Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №6
Дискретное преобразование Фурье

по курсу «Вычислительной математики»

Выполнил:
Студент группы P3230
Пономаренко Алиса Валерьевна

Преподаватель:
Перл Ольга Вячеславовна

Санкт-Петербург
2024

Описание численного метода

Дискретное преобразование Фурье

Дискретное Преобразование Фурье (DFT) является основным инструментом в цифровой обработке сигналов. Оно позволяет преобразовать дискретный сигнал из временной области в частотную. DFT применяется для анализа частотного содержания сигналов, фильтрации и сжатия данных.

DFT преобразует последовательность $x[n]$ длиной N в последовательность $X[k]$, представляющую частотное содержание сигнала. Формула для вычисления DFT определяется как:

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j\frac{2\pi}{N}kn}, \quad k = 0, 1, \ldots, N-1$$

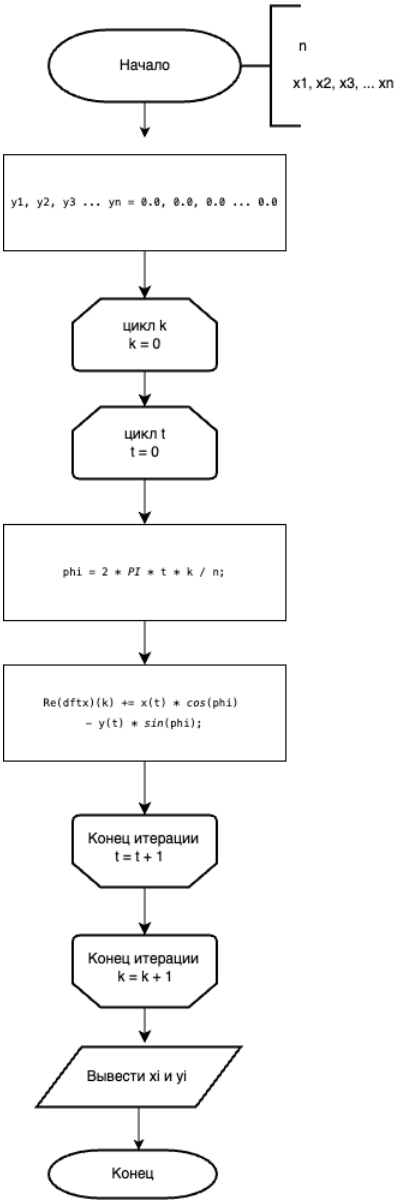
где:

- $x[n]$ — входная последовательность длиной N,
- $X[k]$ — спектральные компоненты,
- j — мнимая единица.

Реализация метода

DFT можно реализовать с использованием вложенных циклов, которые выполняют суммирование для каждой частоты k.

Блок-схема по составленному описанию метода



Код численного метода

```
11

      1 usage   new *
12    public static void printDFT(List<Double> x, int n) {
13        double[][] dftResult = computeDFT(x, n);
14        for (int i = 0; i < n; i++) {
15            System.out.printf(Locale.US,  format: "%.8f %.8f\n", dftResult[0][i], dftResult[1][i]);
16        }
17    }
18

      1 usage   new *
19 @  public static double[][] computeDFT(List<Double> x, int n) {
20        double[] real = new double[n];
21        double[] imag = new double[n];
22
23        for (int i = 0; i < n; i++) {
24            real[i] = x.get(i);
25            imag[i] = 0.0;
26        }
27
28        double[] dftReal = new double[n];
29        double[] dftImag = new double[n];
30
31        for (int k = 0; k < n; k++) {
32            for (int t = 0; t < n; t++) {
33                double angle = 2 * Math.PI * t * k / n;
34                dftReal[k] += real[t] * Math.cos(angle) - imag[t] * Math.sin(angle);
35                dftImag[k] += real[t] * Math.sin(angle) + imag[t] * Math.cos(angle);
36            }
37        }
38
39        return new double[][]{dftReal, dftImag};
40    }
```

Примеры работы программы

1. Входные данные: Случай с нулевыми значениями

4
0.0 0.0 0.0 0.0

Выходные данные:

0.00000000 0.00000000
0.00000000 0.00000000
0.00000000 0.00000000
0.00000000 0.00000000

2. Входные данные: Случай с одинаковыми значениями

4
1.0 1.0 1.0 1.0

Выходные данные:

4.00000000 0.00000000
0.00000000 0.00000000
0.00000000 0.00000000
0.00000000 0.00000000

3. Входные данные: Случай с чередующимися значениями на первой и третьей гармониках

4
1.0 -1.0 1.0 -1.0

Выходные данные:

0.00000000 0.00000000
0.00000000 0.00000000
4.00000000 0.00000000
0.00000000 0.00000000

4. Входные данные: Случай с чередующимися значениями на второй и четвертой гармониках.

