МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

“ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

КАФЕДРА ПРОГРАМНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГИЙ УПРАВЛІННЯ

Лабораторна робота № 4

з курсу «Дослідження операцій»

«Вивчення методів безумовної оптимізації другого порядку»

Виконав:

студент групи КН 36-а

Кулик Володимир

Перевірив:

Гужва В. О.

ХАРКІВ 2018

Мета роботи:

Вивчити методи безумовної оптимізації другого порядку. Скласти блок-схему алгоритму вибраного метода: метод Левенберга-Марквардта. Скласти програму, що реалізує алгоритм вибраного метода. Мінімізувати функцію, що була запропонована.

Завдання:

|  |  |  |
| --- | --- | --- |
| Варіант | Вид функції | Початковий вектор |
| 8 |  | (1, 1) |

Алгоритм методу:

**Крок 1**.Вибрати , максимальну кількість ітерацій М, покласти k=0 та .

**Крок 2**. Обчислити .

**Крок 3**. Якщо або k M, то останов. Інакше перейти до кроку 4.

**Крок 4**. Обчислити -1*f (E –* одинична матриця*)*.

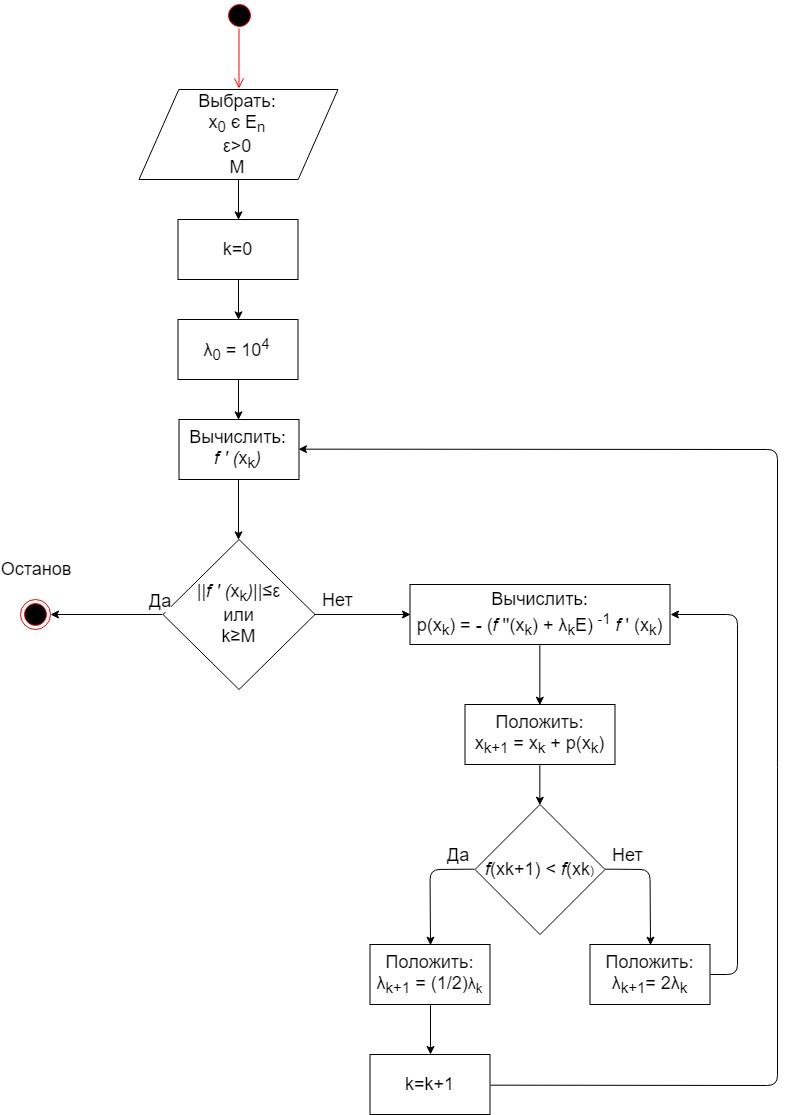
**Крок 5**. Покласти .

