**Министерство образования Российской Федерации**

**МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**им. Н.Э. БАУМАНА**

Факультет: Информатика и системы управления Кафедра: Информационная безопасность (ИУ8)

МЕТОДЫ ОПТИМИЗАЦИИ

Лабораторная работа №1 на тему:

«Исследование методов прямого поиска экстремума унимодальной функции одного переменного»

Вариант 5

**Преподаватель:**

Коннова Н. С.

**Студент**:

Григорьев А.С.

**Группа:**

ИУ8-32

Москва 2020

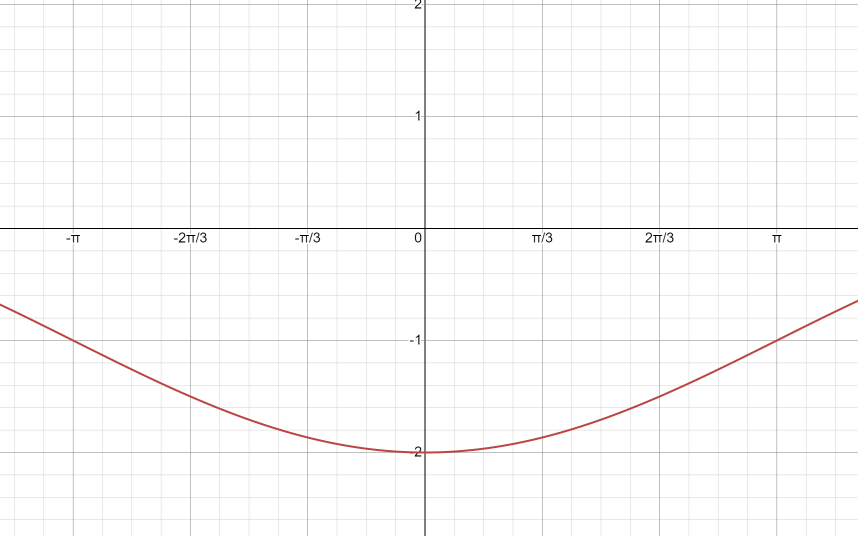
# Цель работы

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

# Постановка задачи

На интервале [-2, 4] задана унимодальная функция одного переменного f(x) = -cos (0,5x) - 1. Используя метод золотого сечения найти интервал нахождения минимума f(x) при заданной наибольшей допустимой длине интервала неопределенности **  0,1. Провести сравнение с методом оптимального пассивного поиска. Результат, в зависимости от числа точек разбиения N, представить в виде таблицы.

**Рис. 1** График функции f(x) = -cos (0,5x) - 1



# Ход работы

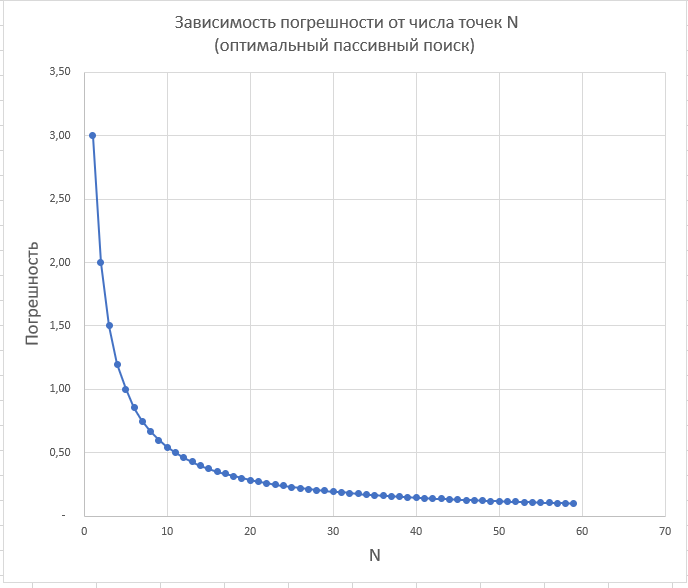
Рассмотрим оптимальный пассивный поиск. Здесь заданный отрезок делится на N+1 частей точками с координатами , где k = 1, …, N и вычисляется значение функции в каждой из этих точек. Для заданной функции получим следующий результат:

**Табл. 1** Оптимальный пассивный поиск

|  |  |
| --- | --- |
| Количество точек (N) | Значение x в минимуме |
| 1 | 1 +-3 |
| 2 | 0 +- 2 |
| 3 | -0.5 +- 1.5 |
| 4 | 0.4 +- 1.2 |
| 5 | 0 +- 1 |
| 6 | -0.285714 +- 0.857143 |
| 7 | 0.25 +- 0.75 |
| 8 | 0 +- 0.666667 |
| 9 | -0.2 +- 0.6 |
| 10 | 0.181818 +- 0.545455 |
| 11 | 0 +- 0.5 |
| 12 | -0.153846 +- 0.461538 |
| 13 | 0.142857 +- 0.428571 |
| 14 | 0 +- 0.4 |
| 15 | -0.125 +- 0.375 |
| 16 | 0.117647 +- 0.352941 |
| 17 | 0 +- 0.333333 |
| 18 | -0.105263 +- 0.315789 |
| 19 | 0.1 +- 0.3 |
| 20 | 0 +- 0.285714 |
| 21 | -0.0909091 +- 0.272727 |
| 22 | 0.0869565 +- 0.26087 |
| 23 | 0 +- 0.25 |
| 24 | -0.08 +- 0.24 |
| 25 | 0.0769231 +- 0.230769 |
| 26 | 0 +- 0.222222 |
| 27 | -0.0714286 +- 0.214286 |
| 28 | 0.0689655 +- 0.206897 |
| 29 | 0 +- 0.2 |
| 30 | -0.0645161 +- 0.193548 |
| 31 | 0.0625 +- 0.1875 |
| 32 | 0 +- 0.181818 |
| 33 | -0.0588235 +- 0.176471 |
| 34 | 0.0571429 +- 0.171429 |
| 35 | 0 +- 0.166667 |
| 36 | -0.0540541 +- 0.162162 |
| 37 | 0.0526316 +- 0.157895 |
| 38 | 0 +- 0.153846 |
| 39 | -0.05 +- 0.15 |
| 40 | 0.0487805 +- 0.146341 |
| 41 | 0 +- 0.142857 |
| 42 | -0.0465116 +- 0.139535 |
| 43 | 0.0454545 +- 0.136364 |
| 44 | 0 +- 0.133333 |
| 45 | -0.0434783 +- 0.130435 |
| 46 | 0.0425532 +- 0.12766 |
| 47 | 0 +- 0.125 |
| 48 | -0.0408163 +- 0.122449 |
| 49 | 0.04 +- 0.12 |
| 50 | 0 +- 0.117647 |
| 51 | -0.0384615 +- 0.115385 |
| 52 | 0.0377358 +- 0.113208 |
| 53 | 0 +- 0.111111 |
| 54 | -0.0363636 +- 0.109091 |
| 55 | 0.0357143 +- 0.107143 |
| 56 | 0 +- 0.105263 |
| 57 | -0.0344828 +- 0.103448 |
| 58 | 0.0338983 +- 0.101695 |
| 59 | 0 +- 0.1 |

При N = 59 достигается заданная неопределённость в методе оптимального пассивного поиска.

**Рис.** **2** Зависимость погрешности от числа точек N (оптимальный пассивный поиск)



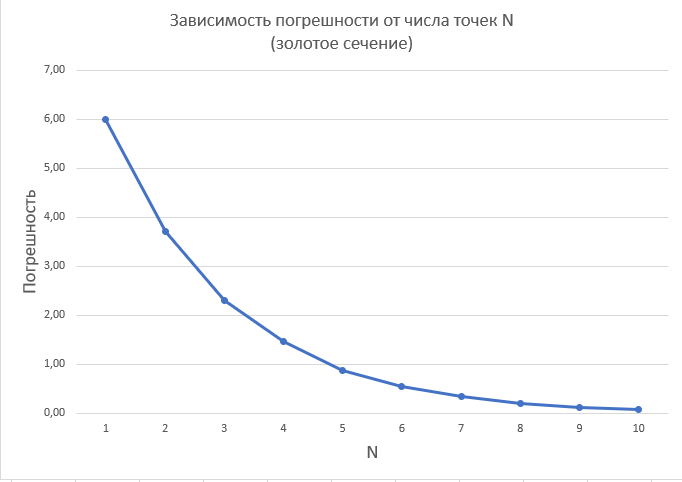
Рассмотрим теперь метод золотого сечения. Получим следующий результат:

**Табл. 2** Метод золотого сечения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Начало интервала () | Конец интервала () | Длина интервала () | f () | f () |
| -2 | 4 | 6 | -1.5403 | -0.583853 |
| -2 | 1.70828 | 3.70828 | -1.5403 | -1.65687 |
| -0.583611 | 1.70828 | 2.29189 | -1.95773 | -1.65687 |
| -0.583611 | 0.832887 | 1.4165 | -1.95773 | -1.91453 |
| -0.583611 | 0.291718 | 0.875329 | -1.95773 | -1.98938 |
| -0.249276 | 0.291718 | 0.540994 | -1.99224 | -1.98938 |
| -0.249276 | 0.0850838 | 0.33436 | -1.99224 | -1.9991 |
| -0.121566 | 0.0850838 | 0.20665 | -1.99815 | -1.9991 |
| -0.0425752 | 0.0850838 | 0.127659 | -1.99977 | -1.9991 |
| -0.0425752 | 0.0363241 | 0.0788992 | -1.99977 | -1.99984 |

Минимальное значение функции достигается при x = -0.00312555 +- 0.0788992

**Рис. 3** Зависимость погрешности от числа точек N (золотое сечение)



Как видно, уже на 10 шаге алгоритма достигается значение допустимого интервала неопределенности ** < 0,1. Заметно, что метод золотого сечения эффективнее метода пассивного поиска при поиске экстремума унимодальной функции одного переменного.

# Выводы

В ходе проделанной работы были освоены методы пассивного поиска и золотого сечения для нахождения экстремума функции. Было выяснено, что метод золотого сечения проделывает работу по поиску экстремума унимодальной функции одного переменного гораздо эффективнее, чем метод пассивного поиска.

# Вопросы

# Что такое интервал неопределенности?

# Приложение А.

*Файл ‘lab1.cpp’.*

Вариант 5. '''

#include <cmath>

#include <iostream>

#include <vector>

#include <algorithm>

double function(double x) {

return -cos(0.5 \* x) - 1;

}

int main() {

const double a = -2;

const double b = 4;

const double epsilon = 0.1;

int N = 1;

double x = 0;

double delta = 0; //погрешность

std::vector<double> x\_vec; //вектор из значений x и f(x)

std::vector<double> f\_vec;

std::cout << "Passive search: " << std::endl;

do {

for (int k = 1; k < N + 1; k++) {

x = (b - a) / (N + 1) \* k + a;

x\_vec.push\_back(x);

f\_vec.push\_back(function(x));

}

auto result = std::min\_element(f\_vec.begin(), f\_vec.end());

int index = std::distance(f\_vec.begin(), result);

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

std::cout << "N = " << N << " " << "x = " << x\_vec[index] << " +- " << delta << std::endl;

N++;

x\_vec.clear();

f\_vec.clear();

} while (delta > epsilon);

std::cout << std::endl;

std::cout << "Golden section search: " << std::endl;

const double t = 1.618;

double ak = a;

double bk = b;

double x1 = b - (b - a) / t;

double x2 = a + (b - a) / t;

double y1 = function(x1);

double y2 = function(x2);

do {

std::cout << "ak = " << ak << " " << "bk = " << bk << " " << "|bk - ak| = " <<

fabs(bk - ak) << " " << "f(ak) = " << function(ak) <<

" " << "f(bk) = " << function(bk) <<std::endl;

if (function(x1) < function(x2)) {

bk = x2;

x2 = x1;

x1 = bk - (bk - ak) / t;

}

else {

ak = x1;

x1 = x2;

x2 = ak + (bk - ak) / t;

}

} while (fabs(bk - ak) > epsilon);

std::cout << std::endl;

std::cout << "ak = " << ak << " " << "bk = " << bk << " " << "Xmin = " << (ak + bk) / 2 << " +- "

<< fabs(bk - ak) << " " << "f(ak) = " << function(ak) <<

" " << "f(bk) = " << function(bk) << std::endl;

}