ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»
(СПбГУТ)

ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ СЕТЕЙ И СИСТЕМ (ИКСС)

КАФЕДРА ПРОГРАММНОЙ ИНЖЕНЕРИИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

(ПИ И ВТ)

Дисциплина: Алгоритмические основы программной инженерии

Отчет по лабораторной работе № 4

«Аппроксимация методом наименьших квадратов»

Выполнила:

Студентка группы ИКПИ-93

Колтунова Е.В.

Подпись 

Принял:

Ерофеев С.А.

Подпись_____

Санкт-Петербург

2020

Содержание

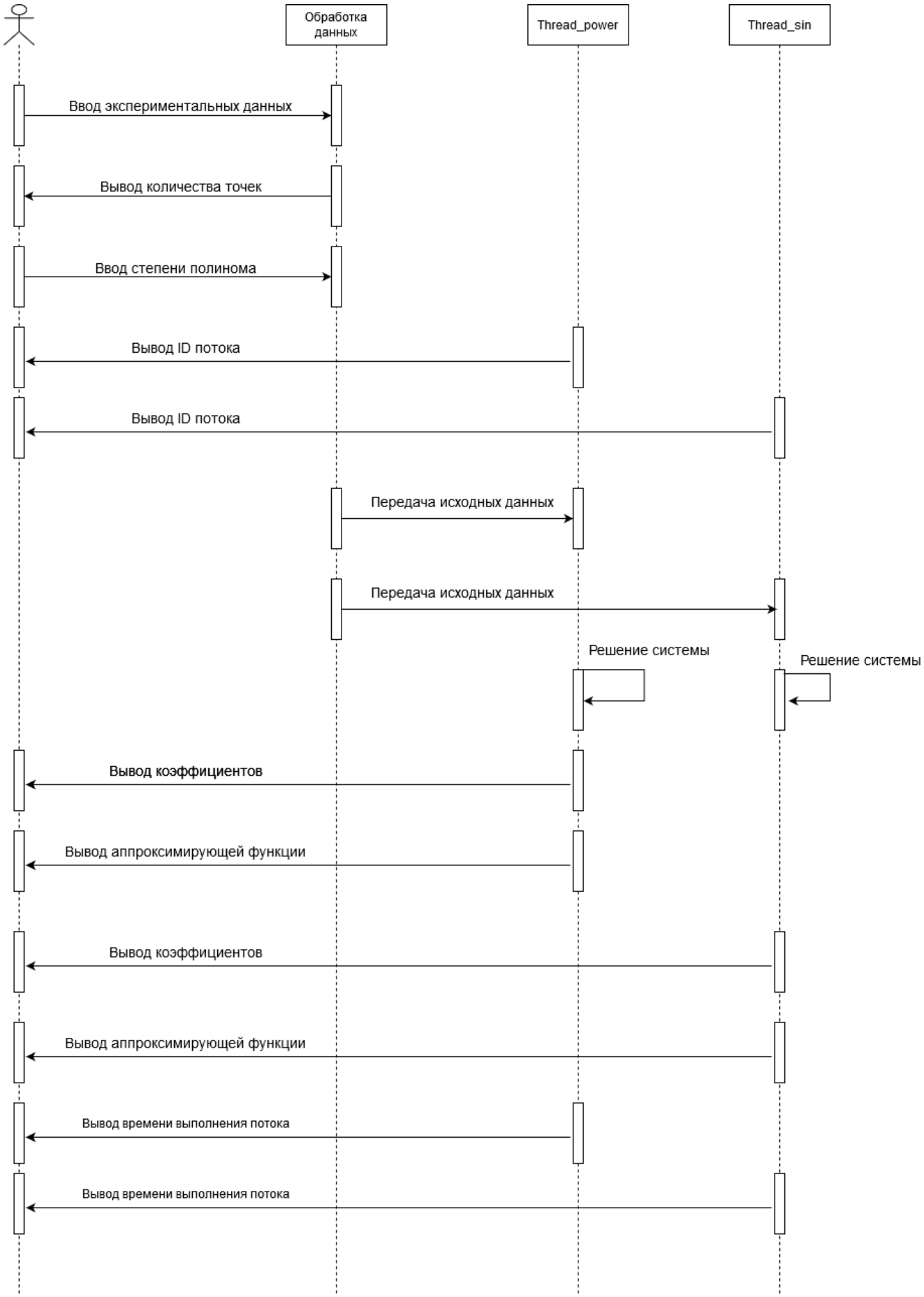
1. Постановка задачи

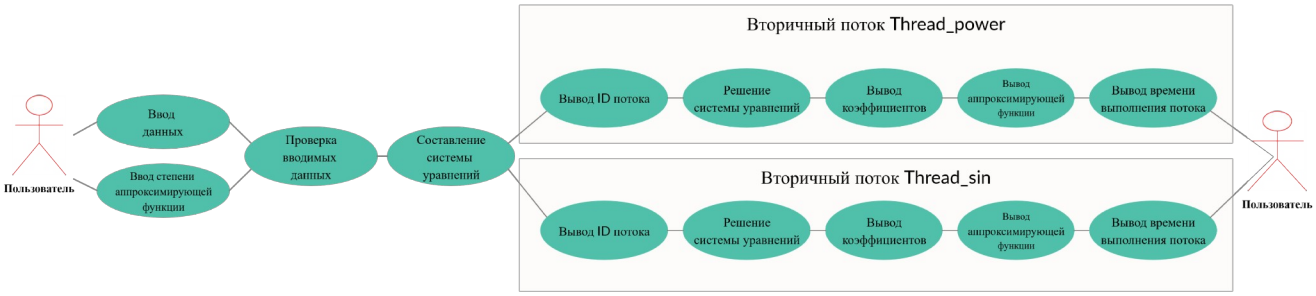
Платформа: Windows;
Тип приложения: консольное;
Язык программирования: C++.

