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 one, 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 $\mathcal{F} : \mathbb{C}^n \to \mathbb{C}^n$. The output of the function for $(x_k)_{0 \leq k \leq n-1}$ is defined by

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

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

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

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 notations. Given $x = (x_k)_{0 \leq 2n-1}$, 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

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

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

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

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

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

$$X_l = X_l^{[0]} + e^{-\frac{2\pi i k}{2n}} X_l^{[1]}, \text{ and } X_{n+l} = X_l^{[0]} - e^{-\frac{2\pi i k}{2n}} X_l^{[1]}, \tag{6}$$

since

$$\sum_{j=0}^{n-1} x_{(2j)} e^{-\frac{4\pi i (n+l) j}{2n}} = \sum_{j=0}^{n-1} x_{(2j)} e^{-\frac{4\pi i l j}{2n}}, \text{ and } e^{-\frac{2\pi i (n+l)}{2n}} = -e^{-\frac{2\pi i l}{2n}}. \tag{7}$$

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. From this point on we assume that input vectors are of the right size. Given an $x_{0 \leq k \leq n}$, where $2 \leq n$, equations 3 and 4 indicate that the input vector should be divided into two parts: the even-indexed and the odd-indexed terms. Once the DFTs of the split vector are calculated, via recursive calls, the DFT of the input vector can be calculated by using equation 6. If the input vector is one dimensional, its DFT is itself. The following Python code is a working implementation of the radix-2 case, illustrating the key ideas used, but due to its performance it is not meant to be used in real world applications.

```
"""A working fft implementation"""
from sympy import exp, pi, I

def fft(terms):
    """Implementation of Cooley-Tukey FFT algorithm"""
    input_length = len(terms)
    if input_length == 1:
        return terms
    minus_nth_root_of_unity = exp(-2*pi*I/input_length)
    running_root = 1
    even_indexed_terms = [terms[i] for i in range(0, input_length, 2)]
    odd_indexed_terms = [terms[i] for i in range(1, input_length, 2)]
    even_fft = fft(even_indexed_terms)
    odd_fft = fft(odd_indexed_terms)
    left_fft, right_fft = [], []
    for k in range(0, int(input_length/2)):
        sub_expr = running_root * odd_fft[k]
        left_fft.append(even_fft[k] + sub_expr)
        right_fft.append(even_fft[k] - sub_expr)
        running_root *= minus_nth_root_of_unity
    return left_fft + right_fft
```

To evaluate the time complexity of the algorithm, apart from the recursive calls in line 13 and 14, the algorithm runs in $\Theta(n)$ time, where n is the length of the input. Using the recurrence relation

$$T(n) = 2T(n/2) + \Theta(n) = \Theta(n \log n). \tag{8}$$

With little modification, the Cooley–Tukey algorithm can compute the inverse-DFT of a transformed vector. Similarly to equation 5, we can derive

$$x_k = \sum_{j=0}^{2n-1} X_j e^{\frac{2\pi i k j}{2n}} = \sum_{j=0}^{n-1} X_{(2j)} e^{\frac{4\pi i k j}{2n}} + e^{\frac{2\pi i k}{2n}} \sum_{j=0}^{n-1} X_{(2j+1)} e^{\frac{4\pi i k j}{2n}} \tag{9}$$

so in the 9th line, instead of $e^{-2\pi i/N}$ we have $e^{2\pi i/N}$ for the N long input, and by multiplying each term of the end result by $1/n$ given the initial input had size of n, we obtain the original x, within the same time complexity.

Importance and Usage

The FFT is widely used among different fields of engineering, computer science and mathematics.

Many popular compression methods, such as `jpeg` or `webm`, use discrete cosine transforms (DCT). Algorithms that compute DCTs in $\mathcal{O}(n \log n)$ are known as fast cosine transform (FCT) algorithms. Specialized FCT algorithms exist, that directly compute a vectors DCT, and tough the theoretical running time of such algorithms are better, on modern hardware computing DCTs via FFTs with some $\mathcal{O}(n)$ pre- and post-processing steps are often faster.

