cdots + 2r_{n-2} + r_{n-1}$$

r		k	
index	binary number	binary number	index
0	$(0,0,0)_2$	$(0,0,0)_2$	0
1	$(0,0,1)_2$	$(1,0,0)_2$	4
2	$(0,1,0)_2$	$(0,1,0)_2$	2
3	$(0,1,1)_2$	$(1,1,0)_2$	6
4	$(1,0,0)_2$	$(0,0,1)_2$	1
5	$(1,0,1)_2$	$(1,0,1)_2$	5
6	$(1,1,0)_2$	$(0,1,1)_2$	3
7	$(1,1,1)_2$	$(1,1,1)_2$	7

Table 9.1.: Bitreversal for $N = 8$.

as shown for the length $N = 8$ in Table 9.1 on the preceding page.

Using the above divide-and-conquer strategy, the fast Fourier transform can also be written as matrix product, where we split the fully populated Fourier matrix $\boldsymbol{F}_N$ into a product of sparse matrices. The successive multiplication with these matrices is then faster than the multiplication with the full matrix. Following the flow chart in Figure 9.2 on the facing page, we may factorize the Fourier matrix $\boldsymbol{F}_8$ into

$$
\begin{aligned}
\boldsymbol{F}_8 = &
\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -1
\end{pmatrix} \\
& \times
\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -i & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -i
\end{pmatrix}
\begin{pmatrix}
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & -1
\end{pmatrix} \\
& \times
\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & w_8 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -i & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & w_8^3
\end{pmatrix}
\begin{pmatrix}
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & -1
\end{pmatrix}
\end{aligned}
$$

Note that the presented Sande–Tukey algorithm is only one possible radix-2-FFT. Where we have splited (9.1) in the middle, it is also possible to split the computation rule with respect to the even- and odd-subscripted elements of the input vector f. Further, one can split N into higher powers of 2, which leads us to the so-called radix-4-FFT, radix-8-FFT, and so on. Moreover, the DFT(N) can be divided into arbitrary smaller discrete Fourier transforms of the same

size, and the size can be chosen for each division step independently. This idea would lead us to the so-called prime-factor FFT, which provides a more or less fast discrete Fourier transform for arbitrary input lengths. The performance here depends on how well the dimension N can be divided into prime factors – the more the better.

10. Non-equispaced FFT

For the discrete Fourier transform and thus the FFT-algorithm, the crucial idea have been the equispaced sampling of the underlying 1-periodic function f. Having the Fourier coefficients $c[f]$ of a function, we then approximate the function f by a Fourier partial sum $S_n[f]$. In order to compute some function values of f approximately, we simply have to evaluate $S_n[f]$. If we are interested in the function values at the equispaced sampling points j/N with $j = 0, \dots, N-1$, then we can also apply the inverse discrete Fourier transform, which again corresponds to a matrix multiplication, or the fast Fourier transform and conjugate the result, see Theorem 8.5.

The crucial observation is here that we may compute the approximated function values of f again with a fast algorithm in certain cases. The main question is thus: can we develop an fast algorithm to compute the function values of a Fourier partial sum in general. We are hence interested in finding a numerical algorithm to solve the following problem.

Problem 10.1 (Non-equispaced sampling points). *For the given even numbers N and M, find a fast numerical method to compute*

$$f(x_j) = \frac{1}{N} \sum_{k=-N/2}^{N/2-1} \hat{f}_k \, \mathrm{e}^{2\pi \mathrm{i} k x_j} \qquad (j = -M/2, \dots, M/2 - 1),$$

where $\hat{f}_k$ denote the approximate Fourier coefficients, and x_j the sampling points.

Remark 10.2 (Fourier partial sums). The function values in Problem 10.1 are approximated by a Fourier-like partial sum, where the minimal and maximal index of the employed Fourier coefficients are non-symmetric. The reason for this is simply that we want to exploit the classical fast Fourier transform for vectors

whose length is a power of two. In order to compute the nth Fourier partial sum nevertheless, one can simply set $\hat{f}_{-N/2} = 0$. If the function f is real-valued and bandlimited with bandwidth N, then the coefficient $\hat{f}_{-N/2}$ vanishes in a natural manner. ○

The other way round, we can also ask whether it is possible to derive fast algorithms to compute the Fourier coefficients of a function when the samples are not equispaced.

Problem 10.3 (Non-equispaced samples). *For the given even numbers N and M, find a fast numerical method to compute*

$$\hat{f}_k = \sum_{j=-M/2}^{M/2-1} f_j \, \mathrm{e}^{-2\pi \mathrm{i} k x_j} \qquad (k = -N/2, \dots, N/2 - 1),$$

where $f_j := f(x_j)$ with $x_j \in \mathbb{T}$.

