МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «Програмна інженерія та інформаційні технології управління»

Звіт з лабораторної роботи №1

з дисципліни «Дослідження операцій»

Виконав:

Студент групи КН-36а

Рубан Ю. Д.

Перевірив:

Гужва В. О.

Харків – 2018

**Ціль роботи:** Дослідити алгоритми золотого поділу та кубічної апроксимації.

**Алгоритми досліджуваних методів:**

**Метод золотого поділу:**

1. Покласти відрізок [a, b], точність eps, та досліджувану функцію.
2. Розрахувати
3. Розрахувати
4. Розрахувати
5. Якщо , то пошук завершено, , інакше перейти на крок 6.
6. Якщо , то

, перейти на крок 5

Інакше, якщо , то

, перейти на крок 5

Інакше, якщо , то алгоритм завершено,

**Метод кубічної апроксимації:**

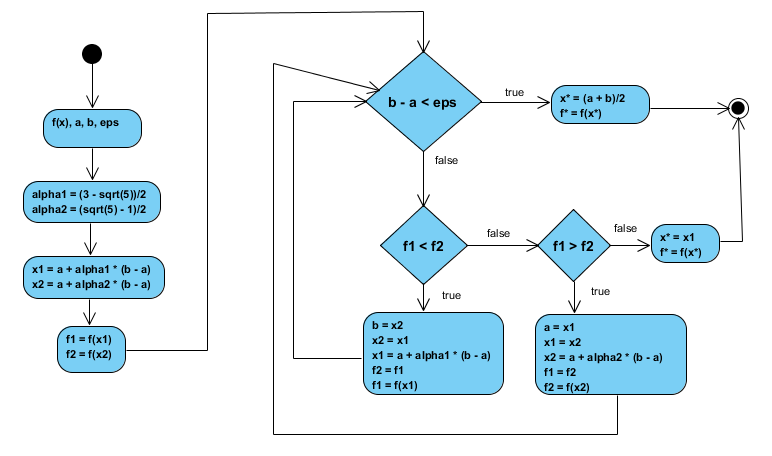
1. Покласти відрізок [a, b], точність eps1, eps2, крок , та досліджувану функцію.
2. Задати
3. Якщо , то , інакше
4. k = k + 1
5. Якщо , то та перейти на крок 3
6. Розрахувати
7. Розрахувати , де
8. Якщо то і перейти на крок 8, інакше
9. Якщо та , то алгоритм завершено,

, інакше перейти на крок 10.

1. Якщо , то та перейти на крок 6, інакше та перейти на крок 6.

**Блок схеми алгоритмів:**

Блок схема алгоритму золотого поділу показана на рисунку 1. Блок схема алгоритму кубічної апроксимації показана на рисунку 2.

Рисунок 1 – Блок схема алгоритму золотого поділу

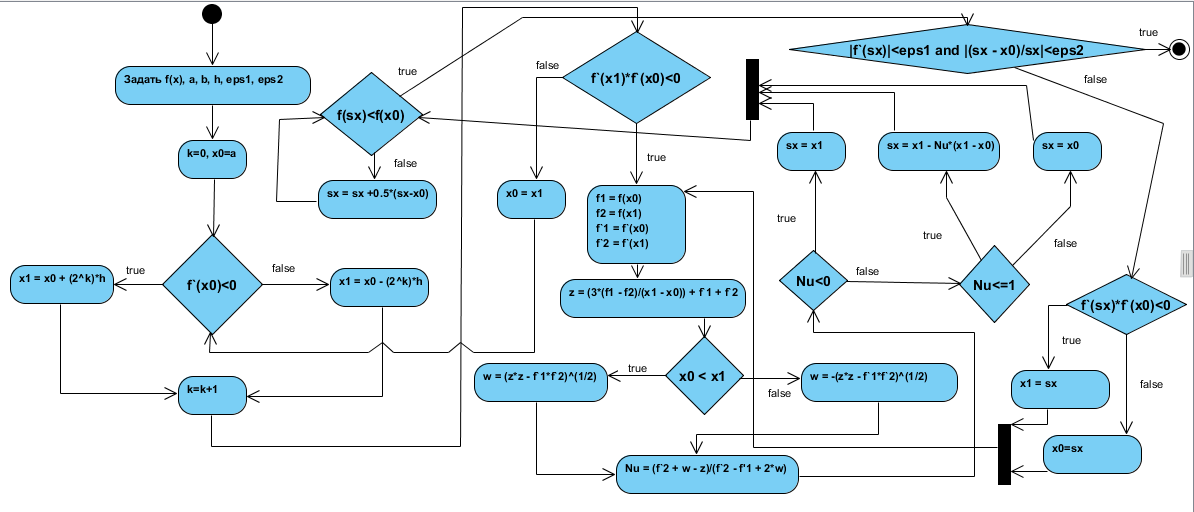


Рисунок 2 – Блок схема алгоритму кубічної апроксимації

**Програми досліджуваних алгоритмів:**

Лістинг програми алгоритму золотого поділу:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace AproximationMethodsLib