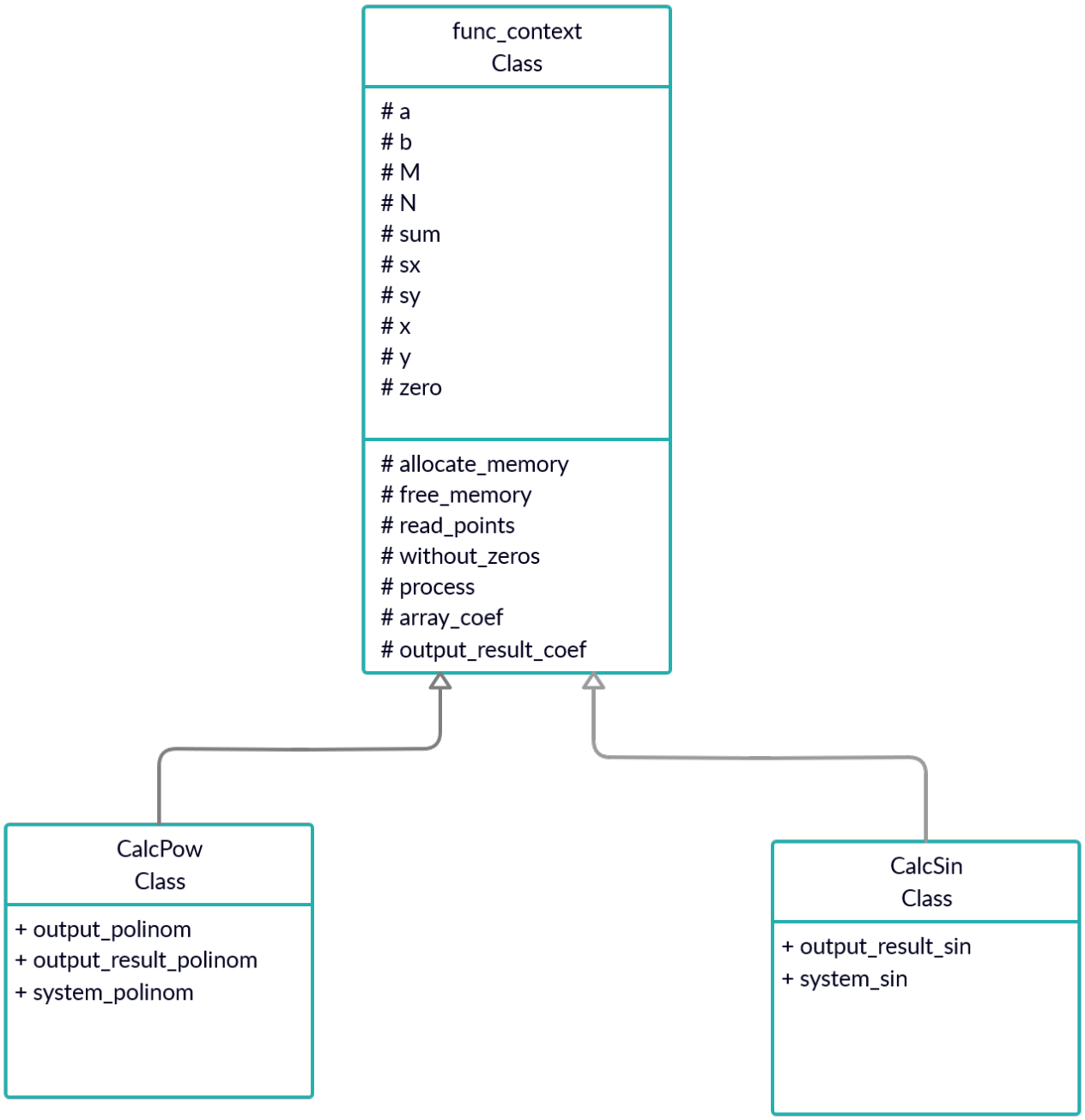
Задача: необходимо разработать программу, которая будет выполнять аппроксимацию функции методом наименьших квадратов двумя способами: с помощью степенной функции и с помощью функций семейства (x^m)*sin(m*x), где степень m задается пользователем.
Координаты точек находятся в текстовом файле. Пользователю будет предложено ввести название файла и степень полинома.
Степень полинома должна быть меньше количества точек. В случае, если введенная степень полинома будет больше количества точек, пользователю будет предложено ввести степень заново.

Требуется разместить способы во вторичные потоки. Каждый отдельный способ аппроксимации должен размещаться в своем вторичном потоке.

Также необходимо оценить время выполнения каждого потока.


Обработка данных
Thread_power
Thread_sin
Ввод экспериментальных данных
Вывод количества точек
Ввод степени полинома
Вывод ID потока
Вывод ID потока
Передача исходных данных
Передача исходных данных
Решение системы
Решение системы
Вывод коэффициентов
Вывод аппроксимирующей функции
Вывод коэффициентов
Вывод аппроксимирующей функции
Вывод времени выполнения потока
Вывод времени выполнения потока


Вторичный поток Thread_power
Вывод ID потока
Решение системы уравнений
Вывод коэффициентов
Вывод аппроксимирующей функции
Вывод времени выполнения потока
Вторичный поток Thread_sin
Вывод ID потока
Решение системы уравнений
Вывод коэффициентов
Вывод аппроксимирующей функции
Вывод времени выполнения потока
Пользователь
Ввод данных
Ввод степени аппроксимирующей функции
Проверка вводимых данных
Составление системы уравнений
Пользователь


func_context
Class
a
b
M
N
sum
sx
sy
x
y
zero
allocate_memory
free_memory
read_points
without_zeros
process
array_coef
output_result_coef
CalcPow
Class
+ output_polinom
+ output_result_polinom
+ system_polinom
CalcSin
Class
+ output_result_sin
+ system_sin

2. Разработка алгоритма

Метод наименьших квадратов применяется при обработке результатов эксперимента для аппроксимации (приближения) экспериментальных данных аналитической формулой. Конкретный вид формулы выбирается, как правило, из физических соображений. Такими формулами могут быть:

$$y = ax + b,$$
$$y = ax^2 + bx + c,$$
$$y = a_0 + a_1 x^1 + \ldots + a_{m-1} x^{m-1} + a_m x^m,$$

и другие.

