МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

“ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

КАФЕДРА ПРОГРАМНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГИЙ УПРАВЛІННЯ

Лабораторна робота № 3

з курсу «Дослідження операцій»

«Вивчення методів безумовної оптимізації першого порядку»

Виконала:

студентка групи КН 36-б

Ликова Маргарита

Перевірив:

Гужва В. О.

ХАРКІВ 2018

Мета роботи:

Вивчити метод безумовної оптимізації першого порядку, скласти блок-схему алгоритму вибраного методу, скласти програму зі складеної блок-схеми. Необхідно мінімізувати функцію, що була запропонована, градієнтним методом з дробленням кроку.

Завдання:

|  |  |  |  |
| --- | --- | --- | --- |
| Номер варіанта | Вид функції | Характер екстремуму | Початкова умова |
| 10 |  | *min* | (-1,1) |

Алгоритм методу:

**Крок 1**.Вибрати та покласти k=0.

**Крок 2**. Обчислити .

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

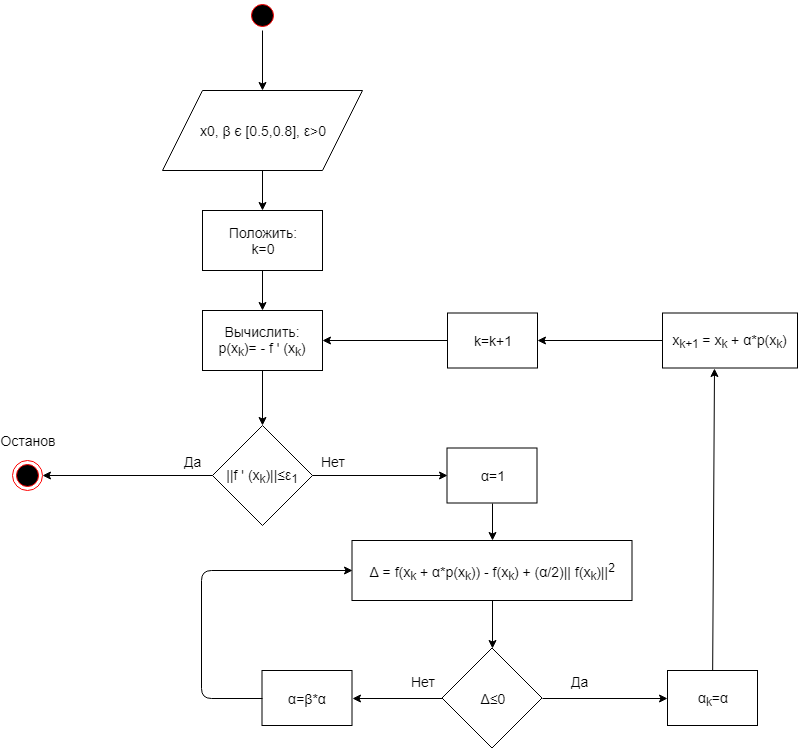
**Крок 4**. Покласти .

**Крок 5**. Обчислити 2.

**Крок 6.** Якщо , то покласти та йти до кроку 7. Інакше та йти до кроку 5.

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

Блок-схема алгоритму:



Лістинг програми:

this.panel1.Enabled = false;

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

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

double x3 = double.Parse(x3Box.Text.Replace(".", ","));

double x4 = double.Parse(x4Box.Text.Replace(".", ","));

double beta = double.Parse(betaBox.Text.Replace(".", ","));

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

double fx(double[] variable)

{

double res = Math.Pow(variable[0]+10\*variable[1], 2) + 5 \* Math.Pow(variable[2]-variable[3], 2) + Math.Pow(variable[1] - 2\*variable[2],4)+10\*Math.Pow(variable[0]-variable[3],4);

return res;

