Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления» (ИУ)

КАФЕДРА «Информационная безопасность» (ИУ8)

Отчёт

по лабораторной работе № 1
по дисциплине «Теория систем и системный анализ»

Тема: «Исследование методов прямого поиска экстремума унимодальной функции одного переменного»

Вариант 7

Выполнила: Кидинова Д.Д.,
студент группы ИУ8-31

Проверила: Коннова Н.С.,
доцент каф. ИУ8

г. Москва,
2020 г.

Цель работы

Исследовать функционирование и провести сравнительный анализ различных алгоритмов прямого поиска экстремума (пассивный поиск, метод дихотомии, золотого сечения, Фибоначчи) на примере унимодальной функции одного переменного.

Условие задачи

На интервале [1,4] задана унимодальная функция одного переменного f(x)=-sqrt(x)*sin(x)+2. Используя метод дихотомии, найти интервал нахождения минимума f(x) при заданной наибольшей допустимой длине интервала неопределенности E=0,1. Провести сравнение с методом оптимального пассивного поиска. Результат, в зависимости от числа точек разбиения N, представить в виде таблицы.

График заданной функции

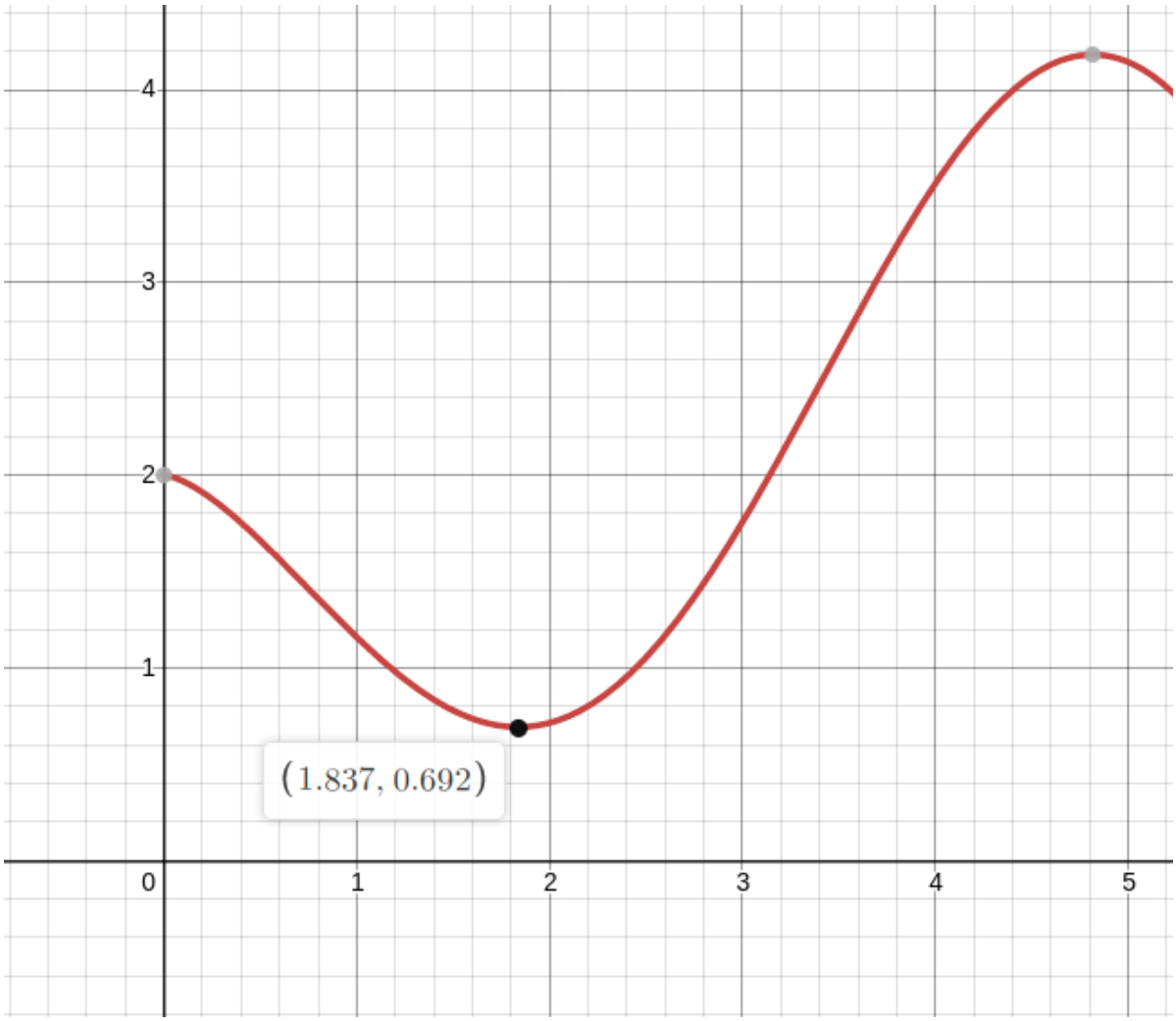

Рисунок 1 - График функции y=-sqrt(x)*sin(x)+2 на [1;4]

Part 1. Sequential search (dichotomy) :