The above two problems are adjoint to each other. More precisely, the matrix versions of Problem 10.1 and 10.3 are given by

$$\boldsymbol{f} = \tfrac{1}{N} \boldsymbol{F} \hat{\boldsymbol{f}} \qquad \text{and} \qquad \hat{\boldsymbol{f}} = \boldsymbol{F}^{\mathrm{H}} \boldsymbol{f},$$

where the occurring vectors and matrices are the following:

- sample vector $\boldsymbol{f} := (f(x_j))_{j=-M/2}^{M/2-1}$,
- approximate Fourier coefficients $\hat{\boldsymbol{f}} := (\hat{f}_k)_{k=-N/2}^{N/2-1}$,
- Fourier-like matrix $\boldsymbol{F} := (\mathrm{e}^{2\pi \mathrm{i} k x_j})_{j=-M/2,k=-N/2}^{M/2-1,N/2-1}$.

Here the $\boldsymbol{F}^{\mathrm{H}}$ denotes the *adjoint* or *Hermitian transpose* $\bar{\boldsymbol{F}}^{\mathrm{T}}$. Thus we may solve both problems simultaneously.

In order to solve Problem 10.1, we approximate the 1-periodic function f of interest by a superposition

$$s(x) := \sum_{\ell=-n/2}^{n/2-1} g_\ell \, \phi(x - \ell/n), \tag{10.1}$$

where $n := \sigma N$ with $\sigma > 1$, and where ϕ is some 1-periodic function. The number of transpositions thus has to be larger than the number of the approximate

Fourier coefficients. Assuming that the function ϕ is known, the main question is now how do we have to chose the coefficients g_ℓ depending on the given approximate Fourier coefficients $\hat{f}_k$. If the structure function ϕ is contained in $L^2(\mathbb{T})$, we can represent s at Fourier series $S_\infty[s]$ with Fourier coefficients

$$c_k[s] = \int_{-1/2}^{1/2} \sum_{\ell=-n/2}^{n/2-1} g_\ell\, \phi(x-\ell/n)\, \mathrm{e}^{-2\pi \mathrm{i} k x}\,\mathrm{d}x = \sum_{\ell=-n/2}^{n/2-1} g_\ell \int_{-1/2}^{1/2} \phi(x-\ell/n)\, \mathrm{e}^{-2\pi \mathrm{i} k x}\,\mathrm{d}x$$

$$= \sum_{\ell=-n/2}^{n/2-1} g_\ell\, \mathrm{e}^{-2\pi \mathrm{i} k \ell/n} \int_{-1/2-\ell/n}^{1/2-\ell/n} \phi(y)\, \mathrm{e}^{-2\pi \mathrm{i} k y}\,\mathrm{d}y = \hat{g}_k\, c_k[\phi]$$

where we used the substitution $y := x - \ell/n$ and the discrete Fourier transform of the coefficients g_ℓ.

Next, we compare the Fourier-like partial sum

$$f(x) = \frac{1}{N} \sum_{k=-N/2}^{N/2-1} \hat{f}_k\, \mathrm{e}^{2\pi \mathrm{i} k x}$$

from Problem 10.1 with the Fourier series of s, which we rearrage to

$$s(x) = \sum_{k\in\mathbb{Z}} \hat{g}_k\, c_k[\phi] \mathrm{e}^{2\pi \mathrm{i} k x} = \sum_{r\in\mathbb{Z}} \sum_{k=-n/2}^{n/2-1} \hat{g}_{k+nr}\, c_{k+nr}[\phi]\, \mathrm{e}^{2\pi \mathrm{i}(k+nr)x}$$

$$= \sum_{k=-n/2}^{n/2-1} \hat{g}_k\, c_k[\phi]\, \mathrm{e}^{2\pi \mathrm{i} k x} + \sum_{\substack{r\in\mathbb{Z}\\ r\neq 0}} \sum_{k=-n/2}^{n/2-1} \hat{g}_k\, c_{k+nr}[\phi]\, \mathrm{e}^{2\pi \mathrm{i}(k+nr)x}, \tag{10.2}$$

where we exploit the n-periodicity of $\hat{g}_k$. Since s should approximate f, we now chose the discrete Fourier coefficients $\hat{g}_k$ as

$$\hat{g}_k := \hat{g}_{k+nr} := \begin{cases} \dfrac{\hat{f}_k}{N c_k[\phi]} & \text{for } k = -N/2, \dots, N/2-1, \\ 0 & \text{for } k = -n/2, \dots, N/2-1 \text{ or } k = N/2, \dots, n/2-1. \end{cases} \tag{10.3}$$

The quality of the approximation then depends on the remaining Fourier coefficients $c_k[\phi]$ with $k \notin \{-n/2, \dots, n/2-1\}$.

