МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



**Дніпропетровський національний університет  
залізничного транспорту імені академіка В. Лазаряна**

Кафедра КИТ

**Звіт  
по навчальній практиці**

Виконав: студент гр. ПЗ1911

Сіньков Г.О.

Дніпро, 2019

**Постановка задачі**

Ознайомитись з методами інтерполяції функцій та розв’язання системи лінійних алгебраїчних рівнянь. Провести програмну реалізацію методів у відповідності до обраного рівня складності та оформити звіт.

Програма повинна забезпечувати введення наступних даних: кількість точок функції/рівнянь, таблиці значень невідомої функції, коефіцієнти рівнянь, степінь багаточлена і точності обчислень(в разі необхідності), вибір метода розрахунку. Відповідь: значення функції в координатах, що задаються користувачем, розв’язок системи рівнянь. Значення виводяться з точністю до шести знаків. В доповнення, для рівнів складності А-С графік функції з позначенням початкових точок та аналітичний вигляд для багаточленів.

|  |  |
| --- | --- |
| Вимоги до оформлення програми | Вимоги до змісту завдання |
| Консольний додаток з контролем вхідної інформації і реалізацією функцій в окремому файлі. Значення функцій та опис рівнянь повинні зчитуватись/записуватись з текстового/бінарного файлу. | Кількість точок для інтерполяції не менше 10, кількість рівнянь не менше 5Допускається поєднання методів середньої складності і простих. |

**Зовнішні специфікації**

Формат вхідних даних

Меню:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Найменування даних | Умовне позначення | Вимоги до даних | Приклад |
| 1 | Номер меню | number\_fun | Ціле число | 2 |

Метод Лагранжа:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Найменування даних | Умовне позначення | Вимоги до даних | Приклад |
| 1 | Ліва границя | a | Дійсне число з області визначення функції | 1 або 2,56 |
| 2 | Права границя | b | Дійсне число з області визначення функції | 4 або 2,56 |
| 3 | Точка з цієї границі | \_x | Дійсне число з області визначення функції | 2 або 2,5 |

Метод Гауса:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Найменування даних | Умовне позначення | Вимоги до даних | Приклад |
| 1 | Кількість рівнянь | equations | Дійсне число з області визначення функції | 1 або 2,56 |
| 2 | Кількість змінних | changing | Дійсне число з області визначення функції | 4 або 2,56 |
| 3 | Елементи матриці | matrix[i][j] | Дійсне число з області визначення функції | 3, 4, 10 або 2.3, 6.4, 0.2 |

Формат вихідних даних

|  |  |  |
| --- | --- | --- |
| № | Вхідні дані | Тестове повідомлення |
| 1 | Вивід результату за методом Лагранжа | Поліндром Лагранжа L(x) равен: 0.14112  x L(x)  2 0.909297  4.5 -0.97753 |
| 2 | Вивід результату за методом Гауса | Масив елемантів із зміною:  x0 = 2  x1 = 1 |

**Функціональні вимоги до програми**

-Вычисление значение определенного интеграла с заданной точностью

-Обеспечивать ввод пользователем исходных данных для вычислений, с контролем входной информации

-Выбор метода численного интегрирования−Построение графика подынтегральной функции

-Сохранение результатов вычислений в соответствии с форматом

-Предоставлять пользователю справочную информацию об использованных методах и порядке работы с программой.

**Вибір методу рішення завдання**

1)Метод Лангранжа

Лагранж запропонував спосіб обчислення таких многочленів:



де базисні поліноми визначаються за формулою:



Очевидно, що мають такі властивості:

* Це поліноми степеня n
* *li*(*xj*) = 1
* *li*(*xj*) = 0 при *i j*

Звідси випливає, що *L*(*x*), як лінійна комбінація *li*(*x*), може мати степінь не більший від *n* та *Li*(*xj*) = *yj*

Застосування

Поліноми Лагранжа використовуються для інтерполяції, а також для численого інтегрування. Нехай для функції f(x) відомі значення *yj = f* (*xj*) у деяких точках. Тоді ця функція може інтерполюватися як



Зокрема,



Значення інтегралів від *f* (*x*) не залежать від *yi = f* (*xj*), тож їм можна обчислювати заздалегідь, знаючи послідовність *x0*.

**Для випадку рівномірного розподілу на відрізки вузлів інтерполяції**

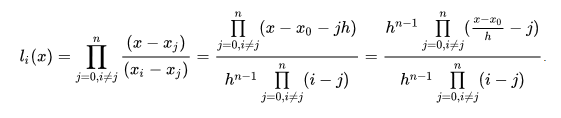
У вказаному випадку можна виразити *xi* через відстань між вузлами інтерполяції *h* та початкову точку *x0*:



і, як наслідок,



Якщо підставити ці вирази у формулу базисного полінома та винести h за знак множення у чисельнику та знаменнику, отримаємо



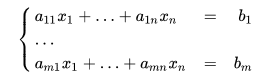
Після цього можна ввести зміну змінної



і отримати поліном від y, який будується з використанням лише цілочисленної арифметики. Недоліком цього підходу є факторіальна складність чисельника та знаменника, що вимагає використання алгоритмів з багатобайтним представлення чисел.

