

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ
Н.Г.ЧЕРНЫШЕВСКОГО»

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ ПОДГОТОВКЕ ПО ДИСЦИПЛИНЕ «ДВОИЧНЫЙ ГАРМОНИЧЕСКИЙ АНАЛИЗ»

фамилия, имя, отчество

инициалы, фамилия

Саратов 2021

СОДЕРЖАНИЕ

Стр.

1	Построение матрицы Уолша и вычисление коэффициентов Фурье-Уолша.....	3
2	Реализация алгоритма.....	6
ПРИЛОЖЕНИЕ А Листинг: построение матрицы Уолша в представлении Пэли и вычисление коэффициентов Фурье- Уолша		
	A.1 BinaryHarmonicAnalysis.cs	7
	A.2 Helpers.cs.....	8

1 Построение матрицы Уолша и вычисление коэффициентов Фурье-Уолша

Введем несколько вспомогательных определений и теорем.

Определение 1 (система Радемахера на $[0, 1)$). Для $t \in [0, 1)$ положим

$$r_0(t) = \begin{cases} 1, & t \in [0, \frac{1}{2}), \\ -1, & t \in [\frac{1}{2}, 1). \end{cases} \quad (1.1)$$

Продолжим $r_0(t)$ периодически на $[0, +\infty)$ с периодом 1, т.е. $r_0(m+t) = r_0(t)$ при $m \in \mathbb{N}$ и $0 \leq t < 1$. Если $k \in \mathbb{N}$, то положим $r_k(t) = r_0(2^k t)$. Очевидно, что при $k = 0$ получим $r_0(t)$.

Определение 2. Если $n \in \mathbb{N}$ имеет двоичное разложение

$$n = 2^{n_1} + 2^{n_2} + \dots + 2^{n_s} \quad (n_1 > n_2 > \dots > n_s \geq 0) \quad (1.2)$$

то положим по определению

$$w_n(t) = r_{n_1}(t)r_{n_2}(t) \dots r_{n_s}(t), \quad w_0(t) \equiv 1. \quad (1.3)$$

Замечание 1. Двоичное разложение (1.2) можно задать в виде

$$n = \sum_{j=0}^{\infty} \varepsilon_j \cdot 2^j, \quad \varepsilon_j = \{0, 1\}, \quad (1.4)$$

причем в сумме (1.4) конечное число слагаемых. Тогда $w_n(t) = \prod_{j=0}^{\infty} r_j^{\varepsilon_j}(t)$.

Теорема 1. Если f - ступенчатая функция, постоянная на двоичных полуинтервалах ранга N ($f(t) = \lambda_j$ на $\Delta_j^{(N)}$), то f есть многочлен по системе Уолша

$$f(t) = \sum_{k=0}^{2^N-1} c_k w_k(t). \quad (1.5)$$

Теорема 2. Система Уолша – ортонормированная система.

Т.к. система Уолша ортонормированная система, то $\int_0^1 w_k(t)w_l(t)dt = \delta_{k,l}$. Умножим обе части равенства (1.5) на $w_l(t)$ и проинтегрируем.

$$\begin{aligned} c_l &= \int_0^1 w_l(t)f(t)dt = \sum_{j=0}^{2^N-1} \int_{\Delta_j^{(N)}} f(t)w_l(t)dt = \sum_{j=0}^{2^N-1} w_l\left(\frac{j}{2^N}\right) \cdot \lambda_j \frac{1}{2^N} = \\ &= \frac{1}{2^N} \sum_{j=0}^{2^N-1} \lambda_j w_l\left(\frac{j}{2^N}\right) = \frac{1}{2^N} \sum_{j=0}^{2^N-1} \lambda_j w_l(\Delta_j^{(N)}). \end{aligned} \quad (1.6)$$

Для вычисления коэффициентов c_l , нужно найти $w_{l,j}$, т.е. нужно получить матрицу Уолша размерности $2^N - 1 \times 2^N - 1$.

Существует несколько способов формирования. Рассмотрим один из них, наиболее наглядный: матрица Адамара, H , может быть сформирована рекурсивным методом с помощью построения блочных матриц по следующей общей формуле:

$$H_{2^N} = \begin{bmatrix} H_{2^{N-1}} & H_{2^{N-1}} \\ H_{2^{N-1}} & -H_{2^{N-1}} \end{bmatrix}, \quad \text{где } H_1 = [1]. \quad (1.7)$$

Построим матрицу Адамара H_4 , размерности $2^2 \times 2^2$:

$$H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}; \quad W_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}. \quad (1.8)$$

Каждая строка матрицы Адамара и является функцией Уолша.