4
0.0 1.0 0.0 -1.0

Выходные данные:

0.00000000 0.00000000
0.00000000 -2.00000000
0.00000000 0.00000000
0.00000000 2.00000000

5. Входные данные: Случай с другим количеством данных

8

1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0

Выходные данные:
36.00000000 0.00000000
-4.00000000 9.65685425
-4.00000000 4.00000000
-4.00000000 1.65685425
-4.00000000 0.00000000
-4.00000000 -1.65685425
-4.00000000 -4.00000000
-4.00000000 -9.65685425

Выводы

1. Первый тест показывает, что если все входные значения равны нулю, то DFT также будет состоять из нулевых значений. Это подтверждает, что метод корректно обрабатывает нулевые входные данные и не вводит никаких артефактов в результате.

2. Второй тест показывает, что при постоянном сигнале все частоты, кроме нулевой, равны нулю. Это логично, поскольку входной сигнал не имеет колебаний.

3. Третий тест показывает, что входной сигнал имеет компоненты на первой и третьей гармониках (периодические колебания). Положительные и отрицательные значения чередуются, создавая высокую частоту.

4. Четвертый тест показывает, что входной сигнал имеет компоненты на второй и четвертой гармониках. Это ожидаемо, поскольку сигнал имеет колебания с удвоенной частотой по сравнению с тестом 3.

5. Пятый тест показывает результаты для большего набора данных. Данные содержат линейно возрастающий сигнал. Как видно из результата, на первой гармонике нет реальной части, что соответствует линейному тренду. Мнимые части на других гармониках показывают наличие высокочастотных компонентов в сигнале.

Реализованный метод успешно вычисляет дискретное преобразование Фурье для различных входных данных. Тесты подтверждают, что метод корректно определяет гармонические компоненты входных сигналов.
Метод демонстрирует ожидаемое поведение для различных типов входных данных, таких как постоянный сигнал, чередующиеся значения, линейно возрастающий сигнал и нулевой сигнал. Вычисленные значения соответствуют теоретическим ожиданиям, что подтверждает правильность реализации метода.

Сравнение с другими методами

Быстрое преобразование Фурье (FFT)

- **Применимость**: FFT является оптимизированной версией DFT и используется для больших наборов данных, поскольку значительно сокращает вычислительное время. Оно применяется в цифровой обработке сигналов, анализе данных и везде, где требуется частотный анализ больших наборов данных.
- **Сложность**: Алгоритмическая сложность FFT составляет $O(N \log N)$, где N — количество входных данных. В отличие от DFT, сложность которого $O(N^2)$, FFT намного эффективнее для больших данных.
- **Точность**: FFT и DFT дают одинаковые результаты, поскольку FFT — это оптимизация DFT. Отличие только в скорости вычислений.

Быстрое косинусное преобразование (DCT)

- **Применимость**: DCT используется в обработке сигналов и изображений, особенно в сжатии данных (например, JPEG). Применимо к вещественным данным и используется для анализа сигналов, где необходимо учитывать только действительную часть.
- **Сложность**: Как и FFT, DCT имеет сложность $O(N \log N)$.
- **Точность**: DCT концентрирует информацию в меньшем числе коэффициентов, что делает его более эффективным для сжатия данных. Результаты отличаются от DFT, так как DCT используется для других целей и не работает с комплексными числами.

Гармонический анализ с использованием волновых преобразований (Wavelet Transform)

- **Применимость**: Волновые преобразования используются для анализа временных рядов с временной локализацией частотных компонент. Применяются в сжатии изображений, денойзинге сигналов и анализе временных рядов.
- **Сложность**: Алгоритмическая сложность варьируется в зависимости от реализации, но часто близка к $O(N \log N)$.
- **Точность**: Волновые преобразования более точны в определении локализованных частотных компонент по сравнению с DFT и FFT, что делает их полезными для анализа данных с временной изменчивостью частотного содержания.

Анализ применимости метода

Применимость DFT:

- **Преимущества**: Простота реализации и понимания. Полезен для обучения и для небольших наборов данных.
- **Недостатки**: Высокая вычислительная сложность ($O(N^2)$), что делает его непригодным для больших наборов данных.

Анализ алгоритмической сложности метода

Алгоритмическая сложность DFT: $O(N^2)$, где N — количество входных данных. Это связано с двойным циклом по всем элементам входного набора данных для вычисления каждого выходного элемента.

Анализ численной ошибки самого численного метода

Численные ошибки в DFT:

- **Округление**: Операции с плавающей запятой подвержены округлениям, которые могут накапливаться, особенно при больших значениях N.
- **Погрешности вычислений**: Из-за многократного использования тригонометрических функций (cos, sin) могут возникать ошибки округления, которые могут искажать результаты, особенно для высокочастотных компонент.