МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «Програмна інженерія та інформаційні технології управління»

Звіт з лабораторної роботи №4

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 Девідона-Флетчера-Пауела.

**Задача:**

**Алгоритм методу:**

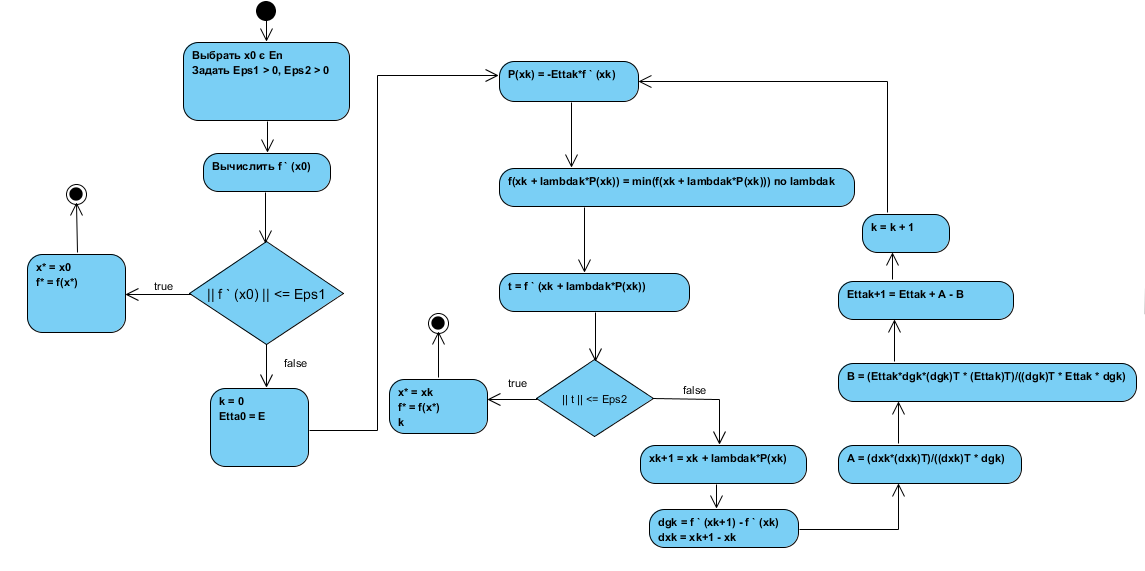
1. Обрати начальну точку , параметр завершення алгоритму та параметр завершення одномірної мінімізації
2. Розрахувати
3. Якщо то алгоритм завершено. Інакше перейти на крок 4.
4. Покласти
5. Покласти
6. Розрахувати таке , що

1. Якщо , то алгоритм завершено

, інакше перейти на крок 8.

1. Покласти
2. Покласти і розрахувати
3. Розрахувати ,
4. Покласти
5. Збільшити k = k+1 і перейти на крок 5.

Блок схема алгоритму показана на рисунку 1.

Рисунок 1 – Блок схема алгоритму Девідона-Флетчера-Пауела

Код програмної реалізації алгоритму показаний у лістингу 1.

Лістинг 1. Код алгоритму Девідона-Флетчера-Пауела

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace AproximationMethodsLib

{

public class DFP : IMultipleExtremumMethod

{

List<double> X0;

double eps1, eps2;

public DFP(List<double>x0,double eps1,double eps2)

{

X0 = x0;

this.eps1 = eps1;

this.eps2 = eps2;

}

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

{

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

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

{

res.Add(new List<double>());

for (int j = 0; j < a.Count; j++)

{

res[i].Add(a[i] \* b[j]);

}

}

return res;

}

private List<List<double>> TransposeMatrix(List<List<double>> A)

{

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

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

{

res.Add(new List<double>());

for (int j = 0; j < A.Count; j++)

{

res[i].Add(A[j][i]);

}

}

return res;

}

private List<List<double>> MatrixMulMatrix(List<List<double>>A, List<List<double>> B)

{

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

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

{

res.Add(new List<double>());

for (int j = 0; j < B[i].Count; j++)

{

res[i].Add(0);

for (int k = 0; k < B.Count; k++)

{

res[i][j] += A[i][k] \* B[k][j];

}

}

}

return res;

}

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

{

double res = 0;

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

{

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

}

return res;

}

private double Norm(IEnumerable<double> a)

{

double res = 0;

foreach (double e in a)

{

res += e \* e;

}

return Math.Sqrt(res);

}

private List<List<double>> MatrixPlusMatrix(List<List<double>> A, List<List<double>> B)

{

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

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

{

res.Add(new List<double>());

for (int j = 0; j < A.Count; j++)

{

res[i].Add(A[i][j] + B[i][j]);

}

}

return res;

}

private List<List<double>> MatrixMinusMatrix(List<List<double>> A, List<List<double>> B)

{

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

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

{

res.Add(new List<double>());

for (int j = 0; j < A.Count; j++)

