МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «Стратегічного управління»

Звіт з лабораторної роботи №1

з дисципліни «Дослідження операцій»

Виконав:

Студент групи КН-27

Перевірив:

Гужва В. О.

Харків – 2019

**Ціль роботи:** Дослідити алгоритми кубічної апроксимації.

**Алгоритми досліджуваних методів:**

**Метод кубічної апроксимації:**

1. Покласти відрізок [a, b], точність eps1, eps2, крок , та досліджувану функцію.
2. Задати
3. Якщо , то , інакше
4. k = k + 1
5. Якщо , то та перейти на крок 3
6. Розрахувати
7. Розрахувати , де
8. Якщо то і перейти на крок 8, інакше
9. Якщо та , то алгоритм завершено,

, інакше перейти на крок 10.

1. Якщо , то та перейти на крок 6, інакше та перейти на крок 6.

**Блок схема алгоритму:**

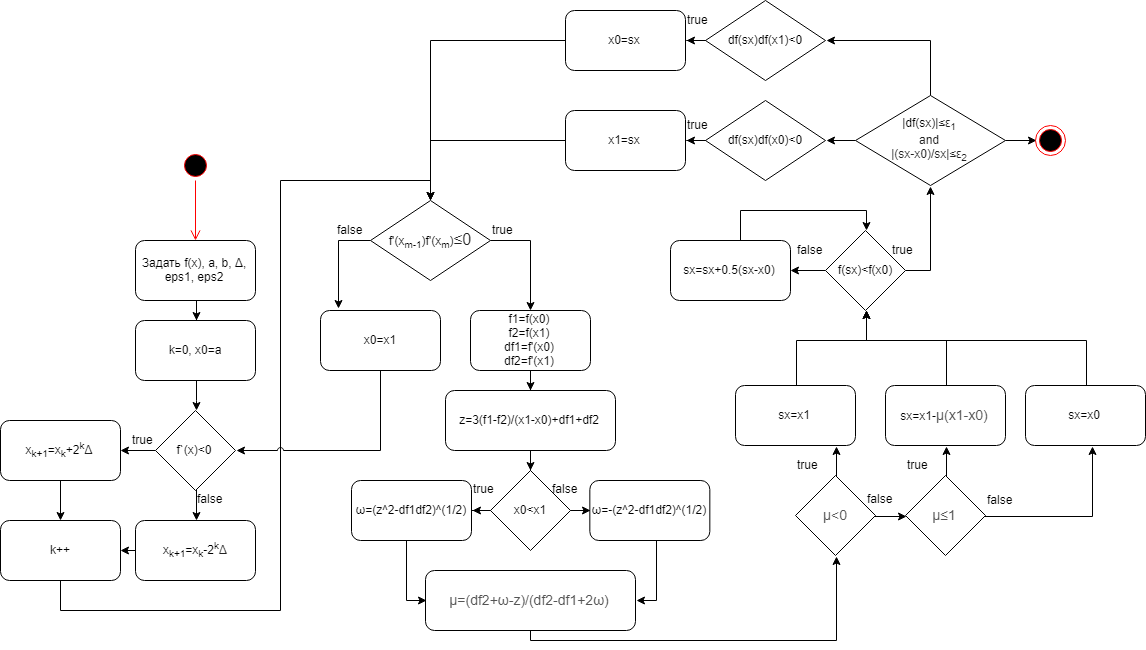


Рисунок 1 – Блок схема алгоритму кубічної апроксимації

**Програми досліджуваних алгоритмів:**

Лістинг програми алгоритму кубічної апроксимації:

