**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Національний університет “Львівська політехніка”**

**Інститут ІКНІ**

**Кафедра ПЗ**



**ЗВІТ**

**До лабораторної роботи №9**

**На тему: “** **НАБЛИЖЕННЯ ФУНКЦІЙ МЕТОДОМ НАЙМЕНШИХ КВАДРАТІВ”**

**З дисципліни : “Чисельні методи”**

**Лектор:**

доц. кафедри ПЗ

Мельник Н.Б.

**Виконав:**

студ. групи ПЗ-18

Кириченко М.К.

**Прийняв:**

проф. каф. ПЗ

Гавриш В.І.

«\_\_\_» \_\_\_\_\_\_ 2023 р.

∑ = \_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

Львів – 2023

**Тема роботи:** Наближення функції методом найменших квадратів.

**Мета роботи:** Ознайомлення на практиці з методом апроксимації (наближення) функції методом найменших квадратів.

**Теоретичні відомості**

Нехай функція  задана таблицею своїх значень: , . Потрібно знайти многочлен фіксованого степеня m, для якого похибка апроксимації - середньоквадратичне відхилення (СКВ)



мінімальне.

Так як многочлен  визначається своїми коефіцієнтами, то фактично треба підібрати набір коефіцієнтів , який мінімізує функцію



Використовуючи необхідну умову екстремуму , , отримуємо так звану ***нормальну систему*** методу найменших квадратів:

, .

Отримана система - це ***система алгебраїчних рівнянь*** відносно невідомих . Можна показати, що визначник цієї системи відмінний від нуля, тобто розв’язок існує і єдиний. Однак при високих степенях m система є погано обумовленою. Тому метод найменших квадратів застосовують для знаходження многочленів, ступінь яких не вищий, ніж 5. Розв’язок нормальної системи можна знайти, наприклад, методом Гаусса.

**Нормальна система методу найменших квадратів**

Запишемо нормальну систему найменших квадратів для двох простих випадків: m = 0 і m = 2.

При m = 0 многочлен прийме вигляд:

.

Для знаходження невідомого коефіцієнта  маємо рівняння:

.

Отримуємо, що коефіцієнт  дорівнює середньому арифметичному значень функції в заданих точках.

Якщо ж використовується многочлен другого порядку

,

то нормальна система рівнянь набуде вигляду:



**Індивідуальне завдання**

Методом найменших квадратів побудувати лінійний, квадратичний і кубічний апроксимаційні поліноми для таблично заданої функції.

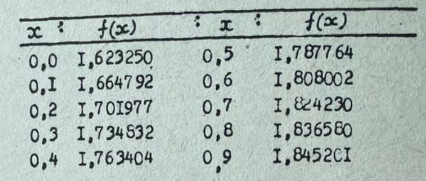


Рис. 1. Таблично задана функція.

