1.10 trigonometric interpolation and FFT (CSE)

convention: In this chapter, $\mathbf{i} = \sqrt{-1}$ with $\mathbf{i}^2 = -1$, that is, *not* an index. Numbering of indices of vectors starts at 0.

1.10.1 trigonometric interpolation

Definition 1.38 *The polynomials* $p: \mathbb{R} \to \mathbb{C},\ p(x) = \sum_{j=0}^{n-1} c_j e^{\mathbf{i}jx},\ c_j \in \mathbb{C}$ *are called* **trigonometric polynomials** *of* **degree** $n-1$.

goal: given distinct knots $x_j \in [0, 2\pi),\ j = 0, \ldots, n-1$ and values $y_j,\ j = 0, \ldots, n-1$ solve:

$$\text{find trigonometric polyomial } p \text{ of degree } n-1 \text{ s.t.} p(x_j) = y_j, \quad j = 0, \ldots, n-1 \tag{1.33}$$

Remark 1.39 *(i) The trigonometric polynomial* $x \mapsto p(x)$ *is a* 2π*-periodic function. The coefficients* c_j *are its Fourier coefficients.*

(ii) The (continuous) Fourier transform is an important tool in signal processing, e.g., when analyzing audio signals. In the simplest setting, a signal is assumed to be periodic (over a given time interval $(0, T)$*) and writing it as a Fourier series decomposes the signal into different frequency components. These components are then analyzed or modified (e.g., low pass or high pass filters).*

For $T = 2\pi$*, the Fourier series is simply the representation*

$$f(x) = \sum_{j=-\infty}^{\infty} a_j e^{\mathbf{i}xj}, \qquad a_j = \frac{1}{2\pi} \int_0^{2\pi} f(x) e^{-\mathbf{i}xj}\, dx, \tag{1.34}$$

and a_j *are the Fourier coefficients. In order to avoid evaluating the integrals, one could proceed as follows: 1) sample the signal in the points* x_j*; 2) approximate* f *by its trigonometric interpolant* p*; 3) interpret the Fourier coefficients of* p *as (good) approximations to the Fourier coefficients of* f*.*[3]

(iii) For convenience, we work in the present section with the complex exponentials $e^{\mathbf{i}xj}$*. However, the Euler identities*

$$e^{\mathbf{i}x} = \cos x + \mathbf{i} \sin x$$

show $e^{\mathbf{i}xj} = \cos(jx) + \mathbf{i}\sin(jx)$ *so that one could also formulate the problem in terms of "true" trigonometric polynomials involving* $\sin(kx)$ *and* $\cos(kx)$*. Also the FFT discussed below can be formulated for these functions ("discrete cosine transform" and "discrete sine transform").* ∎

[3] The procedure described only suggests how to get the coefficients a_j for $j \geq 0$ since the interpolating polynomial is of the form $p(x) = \sum_{j=0}^{n-1} c_j e^{\mathbf{i}jx}$. It would therefore be more natural to seek the interpolating polynomial of the form $\sum_{j=-(n-1)}^{n-1} c_j e^{\mathbf{i}jx}$. For *real valued* signals, one has $a_{-j} = \overline{a_j}$ so that one is lead to seek the interpolating polynomial of the form $\frac{1}{2}(p(x) + \overline{p(x)})$ for a trigonometric polynomial p. For real-valued f, this p is the solution of (1.33).

Theorem 1.40 *Let $x_j \in [0, 2\pi)$, $j = 0, \ldots, n-1$ be distinct. Then (1.33) is uniquely solvable for each sequence $(y_j)_{j=0}^{n-1} \in \mathbb{C}^n$.*

Proof: Set $z_j := e^{\mathbf{i}xj}$, $j = 0, \ldots, n-1$. Then the z_j are distinct. The ansatz $p(x) = \sum_{j=0}^{n-1} c_i\, e^{\mathbf{i}jx}$ yields the linear system of equations:

$$\underbrace{\begin{pmatrix} z_0^0 & z_0^1 & \ldots & z_0^{n-1} \\ z_1^0 & z_1^1 & \ldots & z_1^{n-1} \\ \vdots & \vdots & & \vdots \\ z_{n-1}^0 & z_{n-1}^1 & \ldots & z_{n-1}^{n-1} \end{pmatrix}}_{=:\widetilde{\mathbf{V}}} \underbrace{\begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \end{pmatrix}}_{=:\mathbf{c}} = \underbrace{\begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{pmatrix}}_{=:\mathbf{y}} \tag{1.35}$$

