as rapid summation of the two-dimensional discrete Fourier series (see Section 5.7.2) can then be performed similarly.

We begin with recasting expression (5.102) in the complex form. To do so, we artificially extend the grid function \( v_m \) antisymmetrically with respect to the point \( m = M \): \( v_{M + j} = -v_{M - j}, \quad j = 1, 2, \ldots, M, \) so that for \( m = M + 1, M + 2, \ldots, 2M \) it becomes: \( v_m = -v_{2M - m} \). Then we can write:

\[
c_k = \frac{h}{i \sqrt{2}} \sum_{m=0}^{2M} v_m e^{i k \pi m / M},
\]

(5.116)

because given the antisymmetric extension of \( v_m \) beyond \( m = M \), we clearly have \( \sum_{m=0}^{2M} v_m \cos \frac{k \pi m}{M} = 0 \). Hereafter, we will analyze formula (5.116) instead of (5.102).

First of all, it is clear that a straightforward implementation of the summation formula (5.116) requires \( O(N) \) arithmetic operations per one coefficient \( c_k \) even if all the exponential factors \( e^{i 2k \pi n / N} \) are precomputed and available. Consequently, the computation of all \( c_k \), \( k = 0, 1, 2, \ldots, N - 1 \), requires \( O(N^2) \) arithmetic operations. Let, however, \( N = N_1 \cdot N_2 \), where \( N_1 \) and \( N_2 \) are positive integers. Then we can represent the numbers \( k \) and \( n \) as follows:

\[
k = k_1 N_2 + k_2, \quad k_1 = 0, 1, \ldots, N_1 - 1, \quad k_2 = 0, 1, \ldots, N_2 - 1,
\]

\[
n = n_1 + n_2 N_1, \quad n_1 = 0, 1, \ldots, N_1 - 1, \quad n_2 = 0, 1, \ldots, N_2 - 1.
\]

Consequently, by noticing that \( e^{i 2\pi (k_1 N_2 N_1) / N} = e^{i 2\pi (k_1 n_2)} = 1 \), we obtain from expression (5.116):

\[
i c_k = \frac{h}{\sqrt{2}} \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} v_{n_1 + n_2 N_1} e^{i 2\pi (k_1 N_2 N_1) / N} = \frac{h}{\sqrt{2}} \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} v_{n_1 + n_2 N_1} e^{i 2\pi (k_1 N_2 N_1) / N} \]

\[
= \frac{h}{\sqrt{2}} \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} v_{n_1 + n_2 N_1} e^{i 2\pi (k_1 N_2 N_1) / N} \]

\[
= \frac{h}{\sqrt{2}} \sum_{n_1=0}^{N_1-1} \left( \sum_{n_2=0}^{N_2-1} v_{n_1 + n_2 N_1} e^{i 2\pi (k_1 N_2 N_1) / N} \right) e^{i 2\pi (k_1 n_1) / N}.
\]

Therefore, the sums (5.116) for \( k = 0, 1, 2, \ldots, N - 1 \) can be computed in two stages. First, we evaluate the intermediate quantities:

\[
\bar{v}_{n_1, k_2} = \sum_{n_2=0}^{N_2-1} v_{n_1 + n_2 N_1} e^{i 2\pi (k_2 N_2 N_1) / N}
\]

(5.117a)
for \( n_1 = 0, 1, \ldots, N_1 - 1 \) and \( k_2 = 0, 1, \ldots, N_2 - 1 \), and then obtain
\[
c_k = \frac{\hbar}{i\sqrt{2}} \sum_{n_1=0}^{N_1-1} c_{n_1} e^{2\pi i (k_2/2)}
\]
for \( k = 0, 1, 2, \ldots, N-1 \). The key advantage of computing the coefficients \( c_k \) consecutively by formulae (5.117a)–(5.117b) rather than directly by formula (5.116) is that the cost of calculating \( N_1 \cdot N_2 = N \) intermediate quantities \( v_{n_1,k_2} \) by means of formula (5.117a) is \( \mathcal{O}(N \cdot N_2) \) arithmetic operations, and the cost of subsequently calculating \( N \) coefficients \( c_k \) with the help of formula (5.117b) is \( \mathcal{O}(N \cdot N_1) \) arithmetic operations, which altogether yields \( \mathcal{O}(N \cdot (N_1 + N_2)) \) operations. At the same time, the cost of computing all \( c_k, k = 0, 1, 2, \ldots, N-1 \), by formula (5.116) is \( \mathcal{O}(N^2) \) operations. For example, if \( N \) is a large even number and \( N_1 = 2 \), then one obtains a speed-up by roughly a factor of two.

Assume now that \( N_1 \) is a prime number, whereas \( N_2 \) is a composite number. Then \( N_2 \) can be represented as a product of two factors and a similar transformation can be applied to the sum (5.117a), for which \( n_1 \) is simply a parameter. This will further reduce the computational cost. In general, if \( N = N_1 \cdot N_2 \cdots N_p \), then instead of the \( \mathcal{O}(N^2) \) complexity of the original summation we will obtain an algorithm for computing \( c_k, k = 0, 1, 2, \ldots, N-1 \), at a cost of \( \mathcal{O}(N \cdot (N_1 + N_2 + \cdots + N_p)) \) arithmetic operations. This algorithm is known as the fast Fourier transform (FFT).

Efficient versions of the FFT have been developed for \( N_i = 2, 3, 4 \), although the algorithm can also be built for other prime factors. In practice, the most commonly used and the most convenient to implement version is the one for \( N = 2^p \), where \( p \) is a positive integer. The computational complexity of the FFT in this case is
\[
\mathcal{O}(N(2 + 2 + \ldots + 2)) = \mathcal{O}(N \ln N)
\]

arithmetic operations,

The algorithm of fast Fourier transform was introduced in a landmark 1965 paper by Cooley and Tukey [CT65], and has since become one of the most popular computational tools for solving large linear systems obtained as discretizations of differential equations on the grid. Besides that, the FFT finds many applications in signal processing and in statistics. Practically all numerical software libraries today have FFT as a part of their standard implementation.

Exercises

1. Consider an inhomogeneous Dirichlet problem for the finite-difference Poisson equation on a grid square [cf. formula (5.11)]:
\[
\frac{u_{m_1+1,m_2} - 2u_{m_1,m_2} + u_{m_1-1,m_2}}{h^2} + \frac{u_{m_1,m_2+1} - 2u_{m_1,m_2} + u_{m_1,m_2-1}}{h^2} = f_{m_1,m_2},
\]
\[
m_1 = 1, 2, \ldots, M - 1, \quad m_2 = 1, 2, \ldots, M - 1,
\]
(5.118a)