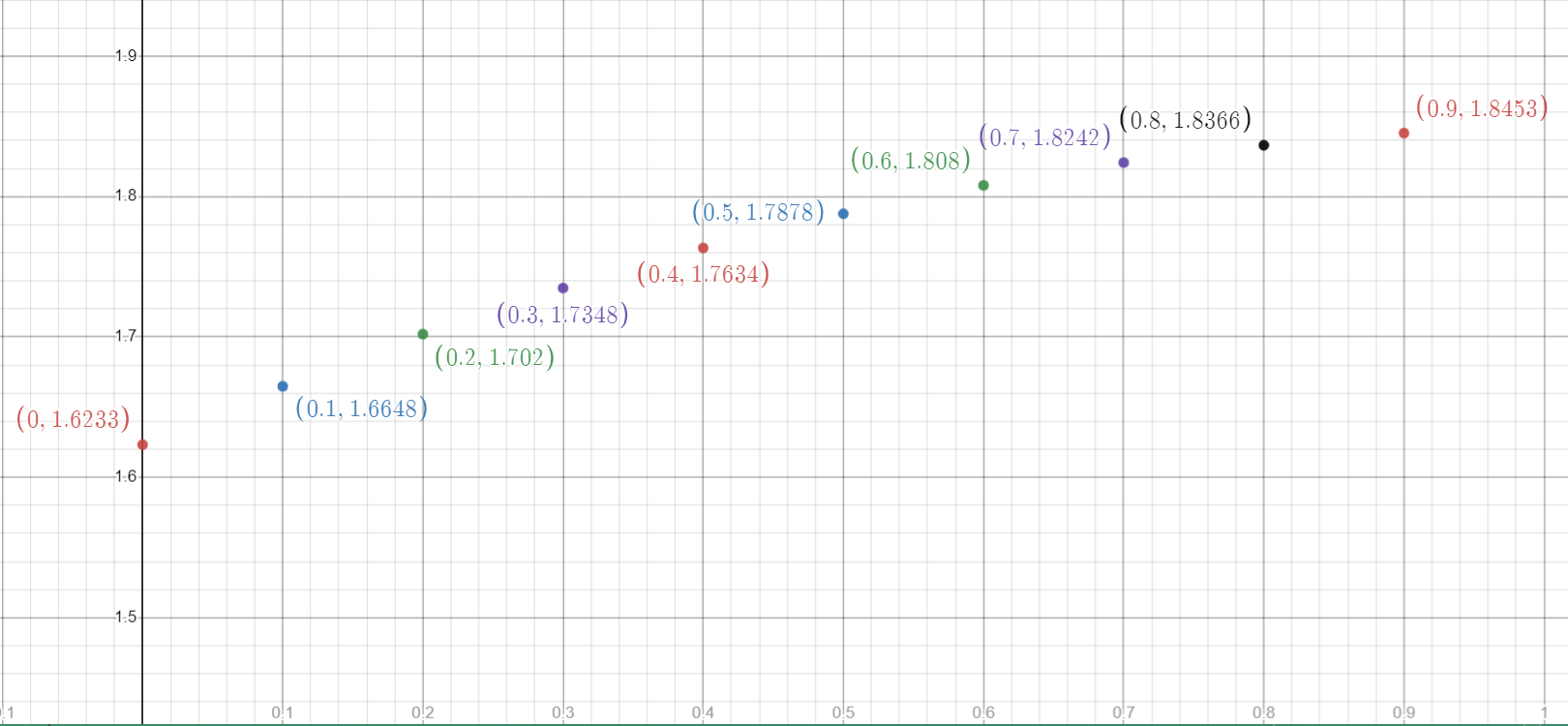


Рис. 2. Геометричне зображення таблично заданої функції.

