

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное бюджетное образовательное  
учреждение высшего образования  
**АДЫГЕЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

Инженерно-физический факультет  
Кафедра автоматизированных систем обработки информации и управления

Отчёт по практике  
Дискретное преобразование Фурье.

2 курс, группа 2ИВТ

Выполнил:

\_\_\_\_\_ К. В. Рябенко  
«\_\_» \_\_\_\_\_ 2025 г.

Руководитель:

\_\_\_\_\_ С. В. Теплоухов  
«\_\_» \_\_\_\_\_ 2025 г.

Майкоп, 2025 г.

## Оглавление

Теория.....	3
Ход выполнения работы.....	5
1. Подключение библиотек.....	5
2. Размер сигнала.....	5
3. Функция <code>dft()</code> .....	6
4. Основная функция <code>main()</code> .....	7
4.2 Входной сигнал.....	7
4.3 Результат <code>dft</code> .....	7
4.4 Вычисление <code>dft</code> .....	7
4.5 Вывод решения.....	8
Код программы.....	8
Пример работы программы.....	9

## Теория

Рассмотрим алгоритм, который позволяет перемножить два полинома длиной  $n$  за время  $O(n \log n)$ , что значительно лучше времени  $O(n^2)$ , достигаемого тривиальным алгоритмом умножения. Очевидно, что умножение двух длинных чисел можно свести к умножению полиномов, поэтому два длинных числа также можно перемножить за время  $O(n \log n)$ .

### Дискретное преобразование Фурье (ДПФ)

Пусть имеется многочлен  $n$ -ой степени:

$$A(x) = a_0x^0 + a_1x^1 + \dots + a_{n-1}x^{n-1}.$$

Не теряя общности, можно считать, что  $n$  является степенью 2. Если в действительности  $n$  не является степенью 2, то мы просто добавим недостающие коэффициенты, положив их равными нулю.

Из теории функций комплексного переменного известно, что комплексных корней  $n$ -ой степени из единицы существует ровно  $n$ . Обозначим эти корни через

$w_{n,k}, k = 0 \dots n-1$ , тогда известно, что  $w_{n,k} = e^{i\frac{2\pi k}{n}}$ . Кроме того, один из этих

корней  $w_n = w_{n,1} = e^{i\frac{2\pi}{n}}$  (называемый главным значением корня  $n$ -ой степени из

единицы) таков, что все остальные корни являются его степенями:  $w_{n,k} = (w_n)^k$ .

Тогда **дискретным преобразованием Фурье (ДПФ)** (discrete Fourier transform, DFT) многочлена  $A(x)$  (или, что то же самое, ДПФ вектора его коэффициентов  $(a_0, a_1, \dots, a_{n-1})$ ) называются значения этого многочлена в точках  $x = w_{n,k}$ , т.е. это вектор:

$$\text{DFT}(a_0, a_1, \dots, a_{n-1}) = (y_0, y_1, \dots, y_{n-1}) = (A(w_{n,0}), A(w_{n,1}), \dots, A(w_{n,n-1})) = (A(w_n^0), A(w_n^1), \dots, A(w_n^{n-1})).$$

Аналогично определяется и **обратное дискретное преобразование Фурье** (InverseDFT). Обратное ДПФ для вектора значений многочлена  $(y_0, y_1, \dots, y_{n-1})$  — это вектор коэффициентов многочлена  $(a_0, a_1, \dots, a_{n-1})$ :

$$\text{InverseDFT}(y_0, y_1, \dots, y_{n-1}) = (a_0, a_1, \dots, a_{n-1}).$$

Таким образом, если прямое ДПФ переходит от коэффициентов многочлена к его значениям в комплексных корнях  $n$ -ой степени из единицы, то обратное ДПФ — наоборот, по значениям многочлена восстанавливает коэффициенты многочлена.

### Применение ДПФ для быстрого умножения полиномов

Пусть даны два многочлена  $A$  и  $B$ . Посчитаем ДПФ для каждого из них:  $\text{DFT}(A)$  и  $\text{DFT}(B)$  — это два вектора-значения многочленов.

Теперь, что происходит при умножении многочленов? Очевидно, в каждой точке их значения просто перемножаются, т.е.

$$(A \times B)(x) = A(x) \times B(x).$$

Но это означает, что если мы перемножим вектора  $\text{DFT}(A)$  и  $\text{DFT}(B)$ , просто умножив каждый элемент одного вектора на соответствующий ему элемент другого вектора, то мы получим не что иное, как ДПФ от многочлена  $A \times B$ :

$$\text{DFT}(A \times B) = \text{DFT}(A) \times \text{DFT}(B).$$

Наконец, применяя обратное ДПФ, получаем:

$$A \times B = \text{InverseDFT}(\text{DFT}(A) \times \text{DFT}(B)),$$

где, повторимся, справа под произведением двух ДПФ понимается попарные произведения элементов векторов. Такое произведение, очевидно, требует для вычисления только  $O(n)$  операций. Таким образом, если мы научимся вычислять ДПФ и обратное ДПФ за время  $O(n \log n)$ , то и произведение двух полиномов (а, следовательно, и двух длинных чисел) мы сможем найти за ту же асимптотику.

## 1. Подключение библиотек

```
#include <iostream>
#include <cmath>
#include <complex>
```

Рисунок 1.