2)Метод Гаусса

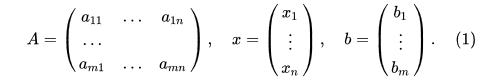
Нехай вихідна система виглядає наступним чином:



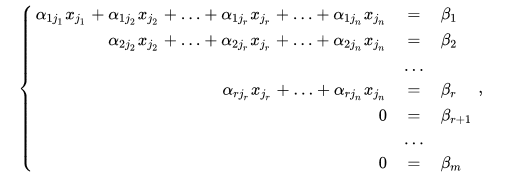
Її можна записати в матричному вигляді:



де



Матриця A називається головного датчика системи, b - стовпцем вільних членів.



Тоді, відповідно до властивості елементарних перетворень над рядками, основну матрицю цієї системи можна привести до східчастого увазі (ці ж перетворення потрібно застосовувати до колонку вільних членів):

де

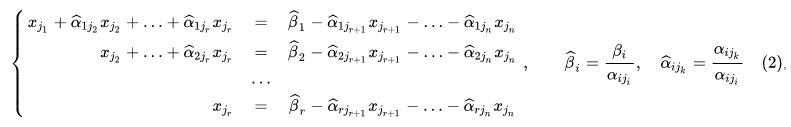
При цьому будемо вважати, що базисний мінор (ненульовий мінор максимального порядку) основної матриці знаходиться у верхньому лівому куті, тобто в нього входять тільки коефіцієнти при змінних *xj1, … , xjr*.

Тоді змінні *xj1, … , xjr* називаються головними змінними. Всі інші називаються вільними.

Якщо хоча б одне число *β i ≠ 0,* де *i> r*, то розглянута система несумісна, тобто у неї немає жодного рішення.

Нехай β i = 0 для будь-яких I > r.

Перенесемо вільні змінні за знаки рівності і поділимо кожне з рівнянь системи на свій коефіцієнт при самому лівому *x (aij, i = 1, … , r,* де *i* - номер рядка):

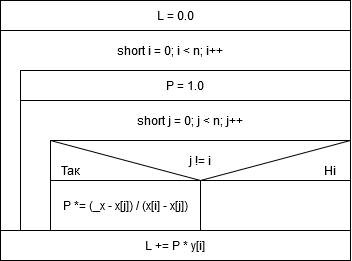


де *i = 1, … , r, k = i + 1, … , n*.

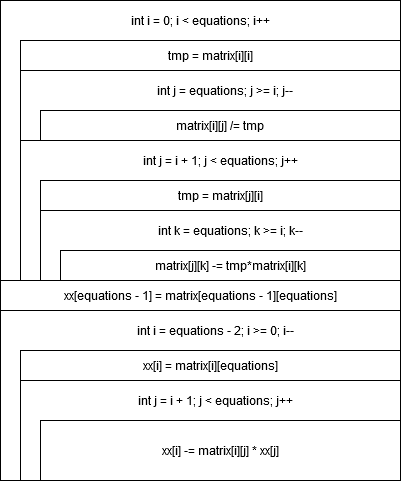
Якщо вільним змінним системи (2) надавати всі можливі значення і вирішувати нову систему щодо головних невідомих знизу вгору (тобто від нижнього рівняння до верхнього), то ми отримаємо всі рішення цієї СЛАР. Так як ця система отримана шляхом елементарних перетворень над вихідною системою (1), то по теоремі про еквівалентність при елементарних перетвореннях системи (1) і (2) еквівалентні, тобто безлічі їх рішень збігаються.

**Алгоритм програми**

1)Метод Лангранжа



2)Метод Гаусса



**Текст програми**

func.h

#ifndef func\_H

#define func\_H

typedef double(\*pointFunction)(double);

double lagrange1(double\* x, double\* y, short n, double \_x);

void gauss(int changing, int equations);

double function1(double x);

double function2(double x);

double function3(double x);

double function4(double x);

double function5(double x);

double input\_a();

double input\_b();

double input\_x(double a, double b);

int input\_quantity\_equations();

int input\_quantity\_changing();

#endif

func.cpp

#include <iostream>

#include "func.h"

#include <math.h>

double lagrange1(double\* x, double\* y, short n, double \_x)

{

double L = 0.0;

for (short i = 0; i < n; i++)

{

double P = 1.0;

for (short j = 0; j < n; j++)

{

if (j != i)

{

P \*= (\_x - x[j]) / (x[i] - x[j]);