{

public class GoldenSectionMethod : IExtremumMethod

{

public Range Range { get; set; }

private double eps;

private double alpha1, alpha2;

public double Eps { get { return eps; } set

{

if (value < 0 || value > 1)

throw new ArgumentException();

eps = value;

}

}

public GoldenSectionMethod(Range range, double eps=0.0001)

{

Range = range;

Eps = eps;

alpha1 = (3d - Math.Sqrt(5)) / 2d;//0.382

alpha2 = (Math.Sqrt(5) - 1) / 2d;//0.618

}

public double FindExtremum(Func<double, double> func, ExtremumType extremumType, Logger logger = null)

{

Func<double, double> f = extremumType == ExtremumType.Min ? func : (arg) => -func(arg);

logger?.Log("Характер экстремума: " + Enum.GetName(typeof(ExtremumType), extremumType));

logger?.Log("Интервал поиска: от " + Range.Begin + ", до " + Range.End);

logger?.Log("Точность: eps = " + Eps);

double a = Range.Begin, b = Range.End;

double x1, x2;

x1 = a + alpha1 \* (b - a);

x2 = a + alpha2 \* (b - a);

double f1 = f(x1), f2 = f(x2);

double k = 0;

while (true)

{

k++;

if (f1 < f2)

{

b = x2;

x2 = x1;

x1 = a + alpha1 \* (b - a);

f2 = f1;

f1 = f(x1);

}

else if(f1 > f2)

{

a = x1;

x1 = x2;

x2 = a + alpha2 \* (b - a);

f1 = f2;

f2 = f(x2);

}

else if (f1 == f2)

{

logger?.Log("Поиску понадобилось " + k + " итераций");

logger?.Log("Алгоритм завершен!");

logger?.Log("Функция принимает экстримальное значение в точке: \nx\* = " + x1 + "\nf(x\*) = " + f(x1));

return x1;//f1==f2==min

}

if (b - a < Eps)

{

double res = (b + a) / 2;

logger?.Log("Поиску понадобилось " + k + " итераций");

logger?.Log("Алгоритм завершен!");

logger?.Log("Функция принимает экстримальное значение в точке: \nx\* = " + res + "\nf(x\*) = " + f(res));

return res;

}

}

}

}

}

Лістинг програми алгоритму кубічної апроксимації:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace AproximationMethodsLib

{

public class CubicAproximationMethod : IExtremumMethod

{

private double step;

private double eps1, eps2;

private Func<double, double> df;

public Range Range { get; set; }

public double Step

{

get

{

return step;

}

set

{

if (value <= 0)

throw new ArgumentException();

step = value;

}

}

public double Eps1

{

get

{

return eps1;

}

set

{

if (value <= 0 || value > 1)

throw new ArgumentException();

eps1 = value;

}

}

public double Eps2

{

get

{

return eps2;

}

set

{

if (value <= 0 || value > 1)

throw new ArgumentException();

eps2 = value;

}

}

public CubicAproximationMethod(Range range, double step=0.0001, double eps1=0.0001, double eps2= 0.0001)

{

Range = range;

Step = step;

Eps1 = eps1;

Eps2 = eps2;

}

public double FindExtremum(Func<double, double> func, ExtremumType extremumType, Logger logger)

{

Func<double, double> f;

if (extremumType == ExtremumType.Max)

{

f = (x) => -func(x);

}

else

{

f = func;

}

logger?.Log("Характер экстремума: " + Enum.GetName(typeof(ExtremumType), extremumType));

logger?.Log("Интервал поиска: от " + Range.Begin + ", до "+ Range.End);

logger?.Log("Точность: eps1 = " + Eps1 + ", eps2 = " + Eps2);

df = Derevation.FirstDer(f);

double x0 = Range.Begin, x1;

int k = 0;

logger?.Log("Поиск значений xm и xm+1 при которых f`(xm)\*f`(xm+1)<0");

do

{

int power =(int) Math.Pow(2, k);

if (df(x0) < 0)

{

x1 = x0 + power \* step;

}

else

{

x1 = x0 - power \* step;

}

k++;

if (df(x1) \* df(x0) < 0)

{

break;

}

else

{

x0 = x1;

}

if(x0 < Range.Begin || x0>Range.End)

{

logger?.Log("Невозможно найти экстремум");

throw new ArithmeticException();

