КН-Н921б

Сторожук О.Д.

Лабораторна робота №1

Тема роботи: Методи безумовної оптимізації в мультисервісних мережах. Унімодальні функції

Мета роботи: навчитися реалізовувати алгоритми знаходження мінімального часу, потрібного на проходження пакета в мультисервісній мережі, якщо залежність часу проходження від комплексного параметру апроксимується унімодальною функцією.

ЗАВДАННЯ ДО ВИКОНАННЯ

Розробити програму реалізації алгоритму одного із методів знаходження мінімуму унімодальної функції на заданому відрізку за методом дихотомії. Для перевірки взяти функцію:

, [0;2]

Також застосувати N=8, ε=0.2.

ХІД РОБОТИ

Графік функції продемонстрований на рис. 1. Згідно нього можна побачити мінімум функції на відрізку [0;2] – (1.775, -0.101).

Chart, line chart

Description automatically generated

Рисунок 1 – Графік функції, що обчислюється

Для введення функції застосовується алгоритм зворотньої польскої анотації. Згідно з ним, функція на вводі матиме наступну форму: x 3 x - / 2 x \* - 2 +.

Фрагмент розпізнавача вхідної послідовності наведений нижче:

vector<string>\* parse(string expression){

vector<string>\* result = new vector<string>();

for(int index = 0; index < expression.length(); index++){

if(isNumber(expression[index])){

string\* number = new string;

while (index < expression.length() && isNumber(expression[index])) {

number->append(1, expression[index++]);

}

result->push\_back(\*number);

delete number;

if(index == expression.length())

break;

}

string\* pushSymbol = new string;

if(expression[index] == '\*' || expression[index] == '/'){

while(operatorsStack->size() != 0 && (operatorsStack->back() == "\*" || operatorsStack->back() == "/")){

result->push\_back(operatorsStack->back());

operatorsStack->pop\_back();

}

pushSymbol->append(1, expression[index]);

operatorsStack->push\_back(\*pushSymbol);

}

else if(expression[index] == '+' || expression[index] == '-'){

while(operatorsStack->size() != 0 && (operatorsStack->back() == "+" || operatorsStack->back() == "-")){

result->push\_back(operatorsStack->back());

operatorsStack->pop\_back();

}

pushSymbol->append(1, expression[index]);

operatorsStack->push\_back(\*pushSymbol);

}

else if(expression[index] == '(' || expression[index] == ')'){

if(expression[index] == '('){

pushSymbol->append(1, expression[index]);

operatorsStack->push\_back(\*pushSymbol);

} else {

while (!operatorsStack->size() != 0 && operatorsStack->back() != "(") {

result->push\_back(operatorsStack->back());

operatorsStack->pop\_back();

}

if (!operatorsStack->size() != 0 && operatorsStack->back() == "(") {

operatorsStack->pop\_back();

}

}

}

delete pushSymbol;

}

while(operatorsStack->size() != 0){

result->push\_back(operatorsStack->back());

operatorsStack->pop\_back();

}

return result;

}

bool isNumber(char symbol){

return symbol - '0' >= 0 && symbol - '0' <= 9;

}

Фрагмент коду програми, що виконує розрахунок за методом дихотомії наведений нижче:

vector<Element>\* calculate(string\* expression, float leftEdge, float rightEdge){

vector<Element>\* result = new vector<Element>();

float x1;

float x2;

float f1;

float f2;

float a = leftEdge;

float b = rightEdge;

float dividedEpsilon = epsilon / 2;

for(int i = 1; i <= N/2; i++){

x1 = ((a + b) / 2) - dividedEpsilon;

x2 = ((a + b) / 2) + dividedEpsilon;

string\* replacedExpression = new string(replaceVariablesWithValue(\*expression, x1));

calculator->calculateExpression(replacedExpression);

f1 = stof(\*replacedExpression);

replacedExpression = new string(replaceVariablesWithValue(\*expression, x2));

calculator->calculateExpression(replacedExpression);

f2 = stof(\*replacedExpression);

if(f1 > f2){

a = x1;

} else {

b = x2;

}

Element element = Element(i, x1, x2, f1, f2, a, b);

result->push\_back(element);

delete replacedExpression;

}

return result;

}

Нижче наведений приклад власного розрахунку для подальшої перевірки програми на правильність отриманих результатів:

Крок 1: a = 0; b = 2;

Крок 2: a = 0.9; b = 2;

Крок 3: a = 1.35; b = 2;

Крок 4: a = 1.575; b = 2;

На рис. 2 наведений результат виконання програми. Згідно до нього можна побачити, що вже на третій ітерації був знайдений абсолютний мінімум функції, який станови f2 = -0.11. Через архітектурну особливість роботи арифметичних операцій для чисел з плаваючою комою, результат дещо відрізняється від власноруч розрахованного значення. Четверта ітерація вже виконала корекцію для змінної b, однак через обмеження загальної кількості ітерацій, розрахунок завершується.

A picture containing text

Description automatically generated

Рисунок 2 – Результат знаходження мінімума функції за методом дихотомії

ВИСНОВКИ

Під час виконання лабораторної роботи були отримані практичні навички із реалізації алгоритму знаходження мінімального часу, потрібного на проходження пакету в мультисервісній мережі, якщо залежність часу проходження від комплексного параметру апроксимується унімодальною функцією.