$\widetilde{\mathbf{V}}$ is a so-called Vandermonde matrix with $\det \widetilde{\mathbf{V}} = \prod_{0 \le j < k \le n-1} (z_k - z_j) \neq 0$ □

In the remainder of the chapter, we consider the uniform knot distribution

$$x_j = \frac{2\pi j}{n}, \quad j = 0, \ldots, n-1. \tag{1.36}$$

It is expedient to introduce the *root*

$$\omega_n := e^{-\frac{2\pi \mathbf{i}}{n}}, \tag{1.37}$$

which satisfies $\omega_n^n = 1$. ⟦ note: $\omega_n^j = e^{-\mathbf{i}x_j}$ ⟧
The matrix $\mathbf{V}$ of (1.35) is easily inverted under the assumption (1.36):

Theorem 1.41 *Assume (1.36) and (1.37). Let $\mathbf{y} := (y_0, \ldots, y_{n-1})^\top \in \mathbb{C}^n$ be given and $p(x) = \sum_{j=0}^{n-1} c_j\, e^{\mathbf{i}jx}$ be the solution to (1.33). Set*

$$\mathbf{V}_n := \left(\omega_n^{j\cdot k}\right)_{j,k=0}^{n-1} \qquad ⟦ \text{ “}\mathbf{DFT\ matrix}\text{”} ⟧ \tag{1.38}$$

Then:

(i) $\frac{1}{n}\mathbf{V}_n\, \mathbf{y} = \mathbf{c}$ ⟦ *i.e.,* $c_k = \frac{1}{n} \sum_{j=0}^{n-1} \omega_n^{j\cdot k} y_j$ ⟧

(ii) $\frac{1}{\sqrt{n}}\mathbf{V}_n$ *symmetric and unitary* $\left(\text{ i.e., } \left(\frac{1}{\sqrt{n}}\mathbf{V}_n\right)^{-1} = \frac{1}{\sqrt{n}}\mathbf{V}_n^H = \frac{1}{\sqrt{n}}\overline{\mathbf{V}_n} \right)$

(iii) $\overline{\mathbf{V}_n} = \left(\overline{\omega_n}^{jk}\right)_{j,k=0}^{n-1} = \left(\omega_n^{-jk}\right)_{j,k=0}^{n-1}$

Proof: ad (iii): ✓
ad (ii): Let v_j, $j = 0, \ldots, n-1$ be the columns of $\frac{1}{\sqrt{n}}\mathbf{V}_n$. Then:

- $v_k^H v_k = \frac{1}{n} \sum_{j=0}^{n-1} \omega_n^{-jk} \omega_n^{jk} = 1$

- $k \neq l$:

$$v_k^H v_l = \frac{1}{n}\sum_{j=0}^{n-1} \omega_n^{-jk}\omega_n^{lj} = \frac{1}{n}\sum_{j=0}^{n-1}\left(\omega_n^{l-k}\right)^j \overset{\text{geometr. series}}{=}$$
$$= \frac{1}{n}\,\frac{1-\left(\omega_n^{l-k}\right)^n}{1-\omega_n^{l-k}} = \frac{1}{n}\,\frac{1-(\omega_n^n)^{l-k}}{1-\omega_n^{l-k}} = 0 \qquad \text{since } \omega_n^n \overset{(1.37)}{=} 1$$

ad(i): For the equidistant points $x_j,\ j = 0,\ldots,n-1$, given by (1.36), the linear system of equations (1.35) has the form $\overline{\mathbf{V}_n}\mathbf{c} = \mathbf{y} \overset{(ii)}{\Rightarrow} \mathbf{c} = \overline{\mathbf{V}_n}^{-1}\mathbf{y} = \frac{1}{\sqrt{n}}\left(\frac{1}{\sqrt{n}}\overline{\mathbf{V}_n}\right)^{-1}\mathbf{y} = \frac{1}{\sqrt{n}}\frac{1}{\sqrt{n}}\mathbf{V}_n\mathbf{y} = \frac{1}{n}\mathbf{V}_n\mathbf{y}$. □

Exercise 1.42 *Show the formula for the geometric series by mulitplying out the right-hand side:*

$$1 - x^{n+1} = (1-x)\sum_{i=0}^{n} x^i, \qquad x \neq 1.$$

■

Definition 1.43 *The linear map*

$$\begin{array}{rccc} \mathcal{F}_n: & \mathbb{C}^n & \to & \mathbb{C}^n \\ & \mathbf{y} = \begin{pmatrix} y_0 \\ \vdots \\ y_{n-1} \end{pmatrix} & \mapsto & \mathbf{V}_n\,\mathbf{y} \end{array}$$

is called the **discrete Fourier transform** (DFT) *of length* n.
The inverse $\mathcal{F}_n^{-1}$ *is called* **IDFT (inverse discrete Fourier transform)**.

