МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «Програмна інженерія та інформаційні технології управління»

Звіт з лабораторної роботи №3

з дисципліни «Дослідження операцій»

Виконав:

Студент групи КН-36а

Рубан Ю. Д.

Перевірив:

Проф. Гужва В. О.

Харків – 2018

**Ціль роботи:** Знайти рішення задачі безумовної мінімізації методом Флетчера Рівса.

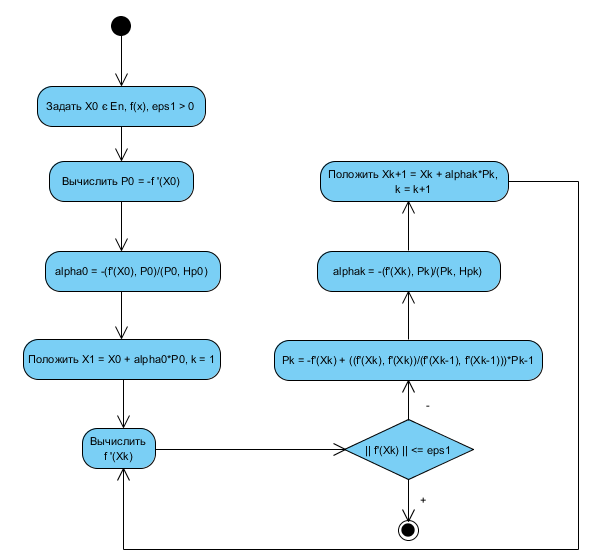
Задача:

n = 3

Алгоритм методу:

1. Обрати початкове наближення є En та Eps>0
2. Розрахувати P0 =
3. Розрахувати крок α0 =
4. Покласти k=1,
5. Розрахувати
6. Якщо то завершити алгоритм. , інакше перейти на крок 7.
7. Розрахувати
8. Розрахувати крок
9. Покласти k=k+1, та перейти на крок 5.

Блок схема алгоритму показана на рисунку 1.

Рисунок 1 – Блок схема алгоритму Флетчера Рівса

Код програми показаний у лістингу 1.

Лістинг 1. Код програми, яка реалізує алгоритм Флетчера Рівса

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace AproximationMethodsLib

{

public class FletcherRivz

{

private double Norm(IEnumerable<double> a)

{

double res = 0;

foreach (double e in a)

{

res += e \* e;

}

return Math.Sqrt(res);

}

private double VecMul(List<double> a, List<double> b)

{

double res = 0;

int n = a.Count;

for (int i = 0; i < n; i++)

{

res += a[i] \* b[i];

}

return res;

}

private List<double> VecSum(List<double> a, List<double> b)

{

List<double> res = new List<double>();

int n = a.Count;

for (int i = 0; i < n; i++)

{

res.Add(a[i] + b[i]);

}

return res;

}

private List<double> MatrixVectorMul(List<List<double>> matrix, List<double> vector)

{

double[] res = new double[vector.Count];

for (int row = 0; row < matrix.Count; row++)

{

for (int col = 0; col < matrix[row].Count; col++)

{

res[col] += matrix[row][col] \* vector[row];

}

}

return res.ToList();

}

IEnumerable<double> X0;

double Eps;

public FletcherRivz(IEnumerable<double> x0, double eps)

{

X0 = x0;

Eps = eps;

}

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

{

var f = func;

if (extremumType == ExtremumType.Max)

{

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

}

var grad = Derevation.Gradient(f);

var pFunc = new Func<IEnumerable<double>, IEnumerable<double>>(delegate (IEnumerable<double> x)

{

return grad(x).Select(y => -y);

});

var H = Derevation.SecondDerMatrix(f);

List<IEnumerable<double>> Pk = new List<IEnumerable<double>>();

List<double> AlphaK = new List<double>();

List<IEnumerable<double>> Xk = new List<IEnumerable<double>>();

Xk.Add(X0.Select(x => x));

int k = 0;

Pk.Add(pFunc(Xk[k]));

AlphaK.Add(-((VecMul(grad(Xk[k]).ToList(),Pk[k].ToList()))/VecMul(MatrixVectorMul(H(Xk[k]).Select(x => x.ToList()).ToList(),Pk[k].ToList()),Pk[k].ToList())));

Xk.Add(VecSum(Xk[k].ToList(), Pk[k].Select(x => x \* AlphaK[k]).ToList()));

logger.LogVector(f, Xk[k], "x" + k.ToString());

var norm = Norm(grad(Xk[k+1]));

k++;

if (Math.Abs(norm) < Eps)

{

return Xk[k];

}

else

{

while (true)

{

Pk.Add(VecSum(pFunc(Xk[k]).ToList(), Pk[k - 1].Select(x => x \* VecMul(grad(Xk[k]).ToList(), grad(Xk[k]).ToList()) / VecMul(grad(Xk[k - 1]).ToList(), grad(Xk[k - 1]).ToList())).ToList()));

AlphaK.Add(-((VecMul(grad(Xk[k]).ToList(), Pk[k].ToList())) / VecMul(MatrixVectorMul(H(Xk[k]).Select(x => x.ToList()).ToList(), Pk[k].ToList()), Pk[k].ToList())));

Xk.Add(VecSum(Xk[k].ToList(), Pk[k].Select(x => x \* AlphaK[k]).ToList()));

norm = Norm(grad(Xk[k + 1]));

logger.LogVector(f, Xk[k], "x" + k.ToString());

k++;

if (Math.Abs(norm) < Eps)

{

logger.Log("Алгоритм завершен");

logger.Log("k = " + k.ToString());

return Xk[k];

}

}

}

}

}

public static class Ex

{

public static void LogVector(this Logger logger, Func<IEnumerable<double>, double> f, IEnumerable<double> vec, string varName)

{

logger?.Log($"{varName.ToUpper()} = ");

for (int i = 0; i < vec.Count(); i++)

{

logger?.Log($"{varName.ToLower()}" + (i + 1).ToString() + ": " + vec.ElementAt(i));

}

logger?.Log($"f({varName.ToUpper()}) = " + f(vec));

}

}

}

public enum ExtremumType

{

Min,

Max

}

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;

}

}

Результати роботи програми показані на рисунку 2.

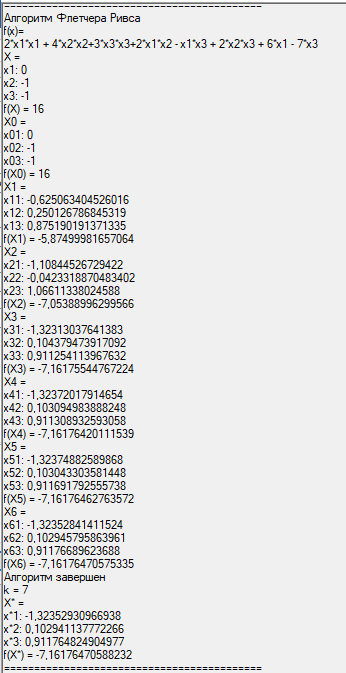


Рисунок 2 – Результат роботи програми

Рішення:

**Висновки:**

У даній лабораторній роботі було програмно реалізовано алгоритм Флетчера Рівса та вирішено задачу безумовної мінімізації.