**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ**

**Московский государственный технический университет**

**им. Н.Э. Баумана**

**(МГТУ им. Н.Э. Баумана)**

**Кафедра «Информационная безопасность» (ИУ8)**

Отчёт

Лабораторная работа № 1

По дисциплине: «Теория систем и системный анализ»

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

# Вариант 6

Выполнил: Гуща Н.В.,

студент группы ИУ8-32

Проверил: Коннова Н.С.

Доцент каф. ИУ8

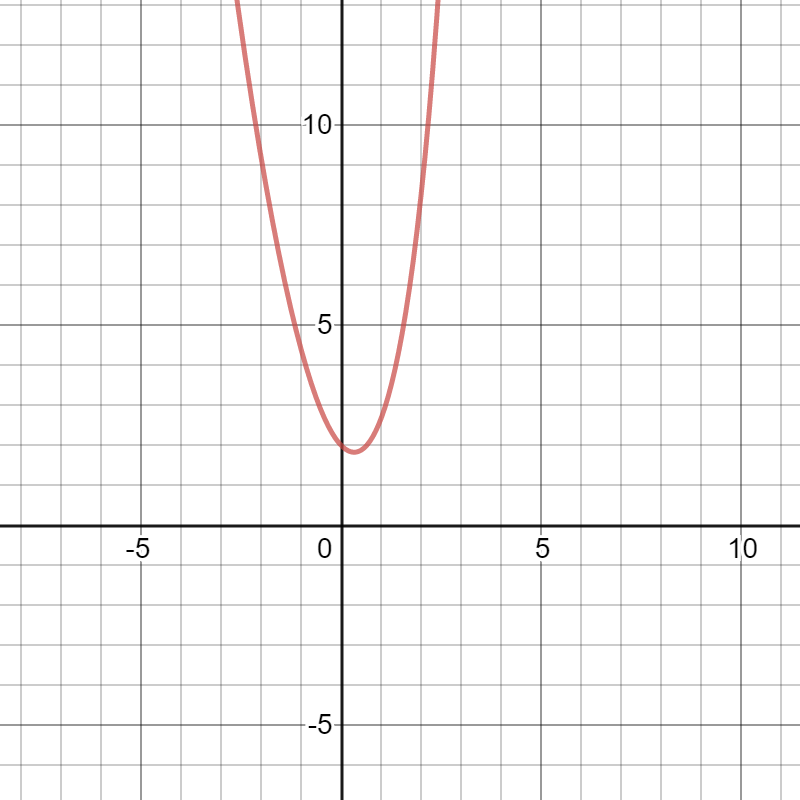
г. Москва 2020 г.