В данном случае функции будут упорядочены по Адамару. Для того чтобы получить представление Уолша(W_{2^N}), нужно сделать перестановку в матрице Адамара: из номера функции по Адамару путем перестановки битов в двоичной записи номера в обратном порядке с последующим преобразованием результата из кода Грея.

Номер по Адамару	Двоичная форма	Перестановка бит	Преобразование из кода Грея	Номер по Уолшу
0	00	00	00	0
1	01	10	11	3
2	10	01	01	1
3	11	11	10	2

Построим теперь матрицу Уолша размерности в представлении Пэли:

$$\begin{aligned}
w_0 &= (1, 1, \dots, 1); \\
w_1 &= w_0 \cdot r_0; \\
w_2 &= w_0 \cdot r_1; \\
w_3 &= w_1 \cdot r_1 = w_0 \cdot r_0 \cdot r_1; \\
w_4 &= w_0 \cdot r_2; \\
w_5 &= w_1 \cdot r_2 = w_0 \cdot r_0 \cdot r_2; \\
w_6 &= w_2 \cdot r_2 = w_0 \cdot r_1 \cdot r_2; \\
w_7 &= w_3 \cdot r_2 = w_0 \cdot r_0 \cdot r_1 \cdot r_2; \\
&\dots
\end{aligned} \tag{1.9}$$

Увидим здесь закономерность и с помощью вложенных циклов построим матрицу Уолша размерности $2^N - 1 \times 2^N - 1$:

$$\begin{aligned}
&w_0 = (1, 1, \dots, 1); \\
&for(k = 1; k \leq N; k++) \{ \\
&\quad for(i = 2^{k-1}; i < 2^k; i++) \{ \\
&\quad \quad for(j = 0; j < 2^N; j++) \{ \\
&\quad \quad \quad w[i, j] = w[i - 2^{k-1}, j] \times r[k - 1, j]; \\
&\quad \quad \quad \} \\
&\quad \quad \} \\
&\quad \} \\
&\}
\end{aligned} \tag{1.10}$$

Значения $r_{k,j}(\Delta_j^N)$ получаем из следующего равенства

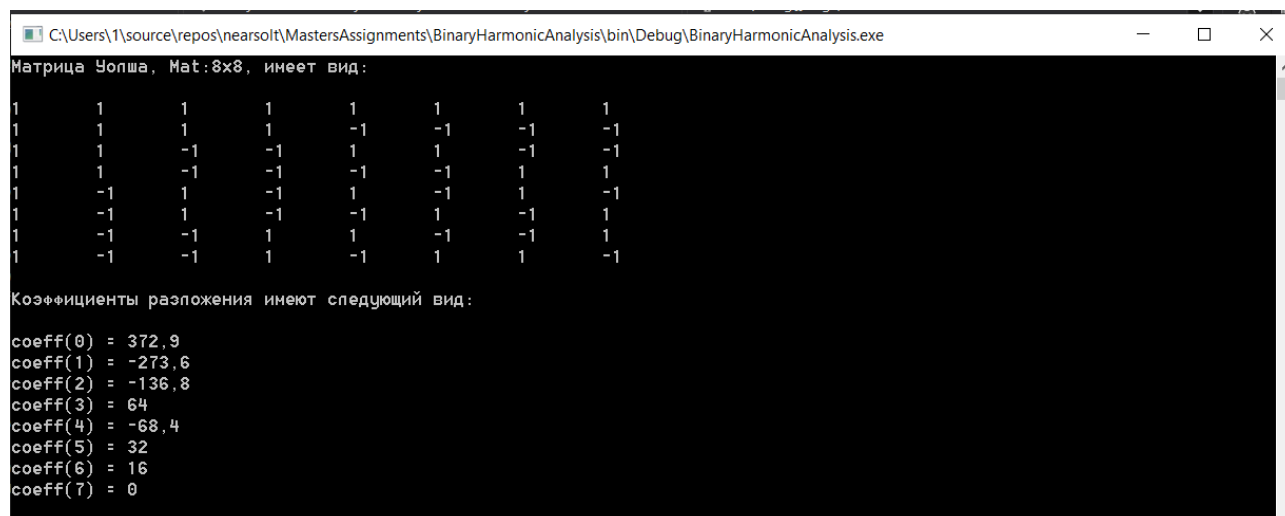
$$r_{k,j}(\Delta_j^N) = (-1)^\theta, \quad \theta = j \operatorname{div} 2^{N-k-1}. \tag{1.11}$$

