|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 4

Выполнил: Велинецкий А. В.,

студент группы ИУ8-32

Проверил: Коннова Н. С.,

доцент каф. ИУ8

г. Москва,

2019 г.

# 1. 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.

# 2. Постановка задачи

Унимодальная функция:

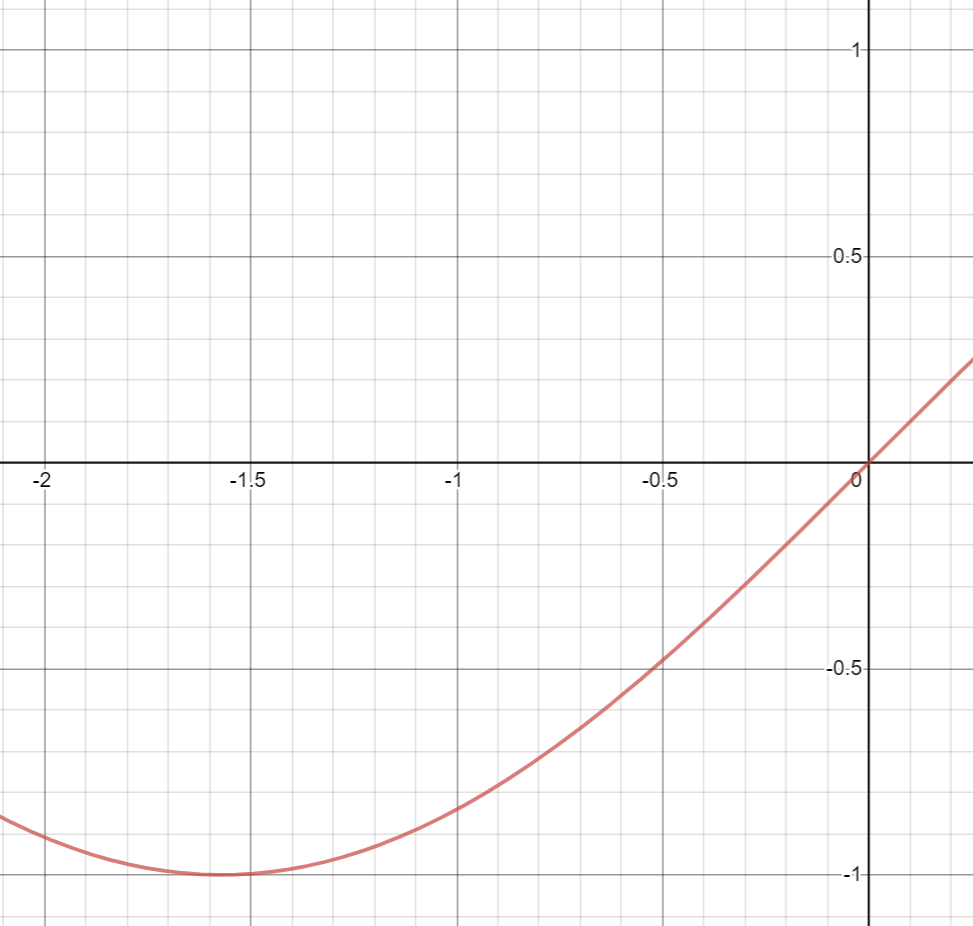
Отрезок поиска:

Методы поиска: Пассивный поиск, дихотомия.

Наибольшая длина интервала неопределенности:

# 3. Ход работы

Рисунок 1 демонстрирует график унимодальной функции.