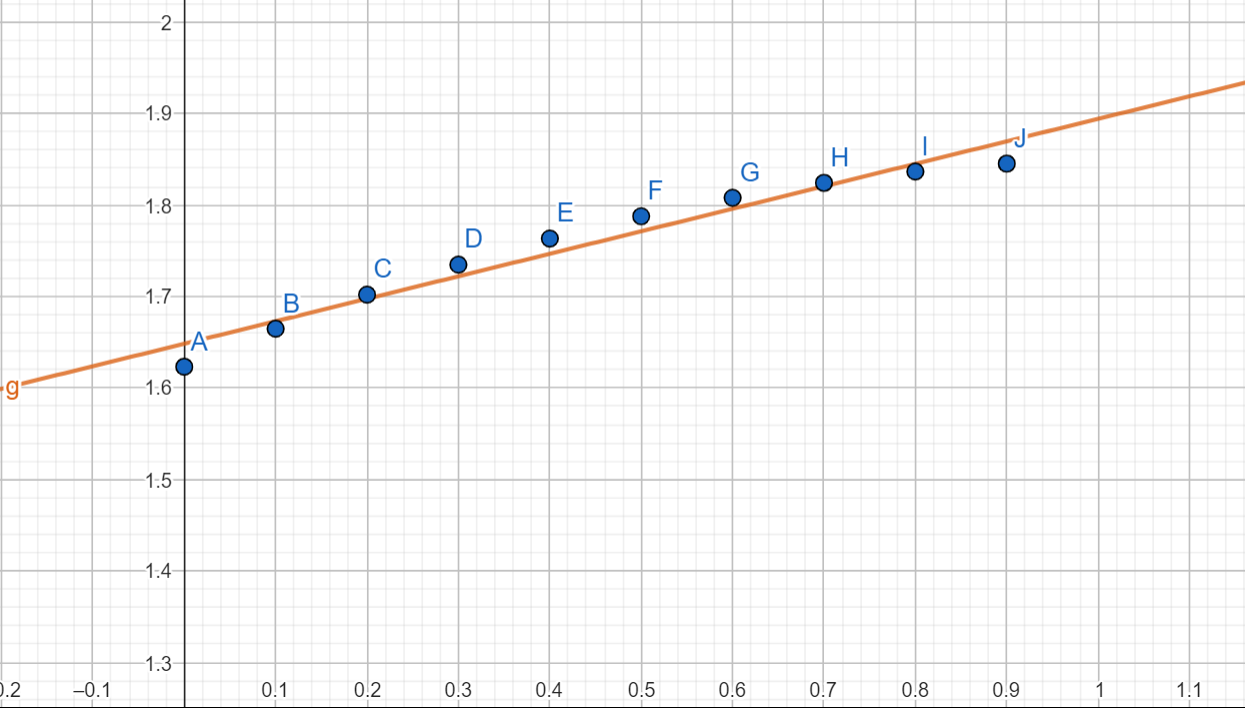
****

Рис. 3. Геометричне зображення лінійного апроксимаційного поліному.

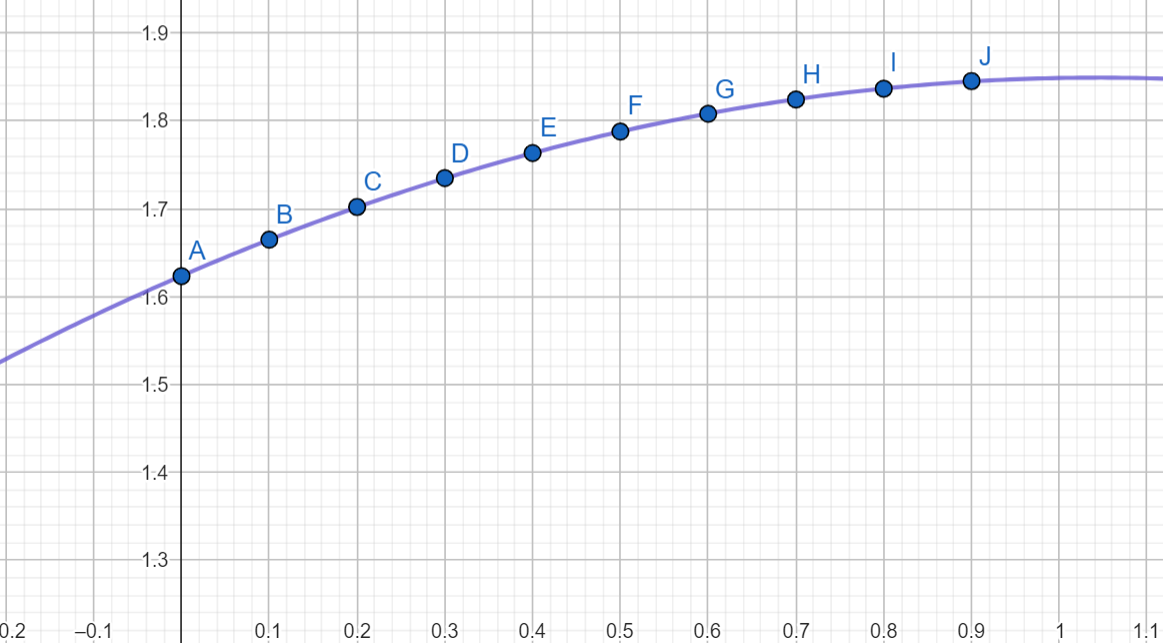


Рис. 4. Геометричне зображення квадратичного апроксимаційного поліному.