Remark 1.44 *Theorem 1.41 yields*

$$\mathcal{F}_n^{-1}\mathbf{y} = \frac{1}{n}\overline{\mathbf{V}_n}\mathbf{y} = \frac{1}{n}\overline{\mathbf{V}_n\overline{\mathbf{y}}} = \frac{1}{n}\overline{\mathcal{F}_n(\overline{\mathbf{y}})} \tag{1.39}$$

1.10.2 FFT

observation: The matrix $\mathbf{V}_n$ is fully populated. A naive realization of the DFT therefore requires $O(n^2)$ arithmetic operations. Exploiting the special structure of $\mathbf{V}_n$ leads to the ***Fast Fourier transform*** (FFT), which only needs $O(n\log n)$ arithmetic operations.

Lemma 1.45 *Let* $n = 2m,\ \omega = e^{\pm\frac{2\pi\mathbf{i}}{n}}$. *Let* $(y_0,\ldots,y_{n-1}) \in \mathbb{C}^n$. *Then the terms*

$$\alpha_k := \sum_{j=0}^{n-1} y_j\,\omega^{kj} \qquad k = 0,\ldots,n-1$$

with $\xi := \omega^2$ *and* $l = 0, \ldots, m-1$ *can be computed as follows:*

$$\begin{aligned}
\alpha_{2l} &= \sum_{j=0}^{m-1} g_j \, \xi^{jl} \qquad \textit{with} \quad g_j := y_j + y_{j+m} \\
\alpha_{2l+1} &= \sum_{j=0}^{m-1} h_j \xi^{jl} \qquad \textit{with} \quad h_j := (y_j - y_{j+m})\, \omega^j \qquad j = 0, \ldots, m-1
\end{aligned}$$

Proof: Since $\omega^{nl} = 1$ we get

$$\begin{aligned}
\alpha_{2l} &= \sum_{j=0}^{n-1} y_j \, \omega^{2lj} = \sum_{j=0}^{\frac{n}{2}-1} y_j \, \omega^{2lj} + y_{j+\frac{n}{2}} \, \omega^{2l(j+\frac{n}{2})} = \\
&= \sum_{j=0}^{\frac{n}{2}-1} \omega^{2lj} (y_j + y_{j+\frac{n}{2}} \, \omega^{ln}) = \sum_{j=0}^{\frac{n}{2}-1} \omega^{2lj} (y_j + y_{j+\frac{n}{2}})
\end{aligned}$$

Since $\omega^{\frac{n}{2}} = -1$ we have:

$$\begin{aligned}
\alpha_{2l+1} &= \sum_{j=0}^{n-1} y_j \, \omega^{(2l+1)j} = \\
&= \sum_{j=0}^{\frac{n}{2}-1} \omega^{(2l+1)j} y_j + y_{j+\frac{n}{2}} \, \omega^{(2l+1)(j+\frac{n}{2})} = \\
&= \sum_{j=0}^{\frac{n}{2}-1} \omega^{2lj} \left(y_j \, \omega^j + y_{j+\frac{n}{2}} \cdot \omega^{ln} \omega^j \omega^{\frac{n}{2}} \right) = \\
&= \sum_{j=0}^{\frac{n}{2}-1} (y_j - y_{j+\frac{n}{2}}) \, \omega^j \omega^{2lj}
\end{aligned}$$

□

Lemma 1.45 shows that the computation of $\hat{\mathbf{y}} = (\hat{y}_0, \ldots, \hat{y}_{n-1})^\top := \mathcal{F}_n(\mathbf{y})$ can be reduced to the computation of $\mathcal{F}_{\frac{n}{2}}(\mathbf{g})$ and $\mathcal{F}_{\frac{n}{2}}(\mathbf{h})$.
With $n = 2m$ we have

$$\begin{aligned}
(\hat{y}_0, \hat{y}_2, \ldots, \hat{y}_{n-2})^\top &= \mathcal{F}_m(\mathbf{g}) \quad , \quad \mathbf{g} = (y_j + y_{j+m})_{j=0}^{m-1} \\
(\hat{y}_1, \hat{y}_3, \ldots, \hat{y}_{n-1})^\top &= \mathcal{F}_m(\mathbf{h}) \quad , \quad \mathbf{h} = ((y_j - y_{j+m})\, \omega_n^j)_{j=0}^{m-1}
\end{aligned}$$