Start of the interval (ak)	End of the interval (bk)	Length of the interval (l)	f(ak)	f(bk)
1	4	3	1.15853	3.5136
1	2.49	1.49	1.15853	1.04303
1.755	2.49	0.735	0.697648	1.04303
1.755	2.1125	0.3575	0.697648	0.754643
1.755	1.92375	0.16875	0.697648	0.698507
1.755	1.82938	0.074375	l < epsilon	

x = 1.792 +- 0.037

Part 2. Optimal passive search :

Number of the points (N)	The value of X in the minimum
1	2.500 +- 1.500
2	2.000 +- 1.000
3	1.750 +- 0.750
4	1.600 +- 0.600
5	2.000 +- 0.500
6	1.857 +- 0.429
7	1.750 +- 0.375
8	2.000 +- 0.333
9	1.900 +- 0.300
10	1.818 +- 0.273
11	1.750 +- 0.250
12	1.923 +- 0.231
13	1.857 +- 0.214
14	1.800 +- 0.200
15	1.750 +- 0.188
16	1.882 +- 0.176
17	1.833 +- 0.167

```
|        18       |     1.789 +- 0.158 |
|        19       |     1.900 +- 0.150 |
|        20       |     1.857 +- 0.143 |
|        21       |     1.818 +- 0.136 |
|        22       |     1.783 +- 0.130 |
|        23       |     1.875 +- 0.125 |
|        24       |     1.840 +- 0.120 |
|        25       |     1.808 +- 0.115 |
|        26       |     1.889 +- 0.111 |
|        27       |     1.857 +- 0.107 |
|        28       |     1.828 +- 0.103 |
|        29       |     1.800 +- 0.100 |
```

```
x = 1.800 +- 0.100
```