When two degree-bound n polynomials are represented in the a "extended" point-value form, over $2n$ distinct points, multiplication can be done in $\mathcal{O}(n)$ time, by simply multiplying the coefficients. Representing such a polynomial over the nth root of unity is the same as having the coefficient vectors transformed. Thus multiplying two polynomials can be done in $\mathcal{O}(n \log n)$ time by using FFTs: first transforming the polynomials to their "extended" point value form in $\mathcal{O}(n \log n)$ time, multiplying them in $\mathcal{O}(n)$, and then transforming back the result to coefficient form. Polynomial representation of the Toeplitz matrices also allow us to multiply them within the same time complexity.

FFts are also widely used in theoretical results. For a long time the **Schönhage–Strassen algorithm** was the fastest known algorithm for multiplying two large integers. With running time $\mathcal{O}(n \log n \log \log n)$, it is asymptotically faster than other many traditional multiplication methods, like for example the **Karatsuba algorithm**, but it only starts to outperform them for numbers with 10000 to 40000 digits, thus it is rarely used in practice. The algorithm employs modular arithmetic instead of the roots of unity (Fourier transforms can be performed in any algebric ring). In 2019 Harvey and van der Hoeven published a $\mathcal{O}(n \log n)$ algorithm, which also builds upon on FFTs. However this is a galactic algorithm, so it only bears theoretical importance.

Answer to Theme B

When studying stochastic processes, the Markov property is a commonly made assumption. This may limit its applicability, as many real life processes have memory. Excited random walks are a class of non-Markovian stochastic processes that generalize other well understood processes. As a novel field of mathematics, results described here will be theoretical in nature, but we also describe some problems and parallels, where models based on ERWs may be helpful. Indeed many established results may be viewed from the perspective of ERWs. Here we are concerned with ERWs on the d dimensional integer lattice, but note that construction of ERWs can be extended to other graphs or continuous time-space processes. As rigorous introduction to the subject is somewhat tedious, we only describe the intuitive meaning behind the ERWs.

Excited random walks

ERWs can be thought of as a generalization of simple random walks (biased or symmetric) or random walks in random environments. On each site of the d dimensional integer lattice, an infinite stack of *cookies* are placed. A cookie is a $2d$ dimensional vector, that describes a probability distribution, meaning its terms are non-negative and sum up to 1. The walker is placed on the node x, and *consumes* the cookie on top, and moves according to the probability distribution described by the cookie. We call a specific configuration of cookies a *cookie-environment*, meaning that on any site of the integer lattice the cookie of a specific location is well defined. We note the set of all cooke-environment with Ω, while a specific cookie environment is usually noted with ω. As per definition, $\omega \in \Omega$.The probability of moving towards the vector e at position z upon the nth arrival is denoted with $\omega(z, e, n)$. That is

$$P_{x,\omega}[X_0 = x] = 1 \tag{10}$$

$$P_{x,\omega}[X_{n+1} = X_n + e \mid (X_i)_{0\le i\le n}] = \omega(X_n, e, \#\{i \in \{0, 1, \dots, n\} \mid X_i = X_n\}), \tag{11}$$

which is called the *quenched* measure. It is easy to see how certain cookie-environment can be equated to the previously given random walks, by putting cookies of the same kind on top of each other on each site. We may assume some probability distribution $\mathbb{P}$ on the cookie-environments itself. This further generalizes the model. If we want to investigate a specific environment $\omega \in \Omega$, we may condition that $\mathbb{P}[\omega] = 1$, but also enables us to create convenient assumptions scenarios, such as the cookie stacks being independent of each other. We also define the *averaged* measure as $P_x[\cdot]$ as $\mathbb{E}[P_{x,w}[\cdot]]$.

Restrictions on ERWs

As ERWs describe very broad settings for random walks to take place in, we often make convenient assumptions on $\mathbb{P}$. We list them here, and refer back to them when the results are stated.

Condition 1 (IID)**.** This condition holds, if the family of cookie stacks $(\omega(z, \cdot, \cdot))_{z\in\mathbb{Z}^d}$ are independent of each other and identically distributed under $\mathbb{P}$. This condition is often weakened however as we can see in the next condition.

Condition 2 (SE)**.** This condition holds, if the family of cookie stacks $(\omega(z, \cdot, \cdot))_{z\in\mathbb{Z}^d}$ is stationary and ergodic with respect to shifts on $\mathbb{Z}^d$ under $\mathbb{P}$. In this context, ergodicity means, that $\forall \mathcal{A} \in \mathcal{F}$ that is invariant under shifts on $\mathbb{Z}^d$ satisfies $\mathbb{P}[A] \in \{0, 1\}$, where $\mathcal{F}$ is the associated sigma algebra.

