МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное учреждения высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт компьютерных технологий и информационной безопасности

Кафедра математического обеспечения и применения ЭВМ

  

**ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ**

по дисциплине

**«Методы оптимизации»**

Выполнил:

Студент группы

КТбо3-1 Самардак А.В

Проверил:

Доцент кафедры

САИТ Липко Ю.Ю

Оценка

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

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

Цель работы

Разработать алгоритм и программу решения задачи нахождения экстремума функции методами дихотомии и «золотого сечения». Сравнить полученные результаты.

Теоретический материал

Общий алгоритм работы метода дихотомии

Дана функция Необходимо найти , доставляющий минимум (или максимум) функции на интервале с заданной точностью ε, т.е. найти

Алгоритм:

На каждом шаге процесса поиска делим отрезок пополам, - координата середины отрезка

Вычисляем значение функции F(x) в окрестности вычисленной точки , т.е. .

Сравниваем и и отбрасываем одну из половинок отрезка

* При поиске минимума:
  + - Если , то отбрасываем отрезок , тогда .
    - Иначе отбрасываем отрезок , тогда .
* При поиске максимума:
  + - Если , то отбрасываем отрезок , тогда .
    - Иначе отбрасываем отрезок , тогда .

Деление отрезка продолжается, пока его длина не станет меньше заданной точности , т.е. .

Общий алгоритм матода золотого сечения

Алгоритм:

Задаются начальные границы отрезка a, b, точность ε и константа Φ = 1.618.

Рассчитывают начальные точки деления:

Значения .

* Если (для поиска максимума изменить знак неравенства), то
* Иначе

Если , то и заканчиваем расчёт. Иначе возвращаемся на пункт 2.

Реализация на ЯП (C#)

Функция реализующая метод дихотомии

public override void Min(double a, double b, double eps, double L\_astr)

{

double x1, x2;

while (Math.Abs(a - b) > L\_astr)

{

x1 = a + (b - a) / 2 - eps / 2;

x2 = a + (b - a) / 2 + eps / 2;

if (Func(x1) < Func(x2))

{

b = x2;

}

if (Func(x1) > Func(x2))

{

a = x1;

}

Console.WriteLine("Fmin = F(" + (x1 + x2) / 2 + ") = " + Func((x1 + x2) / 2));

}

}

public override void Max(double a, double b, double eps, double L\_astr)

{

double x1, x2;

while (Math.Abs(a - b) > L\_astr)

{

x1 = a + (b - a) / 2 - eps / 2;

x2 = a + (b - a) / 2 + eps / 2;

if (Func(x1) < Func(x2))

{

a = x1;

}

if (Func(x1) > Func(x2))

{

b = x2;

}

Console.WriteLine("Fmax = F(" + (x1 + x2) / 2 + ") = " + Func((x1 + x2) / 2));

}

}

Функция реализующая метод золотого сечения

public override void Min(double a, double b, double eps, double L\_astr)

{

const double FI = 1.618;

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

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

while (Math.Abs(b - a) > eps)

{

if (Func(x1) >= Func(x2))

{

a = x1;

x1 = x2;

x2 = b - (x1 - a);

}

else

{

b = x2;

x2 = x1;

x1 = a + (b - x2);

}

Console.WriteLine("Fmin = F(" + (x1 + x2) / 2 + ") = " + Func((x1 + x2) / 2));

}

}

public override void Max(double a, double b, double eps, double L\_astr)

{

const double FI = 1.618;

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

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

while (Math.Abs(b - a) > eps)

{

if (Func(x1) <= Func(x2))

{

a = x1;

x1 = x2;

x2 = b - (x1 - a);

}

else

{

b = x2;

x2 = x1;

x1 = a + (b - x2);

}

Console.WriteLine("Fmax = F(" + (x1 + x2) / 2 + ") = " + Func((x1 + x2) / 2));

}

}

Результат работы программы

Изображение выглядит как текст

Автоматически созданное описание

ЛИСТИНГ ПРОГРАММЫ

**Main.cs**

using System;

namespace MO\_IndividualTask

{

internal class MO\_Main

{

static void Main(string[] args)

{

Console.WriteLine("Dichotomy:");

Search dichotomy = new Dichotomy();

dichotomy.Max(0, 10, 0.5, 1);

Console.WriteLine("GoldenRatio:");

Search goldenRatio = new GoldenRatio();

goldenRatio.Min(0, 10, 0.5, 1);

}

}

}

**Dichotomy.Cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace MO\_IndividualTask

{

internal class Dichotomy : Search

{

public override void Min(double a, double b, double eps, double L\_astr)

{

double x1, x2;

while (Math.Abs(a - b) > L\_astr)

{

x1 = a + (b - a) / 2 - eps / 2;

x2 = a + (b - a) / 2 + eps / 2;

if (Func(x1) < Func(x2))

{

b = x2;

}

if (Func(x1) > Func(x2))

{

a = x1;

}

Console.WriteLine("Fmin = F(" + (x1 + x2) / 2 + ") = " + Func((x1 + x2) / 2));

}

}

public override void Max(double a, double b, double eps, double L\_astr)

{

double x1, x2;

while (Math.Abs(a - b) > L\_astr)

{

x1 = a + (b - a) / 2 - eps / 2;

x2 = a + (b - a) / 2 + eps / 2;

if (Func(x1) < Func(x2))

{

a = x1;

}

if (Func(x1) > Func(x2))

{

b = x2;

}

Console.WriteLine("Fmax = F(" + (x1 + x2) / 2 + ") = " + Func((x1 + x2) / 2));

}

}

}

}

**GoldenRatio.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace MO\_IndividualTask

{

internal class GoldenRatio : Search

{

public override void Min(double a, double b, double eps, double L\_astr)

{

const double FI = 1.618;

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

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

while (Math.Abs(b - a) > eps)

{

if (Func(x1) >= Func(x2))

{

a = x1;

x1 = x2;

x2 = b - (x1 - a);

}

else

{

b = x2;

x2 = x1;

x1 = a + (b - x2);

}

Console.WriteLine("Fmin = F(" + (x1 + x2) / 2 + ") = " + Func((x1 + x2) / 2));

}

}

public override void Max(double a, double b, double eps, double L\_astr)

{

const double FI = 1.618;

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

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

while (Math.Abs(b - a) > eps)

{

if (Func(x1) <= Func(x2))

{

a = x1;

x1 = x2;

x2 = b - (x1 - a);

}

else

{

b = x2;

x2 = x1;

x1 = a + (b - x2);

}

Console.WriteLine("Fmax = F(" + (x1 + x2) / 2 + ") = " + Func((x1 + x2) / 2));

}

}

}

}

**Search.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace MO\_IndividualTask

{

abstract class Search

{

public abstract void Min(double a, double b, double eps, double L\_astr);

public abstract void Max(double a, double b, double eps, double L\_astr);

public virtual double Func(double x)

{

return 12 \* x - 2 \* x \* x;

}

}

}