**Министерство**

**образования**

**Российской**

**Федерации**

**МОСКОВСКИЙ**

**ГОСУДАРСТВЕННЫЙ**

**ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ**

**им**

**.**

**Н**

**.**

**Э**

**.**

**БАУМАНА**

Факультет

:

Информатика

и

системы

управления

Кафедра

:

Информационная

безопасность

(

ИУ

8)

**ТЕОРИЯ СИСТЕМ И**

**СИСТЕМНЫЙ АНАЛИЗ**

**Лабораторная**

**работа**

**№**

**1**

**на**

**тему**

**:**

«Исследование методов прямого поиска экстремума унимодальной функции одного переменного»

Вариант

11

**Преподаватель**

**:**

Коннова Н.С.

**Студент**

:

Меликян Р. О.

**Группа**

**:**

ИУ

8

-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 [a, b] задана унимодальная функция одного переменного

f (x) =-2 \* . Используя метод золотого сечения, найти интервал нахождения минимума f (x) при заданной наибольшей допустимой длине интервала неопределенности ε = 0,1. Провести сравнение с методом оптимального пассивного поиска. Результат, в зависимости от числа точек разбиения N, представить в виде таблицы.

# Ход работы

Найдем для заданной нам функции точку минимума методом оптимального пассивного поиска с погрешностью меньше ε. Результат работы программы (где N - это количество точек, min – значение х в минимуме):

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|Количество| Значение x

|точек (N) | в минимуме

|1 | 4 +- 2

|2 |3.33333 +- 1.33333

|3 | 4 +- 1

|4 | 3.6 +- 0.8

|5 | 4 +- 0.66667

|6 | 3.71429 +- 0.57143

|7 | 3.5 +- 0.5

|8 | 3.77778 +- 0.44444

|9 | 3.6 +- 0.4

|10 | 3.81818 +- 0.3636

|11 | 3.66667 +- 0.33333

|12 | 3.53846 +- 0.30769

|13 | 3.71429 +- 0.28571

|14 | 3.6 +- 0.266667

|15 | 3.75 +- 0.25

|16 | 3.64706 +- 0.23529

|17 | 3.77778 +- 0.22222

|18 | 3.68421 +- 0.21053

|19 | 3.6 +- 0.2

|20 | 3.71429 +- 0.19048

|21 | 3.63636 +- 0.1818

|22 | 3.73913 +- 0.17391

|23 | 3.66667 +- 0.16667

|24 | 3.6 +- 0.16

|25 | 3.69231 +- 0.15385

|26 | 3.62963 +- 0.14815

|27 | 3.71429 +- 0.14286

|28 | 3.65517 +- 0.13793

|29 | 3.73333 +- 0.13333

|30 | 3.67742 +- 0.12903

|31 | 3.625 +- 0.125

|32 | 3.69697 +- 0.1212

|33 | 3.64706 +- 0.11765

|34 | 3.71429 +- 0.11429

|35 | 3.66667 +- 0.11111

|36 | 3.62162 +- 0.10811

|37 | 3.68421 +- 0.10526

|38 | 3.64103 +- 0.10256

|39 | 3.7 +- 0.1



**Рис. 1** График зависимости погрешности от числа измерений(для первых 10 измерений) методом оптимального пассивного поиска.

При N = 39 достигается заданная неопределённость в методе оптимального пассивного поиска.

Теперь рассмотрим последовательный поиск методом золотого сечения. Реализовав его, получаем следующее:

|  |  |  |
| --- | --- | --- |
| Начало интервала () | Конец интервала () | Длина интервала (l) |
| 2 | 6 | 4 |
| 2 | 5.7082 | 3.7082 |
| 2 | 4.2918 | 2.2918 |
| 2.87539 | 4.2918 | 1.41641 |
| 3.41641 | 4.2918 | 0.87539 |
| 3.41641 | 3.95743 | 0.54102 |
| 3.41641 | 3.75078 | 0.334369 |
| 3.54413 | 3.75078 | 0.206651 |
| 3.62306 | 3.75078 | 0.127717 |
| 3.62306 | 3.70199 | 0.078934 |

Минимальное значение функции достигается при x = 3.66253 ± 0.0394669

Видно, что необходимую нам погрешность измерений, мы получаем уже на десятом шаге.



**Рис. 2** график зависимости погрешности от числа измерений методом золотого сечения

# 

**Рис. 3** График заданной функции f(x)

# Выводы

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

**Приложение А.**

*Файл ‘main.cpp’.*

#include <iostream>

#include <cmath>

double func(double x){

return std::sqrt(x) \* std::sin(x \* 0.5) \* (-2);

}

int main(int argc, const char \* argv[]) {

double a\_k = 2;

double b\_k = 6;

double l = b\_k - a\_k;

const double e = 0.1;

const double t = (std::sqrt(5) + 1) / 2;

double x\_k1 = a\_k + (1 - 1 / t) \* b\_k;

double x\_k2 = a\_k + b\_k / t;

double f\_k1 = func(x\_k1);

double f\_k2 = func(x\_k2);

std::cout << "Golden ration method" << std::endl;

while (l > e){

std::cout << "a = " << a\_k << " b = " << b\_k << " l = " << l << std::endl;

if (f\_k1 < f\_k2){

b\_k = x\_k2;

x\_k2 = a\_k + b\_k - x\_k1;

f\_k2 = func(x\_k2);

} else {

a\_k = x\_k1;

x\_k1 = a\_k + b\_k - x\_k2;

f\_k1 = func(x\_k1);

}

if (x\_k1 > x\_k2){

std::swap(x\_k1, x\_k2);

std::swap(f\_k1, f\_k2);

}

l = b\_k - a\_k;

}

std::cout << "a = " << a\_k << " b = " << b\_k << " l = " << l << std::endl;

double s = 0.5 \* (b\_k - a\_k);

double x\_sol = (a\_k + b\_k) / 2;

std::cout << "Minimum x = " << x\_sol << " +- " << s << " l = " << l << std::endl;

a\_k = 2;

b\_k = 6;

double pres = (b\_k - a\_k) / 2;

int n = 1;

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

while(pres >= e){

double x, x\_min, func\_curr;

double func\_min = func(a\_k);

for (auto i = 1; i <= n; i++){

x = (b\_k - a\_k) \* i / (n + 1) + a\_k;

func\_curr = func(x);

if (func\_curr < func\_min){

func\_min = func\_curr;

x\_min = x;

}

}

std::cout << "n = " << n << " minimun x = " << x\_min << " +- " << pres << std::endl;

n++;

pres = (b\_k - a\_k) / (n + 1);

}

return 0;

}

Ссылка на Github - https://github.com/Robert29000/lab01\_ts