Condition 3 (WEL)**.** This condition holds, if for any z, e exists an i such that the equation $\omega(z, e, i) > 0$ holds almost surely under $\mathbb{P}$. This ellipticity condition, along with the next two are often used to escape degenerate situation. We introduce two stricter condition.

Condition 4 (EL)**.** This condition holds, if for any z, e and i the equation $\omega(z, e, i) > 0$ holds almost surely under $\mathbb{P}$.

Condition 5 (UEL)**.** This condition holds, if there is a $\kappa > 0$ such that for all z, e and i the equation $\omega(z, e, i) \le \kappa$ holds almost surely under $\mathbb{P}$.

Condition 6 (BC_F)**.** Given $F \subseteq \mathcal{M}$, the condition holds if $\mathbb{P}\left[\forall e \in F : \sum_{i\ge 1} \omega(0, e, i) < \infty\right] = 0$.

Results concerning the general case

Some of the most elementary properties of ERWs have been proven with minimal assumptions on $\mathbb{P}$. The first such result is connected to the range of the ERWs.

Theorem 2 (The range of one dimensional ERWs)**.** *Assume condition 2 and 4 holds. Let X be a random walk on the ERW model. Assume that 6 also holds for either $F = \{-1\}$ or for $F = \{1\}$ but not for both. Then $\lim_{n\to\infty} X_n \to e\cdot\infty$ P_0 almost surely, where $e \in F$. If 6 holds for $F = \{-1, 1\}$, then the range of the random walk is P_0 almost surely infinite. On the other hand, assume that 6 does not hold for any $e \subseteq \mathcal{M}$. Then the range of the random walk is P_0 almost surely finite.*

A similar theorem can be formulated for the multiplying-dimensional case, by strengthening condition 2 to 1.

Theorem 3. *Assume condition 1 and 4 holds. Let X be a random walk walk in the ERW model. If there is an orthogonal set $F \subset \mathcal{M}$, such that condition 6 holds, then then range of the ERW P_0 almost surely is infinite. On the contrary, if no such F exists, than the range is P_0 almost surely finite.*

The next couple theorems are results regarding transience and recurrence. ERWs are not Markov chains, so the standard definition of recurrence and transience do not apply, however the definitions are analogous of the standard meaning in terms of how many times a walker reaches certain z. As with the range of the ERWs, results concerning the one- and the multi-dimensional ERWs will be discussed separately.

Theorem 4. *Assume 2 and 4. Then the ERW is P_0 almost surely recurrent or transient or has finite range. This is not the same result as 2, as in the case when condition 6 holds for $\{-1, 1\}$, we only claimed that the range is infinite, not that every $z \in \mathbb{Z}$ is visited infinitely often.*

In the subsequent sections we turn our attention to some of the more well established models of ERWs.

The original ERW

The original ERW model was introduced by Benjamini and Wilson in . The model is constructed on $\mathbb{Z}^d$. The walker gets biased at the first arrival to a site, in a direction $e \in \mathcal{E}$, while on subsequent visits to that site the walker jumps to a uniformly chosen neighbor. This model can be generalized by fixing some $\ell \in \mathbb{R}^d$. In , a $\mathbb{P}$ is given, such that

$$\exists \lambda \in \mathbb{R}^+ : \sum_{e \in \mathcal{E}} \omega(0, e, 1) e \cdot \ell \geq \lambda \quad \mathbb{P}\text{-a.s. and} \tag{12}$$

$$\omega(0, e, i) = \omega(0, -e, i) \quad \mathbb{P}\text{-a.s. for all } e \in \mathcal{E}, i \leq 2. \tag{13}$$

Any number of ℓ-positive cookies per site

This model further generalizes the previous one, by lifting the restriction on the number of non-placebo cookies, still requiring every cookie to induce a drift in the same $\ell \in \mathbb{Z}^d \setminus \{0\}$ direction. In the one-dimensional case, assumption 2 is assumed, while in the multi-dimensional case assumptions 1 and 5 are assumed. Then

$$\sum_{e \in \mathcal{E}} \omega(0, e, i) e \cdot \ell \geq 0 \quad \mathbb{P}\text{-a.s. for all } i \in \mathbb{N}. \tag{14}$$

Cookies satisfying the above equation are called ℓ-positive. Negativity is defined by the opposite inequality.