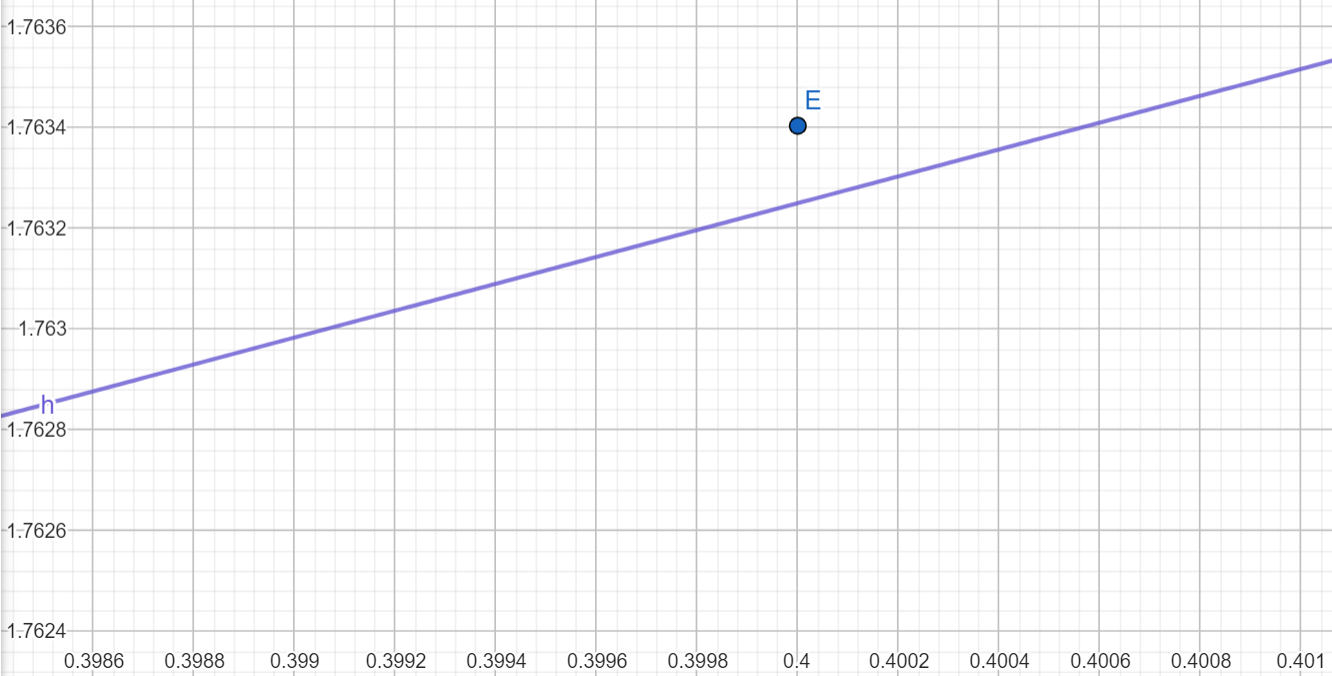


Рис. 5. Геометричне зображення наближення квадратичного апроксимаційного поліному (його точки).