График зависимостей погрешности от числа точек N

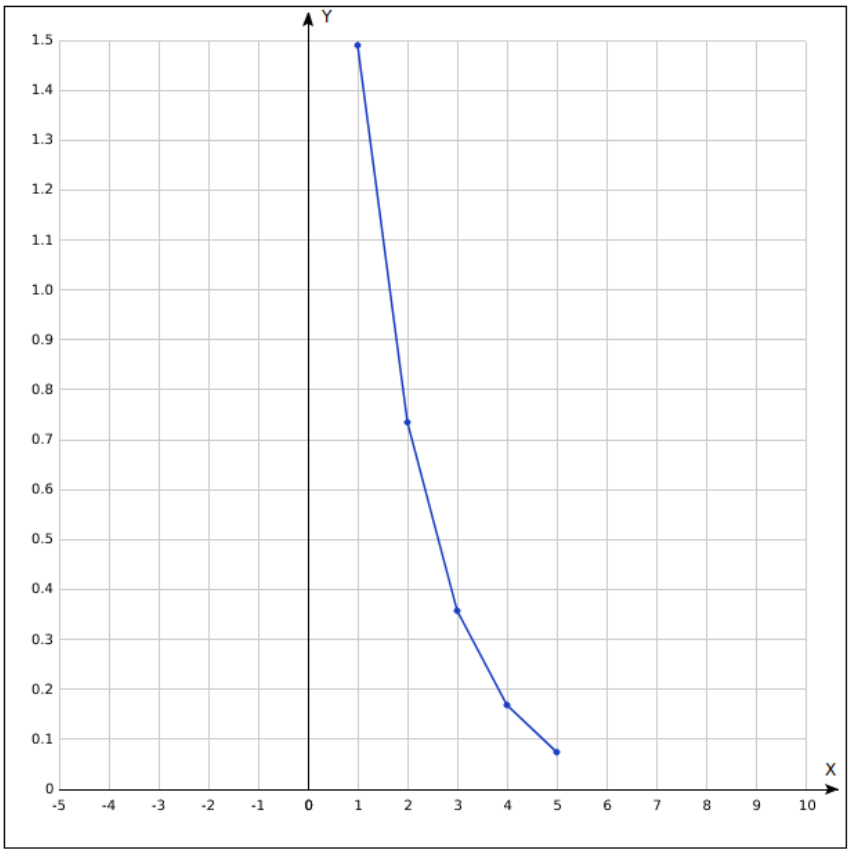

Рисунок 2 - График зависимости погрешности от числа точек N для дихотомии

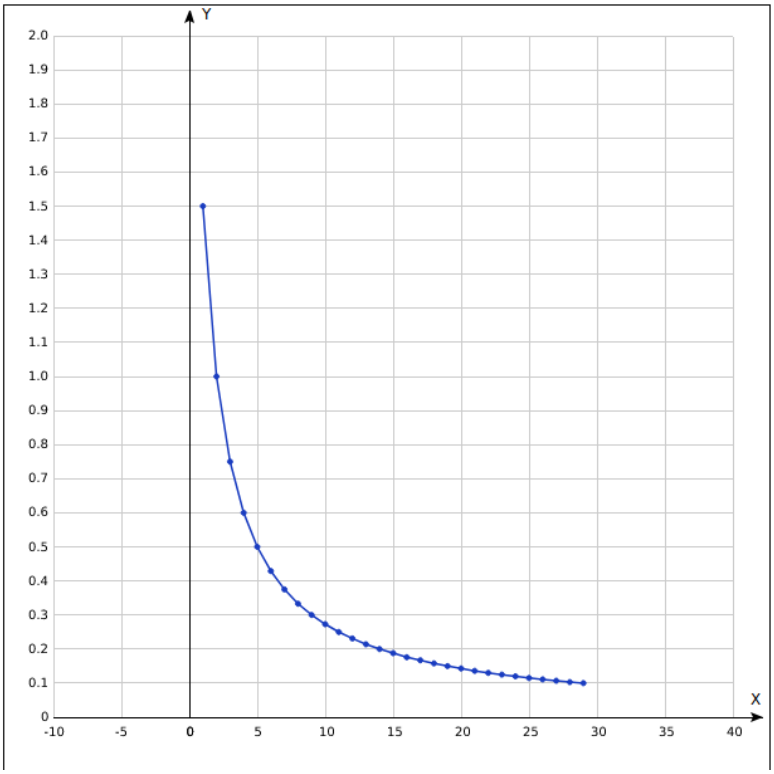

Рисунок 3 - График зависимости погрешности от числа точек N для оптимального пассивного поиска

Выводы

Из полученных таблиц и графиков видно, что метод дихотомии значительно эффективнее метода пассивного поиска при отыскании экстремума унимодальной функции одного переменного.