**Рисунок 1** – график функции

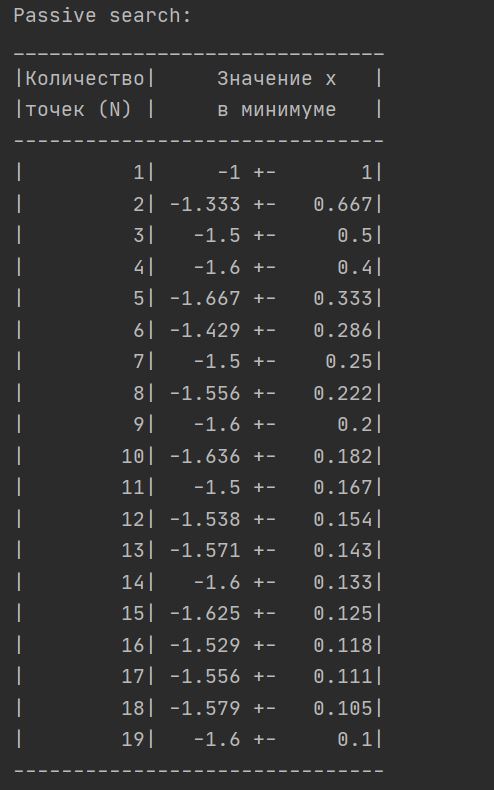
Реализация оптимального пассивного поиска:

Количество точек на 1 меньше, чем количество отрезков.

, где – наибольшая длина интервала неопределенности

Точки расположены равномерно по отрезку, следовательно, координата точки с номером :

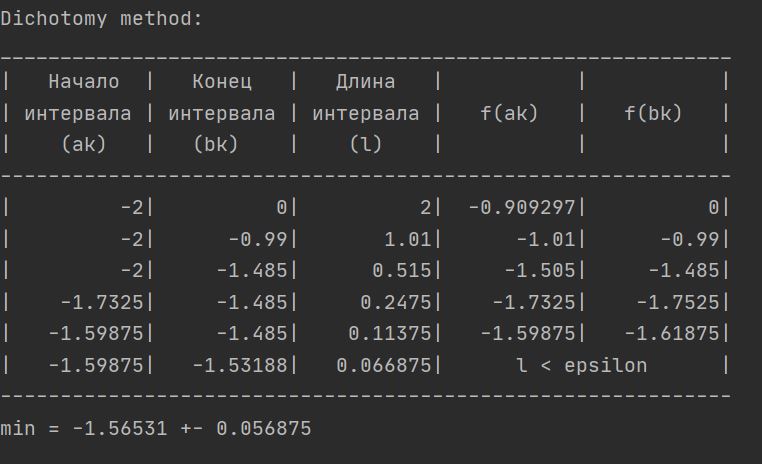
**Таблица 1** – применение оптимального пассивного поиска



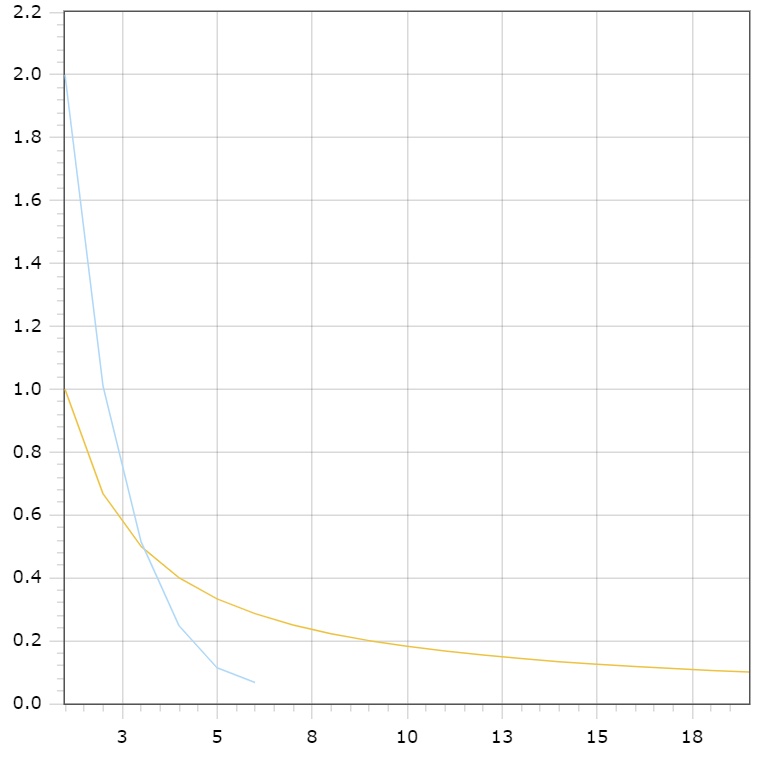
Реализация метода дихотомии:

Отрезок поиска делится пополам точкой x. Вычисляются значения функции на границах окрестности точки x: . Исключается левая половина, если значение функции в точке левой границы окрестности больше, чем в правой. Иначе, исключается правая половина. Эти действия повторяются до тех пор, пока отрезок поиска будет больше отрезка неопределенности.