**Цель:**

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

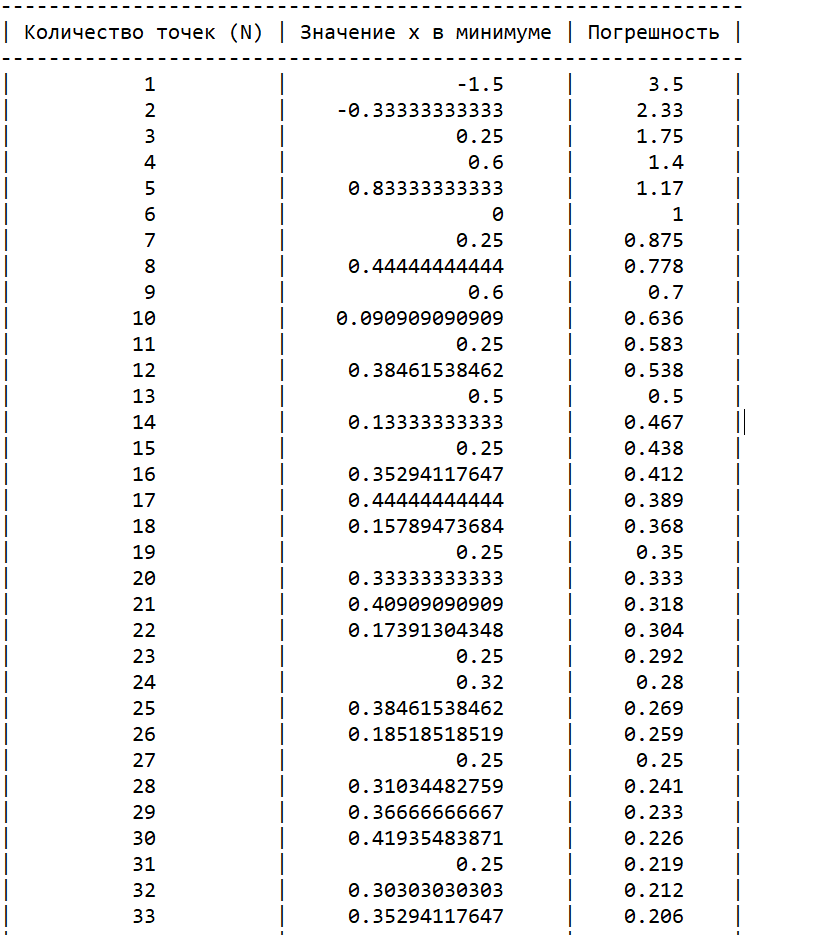
**Условие задачи:**

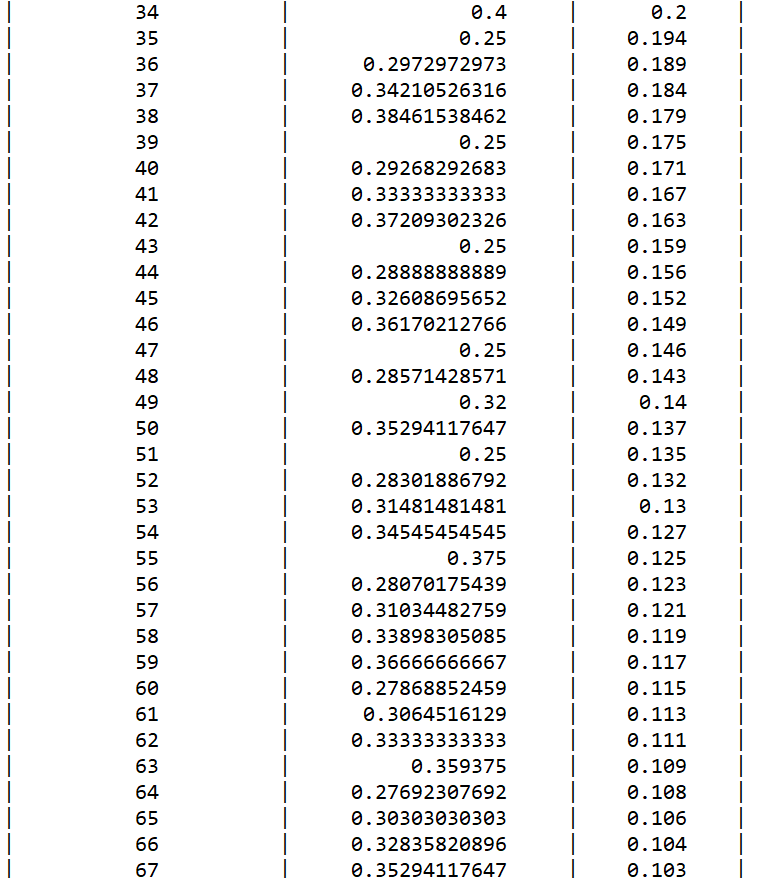
На интервале [-5 ,2] a b задана унимодальная функция одного переменного f x( ) = (1-x)^2 + exp(x) . Используя метод Фибоначчи, найти интервал нахождения минимума f x( ) при заданной наибольшей допустимой длине интервала неопределенности ε = 0,1 . Провести сравнение с методом оптимального пассивного поиска (п. 2.2 в [1]). Результат, в зависимости от числа точек разбиения N, представить в виде таблицы.