This yields

Algorithm 1.46 (FFT)
Input: $n = 2^p, \; p \in \mathbb{N}_0, \; \mathbf{y} = (y_0, \ldots, y_{n-1})^\top \in \mathbb{C}^n$
Output: $\hat{\mathbf{y}} = (\hat{y}_0, \ldots, \hat{y}_{n-1}) = \mathcal{F}_n(\mathbf{y})$

if $n = 1$ ***then***

$\hat{y}_0 := y_0$
else
$\omega := e^{\frac{-2\pi \mathbf{i}}{n}}$
$m := \frac{n}{2}$
$(g_j)_{j=0}^{m-1} := (y_j + y_{j+m})_{j=0}^{m-1}$
$(h_j)_{j=0}^{m-1} := ((y_j - y_{j+m})\,\omega^j\,)_{j=0}^{m-1}$
$(\hat{y}_0, \hat{y}_2, \ldots, \hat{y}_{n-2}) := FFT(m, \mathbf{g})$
$(\hat{y}_1, \hat{y}_3, \ldots, \hat{y}_{n-1}) := FFT(m, \mathbf{h})$
end if
return $(\hat{\mathbf{y}})$

Similarly for the Inverse Fourier Transform: $(\check{y}_0, \ldots, \check{y}_{n-1}) := \mathcal{F}_n^{-1}(\mathbf{y})$ (cf. first equation in (1.39)):

$$(\check{y}_0, \check{y}_2, \ldots, \check{y}_{n-2})^\top = \tfrac{1}{2}\mathcal{F}_{\frac{n}{2}}^{-1}(\mathbf{g}) \quad , \quad \mathbf{g} = (y_j + y_{j+m})_{j=0}^{m-1}$$
$$(\check{y}_1, \check{y}_3, \ldots, \check{y}_{n-1})^\top = \tfrac{1}{2}\mathcal{F}_{\frac{n}{2}}^{-1}(\mathbf{h}) \quad , \quad \mathbf{h} = \big((y_j - y_{j+m})\overline{\omega_n}^j\big)_{j=0}^{m-1}$$

Algorithm 1.47 (IFFT)
Input: $n = 2^p,\ p \in \mathbb{N}_0,\ \mathbf{y} = (y_0, \ldots, y_{n-1})^\top \in \mathbb{C}^n$
Output: $\check{\mathbf{y}} = \mathcal{F}_n^{-1}(\mathbf{y})$

if $n = 1$ ***then***
$\check{y}_0 := y_0$
else
$\omega := e^{\frac{2\pi \mathbf{i}}{n}}$
$m := \frac{n}{2}$
$(g_j)_{j=0}^{m-1} := \frac{1}{2}\,(y_j + y_{j+m})_{j=0}^{m-1}$
$(h_j)_{j=0}^{m-1} := \frac{1}{2}\,((y_j - y_{j+m})\,\omega^j\,)_{j=0}^{m-1}$
$(\check{y}_0, \check{y}_2, \ldots, \check{y}_{n-2}) := IFFT(m, \mathbf{g})$
$(\check{y}_1, \check{y}_3, \ldots, \check{y}_{n-1}) := IFFT(m, \mathbf{h})$
end if
return $(\hat{\mathbf{y}})$

Cost of the FFT: Denote by $A(n)$ the cost of the call of $FFT(n, \mathbf{y})$ and let $n = 2^p,\ p \in \mathbb{N}_0$. Then:

$$A(n) \le 2A(n/2) + \underbrace{C}_{\text{computation of } g,\ h} n \tag{1.40}$$

and thus:

$$\begin{aligned}
A(n) &\overset{(1.40)}{\leq} 2\,A\left(\frac{n}{2}\right) + C\,n \;=\\
&= 2\,A\left(2^{p-1}\right) + C\,2^p \overset{(1.40)}{\leq}\\
&\overset{(1.40)}{\leq} 2\left(2\,A\left(2^{p-2}\right) + C\,2^{p-1}\right) + C\,2^p \;=\\
&= 2^2 A\left(2^{p-2}\right) + 2\,C\,2^p \overset{(1.40)}{\leq}\\
&\overset{(1.40)}{\leq} 2^2\left(2A\left(2^{p-3}\right) + C\,2^{p-2}\right) + 2\,C\,2^p \;=\\
&= 2^3 A\left(2^{p-3}\right) + 3\,C\,2^p \;\leq\; \ldots \;\leq\\
&\leq 2^p\,A\left(2^0\right) + p\,C\,2^p \;=\\
&= n\,A(1) + (\log_2 n)\,C\,n \;\leq\\
&\leq n \cdot \log_2 n \cdot C' \qquad \text{mit} \quad C' = C + A(1)
\end{aligned}$$

slide 9b

1.10.3 Properties of the DFT

The DFT appears very prominently when one is trying to compute efficiently the *convolution* of two sequences, which is defined in the following definition.

Definition 1.48 *(i) A sequence* $\mathfrak{f} = (f_j)_{j\in\mathbb{Z}}$ *is called* n**-periodic**, *if* $f_{j+n} = f_j \;\; \forall j \in \mathbb{Z}$. $\mathbb{C}^n_{per}$ *denotes the space of the* n*-periodic sequences.*

(ii) The DFT $\mathcal{F}_n$ *is defined by:*

$$\begin{aligned}
\mathcal{F}_n : \quad \mathbb{C}^n_{per} &\to \mathbb{C}^n_{per}\\
(f_j)_{j\in\mathbb{Z}} &\mapsto \left(\sum_{j=0}^{n-1} \omega_n^{jk} f_j\right)_{k\in\mathbb{Z}}
\end{aligned}$$

Since $\omega_n^n = 1$ *the DFT* $\mathcal{F}_n$ *is well-defined;* ⟦ *i.e.,* $\mathcal{F}_n((f_j)_{j\in\mathbb{Z}})$ *is again an* n*-periodic sequence* ⟧

(iii) the convolution $*$ *is defined by:*

$$\begin{aligned}
* : \; \mathbb{C}^n_{per} \times \mathbb{C}^n_{per} &\to \mathbb{C}^n_{per}\\
(\mathfrak{f}, \mathfrak{g}) &\mapsto (\mathfrak{f} * \mathfrak{g})_k \;:=\; \left(\sum_{j=0}^{n-1} f_{k-j}\, g_j\right) \qquad \forall k \in \mathbb{Z}
\end{aligned}$$

(iv) the pointwise multiplication $\cdot$ *is defined by:*

$$\begin{aligned}
\cdot : \; \mathbb{C}^n_{per} \times \mathbb{C}^n_{per} &\to \mathbb{C}^n_{per}\\
(\mathfrak{f}, \mathfrak{g}) &\mapsto (\mathfrak{f} \cdot \mathfrak{g})_k \;:=\; f_k \cdot g_k \qquad \forall k \in \mathbb{Z}
\end{aligned}$$

Remark 1.49 *The DFT of Def. 1.43 coincides with the definition of the DFT of Def. 1.48, if one extends the finite sequence* $(f_j)_{j=0}^{n-1}$ *n-periodically.* ∎

Theorem 1.50 *For* $\mathfrak{f}$, $\mathfrak{g} \in \mathbb{C}_{per}^n$ *let* $\hat{\mathfrak{f}} := \mathcal{F}_n(\mathfrak{f}), \hat{\mathfrak{g}} := \mathcal{F}_n(\mathfrak{g})$ *be the Fourier transformations. Then:*

(i) $\mathcal{F}_n : \ \mathbb{C}_{per}^n \to \mathbb{C}_{per}^n$ *is linear.*

(ii) $\mathcal{F}_n^{-1}(\mathfrak{f}) \ = \ \frac{1}{n}\left(\sum_{j=0}^{n-1} \overline{\omega_n}^{jk} f_j\right)_{k\in\mathbb{Z}}$

(iii) **(convolution theorem)**

$$\widehat{\mathfrak{f} * \mathfrak{g}} \ = \ \mathcal{F}_n(\mathfrak{f} * \mathfrak{g}) \ = \ \hat{\mathfrak{f}} \cdot \hat{\mathfrak{g}}$$