After computing the inverse discrete Fourier transform of the coefficients $\hat{g}_k$, we can approximate the wanted function values $f(x_j)$ with the point evaluations

$s(x_j)$ in (10.1). Since we are interested in a fast algorithm, we now chose the structure function ϕ such that ϕ is negligible outside the (1-periodically recurring) interval $[-m/n, m/n]$ with $m \ll n$. We assume that the length of the interval is much smaller than one. In so doing, we now approximate s by the truncated function $\tilde{s}$ given by

$$\tilde{s}(x) = \sum_{\ell=-n/2}^{n/2-1} g_\ell \, \psi(x - \ell/n) = \sum_{\ell=\lceil nx \rceil - m}^{\lfloor nx \rfloor + m} g_\ell \, \psi(x - \ell/n) \tag{10.4}$$

with

$$\psi(x) := \begin{cases} \phi(x) & \text{if } x \in [-m/n, m/n], \\ 0 & \text{if } x \in \mathbb{T} \setminus [-m/n, m/n]. \end{cases}$$

Here we only sum up the shifted functions $\psi(\cdot - \ell/n)$ covering the point x. Moreover, we set $g_\ell := 0$ for $\ell \notin \{-n/2, \dots, n/2 - 1\}$. This leads us to the so-called non-equispaced fast Fourier transform (NFFT) to solve Problem 10.1.

Algorithm 10.4 (Non-equispaced FFT).
Input: $\hat{f}_k$ with $k = -N/2, \dots, N/2 - 1$.

(i) Compute the discrete Fourier coefficients

$$\hat{g}_k := \frac{\hat{f}_k}{N c_k[\phi]} \qquad (k = -N/2, \dots, N/2 - 1).$$

(ii) Compute inverse discrete Fourier transform

$$g_\ell := \frac{1}{n} \sum_{k=-N/2}^{N/2-1} \hat{g}_k \, \mathrm{e}^{2\pi \mathrm{i} k \ell / n} \qquad (\ell = -n/2, \dots, n/2 - 1)$$

using Corollary 8.11 and the FFT(n) with $\hat{g}_\ell := 0$ for $\ell = -n/2, \dots, -N/2 - 1$ and for $\ell = N/2, \dots, n/2 - 1$.

(iii) Approximate the function values by

$$f_j := \sum_{\ell=\lceil nx_j \rceil - m}^{\lfloor nx_j \rfloor + m} g_\ell \, \psi(x_j - \ell/n) \qquad (j = -M/2, \dots, M/2 - 1).$$

Output: $f_j = f(x_j)$ with $j = -M/2, \dots, M/2 - 1$.

Counting the arithmetic operations in Algorithm 10.4, the non-equispaced fast Fourier transformation requires

$$\mathcal{O}(N + n \log n + (2m + 1)M) = \mathcal{O}(n \log n + mM)$$

operations and is thus comparable with the complexity of the FFT.

In order to solve Problem 10.3, we first write the NFFT in matrix-vector form

$$\tfrac{1}{N}\, \boldsymbol{F} \approx \boldsymbol{B}\tilde{\boldsymbol{F}}\boldsymbol{D},$$

where each matrix corresponds to one step of Algorithm 10.4. More precisely, we have

(i) the diagonal matrix

$$\boldsymbol{D} := \operatorname{diag}\Big(\frac{1}{nN\, c_k[\phi]}\Big)_{k=-N/2}^{N/2-1},$$

(ii) the truncated and conjugated Fourier-like matrix

$$\tilde{\boldsymbol{F}} := \Big(\mathrm{e}^{2\pi \mathrm{i} k\ell/n}\Big)_{\ell=-n/2,k=-N/2}^{n/2-1,N/2-1},$$

(iii) the sparse $(2m+1)$-band matrix

$$\boldsymbol{B} := \big(b_{j\ell}\big)_{j=-M/2,\ell=-n/2}^{M/2-1,n/2-1},$$

with

$$b_{j\ell} := \begin{cases} \phi(x_j - \ell/n) & \text{if } \ell \in \{\lceil x_j n\rceil - m, \dots, \lfloor x_j n\rfloor + m\}, \\ 0 & \text{otherwise.} \end{cases}$$

Notice that the division by n has been shifted from the second to the first step. Considering the multiplication with the adjoint $\boldsymbol{F}^{\mathrm{H}}$, we obtain the a numerical method to solve Problem 10.3, which is called NFFT$^{\mathrm{H}}$.