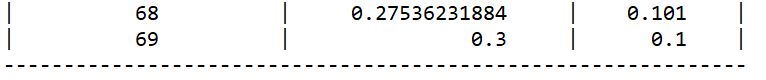


**Рисунок 1.** График функции f(x) на интервале [-5;2].

**Таблица 1.** Оптимальный пассивный поиск





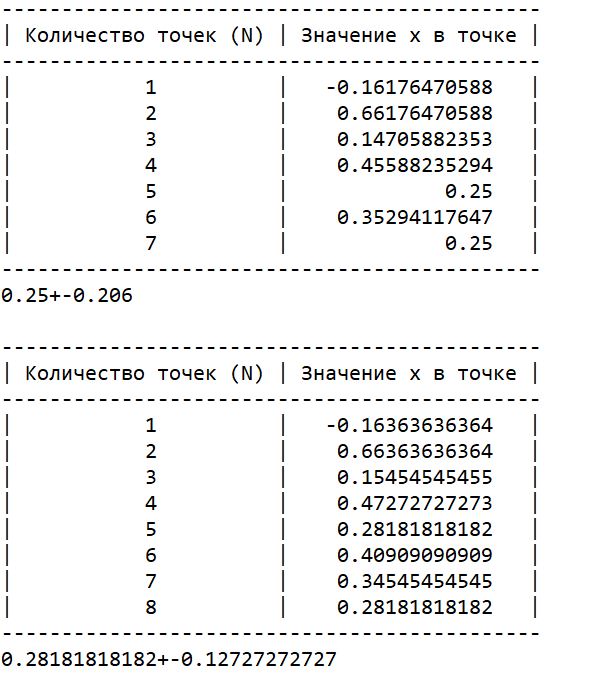


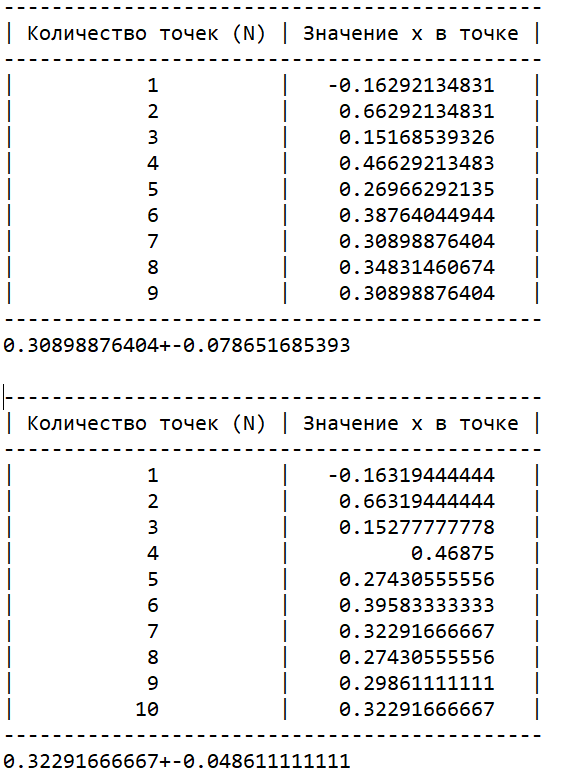
Минимальное значение функции достигается при x = 0.3 +- 0.1