}

} while (true);

logger?.Log("Поиску понадобилось " + k + " итераций");

logger?.Log("Поиск значения sx, при котором выполняються условия завершения алгоритма");

int i = 0;

double sx = 0;

while(true)

{

double f1 = f(x0);

double f2 = f(x1);

double df1 = df(x0);

double df2 = df(x1);

sx = FindSX(x0, x1, f1, f2, df1, df2);

while (true)

{

if (f(sx) < f(x0))

break;

sx = sx + 0.5 \* (sx - x0);

i++;

}

if (Math.Abs(df(sx)) <= Eps1 && Math.Abs((sx - x0) / sx) <= Eps2)

break;

if (df(sx) \* df(x0) < 0)

{

x1 = sx;

}

else

{

x0 = sx;

}

i++;

}

logger?.Log("Поиску понадобилось "+i+" итераций");

logger?.Log("Алгоритм завершен!");

logger?.Log("Всего итераций: " + (i + k));

logger?.Log("Функция принимает экстримальное значение в точке: \nx\* = " + sx + "\nf(x\*) = " + f(sx));

return sx;

}

private static double FindSX(double x1, double x2, double f1, double f2, double df1, double df2)

{

double nu = FindNu(x1, x2, f1, f2, df1, df2);

if (nu < 0) return x2;

else if (nu <= 1) return x2 - nu \* (x2 - x1);

else return x1;

}

private static double FindNu(double x1, double x2, double f1, double f2, double df1, double df2)

{

double z = FindZ(x1, x2, f1, f2, df1, df2);

double w = FindW(x1, x2, f1, f2, df1, df2);

return (df2 + w - z) / (df2-df1+2\*w);

}

private static double FindZ(double x1, double x2, double f1, double f2, double df1, double df2)

{

return ((3 \* (f1 - f2) / (x2 - x1)) + df1 + df2);

}

private static double FindW(double x1, double x2, double f1, double f2, double df1, double df2)

{

double z = FindZ(x1, x2, f1, f2, df1, df2);

double res = Math.Pow(((z \* z) - df1 \* df2), 0.5);

return x1 < x2 ? res : -res;

}

}

}

**Результати роботи програм:**

Результат роботи програми золотого поділу показаний на рисунку 3. Результат роботи програми кубічної апроксимації показаний на рисунку 4.