`<iostream>` — позволяет выводить результаты в консоль (`cout`) или читать входные данные (`cin`).

`<cmath>` — содержит математические функции (`cos`, `sin`) и константу `M_PI` (число  $\pi \approx 3.14159$ ).

`<complex>` — добавляет поддержку комплексных чисел вида  $a + bi$ , где  $a$  — действительная часть,  $b$  — мнимая.

## 2. Размер сигнала

```
const int N = 4;
```

Рисунок 2

❶ **Назначение:** задаёт размер входного сигнала для Дискретного Преобразования Фурье

## 3. Функция `dft()`

```

void dft(double input[], complex<double> output[]) {
    for (int k = 0; k < N; k++) {
        output[k] = complex<double>(0, 0); // Обнуляем выход
        for (int n = 0; n < N; n++) {
            double angle = -2 * M_PI * k * n / N;
            // e^(i*angle) = cos(angle) + i*sin(angle)
            output[k] += input[n] * complex<double>(cos(angle), sin(angle));
        }
    }
}

```

Рисунок 3

### 1. Входные параметры:

- ⑩ **input[]** — массив вещественных чисел (сигнал).
- ⑩ **output[]** — массив комплексных чисел (результат DFT).
- ⑩ **Возвращаемое значение: void** (результат записывается в **output**).

### 2. Внешний цикл по частотам (k)

- ⑩ **k** — индекс частоты (от 0 до **N-1**).
- ⑩ Каждая компонента **output[k]** инициализируется нулём (**0 + 0i**).

### 3. Внутренний цикл по времени (n)

**n** — индекс отсчёта сигнала.

- ⑩ **angle** вычисляет фазу для текущей частоты **k** и времени **n**

**Комплексная экспонента:**

Реализуется как **complex<double>(cos(angle), sin(angle))**.

- ⑩ **Накопление результата:**

- ⑩ Каждый отсчёт **input[n]** умножается на комплексную экспоненту и добавляется к **output[k]**.

### 4. Основная функция main( )

```
int main() {
    double input[N] = {1, 0, -1, 0};
    complex<double> output[N];
    dft(input, output);
    cout << "DFT result:" << endl;
    for (int k = 0; k < N; k++) {
        cout << "X[" << k << "] = " << output[k].real() << " + " << output[k].imag() << "i" << endl;
    }
    return 0;
}
```

Рисунок 4

## 4.2 Входной сигнал

```
double input[N] = {1, 0, -1, 0};
```

Рисунок 5

**input[N]** — тестовый сигнал (можно заменить на свои данные).

## 4.3 Результат dft

```
complex<double> output[N];
```

Рисунок 6

**output[N]** — массив для хранения результатов DFT (комплексные числа).

## 4.4 Вычисление dft

```
dft(input, output);
```

Рисунок 7

**dft(input, output)** — запуск расчёта.

## 4.5 Вывод решения

```

    cout << "DFT result:" << endl;
    for (int k = 0; k < N; k++) {
        cout << "X[" << k << "] = " << output[k].real() << " + " << output[k].imag() << "i" << endl;
    }
    return 0;
}

```

Рисунок 8

**Вывод** в формате:

$X[k] = \text{Re} + \text{Im } i$ , где:

- ⑩ **Re** — действительная часть,
- ⑩ **Im** — мнимая часть.

**Код программы:**

```
#include <iostream>
```

```
#include <cmath>
```

```
#include <complex>
```

```
using namespace std;
```

```
const int N = 4;
```

```
void dft(double input[], complex<double> output[]) {
```

```
    for (int k = 0; k < N; k++) {
```

```
        output[k] = complex<double>(0, 0);
```

```
        for (int n = 0; n < N; n++) {
```

```
            double angle = -2 * M_PI * k * n / N;
```

```
            output[k] += input[n] * complex<double>(cos(angle), sin(angle));
```

```
        }
```

```
    }
```

```
}
```

```
int main() {
```

```
    double input[N] = { 1, 0, -1, 0};
```

```
    complex<double> output[N];
```

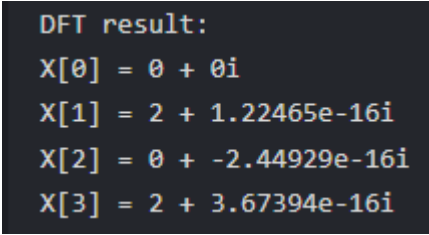
```
    dft(input, output);
```

```
    cout << "DFT result:" << endl;
```



```
for (int k = 0; k < N; k++) {  
    cout << "X[" << k << "] = " << output[k].real() << " + " << output[k].imag() << "i" << endl;  
}  
return 0;  
}
```

### Пример работы программы

A screenshot of a terminal window showing the output of a program. The text is as follows:  
DFT result:  
X[0] = 0 + 0i  
X[1] = 2 + 1.22465e-16i  
X[2] = 0 + -2.44929e-16i  
X[3] = 2 + 3.67394e-16i  
The background of the terminal is dark, and the text is light-colored.

```
DFT result:  
X[0] = 0 + 0i  
X[1] = 2 + 1.22465e-16i  
X[2] = 0 + -2.44929e-16i  
X[3] = 2 + 3.67394e-16i
```

Рисунок 9