**Таблица 2.** Метод Фибоначчи

N задается заранее

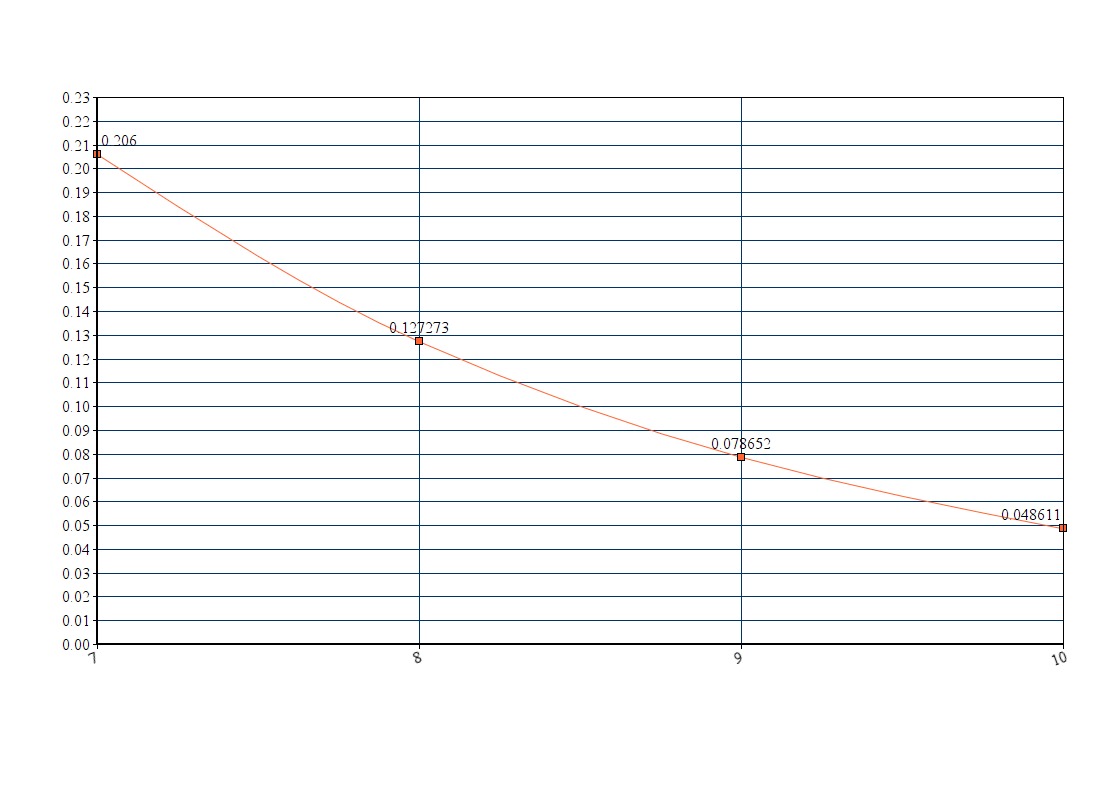
Применим этот метод при N от 7 до 10





**График зависимости погрешности от числа точек N для оптимального пассивного поиска**

**График зависимости погрешности от числа точек N для метода Фибоначчи**

****

**Код программы:**

//Кафедра "Информационная безопасность"

//ИУ8-32

//Гуща Н.В.

//Лабораторная работа №1(Теория систем и системный анализ)

#include <iostream>

#include <cmath>

#include <iomanip>

#include <vector>

#include <algorithm>

using std::cout;

using std::endl;

#define pogr 0.1

double function(const double& x) {

return ((1 - x) \* (1 - x) + exp(x));

}

std::vector<std::pair<double, double>> opt\_passive(const double& a, const double& b) {

std::vector<std::pair<double, double>> values;

size\_t N = 1;

double delta = (b - a) / (N + 1);

double x\_min\_y;

while (delta > pogr) {

std::vector<double> vec\_y;

delta = (b - a) / (N + 1);

for (size\_t k = 1; k <= N; ++k) {

vec\_y.push\_back(function((b - a) / (N + 1) \* k + a));

}

size\_t y\_min\_k = std::min\_element(vec\_y.begin(), vec\_y.end()) - vec\_y.begin() + 1;

x\_min\_y = (b - a) / (N + 1) \* y\_min\_k + a;

values.push\_back({ x\_min\_y, delta });

N++;

}

return values;

}

unsigned int Fib(const size\_t& n) {

if (n < 1)

return 0;

unsigned int f1 = 0, f2 = 1, fn = 0;

for (size\_t i = 1; i < n; ++i) {

fn = f1 + f2;

f1 = f2;

f2 = fn;