Сущность метода наименьших квадратов состоит в следующем. Пусть результаты измерений представлены таблицей:

x	x_1	x_2	...	x_n
y	y_1	y_2	...	y_n

Будем считать, что вид аппроксимирующей функции выбран и представлен в виде:

$f(x) = \varphi(x, a_0, a_1, a_2, \ldots, a_m)$,

где φ - известная функция, $a_0, a_1, a_2, \ldots, a_m$ - неизвестные постоянные параметры, значения которых надо найти.

Требуется определить такие параметры, при которых значения аппроксимирующей функции приблизительно совпадали со значениями экспериментальных дискретных отсчетов.

Параметры $a_0, a_1, a_2, \ldots, a_m$ аппроксимирующей формулы находятся из условия минимума функции $S = S(a_0, a_1, a_2, \ldots, a_m)$,. Так как параметры выступают в роли независимых переменных функции S, то ее минимум найдем, приравнивая к нулю частные производные по этим переменным:

$$\frac{\partial S}{\partial a_0} = 0; \quad \frac{\partial S}{\partial a_1} = 0; \quad \ldots; \quad \frac{\partial S}{\partial a_m} = 0;$$

Полученные соотношения – система уравнений для определения параметров $a_0, a_1, a_2, \ldots, a_m$.

2.1 Аппроксимация МНК с помощью степенной функции

$1, x, x^2, \dots, x^m$ $\quad m - вводит\ пользователь$

$$\left.\begin{array}{ll} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ \multicolumn{2}{c}{\dots} \\ x_n & y_n \end{array}\right\} f(x)$$

$$f(x) = \sum_{k=0}^{m} a_k x^k = a_0 + a_1 x + a_2 x^2 + \dots + a_m x^m$$

$$S(x) = \sum_{k=1}^{n} [f(x_k) - a_0 - a_1 x_k - a_2 x_k^2 - a_m x_k^m]^2$$

Согласно МНК коэффициенты находятся из условия:

$S(x) \to min$

$$\frac{\partial S}{\partial a_0} = 0; \quad \frac{\partial S}{\partial a_1} = 0; \quad \dots; \quad \frac{\partial S}{\partial a_m} = 0;$$

Находим частные производные:

$$\begin{cases} \frac{\partial S}{\partial a_0} = -2 \sum_{k=1}^{n} [f(x_k) - a_0 - a_1 x_k - a_2 x_k^2 - a_m x_k^m] \\ \frac{\partial S}{\partial a_1} = -2 \sum_{k=1}^{n} [f(x_k) - a_0 - a_1 x_k - a_2 x_k^2 - a_m x_k^m] * x_k \\ \frac{\partial S}{\partial a_m} = -2 \sum_{k=1}^{n} [f(x_k) - a_0 - a_1 x_k - a_2 x_k^2 - a_m x_k^m] * x_k^m \end{cases}$$

Приравниваем частные производные к 0:

$$\begin{cases} \sum_{k=1}^{n} f(x_k) - a_0 \sum_{k=1}^{n} 1 - a_1 \sum_{k=1}^{n} x_k - \ldots - a_m \sum_{k=1}^{n} x_k^m = 0 \\ [\sum_{k=1}^{n} f(x_k) - a_0 \sum_{k=1}^{n} 1 - a_1 \sum_{k=1}^{n} x_k - \ldots - a_m \sum_{k=1}^{n} x_k^m] x_k = 0 \\ [\sum_{k=1}^{n} f(x_k) - a_0 \sum_{k=1}^{n} 1 - a_1 \sum_{k=1}^{n} x_k - \ldots - a_m \sum_{k=1}^{n} x_k^m] x_k^m = 0 \end{cases}$$

Получаем систему уравнений и решаем методом Гаусса:

$$\begin{bmatrix} \sum_{k=1}^{n} 1 & \sum_{k=1}^{n} x_k & \ldots & \sum_{k=1}^{n} x_k^m \\ x_k \sum_{k=1}^{n} 1 & x_k \sum_{k=1}^{n} x_k & \ldots & x_k \sum_{k=1}^{n} x_k^m \\ x_k^m \sum_{k=1}^{n} 1 & x_k^m \sum_{k=1}^{n} x_k & \ldots & x_k^m \sum_{k=1}^{n} x_k^m \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \ldots \\ a_m \end{bmatrix} = \begin{bmatrix} \sum_{k=1}^{n} f(x_k) \\ \sum_{k=1}^{n} f(x_k) \\ \sum_{k=1}^{n} f(x_k) \end{bmatrix}$$

2.1 Аппроксимация МНК с помощью функции (x^m) * sin(m*x)

$x\sin x, x^2\sin(2x), \ldots, x^m\sin(mx)$ $\quad m-$ *вводит пользователь*

$$f_0 = x\sin x,$$
$$f_1 = x^2\sin(2x),$$
$$\ldots$$
$$f_{m-1} = x^m\sin(mx)$$

Координаты точек:

$$\left.\begin{array}{ll} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ \ldots & \\ x_n & y_n \end{array}\right\} f(x)$$

Согласно МНК коэффициенты находятся из условия:

$$S(x) = \sum_{k=1}^{n} [f(x_k) - a_0 x\sin x - a_1 x^2\sin(2x) - \ldots - a_{m-1} x^m\sin(mx)]^2 \rightarrow min$$

Вычисляя частные производные и приравнивая их нулю, получим систему линейных алгебраических уравнений относительно коэффициентов a_0, a_1, ..., a_{m-1}.

$$\frac{\partial S}{\partial a_0} = 0; \quad \frac{\partial S}{\partial a_1} = 0; \quad \ldots; \quad \frac{\partial S}{\partial a_{m-1}} = 0;$$