Proof: Exercise □

1.10.4 application: fast convolution of sequence

Example 1.51 *Let* $\mathfrak{f}$, $\mathfrak{g} \in \mathbb{C}_{per}^n$. *The naive evaluation of the convolution* $\mathfrak{h} := \mathfrak{f} * \mathfrak{g}$ *costs* $O(n^2)$ *operations. It is more efficient to proceed with Theorem 1.50:*

1.) compute $\hat{\mathfrak{f}}$ *and* $\hat{\mathfrak{g}}$ *using FFT*	*cost:* $O(n \log n)$
2.) compute $\hat{\mathfrak{h}} := \hat{\mathfrak{f}} \cdot \hat{\mathfrak{g}}$	*cost:* $O(n)$
3.) compute $\mathfrak{h} = \mathcal{F}_n^{-1}\left(\hat{\mathfrak{h}}\right)$ *using IFFT*	*cost:* $O(n \log n)$

The convolution of finite (non-periodic) sequences is defined slightly differently, namely, for two finite sequences $(f_j)_{j=0}^{N-1}$, $(g_j)_{j=0}^{N-1}$, its convolution is given by the sequence $(c_j)_{j=0}^{N-1}$ with entries

$$c_j = \sum_{k=0}^{j} f_{j-k} g_k. \tag{1.41}$$

The sequence $(c_j)_{j=0}^{N-1}$ can also be computed with the aid of the FFT:

Example 1.52 *let* $(f_j)_{j=0}^{N-1}, (g_j)_{j=0}^{N-1}$ *be finite sequences.*

<u>*goal:*</u> *compute* $(h_j)_{j=0}^{N-1}$ *given by* $h_j = \sum_{k=0}^{j} f_{j-k}\, g_k, \qquad j = 0, \ldots, N-1$

<u>*idea:*</u> *periodize the two sequences* $(f_j)_{j=0}^{N-1}$ *and* $(g_j)_{j=0}^{N-1}$, *so that Example 1.51 is applicable.*
<u>*Procedure:*</u> *Choose an* $n \geq 2N$ *of the form* $n = 2^p$ *for a* $p \in \mathbb{N}_0$ *and define* $\mathfrak{f}'$, $\mathfrak{g}' \in \mathbb{C}_{per}^n$ *by*

$$f_j' := \begin{cases} f_j & \text{for } \ j = 0, \ldots, N-1 \\ 0 & \text{for } \ j = N, \ldots, n-1 \end{cases}$$

$$g_j' := \begin{cases} g_j & \text{for } \ j = 0, \ldots, N-1 \\ 0 & \text{for } \ j = N, \ldots, n-1 \end{cases}$$

Then:

$$f'_j = 0 \quad \text{for } N-n \le j \le -1 \tag{1.42}$$
$$g'_j = 0 \quad \text{for } N \le j \le n-1 \tag{1.43}$$

For $k \in \{0,\dots,N-1\}$ we have:

$$(\mathfrak{f}' * \mathfrak{g}')_k = \sum_{j=0}^{n-1} f'_{k-j}\, g'_j \overset{(1.43)}{=} \sum_{j=0}^{N-1} f'_{k-j}\, g_j = \sum_{j=0}^{k} \underbrace{f'_{k-j}}_{=f_{k-j}}\, g_j + \sum_{j=k+1}^{N-1} \underbrace{f'_{k-j}}_{=0 \text{ by (1.42)}}\, g_j = \sum_{j=0}^{k} f_{k-j}\, g_j \qquad \blacksquare$$

The convolution of non-periodic sequence arises, for example, when polynomials are multiplied.

Example 1.53 *Let polynomials $\pi(x) = \sum_{j=0}^{N-1} f_j x^j$ and $\pi_2(x) = \sum_{j=0}^{N-1} g_j x^j$ of degree $N-1$ be given. Then the product $\pi_1\pi_2$ is a polynomial of degree $2N-2$ given by*

$$\pi_1(x)\pi_2(x) = \sum_{j=0}^{2(N-1)} h_j x^j, \qquad c_j = \sum_{k=0}^{j} f_{j-k} g_k,$$

where we implicitly assume that $f_k = g_k = 0$ for $k \in \{N,\dots,2N-2\}$. Hence, Example 1.52 is applicable. ∎

An application that exemplifies the use of the FFT in connection with the computation of the convolution of sequences is the multiplication of very large numbers.