Boundedly many positive or negative cookies per site

It is probably the most widely studied model of ERWs. It was introduced in . We will assume condition 1 and 3 apply, and there is fixed bound after which there are only placebo cookies in the each cookie stack. The model in one dimension is fairly well understood, as a connection with branching processes can be made.

RW perturbated at extrema

This random process is one of the first studied non-Markovian random walk. This model was originally not considered in this setting, however can be viewed as an ERW.

Research plan

Answer to Theme C

Mathematical informatics can help immensely to prepare for and to handle infectious diseases. Though the tools of bioinformatics are useful to investigate properties of the pathogens and to develop vaccines against it, in this section we choose to focus on other methods to combat epidemics. According to Ming et al, epidemic containment efforts have four phases: **preparedness**, **outbreak investigation**, **response** and **evaluation**. These phases contain a wide range of tasks, such as inventory management for essential medical supply, surveillance system design, the scheduling of vehicles used to transport medical supply and identification of potential bottlenecks.

The SEIQRS model

Compartment models are widely used in modeling the spread of infectious diseases. In case of an outbreak, each member of the population is assigned with a label depending on his/her relation with the infectious agent. Individuals with the same label form a compartment. As the epidemic progresses, individuals can also progress between compartments. During the recent COVID-19 outbreak people were divided into 5 groups: susceptible people (S), people during incubation period (E), infectious people (I), quarantined people (Q) and recovered people (R). The name of this model is the **SEIQRS model**. The underlying social structure may taken to be homogenous, but we use small-world networks in our model. Even tough stochastic frameworks have been introduced, we use the more common deterministic approach of utilizing differential equations to mimic the movement between different compartments. The following figure shows the arrangement of the compartments and how people can move between them.

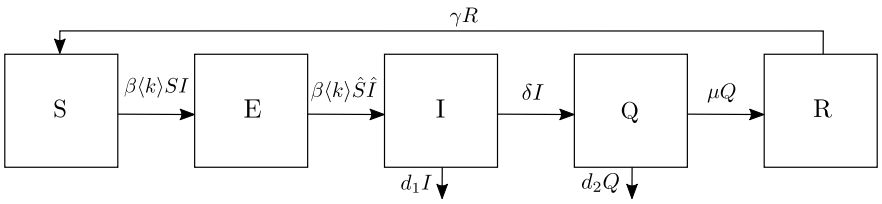

Figure 1: The SEIQRS model

The number of susceptible people at time t is $S(t)$, and the number of people in each compartments is denoted analogously. On the above figure, $\langle k\rangle$ is the average degree distribution of the underlying small-world network, $\hat{S}$ and $\hat{I}$ abbreviate $S(t-\tau)$ and $I(t-\tau)$ respectively, where τ is the incubation period. The speed with which susceptible people progress to other compartments is

$$\frac{\mathrm{d}S}{\mathrm{d}t} = -\beta\langle k\rangle S(t)I(t) + \gamma R(t). \tag{15}$$

Looking at figure 1 the other differential equations can be derived easily. As obtaining the analytical solution for the compartment model remains difficult, we have to rely on computational simulations to investigate further properties of the model. The following figure was created using reasonable input parameters with `jitcdde`, `numpy` and `matplotlib`.

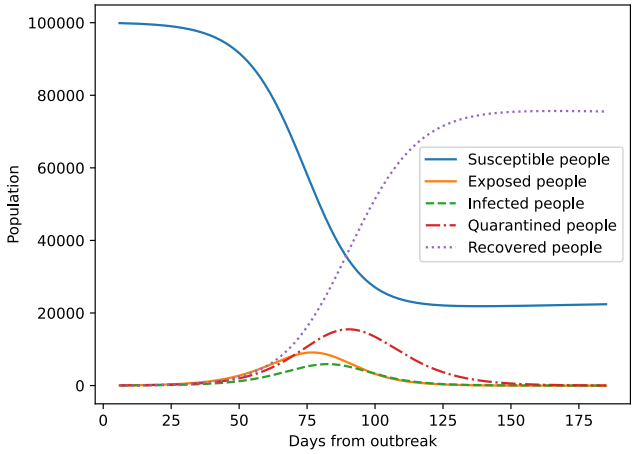

Figure 2: The SEIQRS model

As the capacity of healthcare facilities is limited, during an actual outbreak, the most important task decision-makers have to deal with is to keep the number of infected people to the minimum. This is called flattening the curve. As policies aimed to flatten the curve may only affect parameters $\langle k \rangle$ and δ directly, we investigate their role in relation to the number of people getting infected. We perform sensitivity analysis on a number of possible values for each, leaving all the other parameters fixed.