Находим частные производные:

$$\begin{cases} \frac{\partial S}{\partial a_0} = -2a_0 \sum_{k=1}^{n} [f(x_k) - a_0 x \sin x - a_1 x^2 \sin(2x) - ... - a_{m-1} x^m \sin(mx)] \\ \frac{\partial S}{\partial a_1} = -2a_1 \sum_{k=1}^{n} [f(x_k) - a_0 x \sin x - a_1 x^2 \sin(2x) - ... - a_{m-1} x^m \sin(mx)] \\ ... \\ \frac{\partial S}{\partial a_m} = -2a_{m-1} \sum_{k=1}^{n} [f(x_k) - a_0 x \sin x - a_1 x^2 \sin(2x) - ... - a_{m-1} x^m \sin(mx)] \end{cases}$$

Приравниваем частные производные к 0:

$$\begin{cases} a_0 \sum_{k=1}^{n} f_0^2 + a_1 \sum_{k=1}^{n} f_0 * f_1 + ... + a_{m-1} \sum_{k=1}^{n} f_0 * f_{m-1} = \sum_{k=1}^{n} f_0 * y_k \\ a_0 \sum_{k=1}^{n} f_1 * f_0 + a_1 \sum_{k=1}^{n} f_1^2 + ... + a_{m-1} \sum_{k=1}^{n} f_1 * f_{m-1} = \sum_{k=1}^{n} f_1 * y_k \\ ... \\ a_0 \sum_{k=1}^{n} f_{m-1} * f_0 + a_1 \sum_{k=1}^{n} f_{m-1} * f_1 + ... + a_m \sum_{k=1}^{n} f_{m-1}^2 = \sum_{k=1}^{n} f_{m-1} * y_k \end{cases}$$

Получаем систему уравнений и решаем методом Гаусса:

$$\begin{bmatrix} \sum_{k=1}^{n} f_0^2 & \sum_{k=1}^{n} f_0 * f_1 & ... & \sum_{k=1}^{n} f_0 * f_{m-1} \\ \sum_{k=1}^{n} f_1 * f_0 & \sum_{k=1}^{n} f_1^2 & ... & \sum_{k=1}^{n} f_1 * f_{m-1} \\ & ... & & \\ \sum_{k=1}^{n} f_{m-1} * f_0 & \sum_{k=1}^{n} f_{m-1} * f_1 & ... & \sum_{k=1}^{n} f_{m-1}^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ ... \\ a_{m-1} \end{bmatrix} = \begin{bmatrix} \sum_{k=1}^{n} f_0 * y_k \\ \sum_{k=1}^{n} f_1 * y_k \\ \sum_{k=1}^{n} f_{m-1} * y_k \end{bmatrix}$$

3. Диапазон переменных

Имя	Тип	Диапазон	Количество занимаемой памяти
x	float	От $-3.4 * 10^{38}$ до $+3.4 * 10^{38}$	4 байта
y	float	От $-3.4 * 10^{38}$ до $+3.4 * 10^{38}$	4 байта
a	float	От $-3.4 * 10^{38}$ до $+3.4 * 10^{38}$	4 байта
b	float	От $-3.4 * 10^{38}$ до $+3.4 * 10^{38}$	4 байта
sum	float	От $-3.4 * 10^{38}$ до $+3.4 * 10^{38}$	4 байта
N	int	От −32767 до 32767	2 байта
M	int	От −32767 до 32767	2 байта
file_name	Char [ ]	От -128 до 126	256 байт