}

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

{

double[] res=new double[4];

res[0] = Math.Round(2 \*(x[0]+10\*x[1]+40\*Math.Pow(x[0]-x[3],3)),4);

res[1] = Math.Round(20\*(x[0]+10\*x[1])+4\*Math.Pow(x[1]-2\*x[2],3),4);

res[2] = Math.Round(10\*(x[2]-x[3])-8\*Math.Pow(x[1]-2\*x[2],3),4);

res[3] = Math.Round(-40\*Math.Pow(x[0]-x[3],3)-10\*(x[2]-x[3]),4);

return res;

}

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;

}

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

{

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

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

{

res[i] += var1[i] + var2[i];

}

return res;

}

double[] multVector(double[] var1, double m)

{

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

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

{

res[i] += var1[i] \* m;

}

return res;

}

double[] x\_vect = new double[4];

x\_vect[0] = x1;

x\_vect[1] = x2;

x\_vect[2] = x3;

x\_vect[3] = x4;

int k = 0;

double alpha;

double delta;

double[] p = new double[4];

solutionBox.Text = "Выбрано:"+ Environment.NewLine+"x0=[ ";

solutionBox.SelectionStart = solutionBox.TextLength;

solutionBox.ScrollToCaret();

for (int i = 0; i < x\_vect.Length; i++)

{

solutionBox.Text +=x\_vect[i]+" ";

solutionBox.SelectionStart = solutionBox.TextLength;

solutionBox.ScrollToCaret();

}

solutionBox.Text += "]"+Environment.NewLine+ "β = " + beta+Environment.NewLine+ "ε = " + eps;

solutionBox.SelectionStart = solutionBox.TextLength;

solutionBox.ScrollToCaret();

while (norma(fx\_gradient(x\_vect)) > eps)

{

solutionBox.Text += Environment.NewLine + "||f'(x(k))|| > ε" + Environment.NewLine + "k = " + k;

solutionBox.SelectionStart = solutionBox.TextLength;

solutionBox.ScrollToCaret();

solutionBox.Text += Environment.NewLine + Environment.NewLine + "f'(x(k) = [";

solutionBox.SelectionStart = solutionBox.TextLength;

solutionBox.ScrollToCaret();

for (int i = 0; i < fx\_gradient(x\_vect).Length; i++)

{

solutionBox.Text += Math.Round(fx\_gradient(x\_vect)[i], 4) + " ";

solutionBox.SelectionStart = solutionBox.TextLength;

solutionBox.ScrollToCaret();

}

solutionBox.Text += "]" + Environment.NewLine+ Environment.NewLine;

solutionBox.SelectionStart = solutionBox.TextLength;

solutionBox.ScrollToCaret();

solutionBox.Text += "||f'(x(k))|| = " + Math.Round(norma(fx\_gradient(x\_vect)),4) + Environment.NewLine + "ε = " + eps + Environment.NewLine + Environment.NewLine;

solutionBox.SelectionStart = solutionBox.TextLength;

solutionBox.ScrollToCaret();

alpha = 1;

p = multVector(fx\_gradient(x\_vect), -1);

solutionBox.Text += Environment.NewLine+Environment.NewLine + "p=[ ";

solutionBox.SelectionStart = solutionBox.TextLength;

solutionBox.ScrollToCaret();

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

{

solutionBox.Text += Math.Round(p[i],4) + " ";

solutionBox.SelectionStart = solutionBox.TextLength;

solutionBox.ScrollToCaret();

}

solutionBox.Text += "]"+Environment.NewLine;

solutionBox.SelectionStart = solutionBox.TextLength;

solutionBox.ScrollToCaret();

delta = fx(addVecToVec(x\_vect, multVector(p, alpha))) - fx(x\_vect) + (alpha / 2) \* Math.Pow(norma(fx\_gradient(x\_vect)), 2);

solutionBox.Text += Environment.NewLine + "α = " + Math.Round(alpha,4) + Environment.NewLine + "Δ = " + Math.Round(delta,4) +Environment.NewLine;

solutionBox.SelectionStart = solutionBox.TextLength;

solutionBox.ScrollToCaret();

while (delta > 0)