Algorithm 10.5 (Non-equispaced FFT$^{\mathrm{H}}$).
Input: $f_j = f(x_j)$ with $j = -M/2, \dots, M/2-1$.

(i) Set $g := (g_\ell)_{\ell=-n/2}^{n/2-1} := 0$.
For $j = -M/2, \dots, M/2-1$ do:
For $\ell = \lceil nx_j\rceil - m, \dots, \lfloor nx_j\rfloor + m$ do:

$$g_\ell := g_\ell + \overline{\phi(x_j - \ell/n)}\, f_j.$$

(ii) Compute discrete Fourier transform

$$\hat{g}_k := \sum_{\ell=-n/2}^{n/2-1} g_\ell\, \mathrm{e}^{-2\pi \mathrm{i} k\ell/n}$$

using the FFT(n).

(iii) Compute the approximate FOURIER coefficients

$$\hat{f}_k := \frac{\hat{g}_k}{n\, \bar{c}_k[\phi]} \qquad (k = -N/2, \dots, N/2 - 1).$$

OUTPUT: $\hat{f}_k$ with $k = -N/2, \dots, N/2 - 1$.

To study the numerical performance of the NFFT, which is based on a twofold approximation, we devide the absolute error $E(x_j) := |f(x_j) - \tilde{s}(x_j)|$ into

- the *aliasing error* $E_\mathrm{a} := |f(x_j) - s(x_j)|$,
- the *truncation error* $E_\mathrm{t} := |s(x_j) - \tilde{s}(x_j)|$.

Both errors mainly depend on the chosen structure function ϕ and on the 1-norm of the given approximate FOURIER coefficients $\hat{f}$ given by

$$\|\hat{f}\|_1 := \sum_{k=-N/2}^{N/2-1} |\hat{f}_k|.$$

THEOREM 10.6 (ALIASING ERROR). *For $\phi \in L^2(\mathbb{T})$, for even $N \in \mathbb{N}$ and $M \in \mathbb{N}$, and for the power of two $n = \sigma N$ with $\sigma > 1$, the aliasing error of the NFFT is bounded by*

$$E_\mathrm{a}(x_j) \le \frac{\|\hat{f}\|_1}{N} \left\{ \max_{k=-N/2,\dots,N/2-1} \sum_{\substack{r\in\mathbb{Z}\\ r\neq 0}} \left| \frac{c_{k+nr}[\phi]}{c_k[\phi]} \right| \right\}$$

Proof. In equation (10.2), we have already compared the 1-periodic FOURIER-like partial sum $f(x) := \sum_{k=-N/2}^{N/2-1} \hat{f}_k\, \mathrm{e}^{2\pi \mathrm{i} k x}$ from Problem 10.1 with the structured function s. Due to the choice of the discrete FOURIER coefficients in (10.3), the aliasing error is bounded by

$$\begin{aligned} E_\mathrm{a}(x_j) &= \left| \sum_{\substack{r\in\mathbb{Z}\\ r\neq 0}} \sum_{k=-n/2}^{n/2} \hat{g}_k\, c_{k+nr}[\phi]\, \mathrm{e}^{2\pi \mathrm{i}(k+nr)x_j} \right| \\ &= \left| \sum_{\substack{r\in\mathbb{Z}\\ r\neq 0}} \sum_{k=-N/2}^{N/2} \frac{\hat{f}_k}{N\, c_k[\phi]}\, c_{k+nr}[\phi]\, \mathrm{e}^{2\pi \mathrm{i}(k+nr)x_j} \right| \end{aligned}$$

$$\leq \frac{1}{N} \sum_{k=-N/2}^{N/2} |\hat{f}_k| \sum_{\substack{r\in\mathbb{Z}\\ r\neq 0}} \left| \frac{c_{k+nr}[\phi]}{c_k[\phi]} \right|,$$

where we have used the triangle inequality. Estimating the inner sum by the maximum with respect to $k = -N/2, \dots, N/2 - 1$, we obtain the assertion. □