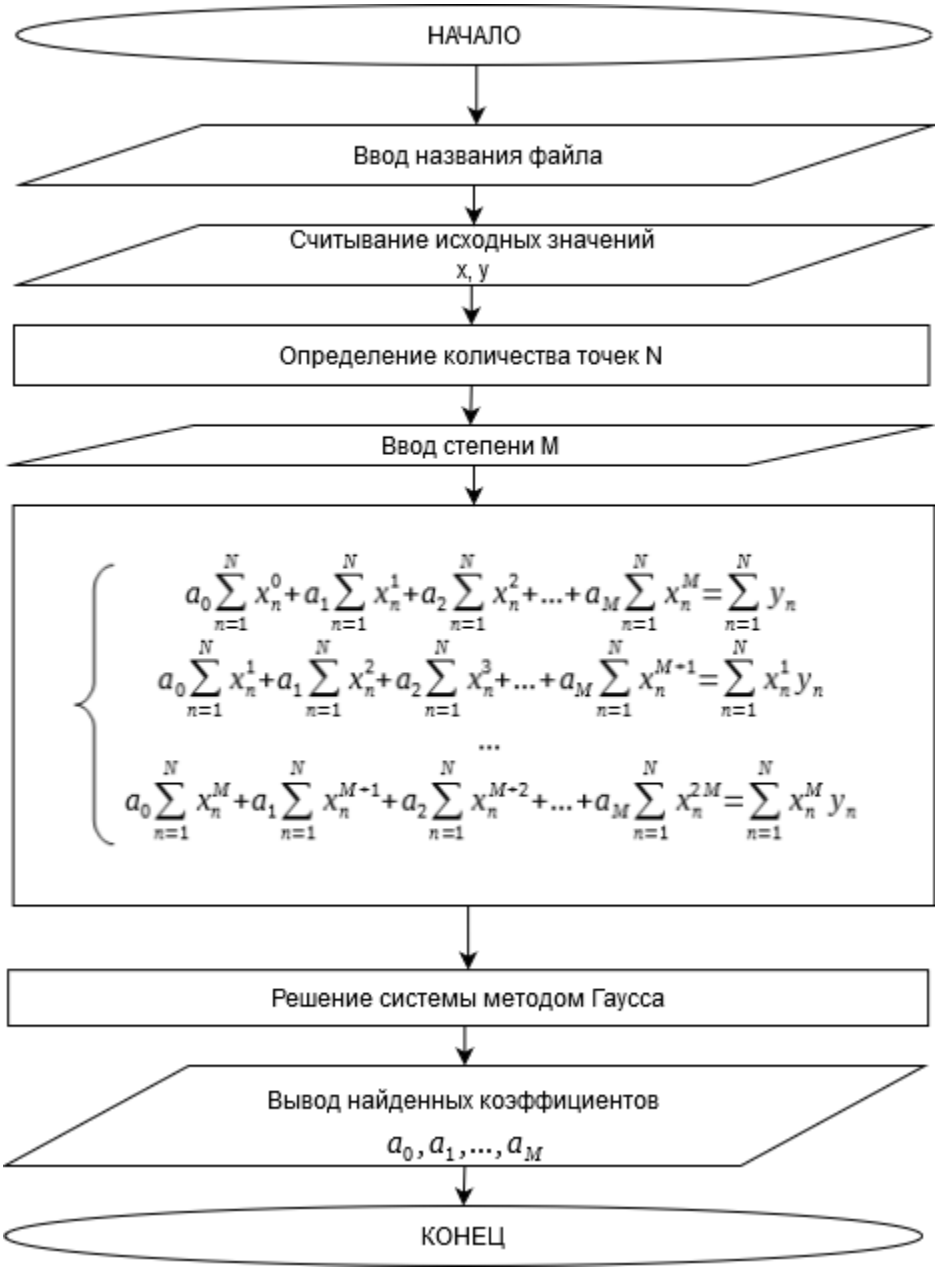
4. Спецификация

Ошибка	Сообщение об ошибке	Реализация

Значение степени больше, чем количество точек	Степень должна быть меньше количества точек!	`printf("\nВведите степень аппроксимирующей функции: M = ");`
		`scanf("%d", &M);`
		`while (M >= N) {`
		`printf("\nОшибка! Степень аппроксимирующей функции должна быть меньше количества точек N = %d.", N);`
		`printf("\nПопробуйте еще раз.\nВведите степень: M = ");`
		`scanf("%d", &M);`
		`}`

Проблема открытия файла	Введите имя файла	`while (NULL == file_data) {`
		`printf("Введите название файла: ");`
		`scanf("%s", file_name);`
		`file_data = fopen(file_name, "r");`
		`}`

5. Блок-схема алгоритма



6. Оценка сложности

1. Функция without_zeros() необходима для избавления от нулей на главной диагонали.

```
	void without_zeros() {
		float temp = 0;

		for (int n = 0; n < (M + 1); n++) {
			if (sum[n][n] == 0)
			{
				for (int m = 0; m < (M + 1); m++) {
					if (n == m) {
						continue;
					}
					if (sum[n][m] != 0 && sum[m][n] != 0) {
						for (int j = 0; j < (M + 1); j++) {
							temp = sum[m][j];
							sum[m][j] = sum[n][j];
							sum[n][j] = temp;
						}
						temp = b[m];
						b[m] = b[n];
						b[n] = temp;
						break;
					}
				}
			}
		}
	}
```

Сложность алгоритма — $T(n^3)$.

2. Функция process() необходима для решения полученной системы методом Гаусса.

```
	void process() {

		for (int n = 0; n < M + 1; n++) {
			for (int m = (n + 1); m < (M + 1); m++) {
				if (sum[m][m] == 0) {
					printf("\nРешение не существует.\n");
					return;
				}

				float K = sum[m][n] / sum[n][n];
				for (int j = n; j < (M + 1); j++) {
					sum[m][j] -= K * sum[n][j];
				}
				b[m] -= K * b[n];
			}
		}
	}
```

Сложность алгоритма — $T(n*(n/2)*(n/2)) = T(n^3)$.

3. Функция array_coef() необходима для вычисления коэффициентов.

```
	void array_coef() {
		for (int n = M; n >= 0; n--) {
			float s = 0;
			for (int m = n; m < (M + 1); m++) {
				s = s + sum[n][m] * a[m];
			}
			a[n] = (b[n] - s) / sum[n][n];
		}
	}
```

Сложность алгоритма — $T(n*(n/2)) = T(n^2)$.

4. Функция array_result_coef() необходима для вывода полученных коэффициентов на экран.

```
	void output_result_coef() {
		printf("\nКоэффициенты аппроксимирующей функции:");
		for (int n = 0; n < (M + 1); n++) {
			printf("\na[%d] = %.3f", n, a[n]);
		}
		printf("\n");
	}
```

Сложность алгоритма — T(n).

5. Функция system_polinom() необходима для составления системы уравнений для степенной функции.

```
	void system_polinom() {

		for (int n = 0; n < (M + 1); n++) {
			for (int m = 0; m < (M + 1); m++) {
				sum[n][m] = 0;
				for (int j = 0; j < N; j++) {
					sum[n][m] += pow(x[j], n + m);
				}
			}
		}
		//free coefficients(b)
		for (int n = 0; n < (M + 1); n++) {
			for (int m = 0; m < N; m++) {
				b[n] += pow(x[m], n) * y[m];
			}
		}
	}
```

Сложность алгоритма относительно степени функции(M) — $T(n^2)$.

Сложность алгоритма относительно количества отсчетных точек(N) — T(n).

6. Функция system_sin() необходима для составления системы уравнений для функции x^M*sin(M*x).

```
void system_sin() {
		for (int n = 0; n <= M; n++) {
			for (int m = 0; m <= M; m++) {
				sum[n][m] = 0;
				for (int j = 0; j < N; j++) {
					sum[n][m] += pow(x[j], n + 1) * sin((n + 1) * x[j]) *
pow(x[j], m + 1) * sin((m + 1) * x[j]);
				}
			}
		}

		//free coefficients(b)
		for (int n = 0; n <= M; n++) {
			b[n] = 0;
			for (int j = 0; j < N; j++) {
				b[n] += (pow(x[j], (n + 1)) * sin((n + 1) * x[j]) * y[j]);
			}
		}
	}
```

Сложность алгоритма относительно степени функции(M) — $T(n^2)$.