**Крок 6.** Якщо *f() < f()*, та йти до кроку 7. Інакше до кроку 8.

**Крок 7**. Покласти та йти до кроку 2.

**Крок 8**. Покласти , та йти до кроку 4.

Блок-схема алгоритму:



Лістинг програми:

//класс с заданной функцией

public class Function

{

//функция

public double fx(double[] variable)

{

double res = Math.Round(2 \* Math.Pow(variable[0], 2) + 2 \* Math.Pow(variable[1], 2) + 2 \* variable[0] \* variable[0] - 4 \* variable[0] - 6 \* variable[1]);

return res;

}

//градиент

public double[] fx\_gradient(double[] x)

{

double[] res = new double[3];

res[0] = Math.Round(4 \* x[0] - 4);

res[1] = Math.Round(4 \* x[1] - 6);

res[2] = 0;

return res;

}

//f''

public double[,] fx\_two\_strokes()

{

double[,] res = { { 4,1 },{ 1,4 } };

return res;

}

}

//класс с реализацией вычислений над матрицами и векторами

public class MathActionsWithVectors

{

//вычисление нормы

public double norma(double[] var)

{

double norm;

double norm\_sum = 0;

for (int i = 0; i < var.Length; i++)

{

norm\_sum += Math.Pow(var[i], 2);

}

norm = Math.Round(Math.Sqrt(norm\_sum), 4);

return norm;

}

//сложение векторов

public double[] addVecToVec(double[] var1, double[] var2)

{

double[] res = new double[var1.GetLength(0)];

for (int i = 0; i < var1.GetLength(0); i++)

{

res[i] += Math.Round(var1[i] + var2[i],4);

}

return res;

}

//умножение матрицы на число

public double[,] multUnitVector(double[,] var1, double m)

{

double[,] res = new double[2,2];

int rows = res.GetUpperBound(0) + 1;

int columns = res.Length / rows;

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

{

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

{

res[i,j]=var1[i, j]\*m;

}

}

return res;

}

//сложение матриц

public double[,] matrix\_on\_matrix(double[,] var1, double[,] var2)

{

double[,] res = new double[2, 2];

for (int i = 0; i < 2; i++)

{

for (int j = 0; j < 2; j++)

{

res[i, j] = Math.Round(var1[i, j] + var2[i,j],4);

}

}

return res;

}

//матрица в минус первой степени

public double[,] matrix\_in\_minus\_first(double[,] var1)

{

double[,] res = new double[2, 2];

double det = var1[0, 0] \* var1[1, 1] - var1[0, 1] \* var1[1, 0];

double[,] transpon = new double[2, 2];

transpon[0, 0] = var1[0, 0];

transpon[0, 1] = var1[1, 0];

transpon[1, 0] = var1[0, 1];

transpon[1, 1] = var1[1, 1];

res[0, 0] = transpon[1, 1] \* 1/det;

res[0, 1] = transpon[1, 0] \* 1 / det\*(-1);

res[1, 0] = transpon[0, 1] \* 1 / det\*(-1);

res[1, 1] = transpon[0, 0] \* 1 / det;

return res;

}

//умножение матрицы на вектор

public double[] matrix\_mult\_vector(double[,] matrix, double[] vector)

{

double[] res = new double[2];

for (int i = 0; i < 2; i++)

{

for (int j = 0; j < 2; j++)

{

res[i] += Math.Round(matrix[i,j]\*vector[j],4);

}

}

return res;

}

}

//класс с формой для отображения результатов и элементов

public partial class Form1 : Form

{

//инициализация формы

public Form1()

{

InitializeComponent();

}

//новые вычисления

private void button3\_Click(object sender, EventArgs e)

{

this.solutionBox.Clear();

this.epsBox.Clear();

this.mBox.Clear();

this.panel1.Enabled = true;