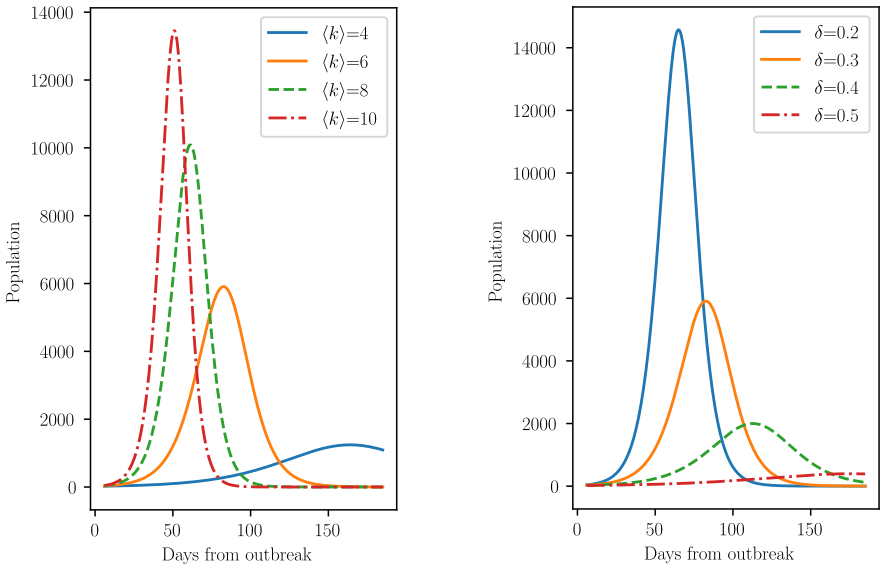

Figure 3: Sensitivity analysis with regards to $\langle k \rangle$ and δ

These figures seem to justify the decisions made by authorities that implemented strict measures during the COVID-19 pandemic. Declaring curfew and promoting self-isolation are some of the most efficient ways to minimize $\langle k \rangle$. In Wuhan, following the outbreak, authorities were quick to declare curfew which lead to the sharp decline of new cases. Many other countries have also called for social-distancing which is aimed to decrease the interactions between people. More efficient quarantining of individuals who came into contact with the virus also helps immensely. This can be achieved via increasing testing capacities and the quality of testing-kits. Following the COVID-19 outbreak, South Korea was quick to organize large scale testings and has provided drive-through testing facilities to minimize social contacts. These measures resulted in a sharp decline of new COVID-19 cases on the peninsula. On the other hand, countries that failed or were late to put protective measures in place have seen a steep upsurge of newly infected people. The United States Government was particularly slow to act and as a result the country has the most cases of COVID-19 infections. The US President had repeatedly downplayed the significance of the virus and had had advocated for slowing down testings.

Other tasks and conclusion

Once an epidemic outbreak has been confirmed, having protective measures in place is just one of the many issues public officials face. Another important challenge is to ensure that health-care facilities are provided the appropriate amount of medical supply. Solutions to such problems usually use techniques from operations research. As it turns out, the epidemic diffusion model can help us predict the emergency demand of health-care facilities. A specific such facility has demand at time t denoted by d_t, and we formulate it as

$$d_t = \langle k \rangle I(t) + Q(t) \tag{16}$$

where I and Q refer to the number of infected and quarantined people in the epidemic area. To supply demand, multiple distribution models have been proposed such as the point-to-point distribution model and the multi-depots multiple travelling salesmen model. These two are models considered as pure models, and optimize replenishment efforts according timeliness and cost respectively. In practice, due to their pure nature, they might not be applicable. Other hybrid models that try to balance between cost and timeliness of delivery have also been proposed. Other arising logistical problems are also connected to the diffusion model.