Сложность алгоритма относительно количества отсчетных точек(N) — T(n).

7. Функция output_result_polinom необходима для вывода аппроксимирующей степенной функции.

```
	void output_result_polinom() {
		printf("\nРезультат:");
		printf("\nf(x) = ");
		for (int n = 0; n <= M; n++) {
			printf("%.3f*x^%i", a[n], n);
			if (n < M) {
				printf(" + ");
			}
		}
		printf("\n");
	}
```

Сложность алгоритма — T(n).

8. Функция output_result_polinom необходима для вывода аппроксимирующей функции $x^M*\sin(M*x)$.

```
	void output_result_sin() {
		printf("\nРезультат:");
		printf("\nf(x) = ");
		for (int n = 0; n < (M + 1); n++) {
			printf("%.3f*(x^%i) * sin(%i * x)", a[n], M + 1, M + 1);
			if (n < M) {
				printf(" + ");
			}
		}
		printf("\n\n");
	}
```

Сложность алгоритма — T(n).

Время выполнения потока для аппроксимации с помощью степенной функции:

```
P time:16285 uSec
```

Время выполнения потока для аппроксимации с помощью функции $x^M*\sin(M*x)$:

```
S time:33484 uSec
```

Проанализировав сложность алгоритмов, можно сделать вывод о том, что сложность потоков для аппроксимации с помощью степенной функции и с помощью функции $x^M*\sin(M*x)$ одинакова.

Время выполнения первого потока больше, чем время выполения второго потока потому, что при аппроксимации с помощью функции $x^M*\sin(M*x)$ требуется больше вычислений при составлении системы уравнений.