}

}

L += P \* y[i];

}

return L;

}

void gauss(int changing, int equations)

{

changing += 1;

float \*\*matrix = new float \*[equations];//створюємо масив

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

{

matrix[i] = new float[changing];

}

//ввод масива

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

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

{

std::cout << " Елементи " << "[" << i + 1 << " , " << j + 1 << "]: ";

std::cin >> matrix[i][j];

}

//виводимо масив

std::cout << "Масив елементів: " << std::endl;

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

{

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

{

std::cout << matrix[i][j] << " ";

}

std::cout << std::endl;

}

std::cout << std::endl;

float tmp;

int k;

float \*xx = new float[changing-1];

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

{

tmp = matrix[i][i];

for (int j = equations; j >= i; j--)

{

matrix[i][j] /= tmp;

}

for (int j = i + 1; j < equations; j++)

{

tmp = matrix[j][i];

for (int k = equations; k >= i; k--)

{

matrix[j][k] -= tmp\*matrix[i][k];

}

}

}

//обратный ход

xx[equations - 1] = matrix[equations - 1][equations];

for (int i = equations - 2; i >= 0; i--)

{

xx[i] = matrix[i][equations];

for (int j = i + 1; j < equations; j++)

{

xx[i] -= matrix[i][j] \* xx[j];

}

}

//вивід масива

std::cout << "Масив елемантів із зміною: " << std::endl;

for (int i = 0; i < changing-1; i++)

{

std::cout << "x" << i << " = " << xx[i] << std::endl;

}

std::cout << std::endl;

delete[] matrix;

}

double function1(double x)

{

return sin(x);

}

double function2(double x)

{

return tan(x);

}

double function3(double x)

{

return x\*x + 2 \* x + 1;

}

double function4(double x)

{

return x\*x\*x + 1;

}

double function5(double x)

{

return -(x\*x) + 3;

}

double input\_a()

{

double a;

std::cout << "Введіть ліву границю: ";

while (!(std::cin >> a) || (std::cin.peek() != '\n')) // цикл для перевірки введеної зміної на символи

{

//std::system("cls");

std::cin.clear();

while (std::cin.get() != '\n');

std::cerr << "Помилка вводу! Повторить ввод. n повино бути цілим" << std::endl;

std::cout << std::endl;

std::cout << "Введіть ліву границю: ";

}

return a;

}

double input\_b()

{

double b;

std::cout << "Введіть праву границю: ";

while (!(std::cin >> b) || (std::cin.peek() != '\n')) // цикл для перевірки введеної зміної на символи

{

//std::system("cls");

std::cin.clear();

while (std::cin.get() != '\n');

std::cerr << "Помилка вводу! Повторить ввод. n повино бути цілим" << std::endl;

std::cout << std::endl;

std::cout << "Введіть праву границю: ";

}

return b;

}

double input\_x(double a, double b)

{

double \_x;

std::cout << "Введіть точку \_х з проміжка: ";

while (!(std::cin >> \_x) || (std::cin.peek() != '\n') || (\_x < a) || (\_x > b)) // цикл для перевірки введеної зміної на символи

{

//std::system("cls");

std::cin.clear();

while (std::cin.get() != '\n');

std::cerr << "Помилка вводу! Точка \_x повина бути дійсним числом та вхадити у проміжок." << std::endl;

std::cout << std::endl;

std::cout << "Введіть точку \_х з проміжка: ";

};

return \_x;

}

int input\_quantity\_equations()

{

double equations;

std::cout << "Кількість рівнянь: ";

while (!(std::cin >> equations) || (std::cin.peek() != '\n')) // цикл для перевірки введеної зміної на символи

{

//std::system("cls");

std::cin.clear();

while (std::cin.get() != '\n');

std::cerr << "Помилка вводу! Повторить ввод. Кількість рівнянь повино бути цілим числом" << std::endl;

std::cout << std::endl;

std::cout << "Кількість рівнянь: ";

}

return equations;

}

int input\_quantity\_changing()

{

double changing;

std::cout << "Кількість зміних: ";

while (!(std::cin >> changing) || (std::cin.peek() != '\n')) // цикл для перевірки введеної зміної на символи

{

//std::system("cls");

std::cin.clear();

while (std::cin.get() != '\n');

std::cerr << "Помилка вводу! Повторить ввод. Кількість зміних повино бути цілим числом" << std::endl;

std::cout << std::endl;

std::cout << "Кількість зміних: ";

}

return changing;

}

source.cpp

#include <iostream>

#include <Windows.h>

#include "func.h"