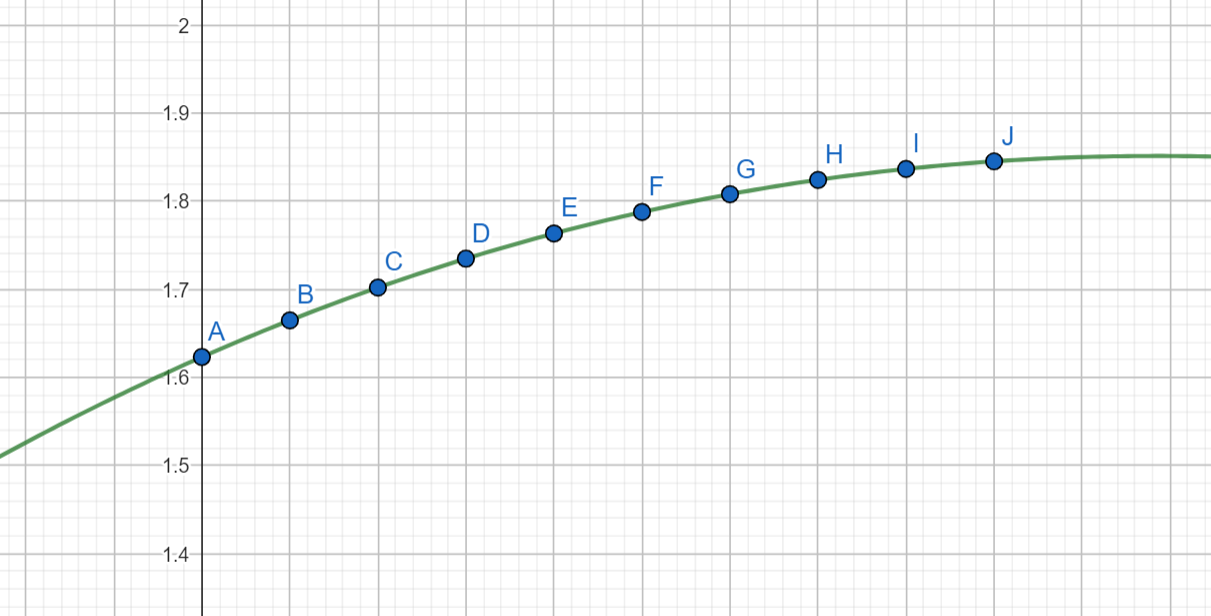
****

Рис. 6. Геометричне зображення кубічного апроксимаційного поліному.