Theorem 10.7 (Truncation error). *For $\phi \in L^2(\mathbb{T})$, for even $N \in \mathbb{N}$ and $M \in \mathbb{N}$, for the power of two $n = \sigma N$ with $\sigma > 1$, and for $m \in \mathbb{N}$ with $m \ll n$, the truncation error of the NFFT is bounded by*

$$E_t(x_j) \leq \frac{\|\hat{f}\|_1}{nN} \left\{ \max_{k=-N/2,\dots,N/2-1} |c_k[\phi]|^{-1} \right\} \left\{ \left(\sum_{\ell=-n/2}^{\lceil nx_j\rceil - m - 1} + \sum_{\ell=\lfloor nx_j\rfloor + m + 1}^{n/2-1} \right) |\phi(x_j - \ell/n)| \right\}.$$

Proof. Plugging in the definitions of s and $\tilde{s}$, we write the truncation error as

$$E_t(x_j) = \left| \sum_{\ell=-n/2}^{n/2} g_\ell \left(\phi(x_j - \ell/n) - \psi(x_j - \ell/n) \right) \right|,$$

where the coefficients g_ℓ are given by the inverse discrete Fourier transform

$$g_\ell = \frac{1}{n} \sum_{k=-N/2}^{N/2-1} \frac{\hat{f}_k}{N\, c_k[\phi]} \mathrm{e}^{2\pi \mathrm{i} k\ell/n} \qquad (\ell = -N/2, \dots, N/2 - 1).$$

Considering that the structure functions ϕ and ψ coincide on the (1-periodically recurring) interval $[-m/n, m/n] \subset \mathbb{T}$, we only have to sum up the summands with $x_j - \ell/n \notin [-m/n, m/n]$, cf. (10.4). Consequently, the truncation error is given by

$$E_t(x_j) = \frac{1}{nN} \left| \sum_{k=-N/2}^{N/2} \frac{\hat{f}_k\, \mathrm{e}^{2\pi \mathrm{i} k\ell/n}}{c_k[\phi]} \left(\sum_{\ell=-n/2}^{\lceil nx_j\rceil - m - 1} + \sum_{\ell=\lfloor nx_j\rfloor + m + 1}^{n/2-1} \right) \phi(x_j - \ell/n) \right|.$$

Estimating $\hat{f}_k$ upwards by the 1-norm and $c_k[\phi]$ downwards by the minimal Fourier coefficient, we establish the assertion. □

One possibility to construct the 1-periodic structure function ϕ is to take an arbitrary function $\tilde{\phi} \in L^2(\mathbb{R})$ that is small outside the interval $[-m/n, m/n]$ and to define ϕ as *periodization*

$$\phi(x) := \sum_{r\in\mathbb{Z}} \tilde{\phi}(x - r) \qquad (x \in \mathbb{T}).$$

If the function $\tilde{\phi}$ is monotonically decreasing on $(0, \infty)$, then we can estimate the truncation error as follows.

Corollary 10.8 (Truncation error). *For the periodization ϕ of the even function $\tilde{\phi} \in L^2(\mathbb{R})$ monotonically decreasing on $(0, \infty)$, for even $N \in \mathbb{N}$ and $M \in \mathbb{N}$, for the power of two $n = \sigma N$ with $\sigma > 1$, and for $m \in \mathbb{N}$ with $m \ll n$, the truncation error of the NFFT is bounded by*

$$E_t(x_j) \leq \frac{2\,\|\hat{f}\|_1}{nN} \left\{ \max_{k=-N/2,\dots,N/2-1} |c_k[\phi]|^{-1} \right\} \left\{ \tilde{\phi}(m/n) + \int_m^\infty \tilde{\phi}(x/n)\,\mathrm{d}x \right\}.$$

Proof. Using the triangle inequality and the definition of the periodization, we obtain

$$\left(\sum_{\ell=-n/2}^{\lceil nx_j \rceil - m - 1} + \sum_{\ell=\lfloor nx_j \rfloor + m + 1}^{n/2-1} \right) \left| \phi(x_j - \ell/n) \right| \leq \left(\sum_{\ell=-n/2}^{\lceil nx_j \rceil - m - 1} + \sum_{\ell=\lfloor nx_j \rfloor + m + 1}^{n/2-1} \right) \sum_{r \in \mathbb{Z}} \left| \tilde{\phi}(x_j - \ell + nr/n) \right|$$

$$\leq \sum_{\substack{r \in \mathbb{Z} \\ |x_j - r/n| \geq m/n}} \left| \tilde{\phi}(x_j - r/n) \right| \leq 2 \left\{ \tilde{\phi}(m/n) + \int_m^\infty \tilde{\phi}(x/n)\,\mathrm{d}x \right\}. \qquad \square$$

Remark 10.9 (Fourier coefficients). The required Fourier coefficients of the periodization ϕ can be easily computed using the Fourier transform of the function $\tilde{\phi} \in L^2(\mathbb{R})$, which we will study in the following chapter. ○