**Таблица 4** – последовательный поиск (метод дихотомии)



На Рисунке 2 приведен график зависимостей погрешности от количества итераций.



Метод дихотомии; Пассивный поиск

**Рисунок 2** – график зависимости погрешности от количества итераций

# 4. Выводы

В ходе данной лабораторной работы мы научились применять алгоритмы пассивного поиска и дихотомии для поиска экстремума функций на примере унимодальной функции одного переменного. В результате дихотомии оказался эффективнее метода пассивного поиска при поиске экстремума унимодальной функции с наименьшей погрешностью.

# Приложение 1. Исходный код программы. Файл main.cpp

#include <cmath>  
#include <vector>  
#include <iostream>  
#include <algorithm>  
#include <iomanip>  
using namespace std;  
  
double myfun(double x) {  
 return (cos(x)\*tan(x));  
}  
  
int main() {  
 const float eps = 0.1;  
 double b = 0;  
 double a = -2;  
 unsigned int N = 1;  
 double x;  
 double current\_error = 1;  
  
 cout << "Passive search:" << endl;  
 cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n"  
 "|Количество| Значение x |\n"  
 "|точек (N) | в минимуме |\n"  
 "-------------------------------" << endl;  
 while (current\_error > eps){  
 double min = 1.7976931348623158\*pow(10, 308);  
 double minx = 0;  
 for (int k = 1; k < N + 1; k++) {  
 x = ((b - a) \* k / (N + 1)) + a;  
 if (myfun(x) < min){  
 min = myfun(x);  
 minx = x;  
 }  
 }  
 current\_error = (b - a) / (N + 1);  
 cout << "|" << setw(10) << N << "|" << setw(7) << round(minx\*1000)/1000 << " +- "  
 << setw(7) << round(current\_error\*1000)/1000 << "|" << endl;  
 N++;  
 }  
 cout << "-------------------------------" << endl;  
  
 const double delta = 0.01;  
 double F\_ak, F\_bk;  
 cout << "Dichotomy method:" << endl;  
 cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n"  
 "| Начало | Конец | Длина | | |\n"  
 "| интервала | интервала | интервала | f(ak) | f(bk) |\n"  
 "| (ak) | (bk) | (l) | | |\n"  
 "-------------------------------------------------------------" << endl;  
 cout << "|" << setw(11) << a << "|" << setw(11) << b << "|"  
 << setw(11) << b - a << "|" << setw(11) <<  
 myfun(a) << "|" << setw(11) << myfun(b) << "|" << endl;  
 double ak, bk;  
 while ((b - a) > eps){  
 F\_ak = (b + a)/2 - delta;  
 F\_bk = (b + a)/2 + delta;  
 ak = (myfun(F\_ak) < myfun(F\_bk))? F\_ak: F\_bk;  
 bk = (myfun(F\_ak) < myfun(F\_bk))? F\_bk: F\_ak;  
 a = (ak>bk)?ak:a;  
 b = (ak<=bk)?bk:b;  
 if ((b - a) < eps){  
 cout << "|" << setw(11) << a << "|" << setw(11) << b << "|"  
 << setw(11) << (b - a) << "|" << " l < epsilon |" << endl;  
 break;  
 }  
 cout << "|" << setw(11) << a << "|" << setw(11) << b << "|"  
 << setw(11) << (b - a) << "|" << setw(11) <<  
 ak << "|";  
 current\_error = 0.5 \* (b - a);  
 cout << setw(11) << bk << "|" << endl;  
 }  
 cout << "-------------------------------------------------------------" << endl;  
 cout << "min = " << (a + b) / 2 << " +- " << current\_error << "\n";  
 return 0;  
}