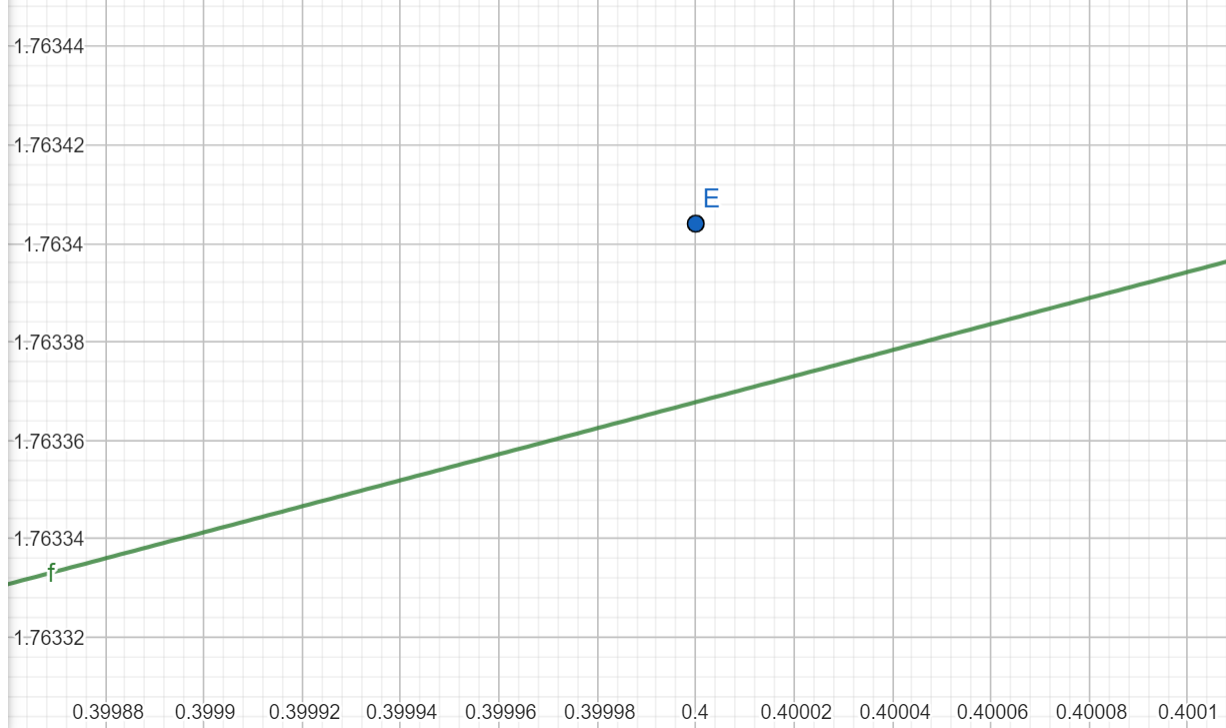
****

Рис. 7. Геометричне зображення наближення кубічного апроксимаційного