Приложение. Исходный код программы

```
#include <cmath>
#include <iomanip>
#include <iostream>
#include <vector>
#include <algorithm>

using std::cin;
using std::cout;

double FunctionFromTask(double x) {
	return -sqrt(x) * std::sin(x) + 2;
}

void PrintPart1(double ak, double bk) {
	cout << '|' << std::setw(13) << ak << ' '
		<< '|' << std::setw(13) << bk << ' '
		<< '|' << std::setw(13) << bk - ak << ' '
		<< '|' << std::setw(13) << FunctionFromTask(ak) << ' '
		<< '|' << std::setw(13) << FunctionFromTask(bk) << ' ' << '|' << '\n';
}

void PrintPart2(uint N, double xk, double delta) {
	cout << '|' << std::setw(10) << N << "      "
		<< '|' << std::setw(10) << xk << " +- " << delta << " |" << '\n';
}

const double LOWER_EDGE = 1.;
const double UPPER_EDGE = 4.;
const double EPSILON = 0.1;

void Dichotomy(double lower, double upper) {
	cout << "Part 1. Sequential search (dichotomy) :\n"
		<< std::string(76, '_') << '\n'
		<< '|' << std::string(3, ' ') << "Start of" << std::string(3, ' ')
		<< '|' << std::string(4, ' ') << "End of" << std::string(4, ' ')
		<< '|' << std::string(2, ' ') << "Length of" << std::string(3, ' ')
		<< '|' << std::string(14, ' ')
		<< '|' << std::string(14, ' ') << '|' << '\n'
		<< '|' << std::string(1, ' ') << "the interval" << std::string(1, ' ')
		<< '|' << std::string(1, ' ') << "the interval" << std::string(1, ' ')
		<< '|' << std::string(1, ' ') << "the interval" << std::string(1, ' ')
		<< '|' << std::string(4, ' ') << "f(ak)" << std::string(5, ' ')
		<< '|' << std::string(4, ' ') << "f(bk)" << std::string(5, ' ') << '|' << '\n'
		<< '|' << std::string(5, ' ') << "(ak)" << std::string(5, ' ')
		<< '|' << std::string(5, ' ') << "(bk)" << std::string(5, ' ')
		<< '|' << std::string(5, ' ') << "(l)" << std::string(6, ' ')
		<< '|' << std::string(14, ' ')
		<< '|' << std::string(14, ' ') << '|' << '\n'
		<< std::string(76, '_') << '\n';
	const double DELTA = .01;
	while (upper - lower > EPSILON) {
		PrintPart1(lower, upper);
		double x1 = lower + (upper - lower) / 2 - DELTA,
			x2 = lower + (upper - lower) / 2 + DELTA;
		FunctionFromTask(x1) < FunctionFromTask(x2)
		? upper = x1
```

```
                : lower = x2;
            }
            cout << '|' << std::setw(13) << lower << ' '
                 << '|' << std::setw(13) << upper << ' '
                 << '|' << std::setw(13) << upper - lower << ' '
                 << '|' << std::string(9, ' ') << "l < epsilon" << std::string(9, ' ') << '|' << '\n'
                 << std::string(76, '_') << '\n'
                 << "x = " << std::fixed << std::setprecision(3)
                 << (lower + upper) / 2 << " +- " << (lower + upper) / 2 - lower << "\n\n";
        }
        void OptimalPassiveSearch(double lower, double upper) {
            cout << "Part 2. Optimal passive search :\n"
                 << std::string(39, '_') << '\n'
                 << '|' << std::string(4, ' ') << "Number of" << std::string(3, ' ') ///16|20
                 << '|' << std::string(3, ' ') << "The value of X" << std::string(3, ' ') <<"|\n"
                 << '|' << std::string(1, ' ') << "the points (N)" << std::string(1, ' ')
                 << '|' << std::string(3, ' ') << "in the minimum" << std::string(3, ' ') << "|\n"
                 << std::string(39, '_') << '\n';
            uint N = 1;
            double delta = (upper - lower) / (N + 1);
            double xForMinY;
            while (delta > EPSILON) {
                std::vector<double> VectorOfYk;
                delta = (upper - lower) / (N + 1);
                for (uint k = 1; k <= N; ++k) {
                    VectorOfYk.push_back(FunctionFromTask((upper - lower) / (N + 1) * k + lower));
                }
                    uint kForMinY = std::min_element(VectorOfYk.begin(), VectorOfYk.end()) -
VectorOfYk.begin() + 1;
                xForMinY = (upper - lower) / (N + 1) * kForMinY + lower;
                PrintPart2(N, xForMinY, delta);
                N++;
            }
            cout << std::string(39, '_') << '\n'
                 << "x = " << std::fixed << std::setprecision(3)
                 << xForMinY << " +- " << delta << "\n\n";
        }

        int main() {
            cout << "Variant 7: \t -sqrt(x) * sin(x) + 2 \t
         [" << LOWER_EDGE << "; " << UPPER_EDGE << "]\n";
            Dichotomy(LOWER_EDGE, UPPER_EDGE);  // Part 1. Dichotomy
           OptimalPassiveSearch(LOWER_EDGE, UPPER_EDGE); //Part 2.Optimal Passive Search
            return 0;
        }
```