{

alpha \*= beta;

delta = fx(addVecToVec(x\_vect, multVector(p, alpha))) - fx(x\_vect) + (alpha / 2) \* Math.Pow(norma(fx\_gradient(x\_vect)), 2);

solutionBox.Text += Environment.NewLine+ "Δ>0 =>"+Environment.NewLine + "α = αβ = " + Math.Round(alpha,4) + Environment.NewLine + "Δ = " + Math.Round(delta,4) +Environment.NewLine;

solutionBox.SelectionStart = solutionBox.TextLength;

solutionBox.ScrollToCaret();

}

solutionBox.Text += Environment.NewLine + "Δ≤0 =>" + Environment.NewLine;

solutionBox.SelectionStart = solutionBox.TextLength;

solutionBox.ScrollToCaret();

x\_vect = addVecToVec(x\_vect, multVector(p, alpha));

k++;

solutionBox.Text += Environment.NewLine + "Значение нового вектора x"+k+"=[ ";

solutionBox.SelectionStart = solutionBox.TextLength;

solutionBox.ScrollToCaret();

for (int i = 0; i < x\_vect.Length; i++)

{

solutionBox.Text += Math.Round(x\_vect[i],4) + " ";

solutionBox.SelectionStart = solutionBox.TextLength;

solutionBox.ScrollToCaret();

}

solutionBox.Text += "]" + Environment.NewLine;

solutionBox.SelectionStart = solutionBox.TextLength;

solutionBox.ScrollToCaret();

}

solutionBox.Text += Environment.NewLine + "||f'(x(k))|| ≤ ε => Решение найдено!" + Environment.NewLine;

solutionBox.SelectionStart = solutionBox.TextLength;

solutionBox.ScrollToCaret();

solutionBox.Text += Environment.NewLine + Environment.NewLine + "f'(x(k) = [";

solutionBox.SelectionStart = solutionBox.TextLength;

solutionBox.ScrollToCaret();

for (int i = 0; i < fx\_gradient(x\_vect).Length; i++)

{

solutionBox.Text += Math.Round(fx\_gradient(x\_vect)[i], 4) + " ";

solutionBox.SelectionStart = solutionBox.TextLength;

solutionBox.ScrollToCaret();

}

solutionBox.Text += "]" + Environment.NewLine + Environment.NewLine;

solutionBox.SelectionStart = solutionBox.TextLength;

solutionBox.ScrollToCaret();

solutionBox.Text += "||f'(x(k))|| = " + Math.Round(norma(fx\_gradient(x\_vect)), 4) + Environment.NewLine + "ε = " + eps + Environment.NewLine + Environment.NewLine;

solutionBox.SelectionStart = solutionBox.TextLength;

solutionBox.ScrollToCaret();

solutionBox.Text += Environment.NewLine + "Значение вектора x\*=[ ";

solutionBox.SelectionStart = solutionBox.TextLength;

solutionBox.ScrollToCaret();

for (int i = 0; i < x\_vect.Length; i++)

{

solutionBox.Text += Math.Round(x\_vect[i],4) + " ";

solutionBox.SelectionStart = solutionBox.TextLength;

solutionBox.ScrollToCaret();

}

solutionBox.Text += "]" + Environment.NewLine;

solutionBox.SelectionStart = solutionBox.TextLength;

solutionBox.ScrollToCaret();

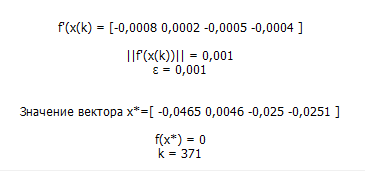
solutionBox.Text += Environment.NewLine+"f(x\*) = "+ Math.Round(fx(x\_vect),4)+Environment.NewLine+"k = "+k;

solutionBox.SelectionStart = solutionBox.TextLength;

solutionBox.ScrollToCaret();

}

Результат оптимізації:



Висновки:

Виконуючи лабораторну роботу отримані навички у сфері безумовної оптимізації першого порядку,а саме градієнтним методом з дробленням кроку. Отримані навички були закріплено за допомогою програмування алгоритмів мовою програмування c#. Закріплено навички у складанні блок-схем алгоритмів.