поліному (його точки).

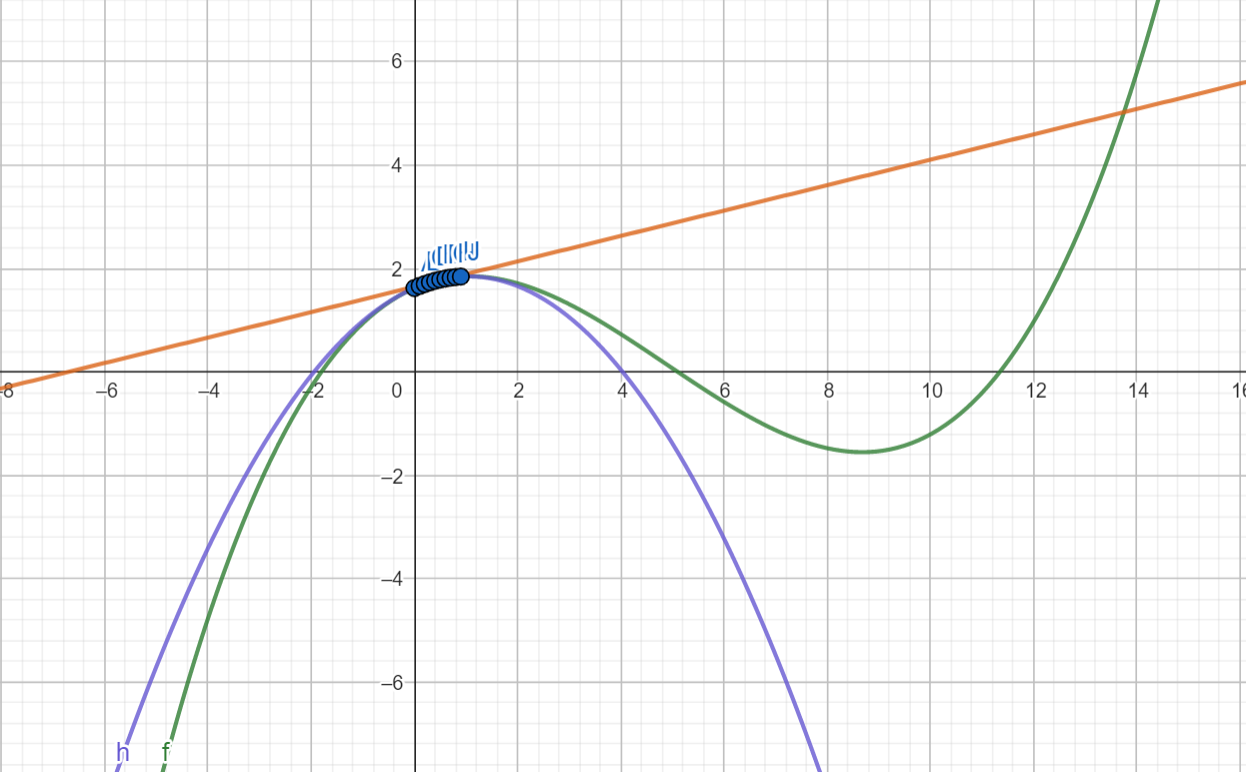
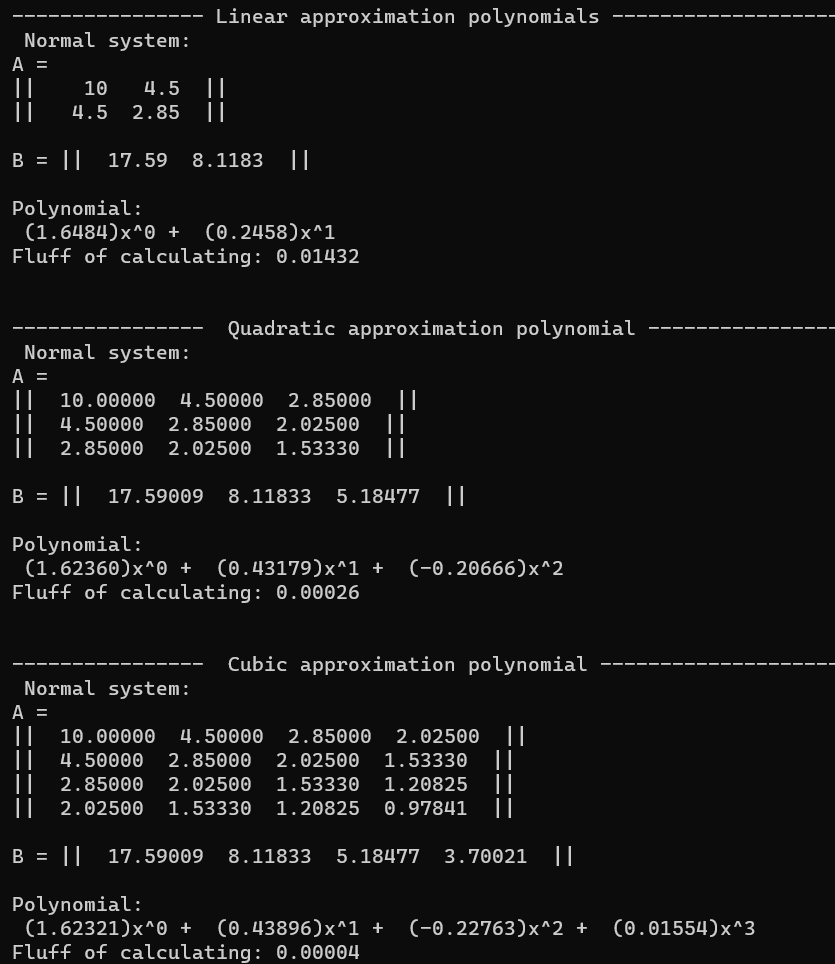
****