}

return fn;

}

std::vector<double> fib(size\_t N, double& a, double& b) {

std::vector<double> values;

double x1 = a + (b - a) \* Fib(N) / Fib(N + 2);

double x2 = a + b - x1;

double y1 = function(x1);

double y2 = function(x2);

while (N--) {

if (y1 > y2) {

a = x1;

x1 = x2;

x2 = b - (x1 - a);

y1 = y2;

y2 = function(x2);

values.push\_back(((b - a) / 2) + a);

}

else {

b = x2;

x2 = x1;

x1 = a + (b - x2);

y2 = y1;

y1 = function(x1);

values.push\_back(((b - a) / 2) + a);

}

}

cout <<"Погрешность равна "<<(b-a) <<endl;

return values;

}

void PrintValues(const std::vector<double>& values) {

cout << std::string(45, '-') << endl;

cout << std::setw(23) << std::left << "| Количество точек (N) " << "|";

cout << std::setw(24) << std::left << " Значение x в точке |" << endl;

cout << std::string(45, '-') << endl;

for (size\_t i = 0; i < values.size(); i++) {

cout << "|";

cout << std::setw(12) << std::right << i + 1;

cout << std::setw(11) << "|";

cout << std::setw(17) << std::right << std::setprecision(11) << values[i];

cout << std::setw(4) << "|" << endl;

}

cout << std::string(45, '-') << endl;

cout << values[values.size() - 1]<<endl<<endl;

}

void PrintValues(const std::vector<std::pair<double, double>>& values) {

cout << std::string(62, '-') << endl;

cout << std::setw(23) << std::left << "| Количество точек (N) " << "|";

cout << std::setw(23) << std::left << " Значение x в минимуме |";

cout << std::setw(15) << std::left << " Погрешность |" << endl;

cout << std::string(62, '-') << endl;

int i = 0;

double ans1, ans2;

for (auto const& val : values)

{

cout << "|";

cout << std::setw(12) << std::right << i + 1;

cout << std::setw(11) << "|";

cout << std::setw(18) << std::setprecision(11) << val.first << std::setw(6) << "|";

cout << std::setw(9) << std::setprecision(3) << val.second;

cout << std::setw(5) << "|" << endl;

i++;

ans1 = val.first;

ans2 = val.second;

}

cout << std::string(62, '-') << endl;

cout << ans1 << " +- " << ans2 << endl;

}

int main() {

setlocale(LC\_ALL, "Russian");

size\_t N = 7;

double a = -5.0, b = 2.0;

cout << "Вариант №6:" << endl;

cout << "Функция (1-x)^2 + exp(x) для интервала [-5, 2]" << endl << endl;

cout << "Первый метод: метод оптимального пассивного поиска" << endl;

cout << "Для погрешности 0,1" << endl;

PrintValues(opt\_passive(a, b));

cout << endl;

cout << "Второй метод: метод Фибоначчи" << endl;

cout << "N задается заранее" << endl;

cout << "Применим этот метод при N от 7 до 10 "<< endl;

PrintValues(fib(N, a, b));

N = 8;

a = -5.0, b = 2.0;

PrintValues(fib(N, a, b));

N = 9;

a = -5.0, b = 2.0;

PrintValues(fib(N, a, b));

N = 10;

a = -5.0, b = 2.0;

PrintValues(fib(N, a, b));

return 0;

}

**Вывод**

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

**Контрольный вопрос**

В чем состоит сущность метода оптимального пассивного поиска?

Минимаксный метод поиска, в котором информация о значениях функции, вычисленных в предшествующих точках, не может быть использована, называют оптимальным пассивным поиском. Оптимальный пассивный поиск состоит в выборе точек, равномерно расположенных на отрезке. При этом дает оценку скорости сходимости пассивного поиска с ростом числа N точек, так как скорость сходимости любого метода прямого поиска можно характеризовать скоростью уменьшения интервала неопределенности с возрастанием N.