}

//выполение вычислений при нажатие на кнопку

private void button1\_Click(object sender, EventArgs e)

{

//вызов методов из класса с функцией

Function func = new Function();

//вызов методов из класса с действиями с векторами и матрицами

MathActionsWithVectors mawv = new MathActionsWithVectors();

this.panel1.Enabled = false;

//значения начального вектора приближения x0

double x1 = double.Parse(x1Box.Text.Replace(".", ","));

double x2 = double.Parse(x2Box.Text.Replace(".", ","));

//максимальное (допустимое) количество операций

double m = double.Parse(mBox.Text.Replace(".", ","));

//параметр окончания поиска оптимального решения

double eps = double.Parse(epsBox.Text.Replace(".", ","));

//массив со значениями лямбда

double[] lyambda = new double[1000];

lyambda[0] = Math.Pow(10, 4);

//вектор х(k)

double[] x = new double[2];

x[0] = x1;

x[1] = x2;

//вектор х(k+1)

double[] x\_k\_plus\_one = new double[2];

x\_k\_plus\_one[0] = 0;

x\_k\_plus\_one[1] = 0;

//массив со значениями p

double[] p = new double[2];

p[0] = 0;

p[1] = 0;

//единичная матрица E

double[,] E = { {1,0 },{ 0,1} };

int k = 0;

solutionBox.Text += "Выбрано:"+Environment.NewLine+"x("+k+") = ["+x1+", "+x2+"]"+Environment.NewLine+ "Критерий останова ε = "+eps+Environment.NewLine+"Максимальное число итераций: M = "+m+Environment.NewLine+ "λ("+k+") = " + lyambda[k]+Environment.NewLine+Environment.NewLine;

solutionBox.SelectionStart = solutionBox.TextLength;

solutionBox.ScrollToCaret();

while (mawv.norma(func.fx\_gradient(x)) > eps && k<m)

{

solutionBox.Text += "f'(x) = [ " + func.fx\_gradient(x)[0] + ", " + func.fx\_gradient(x)[1] + " ]" + Environment.NewLine+ "||f'(x)|| > > ε" + Environment.NewLine+ mawv.norma(func.fx\_gradient(x)) +">"+eps+Environment.NewLine;

solutionBox.SelectionStart = solutionBox.TextLength;

solutionBox.ScrollToCaret();

p = mawv.matrix\_mult\_vector(mawv.multUnitVector(mawv.matrix\_in\_minus\_first(mawv.matrix\_on\_matrix(func.fx\_two\_strokes(),mawv.multUnitVector(E, lyambda[k]))),-1), func.fx\_gradient(x));

x\_k\_plus\_one = mawv.addVecToVec(x, p);

solutionBox.Text += "p = [" + p[0] + ", " + p[1] + "]" + Environment.NewLine + "x(" + (k + 1) + ") = [" + x\_k\_plus\_one[0] + ", " + x\_k\_plus\_one[1] + "]" + Environment.NewLine + Environment.NewLine;

solutionBox.SelectionStart = solutionBox.TextLength;

solutionBox.ScrollToCaret();

if (func.fx(x\_k\_plus\_one) < func.fx(x))

{

solutionBox.Text += "f(x) = " + func.fx(x) + Environment.NewLine + "f(x+1) = " + func.fx(x\_k\_plus\_one) + Environment.NewLine + Environment.NewLine;

solutionBox.SelectionStart = solutionBox.TextLength;

solutionBox.ScrollToCaret();

lyambda[k+1] = Math.Round(0.5 \* lyambda[k],4);

k++;

x = x\_k\_plus\_one;

solutionBox.Text += "λ("+k+") = " + lyambda[k] + Environment.NewLine + "x(" + k + ") = [" + x[0] + ", " + x[1] + "]" + Environment.NewLine +"k = "+k+ Environment.NewLine+Environment.NewLine;

solutionBox.SelectionStart = solutionBox.TextLength;

solutionBox.ScrollToCaret();

}

else if (func.fx(x\_k\_plus\_one) >= func.fx(x))