|  |
| --- |
| const delta = 0.00000001  var eq;  document.addEventListener("DOMContentLoaded", () => {      document.getElementById("run").addEventListener("click", () =>{  document.getElementById("answer").innerHTML = "";  eq = document.getElementById("equation").value;  var start = parseFloat(document.getElementById("start").value);  var end = parseFloat(document.getElementById("end").value);  var epsilon = parseFloat(document.getElementById("epsilon").value);  var step = parseFloat(document.getElementById("step").value);    var x;  var k = 0;  var x0 = start;  var x1;    var eps1 = 0.00000001;  var eps2 = 0.5;    var f = Function('x', 'return eval(eq)');    do{  if (firstDer(f, x0) < 0 ){  x1 = x0 + Math.pow(2,k) \* step;  }  else{  x1 = x0 - Math.pow(2,k) \* step;  }    k += 1;    if (firstDer(f, x1)\*firstDer(f, x0) <= 0){  break;  }  x0=x1;  }  while(true)    document.getElementById("answer").innerHTML += "Понадобилось " + k + " итераций<br>";    var i = 0;  var sx = 0;  while(true)  {  var f1 = f(x0);  var f2 = f(x1);  var df1 = firstDer(f, x0);  var df2 = firstDer(f, x1);    sx = findSX(x0, x1, f1, f2, df1, df2);  console.log(x0 + " " + x1 + " " + f1 + " " + f2 + " " + df1 + " " + df2 + " " + sx);  while (true)  {  if (f(sx) <= f(x0))  break;  sx = sx + 0.5 \* (sx - x1);  i++;  }  console.log(Math.abs(firstDer(f, sx)) + " " + Math.abs((sx - x1) / sx));  //if (Math.abs(firstDer(f, sx)) <= eps1 && Math.abs((sx - x1) / sx) <= eps2 )    if (Math.abs(firstDer(f, sx)) <= eps1 )  break;  if (firstDer(f, sx) \* firstDer(f, x0) < 0)  {  x1 = sx;  }  if (firstDer(f, sx) \* firstDer(f, x1) < 0)  {  x0 = sx;  }  // else  // {  // x0 = sx;  // }  i++;  }  document.getElementById("answer").innerHTML += "Поиску понадобилось "+i+" итераций <br>";  document.getElementById("answer").innerHTML += "Алгоритм завершен!";  document.getElementById("answer").innerHTML += "Всего итераций: " + (i + k) + "<br>";  document.getElementById("answer").innerHTML += "Функция принимает экстремальное значение в точке: \nx\* = " + sx + "\nf(x\*) = " + f(sx) + "<br>";  var arrX = [];  var arrY = [];  for (var i = start; i <= end; i += step){  var x = i;  var y = f(x);  arrX.push(x);  arrY.push(y);  //arr.push({x: x, y: y});  }  var ctx = document.getElementById('myChart').getContext('2d');  var chart = new Chart(ctx, {  type: 'line',  data: {  labels: arrX,  datasets: [{  label: 'My First dataset',  data: arrY  }]  },  options: {}  });      });  document.getElementById("clean").addEventListener("click", () =>{  document.getElementById("answer").innerHTML = "";  });    });  // Four copypasted functions  function findSX(x1, x2, f1, f2, df1, df2){  var mu = findMu(x1, x2, f1, f2, df1, df2);  if (mu < 0) return x2;  else if (mu <= 1) return x2 - mu \* (x2 - x1);  else return x1;  }  function findZ(x1, x2, f1, f2, df1, df2){  return ((3 \* (f1 - f2) / (x2 - x1)) + df1 + df2);  }  function findMu(x1, x2, f1, f2, df1, df2){  var z = findZ(x1, x2, f1, f2, df1, df2);  var w = findW(x1, x2, f1, f2, df1, df2);  return (df2 + w - z) / (df2-df1+2\*w);  }  function findW(x1, x2, f1, f2, df1, df2){  var z = findZ(x1, x2, f1, f2, df1, df2);  var res = Math.pow(((z \* z) - df1 \* df2), 0.5);  return x1 < x2 ? res : -res;  }  // Finding derivative in a silly way  function firstDer (func, argument){  return Math.round((func(argument+delta)-func(argument))/delta\*100)/100;  } |

**Результати роботи програм:**

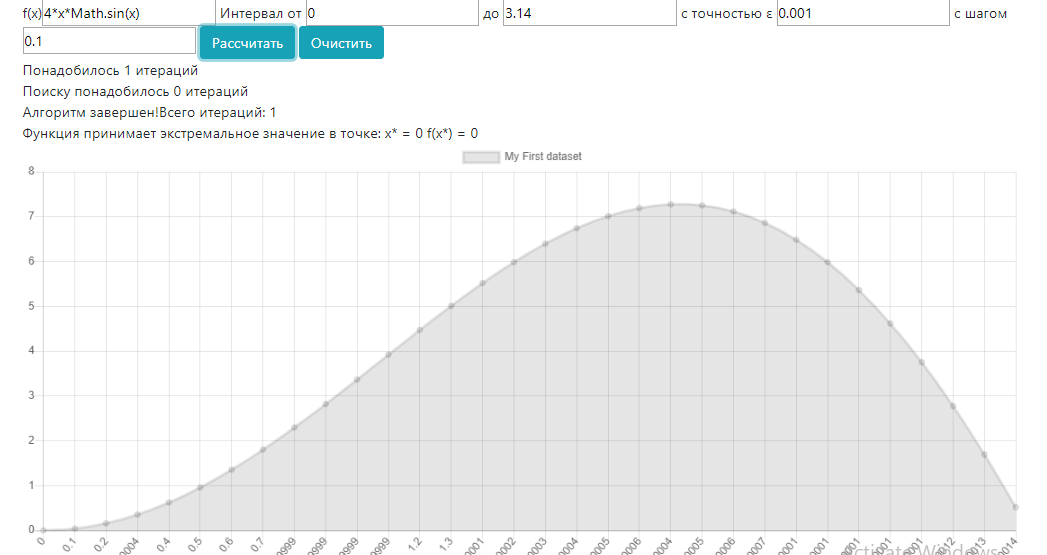


Рисунок 2. Приклад виконання програми

**Висновок:**

У даній лабораторній роботі було досліджено алгоритм кубічної апроксимації. Алгоритм кубічної апроксимації дає точний результат за менший час ніж інші схожі алгоритми одномірної оптимізації.