

Examinee number: 11070  
Name: Gábor Ádám Fehér

## Answer to Theme A

From the fundamental concepts related to mathematical informatics, we present the **Cooley–Tukey algorithm**. The algorithm allows us to compute the discrete Fourier transform of a  $n$  long sequence in  $\mathcal{O}(n \log n)$  time. Algorithm with such properties are called fast Fourier transform (FFT) algorithms. Among the many variants of the Cooley–Tukey algorithm, the **radix-2 DIT** is the simplest and most common, and thus we present it in great detail. Other forms of the algorithm, as well as other types of FFT algorithms are also mentioned, but no rigorous construction is given.

### Mathematical overview

**Definition 1** (Discrete Fourier transform). The discrete Fourier transform (DFT) over the  $n$  dimensional complex field is an invertible linear transformation  $\mathcal{F} : \mathbb{C}^n \rightarrow \mathbb{C}^n$ . The output of the function for  $(x_k)_{0 \leq k \leq n-1}$  is defined by

$$X_k = (\mathcal{F}(x))_k = \sum_{j=0}^{n-1} x_j e^{-\frac{2\pi i k j}{n}} = \sum_{j=0}^{n-1} x_j \left[ \cos\left(\frac{2\pi i k j}{n}\right) - i \sin\left(\frac{2\pi i k j}{n}\right) \right], \quad (1)$$

where the right side of the equation is due to Euler's formula and the trigonometric properties of sine and cosine.

We should mention, that the sign of the exponential is sometimes taken as positive. The resulting equation is equal to the inverse of the previously defined Fourier transformation multiplied by the constant  $n$ , that is

$$x_k = (\mathcal{F}^{-1}(X))_k = \frac{1}{n} \sum_{j=0}^{n-1} X_j e^{\frac{2\pi i k j}{n}}. \quad (2)$$

With some minor adjustments the Cooley–Tukey algorithm is capable of computing the inverse-DFT of the  $n$  long sequence, so no matter which convention we use, the algorithm remains useful.

We introduce some notation. Given  $x = (x_k)_{0 \leq k \leq 2n-1}$ , we define  $X^{[0]}$  to be the DFT of the even-indexed terms, while  $X^{[1]}$  to be the DFT of the odd-indexed terms in the sequence. That is

$$X_k^{[0]} = \left(X^{[0]}\right)_k = (\mathcal{F}(x_0, x_2, \dots, x_{2n-2}))_k = \sum_{j=0}^{n-1} x_{(2j)} e^{-\frac{2\pi i k j}{n}} = \sum_{j=0}^{n-1} x_{(2j)} e^{-\frac{4\pi i k j}{2n}} \quad (3)$$

$$X_k^{[1]} = \left(X^{[1]}\right)_k = (\mathcal{F}(x_1, x_3, \dots, x_{2n-1}))_k = \sum_{j=0}^{n-1} x_{(2j+1)} e^{-\frac{2\pi i k j}{n}} = \sum_{j=0}^{n-1} x_{(2j+1)} e^{-\frac{4\pi i k j}{2n}}. \quad (4)$$

The key observation to make, in order to verify the validity of the algorithm is

$$X_k = \sum_{j=0}^{2n-1} x_j e^{-\frac{2\pi i k j}{2n}} = \sum_{j=0}^{n-1} x_{(2j)} e^{-\frac{4\pi i k j}{2n}} + e^{-\frac{2\pi i k}{2n}} \sum_{j=0}^{n-1} x_{(2j+1)} e^{-\frac{4\pi i k j}{2n}}. \quad (5)$$

Thus for  $0 \leq l \leq n-1$  we have

$$X_l = X_l^{[0]} + e^{-\frac{2\pi i k}{2n}} X_l^{[1]} \quad (6)$$

$$X_{n+l} = X_l^{[0]} - e^{-\frac{2\pi i k}{2n}} X_l^{[1]}, \quad (7)$$

since

$$\sum_{j=0}^{n-1} x_{(2j)} e^{-\frac{4\pi i (n+l) j}{2n}} = \sum_{j=0}^{n-1} x_{(2j)} e^{-\frac{4\pi i l j}{2n}}, \text{ and } e^{-\frac{2\pi i (n+l)}{2n}} = -e^{-\frac{2\pi i l}{2n}}. \quad (8)$$

### Interpretation

#### The algorithm

This basis of all variants of the Cooley–Tukey algorithm is the divide-and-conquer technique. In the radix-2 case the size of the input has to be a power of two. We may add padding zeros to fit the size constraint.

#### Importance and Usage