Example 1.54 (multiplication of numbers with many digits) *The fast realization of the multiplication of numbers with many digits is nowadays done by FFT[4]. Consider the multiplication of two integers with n digits that are written as*

$$x = \sum_{j=0}^{n} f_j b^j, \qquad y = \sum_{j=0}^{n} g_j b^j,$$

where $b \in \mathbb{N}$ (e.g., $b = 10$) and the coefficients ("digits") satisfy $f_j,\ g_j \in \{0,\dots,b-1\}$. We seek the representation of $z = xy$ in the form $z = \sum_{j=0}^{2n} c_j b^j$ with $c_j \in \{0,\dots,b-1\}$. This is very similar to Example 1.53, and a formal multiplication yields

$$xy = \sum_{j=0}^{2n} h_j b^j, \qquad h_j = \sum_{k=0}^{j} f_{j-k} g_k,$$

where we again assumed that $f_j = 0 = g_j$ for $j \in \{n+1,\dots,2n\}$. The sequence $(h_j)_j$ can be calculated with cost $O(n \log n)$ using the FFT as described in Example 1.52. The sought coefficients $(c_j)_j$ of z are obtained from the sequence $(h_j)_j$ by one more sweep through the sequence with cost $O(n)$ that ensures that the coefficients c_j satisfy $c_j \in \{0,\dots,b-1\}$. The following loop overwrites the h_j with the sought c_j:

for $j = 0 : 2n$ ***do***

[4]This is also a building block of arbitrary precision arithmetic

```
    if h_j ≥ b then                                   ▷ carrying over is necessary
        h_j := h_j − ⌊h_j/b⌋b
        h_{j+1} := h_{j+1} + ⌊h_j/b⌋
    end if
end for
```

■

Example 1.55 (solving linear systems with circulant matrices) *A matrix* $\mathbf{C} \in \mathbb{C}^{n\times n}$ *is called* circulant*, if it has the form*

$$\mathbf{C} = \begin{pmatrix} c_0 & c_{n-1} & \cdots & c_2 & c_1 \\ c_1 & c_0 & c_{n-1} & & c_2 \\ \vdots & c_1 & c_0 & \ddots & \vdots \\ c_{n-2} & & \ddots & \ddots & c_{n-1} \\ c_{n-1} & c_{n-2} & \cdots & c_1 & c_0 \end{pmatrix}.$$

Introduce the vector $\mathbf{c} := (c_0, \ldots, c_{n-1})^T$*. Observe that the matrix-vector product* $\mathbf{Cx}$ *is a convolution, i.e., the entries* $\mathbf{y}_j$ *of the vector* $\mathbf{y} = \mathbf{Cx}$ *are given by*

$$\mathbf{y}_j = \sum_{k=0}^{n-1} \mathbf{c}_{j-k}\mathbf{x}_k,$$

where we view the sequence $(c_j)_{j=0}^{n-1}$ *as an element of* $\mathbb{C}^n_{per}$ *(i.e., extend the sequence* $(c_j)_{j=0}^{n-1}$ *periodically). That is,*

$$(\mathbf{Cx})_j = (\mathbf{c} * \mathbf{x})_j, \qquad j = 0, \ldots, n-1.$$

Hence, given $\mathbf{b} \in \mathbb{C}^n$*, the linear system of equations* $\mathbf{Cx} = \mathbf{b}$ *can also be written as*

$$\mathbf{c} * \mathbf{x} = \mathbf{b}. \tag{1.44}$$

Solving for $\mathbf{x}$ *can be achieved with the FFT. To that end, write* $\widehat{\mathbf{c}} = \mathcal{F}_n(\mathbf{c})$*,* $\widehat{\mathbf{x}} = \mathcal{F}_n(\mathbf{x})$*,* $\widehat{\mathbf{b}} = \mathcal{F}_n(\mathbf{b})$ *and observe:*

1. *Applying DFT on both sides of (1.44) gives by the convolution theorem* $\widehat{c}_j\widehat{x}_j = \widehat{b}_j$*,* $j = 0, \ldots, n-1$*.*
2. *Hence,* $\widehat{x}_j = \widehat{b}_j/\widehat{c}_j$*.*
3. *an inverse DFT of* $\widehat{\mathbf{x}} = (\widehat{x}_j)_{j=0}^{n-1}$ *gives* $\mathbf{x}$*.*

Hence, the work to solve $\mathbf{Cx} = \mathbf{x}$ *is 2 FFTs of length* n *and* n *divisions.*
Remark: *a much more common type matrices that can be treated with similar techniques are* → Toeplitz matrices.

■