МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«Харківський Політехнічний Інститут»

Кафедра стратегічного управління

ЗВІТ

з лабораторної роботи №3

з дисципліни «Дослідження операцій»

Перевірив

Лисицький В. Л.

Виконав:

ст. гр. КН-27

Харків – 2019

**Ціль роботи:** Знайти рішення задачі безумовної мінімізації методом Флетчера Рівса (метод сполучених градієнтів).

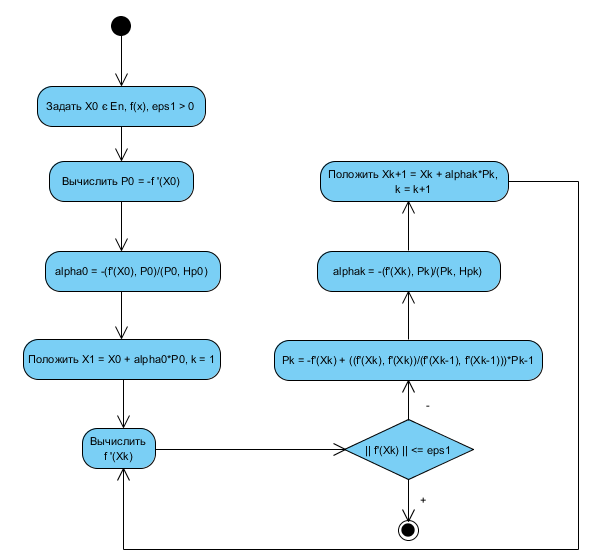
Задача (варіант №2):

n = 2

**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 таблиці 1.

Таблиця 1. Код програми, що реалізує алгоритм Флетчера Рівса

|  |
| --- |
| "use strict"  let log = ""  document.addEventListener("DOMContentLoaded", () => {  let eps = document.getElementById('epsilon').value  let x\_start = document.getElementById('start').value.split(',').map(Number)  document.getElementById('run').addEventListener('click', () => FletcherRivz(eps, x\_start))  });  // const myFunction = x\_vector => x\_vector[0]\*\*3 + x\_vector[1]\*\*3  const myFunction = x\_vector => x\_vector[0] + x\_vector[1] \* x\_vector[1] + ((x\_vector[0]+x\_vector[1]-10)/3)\*\*2  //const myFunction = x\_vector => (x\_vector[0] - x\_vector[1])\*\*2 + ((x\_vector[0]+x\_vector[1]-10)/3)\*\*2  const eps = 0.0001  function findDerivative(func,x\_vector,num){  let arr\_second = [...x\_vector]  arr\_second[num] += eps  return (func(arr\_second)-func(x\_vector))/eps  }  function gradient (x\_vector){  let grad = []  for (let i = 0; i < x\_vector.length; i++){  grad.push(findDerivative(myFunction, x\_vector, i))  }  return grad  }  const getScalar = (vector1, vector2) => vector1.reduce((total, value, index, array) => total + value\*vector2[index],0)  function matrixVectorMul(matrix, p\_k){  let res = new Array(p\_k.length).fill(0)  for (let row = 0; row < matrix.length; row++){  for (let col = 0; col < matrix[row].length; col++){  res[col] += matrix[row][col] \* p\_k[row]  }  }  console.log(res)  return res  }  function getStep(x\_k,p\_k){  return -getScalar(gradient(x\_k),p\_k)/getScalar(p\_k, matrixVectorMul(secondDerivativeMatrix(myFunction,x\_k), p\_k))  }  function getNorm(x\_vector){  return Math.pow(x\_vector.reduce((t,v,i,a) => t+v\*v,0),1/2)  }  function getSum(a,b){  return a.map((v,i,a) => v += b[i])  }  function secondDerivativeMatrix(func,x\_k){  let matrix = []  let grad = gradient(x\_k)  for (let i = 0; i < x\_k.length; i++){  let x\_k\_new = [...x\_k]  x\_k\_new[i] += eps  let gradI = gradient(x\_k\_new)  let row = []  for (let j = 0; j < x\_k.length; j++){  row.push((gradI[j] - grad[j])/eps)  }  matrix.push(row)  }  console.log(matrix)  return matrix  }  function FletcherRivz(e1, x\_start){  // Step 1  let max\_iterations = 1000  //Step 2  let p\_0 = gradient([...x\_start]).map(el=>-el)  log += 'x\_0: ' + x\_start + "<br>"  log += 'p\_0: ' + p\_0 + "<br>"  log += 'z\_0: ' + myFunction(x\_start).toFixed(2) + "<br>"  secondDerivativeMatrix(myFunction,x\_start)  log += 'a\_0: ' + getStep(x\_start, p\_0) + "<br>"  let k = 0  let a\_k = 0  let p\_k = [...p\_0]  let p\_k\_1 = [...p\_0]  let x\_k = [...x\_start]  let x\_k\_1 = [...x\_start]  while(k<max\_iterations){    k++  log += '===' + '<br>'  //Step 3  a\_k = getStep(x\_k, p\_k)    log+='a\_k: '+a\_k + '<br>'  //Step 4  x\_k = getSum(x\_k,p\_k.map(el=>el\*a\_k))  log+='x\_k: '+x\_k.map(el=>el.toFixed(2)) + '<br>'  log+='z(x\_k): '+myFunction(x\_k).toFixed(2) + '<br>'  log+='Norm: '+getNorm(gradient(x\_k)) + '<br>'  //Step 5-6  if (getNorm(gradient(x\_k))<e1){  break  }  //Step 7  let temp = getScalar(gradient(x\_k),gradient(x\_k))/getScalar(gradient(x\_k\_1),gradient(x\_k\_1))  p\_k = getSum(gradient(x\_k).map(el => -el),p\_k\_1.map(el=>el\*temp))    log+='p\_k: '+p\_k + '<br>'  // console.log('p\_k: '+p\_k)  x\_k\_1 = x\_k  p\_k\_1 = p\_k  }  // console.log('Finished')  log+='Iterations: '+k  console.log('Iterations: '+k)  document.getElementById("answer").innerHTML = log  } |

Результати роботи програми показано у таблиці 2.

Таблиця 2. Результат виконання.

|  |
| --- |
| x\_0: -1,-1  p\_0: 1.6666555555389095,4.666555555541407  z\_0: 16.00  a\_0: 0.4680011862111002  ===  a\_k: 0.4680011862111002  x\_k: -0.22,1.18  z(x\_k): 10.25  Norm: 1.070359286021796  p\_k: 1.0857631529589105,-0.1422738503760103  ===  a\_k: 4.807679724156123  x\_k: 5.00,0.50  z(x\_k): 7.50  Norm: 0.000008400543101831373  Iterations: 2 |

Рішення:

Керівництво користувача

Користувач вводе точність та початковий вектор х у відповідні поля – Рисунок 1

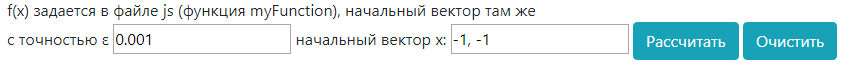


Рисунок 1- Форма вводу даних

Функція задається у файлі lab3.js – рисунок 2



Рисунок 2 – Функція для оптимізації

**Висновки:**

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