{

solutionBox.Text += "f(x) = " + func.fx(x) + Environment.NewLine + "f(x+1) = " + func.fx(x\_k\_plus\_one) + Environment.NewLine + Environment.NewLine;

solutionBox.SelectionStart = solutionBox.TextLength;

solutionBox.ScrollToCaret();

lyambda[k+1] = Math.Round(2 \* lyambda[k],4);

k++;

x = x\_k\_plus\_one;

solutionBox.Text += "λ(" + k + ") = " + lyambda[k] + Environment.NewLine + "x(" + k + ") = [" + x[0] + ", " + x[1] + "]" + Environment.NewLine + "k = " + k + Environment.NewLine + Environment.NewLine;

solutionBox.SelectionStart = solutionBox.TextLength;

solutionBox.ScrollToCaret();

}

}

solutionBox.Text += " Решение найдено!! " + Environment.NewLine + Environment.NewLine;

solutionBox.SelectionStart = solutionBox.TextLength;

solutionBox.ScrollToCaret();

solutionBox.Text += "f'(x) = [ " + func.fx\_gradient(x)[0] + ", " + func.fx\_gradient(x)[1] + " ]" + Environment.NewLine + "||f'(x)|| <= ε" + Environment.NewLine + mawv.norma(func.fx\_gradient(x)) + "<" + eps + Environment.NewLine;

solutionBox.SelectionStart = solutionBox.TextLength;

solutionBox.ScrollToCaret();

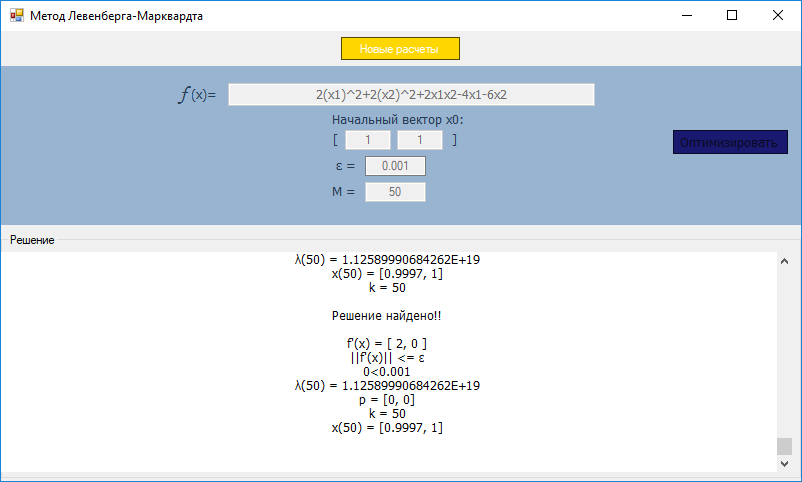
solutionBox.Text += "λ(" + k + ") = " + lyambda[k] +Environment.NewLine+ "p = [" + p[0] + ", " + p[1] + "]"+Environment.NewLine+ "k = " + k + Environment.NewLine + "x(" + k + ") = [" + x[0] + ", " + x[1] + "]" + Environment.NewLine + Environment.NewLine;

solutionBox.SelectionStart = solutionBox.TextLength;

solutionBox.ScrollToCaret(); }

}

Результат оптимізації:



Висновки:

Виконуючи лабораторну роботу отримані навички у сфері безумовної оптимізації другого порядку, а саме методом Левенберга-Марквардта. Отримані навички були закріплено за допомогою програмування алгоритмів мовою програмування C#. Закріплено навички у складанні блок-схем алгоритмів.

Список літератури:

1) Методичні вказівки до виконання лабораторних робіт з курсу «Дослідження операцій» для студентів, які навчаються за спеціальностями 121 Інженерія програмного забезпечення та 122 Комп’ютерні науки та інформаційні технології / укладачі: В. О. Гужва, Н. К. Стратієнко, І. О. Бородіна. – Харків : НТУ «ХПІ», 2018.