{

res[i].Add(A[i][j] - B[i][j]);

}

}

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> VecDif(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();

}

private List<double> MatrixVectorTMul(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[row] += matrix[col][row] \* vector[col];

}

}

return res.ToList();

}

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 gradX0 = grad(X0);

if (Norm(gradX0) <= eps1)

{

return X0;

}

int k = 0;

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

Ettak.Add(new List<List<double>>());

Ettak[k].AddRange(Enumerable.Range(0, n).Select(x => new List<double>()).ToList());

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

{

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

{

Ettak[k][i].Add(i == j ? 1 : 0);

}

}

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

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

Xk.Add(X0);

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

GoldenSectionMethod gsm = new GoldenSectionMethod(new Range(0,1),0.00000001);

Pk.Add(gradX0.Select(x => -x).ToList());

while (true)

{

var fOne = new Func<double, double>(x =>

{

var arg = VecSum(Xk[k], Pk[k].Select(y => y \* x).ToList());

return f(arg);

});

var l = gsm.FindExtremum(fOne, ExtremumType.Min, null);

lambdak.Add(l);

var s = Pk[k].Select(y => y \* l).ToList();

Xk.Add(VecSum(Xk[k], s));

if (Norm(grad(Xk[k+1])) <= eps2)

{

logger?.Log("Алгоритм завершен");

logger?.Log("k = " + (k + 1));

logger?.LogVector(f, Xk[k + 1], "X" + (k + 1));

return Xk[k + 1];

}

logger?.LogVector(f, Xk[k + 1], "X" + (k + 1));

var gradxk1 = grad(Xk[k + 1]).ToList();

var gradxk = grad(Xk[k]).ToList();

var p = VecDif(gradxk1,gradxk);

var v = MatrixVectorMul(Ettak[k], p);

var sst = VecMulVecT(s, s);

var stp = VecMulVec(s, p);

var A = sst.Select(x => x.Select(y => y / stp).ToList()).ToList();

var vvt = VecMulVecT(v, v);

var vtp = VecMulVec(v, p);

var B = vvt.Select(x => x.Select(y => y / vtp).ToList()).ToList();

Ettak.Add(MatrixMinusMatrix(MatrixPlusMatrix(Ettak[k], A), B));

logger?.LogMatrix(Ettak[k + 1], "Etta" + (k + 1));

Pk.Add(MatrixVectorMul(Ettak[k+1], grad(Xk[k+1]).ToList()).Select(x => -x).ToList());

k++;

}

}

}

public static class LoggerExMatrix

{

public static void LogMatrix(this Logger logger, List<List<double>> matrix,string title)

{

logger.Log(title);

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

{

string buf = "";

for (int j = 0; j < matrix[i].Count; j++)

{

buf += matrix[i][j] + " ";

}

logger.Log(buf);

}

}

}

}

Результати роботи програми:

Алгоритм Девидона Флетчера Пауэлла

f(x)=

x1 \* x1 + 4 \* x1 \* x2 + 5 \* x2 \* x2 + 3 \* x3 \* x3 - 2 \* x2 \* x3 - 2 \* x1 \* x3

X =

x1: 2

x2: 2

x3: 3

f(X) = 43

X1 =

x11: 1,31718064707841

x12: -0,503671014620551

x13: 1,86196769122264

f(X1) = 7,72099820527326

Etta1

0,893517865518514 -0,292423306081466 0,0053586041586594

-0,292423306081466 0,202212708583017 0,0245488664214927

0,0053586041586594 0,0245488664214927 1,01807261889994

X2 =

x21: 1,58579685070784

x22: -0,572498181657286

x23: 0,344003448162091

f(X2) = 0,17991788238636

Etta2

0,86072805210277 -0,283621028375268 0,16822214736793

-0,283621028375268 0,199850399903818 -0,0192060135906489

0,16822214736793 -0,0192060135906489 0,211100773455182

X3 =

x31: 1,39307207807379

x32: -0,502922264192959

x33: 0,30219587735077

f(X3) = 0,138843815962968

Etta3

6,99983759598312 -2,49994560485823 1,49996588813825

-2,49994560485823 0,999981849576756 -0,499988505406076

1,49996588813825 -0,499988505406076 0,499993059552959

X4 =

x41: 3,38626010660814E-05

x42: -1,14087531490892E-05

x43: 7,11349806076234E-06

f(X4) = 8,45085236018659E-11

Etta4

7,00000885544141 -2,50000330565211 1,50000190995877

-2,50000330565211 1,00000119070114 -0,500000614680803

1,50000190995877 -0,500000614680803 0,500000628750109

Алгоритм завершен

k = 5

X5 =

x51: 2,8520116645341E-07

x52: -1,02662929301304E-07

x53: 5,28686306965185E-08

f(X5) = 0

Рішення:

**Висновки:**

У даній лабораторній роботі було програмно реалізовано алгоритм Девідона-Флетчера-Пауела та вирішено задачу безумовної мінімізації.