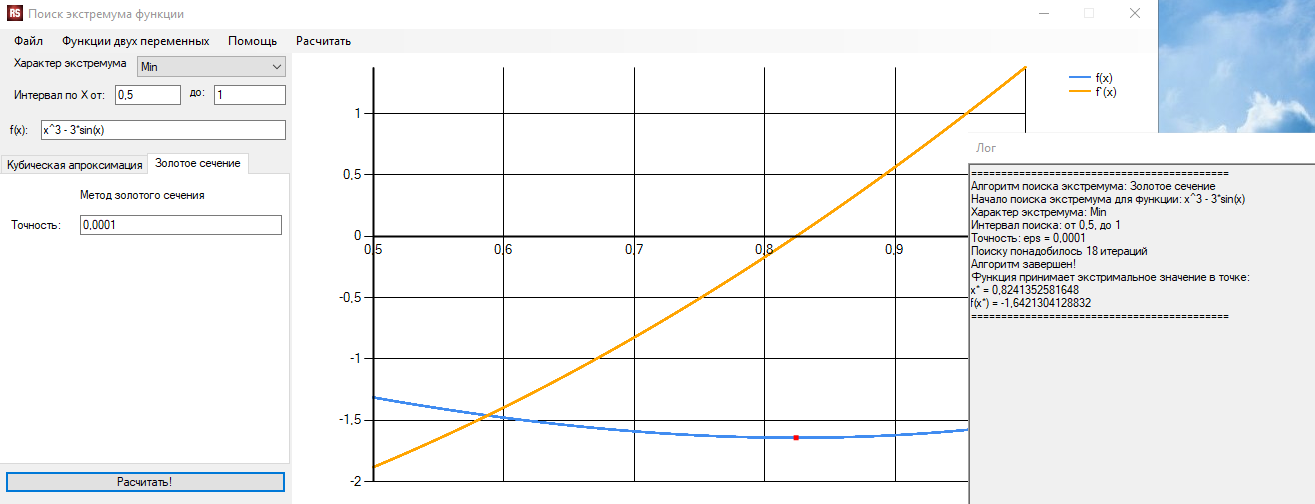


Рисунок 3 – Результат роботи програми методом золотого поділу

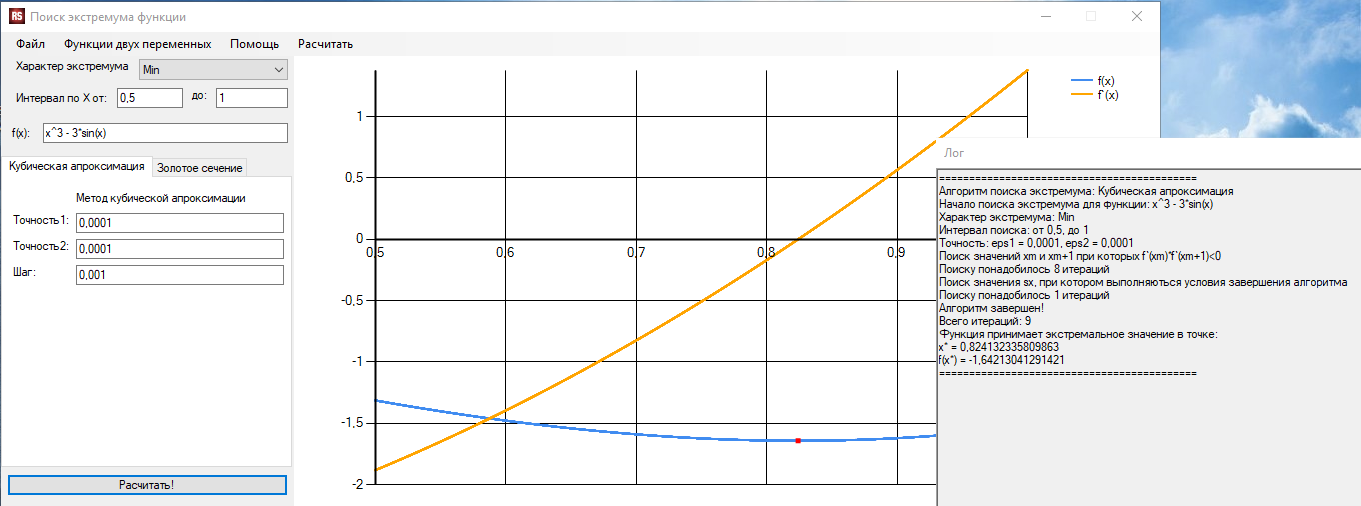


Рисунок 4 – Результат роботи програми методом кубічної апроксимації

**Порівняння алгоритмів:**

Порівняння алгоритмів показано у таблиці 1.

Таблиця 1 – Порівняння алгоритмів.

|  |  |  |
| --- | --- | --- |
| Алгоритм: | Золотий поділ | Кубічна апроксимація |
| Задана функція | x^3 – 3\*sin(x) | x^3 – 3\*sin(x) |
| Значення мінімуму | x\* = 0,824135, f(x\*) = -1,64213 | x\* = 0,824135, f(x\*) = -1,64213 |
| Точність | eps = 0,0001 | eps1 = eps2 = 0,0001 |
| Кількість ітерацій | 18 | 10 |

**Висновок:**

У даній лабораторній роботі було досліджено 2 алгоритми: алгоритм золотого поділу, та алгоритм кубічної апроксимації. Практичним засобом було встановлено, що алгоритми знаходять однаковий результат при однаковій точності, але алгоритм золотого поділу потребує більшої кількості ітерації. Отже алгоритм кубічної апроксимації дає точний результат за менший час.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace AproximationMethodsLib

{

public interface IExtremumMethod

{

double FindExtremum(Func<double, double> func, ExtremumType extremumType, Logger logger = null);

}

public interface IMultipleExtremumMethod

{

IEnumerable<double> FindExtremum(Func<IEnumerable<double>,double>func,int n , ExtremumType extremumType, Logger logger = null);

}

}

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace AproximationMethodsLib

{

public class Logger

{

private Stream stream;

public Stream Stream { get { return stream; } set

{

Writer = new StreamWriter(value);

Reader = new StreamReader(value);

stream = value;

} }

private StreamWriter Writer { get; set; }

private StreamReader Reader { get; set; }

public Logger(Stream stream)

{

Stream = stream;

}

public void LogBeginOrEnd()

{

Log("===========================================");

}

public void Log(string msg)

{

Writer.WriteLine(msg);

Writer.Flush();

}

public string GetLog()

{

Stream.Position = 0;

if (Stream.CanRead)

{

string log = Reader.ReadToEnd();

return log;

}

return null;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace AproximationMethodsLib

{

public struct Range

{

public double Begin { get; }

public double End { get; }

public Range(double begin, double end)

{

if (begin > end)

throw new ArgumentException();

Begin = begin;

End = end;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace AproximationMethodsLib

{

public static class Derevation

{

public static Func<double,double> FirstDer(Func<double,double>func)

{

double eps = 0.00000001;

return (x) => ((func(x + eps) - func(x)) / eps);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace AproximationMethodsLib

{

public enum ExtremumType

{

Min,

Max

}

}