7. Код программы

```
// IKPI93_Koltunova_EV_AOPI_LR_1.c : Defines the entry point for the console application.
//

#include "stdafx.h"
#include <stdio.h>
#include <conio.h>
#include <iostream>
#include <fstream>
#include <string>
#include <math.h>
#include <windows.h>

using namespace std;

class func_context {

public:

	func_context(int M, int N, ifstream& file_data) : M(M), N(N) {
		allocate_memory();
		read_points(file_data);
	}

	~func_context() {
		free_memory();
	}

	virtual void run() = NULL;

protected:

	//выделение памяти
	int allocate_memory() {
		a = new float[M + 1];
		b = new float[M + 1];
		x = new float[N];
		y = new float[N];
		sum = new float*[M + 1];
		if (NULL == x || NULL == y || NULL == a || NULL == sum) {
			printf("\nНедостаточно памяти. \nN = %d, M = %d\n", N, M);
			return -1;
		}

		for (int n = 0; n < (M + 1); n++) {
			sum[n] = new float[M + 1];
			if (NULL == sum[n]) {
				printf("\nНедостаточно памяти.\n", M + 1);
			}
		}
		for (int n = 0; n < M + 1; n++) {
			a[n] = 0;
			b[n] = 0;
			for (int m = 0; m < M + 1; m++) {
				sum[n][m] = 0;
			}
		}
```

```
		for (int n = 0; n < N; n++) {
			x[n] = 0;
			y[n] = 0;
		}
	}

	//освобождение памяти
	void free_memory() {
		for (int n = 0; n < (M + 1); n++) {
			delete[] sum[n];
		}
		delete[] a;
		delete[] b;
		delete[] x;
		delete[] y;
		delete[] sum;
	}


	//чтение точек из файла
	void read_points(ifstream& file_data) {
		//read points
		file_data.clear();
		file_data.seekg(0, file_data.beg);

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

			try
			{
				file_data >> sx >> sy;

				//fscanf(file_data, "%c", &sx[n]);
				//	fscanf(file_data, "%c", &sy[n]);
				// Блок проверки на наличие нуля
				if (sx[0] == '0')
				{
					x[n] = 0;
					zero = false;
				}
				if (sy[0] == '0')
				{
					y[n] = 0;
					zero = false;
				}

				// Блок проверки корректности ввода данных
				if ((!atof(sx) || !atof(sy)) && zero)
				{
					throw 3; // передать код ошибки 3
				}
				else
				{
					x[n] = atof(sx);
					y[n] = atof(sy);
				}

				// Проверка на выход за пределы максимального значения
				if (x[n] > (pow(2, sizeof(float) * 8.0 - 1) - 1) || y[n] > (pow(2,
sizeof(float) * 8.0 - 1) - 1))
				{
```

```
                                throw 5;
                        }

                        // Проверка на выход за пределы минимального значения
                        if (x[n] < (-1 * (pow(2, sizeof(float) * 8.0 - 1))) || y[n] < (-1 *
(pow(2, sizeof(float) * 8.0 - 1))))
                        {
                                throw 7;
                        }
                }
                catch (int i) // Получение кода ошибки
                {
                        if (i == 3)
                        {
                                cout << "Ошибка: Введена буква" << endl;
                        }
                        else if (i == 5)
                        {
                                cout << "Ошибка: Выход за пределы диапазона" << endl;
                        }
                        else if (i == 7)
                        {
                                cout << "Ошибка: Выход за пределы диапазона" << endl;
                        }
                        continue;
                }
        }


}


//избавление от нулей на главной диагонали
void without_zeros() {
        float temp = 0;

        for (int n = 0; n < (M + 1); n++) {
                if (sum[n][n] == 0)
                {
                        for (int m = 0; m < (M + 1); m++) {
                                if (n == m) {
                                        continue;
                                }
                                if (sum[n][m] != 0 && sum[m][n] != 0) {
                                        for (int j = 0; j < (M + 1); j++) {
                                                temp = sum[m][j];
                                                sum[m][j] = sum[n][j];
                                                sum[n][j] = temp;
                                        }
                                        temp = b[m];
                                        b[m] = b[n];
                                        b[n] = temp;
                                        break;
                                }
                        }
                }
        }
}

//вычисление коэффициентов
```

```
	void array_coef() {
		for (int n = M; n >= 0; n--) {
			float s = 0;
			for (int m = n; m < (M + 1); m++) {
				s = s + sum[n][m] * a[m];
			}
			a[n] = (b[n] - s) / sum[n][n];
		}
	}

	//вывод коэффициентов
	void output_result_coef() {
		printf("\nКоэффициенты аппроксимирующей функции:");
		for (int n = 0; n < (M + 1); n++) {
			printf("\na[%d] = %.3f", n, a[n]);
		}
		printf("\n");
	}

	//решение системы методом Гаусса
	void process() {

		for (int n = 0; n < M + 1; n++) {
			for (int m = (n + 1); m < (M + 1); m++) {
				if (sum[m][m] == 0) {
					printf("\nРешение не существует.\n");
					return;
				}

				float K = sum[m][n] / sum[n][n];
				for (int j = n; j < (M + 1); j++) {
					sum[m][j] -= K * sum[n][j];
				}
				b[m] -= K * b[n];
			}
		}
	}


	float* a;
	float* b;
	float* x;
	float* y;
	float** sum;
	char sx[64], sy[64];
	bool zero = true;
	int N, M;
};

char file_name[256];

CRITICAL_SECTION csFlag;

HANDLE hEvent;

//подсчет точек в файле
void amount_points(ifstream& file_data, int& N) {

	string s;
```

```
	N = 0;

	while (file_data.peek() != EOF) {
		getline(file_data, s);
		N++;
	}

	cout << "\nКоличество точек: N = " << N << endl;
}

//ввод степени
int input_power(int N) {
	printf("\nВведите степень аппроксимирующей функции: M = ");
	int M = 0;
	scanf("%d", &M);
	while (M >= N) {
		printf("\nОшибка! Степень функции должна быть меньше количества точек N = %d.",
N);
		printf("\nПопробуйте еще раз.\nВведите степень: M = ");
		scanf("%d", &M);
	}
	return M;
}


class CalcPow : public func_context {

public:


	CalcPow(int M, int N, ifstream& file_data) : func_context(M, N, file_data) {
	}

	virtual void run() {
		system_polinom();
		without_zeros();
		process();
		array_coef();

		EnterCriticalSection(&csFlag);

		cout << endl << "Аппроксимация МНК с помощью степенной функции" << endl;
		output_result_coef();
		output_result_polinom();
		LeaveCriticalSection(&csFlag);

	}


	//составление системы уравнений для полинома
	void system_polinom() {

		for (int n = 0; n < (M + 1); n++) {
			for (int m = 0; m < (M + 1); m++) {
				sum[n][m] = 0;
				for (int j = 0; j < N; j++) {
					sum[n][m] += pow(x[j], n + m);
				}
			}
		}
```

```
		//free coefficients(b)
		for (int n = 0; n < (M + 1); n++) {
			for (int m = 0; m < N; m++) {
				b[n] += pow(x[m], n) * y[m];
			}
		}
	}


	//вывод вида полинома
	void output_polinom() {

		printf("\nАппроксимирующий полином %iй степени:", M);
		printf("\nf(x) = ");
		for (int n = 0; n <= M; n++) {
			printf("a[%i]*x^%i", n, n);
			if (n < M) {
				printf(" + ");
			}
		}
		printf("\n");
	}

	//вывод итогового полинома
	void output_result_polinom() {
		printf("\nРезультат:");
		printf("\nf(x) = ");
		for (int n = 0; n <= M; n++) {
			printf("%.3f*x^%i", a[n], n);
			if (n < M) {
				printf(" + ");
			}
		}
		printf("\n");
	}


};


class CalcSin : public func_context {

public:

	CalcSin(int M, int N, ifstream& file_data) : func_context(M - 1, N, file_data) {
	}

	virtual void run() {
		system_sin();
		without_zeros();

		process();
		array_coef();

		EnterCriticalSection(&csFlag);
		cout << endl << "Аппроксимация МНК с помощью функции (x^M)*sin(M*x)" << endl;
		output_result_coef();
		output_result_sin();
		LeaveCriticalSection(&csFlag);
```

```
	}

	//вывод аппроксимирующей функции (x^M)*sin(M*x)
	void output_result_sin() {
		printf("\nРезультат:");
		printf("\nf(x) = ");
		for (int n = 0; n < (M + 1); n++) {
			printf("%.3f*(x^%i) * sin(%i * x)", a[n], M + 1, M + 1);
			if (n < M) {
				printf(" + ");
			}
		}
		printf("\n\n");
	}

	//составление системы уравнений для функции (x^M)*sin(M*x)
	void system_sin() {
		for (int n = 0; n <= M; n++) {
			for (int m = 0; m <= M; m++) {
				sum[n][m] = 0;
				for (int j = 0; j < N; j++) {
					sum[n][m] += pow(x[j], n + 1) * sin((n + 1) * x[j]) *
pow(x[j], m + 1) * sin((m + 1) * x[j]);
				}
			}
		}

		//free coefficients(b)
		for (int n = 0; n <= M; n++) {
			b[n] = 0;
			for (int j = 0; j < N; j++) {
				b[n] += (pow(x[j], (n + 1)) * sin((n + 1) * x[j]) * y[j]);
			}
		}
	}

};



DWORD WINAPI Thread_power(LPVOID lpParam) {

	func_context& context = *reinterpret_cast<func_context*>(lpParam);

	context.run();

	SetEvent(hEvent);
	return 0;
}

DWORD WINAPI Thread_sin(LPVOID lpParam) {

	WaitForSingleObject(hEvent, INFINITE);

	func_context& context = *reinterpret_cast<func_context*>(lpParam);

	context.run();

	return 0;
}
```

```
void print_times(HANDLE hThread) {
	FILETIME  ftimeCreate, ftimeExit, ftimeKernel, ftimeUser;
	BOOL ok = GetThreadTimes(hThread, &ftimeCreate, &ftimeExit, &ftimeKernel, &ftimeUser);
	cout << (((uint64_t)(ftimeExit.dwHighDateTime - ftimeCreate.dwHighDateTime) << MAXDWORD)
+
		ftimeExit.dwLowDateTime - ftimeCreate.dwLowDateTime) / 10 << " uSec "
		<< endl;
}

int main()
{
	setlocale(LC_ALL, "Russian");

	cout << "Введите название файла: ";
	cin >> file_name;

	ifstream file_data(file_name);

	while (!(file_data.is_open())) {
		cout << "Введите название файла: ";
		cin >> file_name;
		ifstream file_data(file_name);
	}

	int N = 0;
	amount_points(file_data, N);


	int M = input_power(N);

	CalcPow context_power(M, N, file_data);
	CalcSin context_sin(M, N, file_data);

	file_data.close();

	HANDLE fThread, sThread;
	DWORD ThreadID, dwRet;
	InitializeCriticalSection(&csFlag); //инициализациия критической секции

	hEvent = CreateEvent(nullptr, FALSE, FALSE, nullptr);

	fThread = CreateThread(
		NULL,
		0,
		Thread_power,
		&context_power,
		CREATE_SUSPENDED,
		&ThreadID);

	if (fThread == NULL) {
		cout << "Thread Power Creation Failed & Error No ---> " << GetLastError() <<
endl;
	}

	cout << "\nThread Power Creation Success" << endl;
	cout << "Thread Power ID --> " << ThreadID << endl;

	sThread = CreateThread(
```

```
		NULL,
		0,
		Thread_sin,
		&context_sin,
		CREATE_SUSPENDED,
		&ThreadID);

	if (sThread == NULL) {
		cout << "Thread Sin Creation Failed & Error No ---> " << GetLastError() << endl;
	}
	cout << "\nThread Sin Creation Success" << endl;
	cout << "Thread Sin ID --> " << ThreadID << endl;


	ResumeThread(fThread);
	ResumeThread(sThread);

	dwRet = WaitForSingleObject(fThread, INFINITE);
	dwRet = WaitForSingleObject(sThread, INFINITE);

	cout << "P time:"; print_times(fThread);
	cout << "S time:"; print_times(sThread);

	CloseHandle(fThread);
	CloseHandle(sThread);
	CloseHandle(hEvent);

	return 0;
}
```