2 Реализация алгоритма

В листингах приведены два класса: `BinaryHarmonicAnalysis.cs` – включает в себя точку входа в программу и в методе `Main` объявляем `powerOfTwo` (степень двойки) для того, чтобы инициализировать размерность матрицы Уолша, а также два значения (`start`, `end`) – инициализация интервала, которым будет определяться функция f ; `Helpers.cs` – класс, включающий в себя метод `FuncValues`, в котором инициализируем функцию f , методы вычисления матрицы Уолша в представлении Пэли, коэффициентов разложения и вспомогательные для них методы.

В методе `Main` инициализируем переменные `powerOfTwo`, `start`, `end`. С помощью метода `CalcWalshMatrix` вычисляем матрицу Уолша в представлении Пэли и, используя `PrintWalshMatrix`, выводим ее в консоль. Затем, выполняем метод `CalcCoeff` для вычисления коэффициентов разложения и выводим их в консоль.

При значениях `powerOfTwo = 3`, `start = 1`, `end = 33` и функции $f = x^2 + 0.2x - 3.5$ имеем следующий результат выполнения программы:



```
C:\Users\1\source\repos\nearsolt\MastersAssignments\BinaryHarmonicAnalysis\bin\Debug\BinaryHarmonicAnalysis.exe
Матрица Уолша, Mat:8x8, имеет вид:
1      1      1      1      1      1      1      1
1      1      1      1      -1     -1     -1     -1
1      1      -1     -1      1      1      -1     -1
1      1      -1     -1     -1     -1     1      1
1     -1      1      -1      1     -1      1     -1
1     -1      1      -1     -1     1      -1     1
1     -1     -1      1      1     -1     -1     1
1     -1     -1      1     -1      1      1     -1

Коэффициенты разложения имеют следующий вид:
coeff(0) = 372,9
coeff(1) = -273,6
coeff(2) = -136,8
coeff(3) = 64
coeff(4) = -68,4
coeff(5) = 32
coeff(6) = 16
coeff(7) = 0
```

ПРИЛОЖЕНИЕ А

Листинг: построение матрицы Уолша в представлении Пэли и вычисление коэффициентов Фурье-Уолша

A.1 BinaryHarmonicAnalysis.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace BinaryHarmonicAnalysis {
8      class BinaryHarmonicAnalysis {
9
10         #region task 1: Walsh Main
11         static void Main(string[] args) {
12
13             int powerOfTwo = 3;
14             double start = 1, end = 33;
15
16             if (powerOfTwo < 0) {
17                 Console.WriteLine(string.Format("Степень_числа_не_может_быть_отрицательной: {0} = {1}, введите корректные данные", nameof(powerOfTwo), powerOfTwo));
18                 Console.ReadKey();
19                 return;
20             }
21
22             int partitionsNumber = Convert.ToInt32(Math.Pow(2, powerOfTwo));
23
24             int[,] walshMatrix = new int[partitionsNumber, partitionsNumber];
25
26             Helpers.CalcWalshMatrix(walshMatrix, powerOfTwo, partitionsNumber);
27             Helpers.PrintWalshMatrix(walshMatrix, partitionsNumber);
28
29             double[] coeff = Helpers.CalcCoeff(walshMatrix, partitionsNumber, start, end);
30
31             Console.WriteLine(string.Format("Коэффициенты{0}_разложения_имеют_следующий_вид:{0}", Environment.NewLine));
```

```

32         for (int i = 0; i < partitionsNumber; i++) {
33             Console.WriteLine(string.Format("coeff({0})={1}", i, coeff[i])
34         );
35         }
36         Console.ReadKey();
37     }
38     #endregion
39 }
40 }

```

A.2 Helpers.cs

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace BinaryHarmonicAnalysis {
8      class Helpers {
9
10         #region task 1: Walsh
11
12         #region Initial Data
13
14             /// <summary>
15             /// Получение значения функции в точке
16             /// </summary>
17             /// <param name="valueточка"></param>
18             /// <returns></returns>
19             static double FuncValues(double value) {
20                 return Math.Pow(value, 2.0) + value * 0.2 - 3.5;
21             }
22
23         #endregion
24
25         /// <summary>
26         /// Получение значений массива на  $\delta_j^n, (\lambda_j^n = f(\delta)) \quad j=0, 2^n$ 
27         -1
28         /// </summary>
29         /// <param name="partitionsNumberколичество"> разбиений интервала</param>
30         /// <param name="startначало"> интервала</param>
31         /// <param name="endконец"> интервала</param>
32         /// <returns></returns>