#include <math.h>

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int number\_fun, number\_metod;

pointFunction mf; pointFunction mP1;pointFunction mP2;

mf = 0;

do

{

system("cls");

std::cout << "За методом Лангранджа(всі наведені приклади):" << std::endl;

std::cout << "1. sin(x)" << std::endl;

std::cout << "2. tan(x)" << std::endl;

std::cout << "3. x\*x + 2 \* x + 1" << std::endl;

std::cout << "4. x\*x\*x + 1" << std::endl;

std::cout << "5. -(x\*x) + 3" << std::endl;

std::cout << "6. Розв'язання системи алгебраїчних рівнянь" << std::endl;

std::cout << "7. Вихід" << std::endl;

std::cin >> number\_fun;

switch (number\_fun)

{

case 1:

{

mf = function1;

}

break;

case 2:

{

mf = function2;

}

break;

case 3:

{

mf = function3;

}

break;

case 4:

{

mf = function4;

}

break;

case 5:

{

mf = function5;

}

break;

case 6:

{

system("cls");

int equations, changing;

equations = input\_quantity\_equations();

changing = input\_quantity\_equations();

gauss(changing, equations);

system("pause");

}

break;

default:

break;

}

if ((number\_fun >= 1) && (number\_fun <= 5))

{

system("cls");

double x[10], y[10];

double a, b, \_x, n, L, h;

a = input\_a();

b = input\_b();

n = (b - a) / 10;

\_x = input\_x(a, b);

std::cout << "";

for (int i = 0; i < 10; i++)

{

x[i] = a + i\*n;

y[i] = mf(x[i]);

};

L = lagrange1(x, y, 10, \_x);

std::cout << "Поліндром Лагранжа L(x) равен: " << L << std::endl;

h = (b - a) / (4 \* n);

std::cout << "x " << "\t" << "L(x) " << std::endl;

for (\_x = a; \_x <= b;\_x = \_x + h)

{

std::cout << \_x << "\t" << lagrange1(x, y, 10, \_x) << std::endl;

}

system("pause");

}

} while (number\_fun != 7);

return 0;

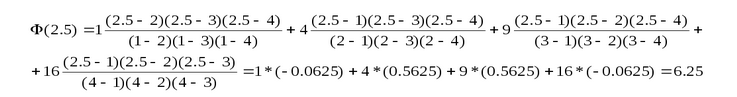
}

**Розробка тестів**

1)Метод Лагранжа

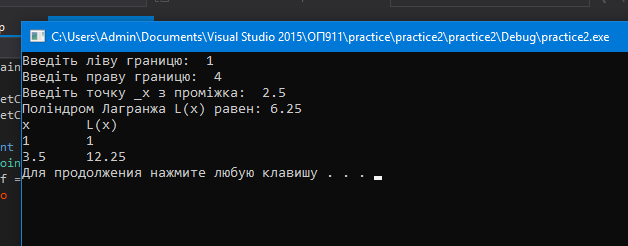
Контрольний приклад

Функція x2 на інтервалі [1; 4] в точці 2,5



Для всіх тестів значення для полиндрома Лагранжа a = 1, b = 4, \_x = 2,5

|  |  |  |
| --- | --- | --- |
| № | Функція | Метод Лагранжа |
| 1 | x2 | 6,25 |

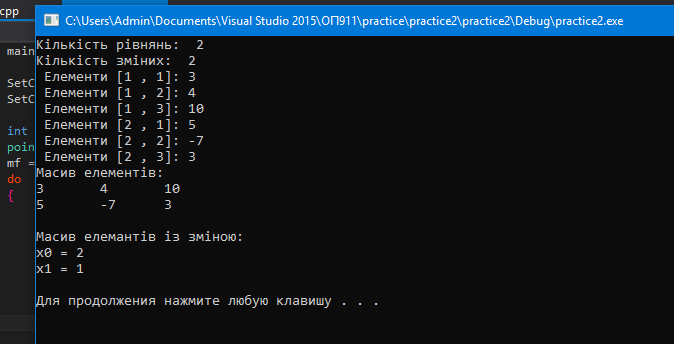


2)Метод Гаусса

Контрольний приклад



Тестів для метода Гауса equations = 2, changing = 2, елементи матриці = 3 4 10 5 -7 3



Висновок: було розроблено програму, що працює з функціями та значень величини за наявним [дискретним](https://uk.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D0%BA%D1%80%D0%B5%D1%82%D0%BD%D1%96%D1%81%D1%82%D1%8C) набором відомих значень (проміжок функції, точки яка знаходиться в цьому проміжку). Був розроблений метод Гауса який знаходить розв’язки системи алгебраїчних рівнянь. При початку роботи програми з’являється меню в якому є запропоновані для вибору функції, після чого він повинен ввести всі необхідні дані, а якщо дані були некоректні програма виведемо повідомлення про помилку та запропонує ввести ще раз. Після введення необхідних даних на екрані з’явиться відповідь.

Контекстне меню було реалізовано за допомогою операторів switch case, функції та методи були реалізовані в окремому .срр файлі, прототипи функцій були винесені в файл .h.