8. Тесты

№	Содержимое файла	Результат выполнения программы
1	aopi_data.txt 1 1,2 7,4 2 2,9 9,5 3 4,1 11,1 4 5,5 12,9 5 6,7 14,6 6 7,8 17,3 7 9,2 18,2 8 10,3 20,7	Введите название файла: aopi_data.txt Количество точек: N = 8 Введите степень аппроксимирующей функции: M = 1 Thread Power Creation Success Thread Power ID --> 13500 Thread Sin Creation Success Thread Sin ID --> 7848 Аппроксимация МНК с помощью степенной функции Коэффициенты аппроксимирующей функции: a[0] = 5,291 a[1] = 1,454 Результат: f(x) = 5,291*x^0 + 1,454*x^1 Аппроксимация МНК с помощью функции (x^M)*sin(M*x) Коэффициенты аппроксимирующей функции: a[0] = -0,150 Результат: f(x) = -0,150*(x^1) * sin(1 * x) P time:10500 uSec S time:13850 uSec Press any key to continue . . .

2	aopi_data.txt 1 1,2 7,4 2 2,9 9,5 3 4,1 11,1 4 5,5 12,9 5 6,7 14,6 6 7,8 17,3 7 9,2 18,2 8 10,3 20,7	Введите название файла: aopi_data.txt Количество точек: N = 8 Введите степень аппроксимирующей функции: M = 7 Thread Power Creation Success Thread Power ID --> 11232 Thread Sin Creation Success Thread Sin ID --> 12292 Аппроксимация МНК с помощью степенной функции Коэффициенты аппроксимирующей функции: a[0] = 4,996 a[1] = 2,029 a[2] = 0,157 a[3] = -0,214 a[4] = 0,045 a[5] = -0,002 a[6] = -0,000 a[7] = 0,000 Результат: f(x) = 4,996*x^0 + 2,029*x^1 + 0,157*x^2 + -0,214*x^3 + 0,045*x^4 + -0,002*x^5 + -0,000*x^6 + 0,000*x^7 Аппроксимация МНК с помощью функции (x^M)*sin(M*x) Коэффициенты аппроксимирующей функции: a[0] = -2,922 a[1] = -0,186 a[2] = 0,091 a[3] = -0,008 a[4] = 0,001 a[5] = 0,000 a[6] = -0,000 Результат: f(x) = -2,922*(x^7) * sin(7 * x) + -0,186*(x^7) * sin(7 * x) + 0,091*(x^7) * sin(7 * x) + -0,008*(x^7) * sin(7 * x) + 0,001*(x^7) * sin(7 * x) + 0,000*(x^7) * sin(7 * x) + -0,000*(x^7) * sin(7 * x) P time:16285 uSec S time:33484 uSec Press any key to continue . . .
3		Введите название файла: aopi_datajudkhcd Введите название файла:
4		Введите название файла: aopi_data.txt Количество точек: N = 8 Введите степень аппроксимирующей функции: M = 13 Ошибка! Степень функции должна быть меньше количества точек N = 8. Попробуйте еще раз. Введите степень: M =

9. Вывод

Разработана программа на языке C++, выполняющая аппроксимацию функции методом наименьших квадратов двумя способами: с помощью степенной функции и с помощью функций семейства (x^m)*sin(m*x), где степень m задается пользователем. Каждый отдельный способ аппроксимации размещен в своем вторичном потоке. Оценена сложность алгоритмов, а также время выполнения каждого потока. Проведенные тесты доказывают, что программа работает в полном соответствии с ТЗ.