```



```

32     static double[] DeltaValues(int partitionsNumber, double start, double
end) {
33         double step = (end - start) / partitionsNumber;
34         double[] intervalSplit = new double[partitionsNumber + 1];
35         double[] funcValues = new double[partitionsNumber];
36
37         intervalSplit[0] = start;
38         intervalSplit[partitionsNumber] = end;
39
40         for (int i = 1; i < partitionsNumber; i++) {
41             intervalSplit[i] = intervalSplit[0] + i * step;
42         }
43
44         for (int i = 0; i < partitionsNumber; i++) {
45             funcValues[i] = FuncValues((intervalSplit[i] + intervalSplit[i +
1]) * 0.5);
46         }
47
48         return funcValues;
49     }
50
51     /// <summary>
52     /// Вычисление матрицы Уолша в представлении Пэли
53     /// </summary>
54     /// <param name="walshMatrixматрица"> Уолша</param>
55     /// <param name="powerOfTwoстепеней"> размерности по основанию 2</param>
56     /// <param name="partitionsNumberразмерность"> матрицы Уолша</param>
57     internal static void CalcWalshMatrix(int[, ] walshMatrix, int powerOfTwo,
int partitionsNumber) {
58         int[] w_0_Vector = Enumerable.Repeat(1, partitionsNumber).ToArray();
59
60         int[, ] rMatrix = new int[powerOfTwo, partitionsNumber];
61
62         for (int i = 0; i < powerOfTwo; i++) {
63             for (int j = 0; j < partitionsNumber; j++) {
64                 rMatrix[i, j] = Convert.ToInt32(Math.Pow(-1, j / Convert.
ToInt32(Math.Pow(2, powerOfTwo - i - 1))));
65             }
66         }
67
68         if (powerOfTwo > 0) {
69             for (int j = 0; j < partitionsNumber; j++) {
70                 walshMatrix[0, j] = w_0_Vector[j];
71             }
72             for (int k = 1; k <= powerOfTwo; k++) {

```

```

73         for (int i = Convert.ToInt32(Math.Pow(2, k - 1)); i <
Convert.ToInt32(Math.Pow(2, k)); i++) {
74             for (int j = 0; j < partitionsNumber; j++) {
75                 walshMatrix[i, j] = walshMatrix[i - Convert.ToInt32(
Math.Pow(2, k - 1)), j] * rMatrix[k - 1, j];
76             }
77         }
78     }
79 } else {
80     walshMatrix[0, 0] = w_0_Vector[0];
81 }
82 }
83
84     /// <summary>
85     /// Вывод матрицы Уолша
86     /// </summary>
87     /// <param name="walshMatrixматрица"> Уолша</param>
88     /// <param name="partitionsNumberразмерность"> матрицы Уолша</param>
89     internal static void PrintWalshMatrix(int[,] walshMatrix, int
partitionsNumber) {
90         Console.WriteLine(string.Format("Матрица_Уолша,_Mat:{0}x{0},_имеет_
вид:{1}", partitionsNumber, Environment.NewLine));
91         for (int i = 0; i < partitionsNumber; i++, Console.WriteLine("")) {
92             for (int j = 0; j < partitionsNumber; j++, Console.Write("\t"))
{
93                 Console.Write(walshMatrix[i, j].ToString());
94             }
95         }
96     }
97
98     /// <summary>
99     /// Вычисление коэффициентов разложения
100    /// </summary>
101    /// <param name="walshMatrixматрица"> Уолша</param>
102    /// <param name="partitionsNumberразмерность"> матрицы Уолша</param>
103    /// <param name="startначало"> интервала</param>
104    /// <param name="endконец"> интервала</param>
105    /// <returns></returns>
106    internal static double[] CalcCoeff(int[,] walshMatrix, int
partitionsNumber, double start, double end) {
107        double[] coeff = new double[partitionsNumber];
108        double[] deltaValues = DeltaValues(partitionsNumber, start, end);
109        for (int i = 0; i < partitionsNumber; i++) {
110            for (int j = 0; j < partitionsNumber; j++) {
111                coeff[i] += deltaValues[j] * walshMatrix[i, j] * 1.0;
112            }

```

```

113             coeff[i] /= partitionsNumber;
114         }
115         return coeff;
116     }
117
118     #endregion
119 }
120 }

```