Рис. 8. Геометричне зображення всіх знайдених апроксимаційних поліномів у лабораторній роботі.

**Результат виконання програми**

****

**Висновки**

У результаті виконання лабораторної роботи, реалізовано програму побудови лінійного, квадратичного і кубічного апроксимаційних поліномів для таблично заданої функції методом найменших квадратів. Знайдено похибки апроксимації, для лінійного апроксимаційного поліному 0,01432, для квадратичного 0,00026, для кубічного 0,00004. Нормальну систему рівнянь для визначення коефіцієнтів апроксимаційних поліномів розв’язано методом LU - розкладу.

**Додаток**

#include <iostream>

#include <vector>

#include <cmath>

#include <Eigen/Dense>

#include <iomanip>

using std::cout, std::endl, std::vector;

struct NormalSystem

{

vector<vector<double>> A;

vector<double> B;

NormalSystem() = default;

NormalSystem(const NormalSystem& syst)

{

A.resize(syst.A.size(), vector<double>(syst.A.size(), 0));

for (size\_t i = 0; i < syst.A.size(); i++)

{

for (size\_t j = 0; j < syst.A.size(); j++)

{

A.at(i).at(j) = syst.A.at(i).at(j);

}

}

B.resize(syst.B.size());

for (double i : syst.B)

{

B.push\_back(i);

}

}

NormalSystem& operator=(NormalSystem&& syst) noexcept

{

A.resize(syst.A.size(), vector<double>(syst.A.size(), 0));

for (size\_t i = 0; i < syst.A.size(); i++)

{

for (size\_t j = 0; j < syst.A.size(); j++)

{

A.at(i).at(j) = syst.A.at(i).at(j);

}

}

syst.A.clear();

for (double i : syst.B)

{

B.push\_back(i);

}

syst.B.clear();

return \*this;

}

};

class CApproximation {

public:

CApproximation() = default;

CApproximation(vector<double> x\_vector, vector<double> y\_vector);

~CApproximation();

void show\_polinom(std::vector<double> A) const;

\_NODISCARD double get\_fluff(std::vector<double> A) const;

NormalSystem get\_system(size\_t rang);

Eigen::VectorXd solve\_linear\_system(Eigen::MatrixXd& A, Eigen::VectorXd& b);

void show\_coeff(NormalSystem& coeff);

private:

vector<double> m\_X;

vector<double> m\_Y;

};

int main (void)

{

cout << "---------------- Linear approximation polynomials --------------------" << endl;

CApproximation obj{ { 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 }, { 1.623250, 1.664792, 1.701977, 1.734832, 1.763404, 1.787764, 1.808002, 1.824230, 1.836580, 1.845261 } };

NormalSystem system{};

system = obj.get\_system(2);

Eigen::MatrixXd A(system.A.size(), system.A.size() );

Eigen::VectorXd B(system.B.size());

for (size\_t i = 0; i < system.A.size(); i++)

{

for (size\_t j = 0; j < system.B.size(); j++)

{

A(i, j) = system.A.at(i).at(j);

}

B(i) = system.B.at(i);

}

Eigen::VectorXd solutions = obj.solve\_linear\_system(A, B);

vector<double> coefficients;

for (double i : solutions)

{

coefficients.push\_back(i);

}

obj.show\_coeff(system);

obj.show\_polinom(coefficients);

cout << std::fixed << "\nFluff of calculating: " << obj.get\_fluff(coefficients);

///////////////////////

cout << "\n\n\n";

cout << "---------------- Quadratic approximation polynomial --------------------" << endl;

///

///

CApproximation obj1{ { 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 }, { 1.623250, 1.664792, 1.701977, 1.734832, 1.763404, 1.787764, 1.808002, 1.824230, 1.836580, 1.845261 } };

NormalSystem system1{};

system1 = obj1.get\_system(3);

Eigen::MatrixXd A1(system1.A.size(), system1.A.size());

Eigen::VectorXd B1(system1.B.size());

for (size\_t i = 0; i < system1.A.size(); i++)

{

for (size\_t j = 0; j < system1.B.size(); j++)

{

A1(i, j) = system1.A.at(i).at(j);

}

B1(i) = system1.B.at(i);

}

Eigen::VectorXd solutions1 = obj1.solve\_linear\_system(A1, B1);

vector<double> coefficients1;

for (double i : solutions1)

{

coefficients1.push\_back(i);

}

obj1.show\_coeff(system1);

obj1.show\_polinom(coefficients1);

cout << std::fixed << "\nFluff of calculating: " << obj1.get\_fluff(coefficients1);

///////////////////////

cout << "\n\n\n";

cout << "---------------- Cubic approximation polynomial --------------------" << endl;

///

///

CApproximation obj2{ { 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 }, { 1.623250, 1.664792, 1.701977, 1.734832, 1.763404, 1.787764, 1.808002, 1.824230, 1.836580, 1.845261 } };

NormalSystem system2{};

system2 = obj2.get\_system(4);

Eigen::MatrixXd A2(system2.A.size(), system2.A.size());

Eigen::VectorXd B2(system2.B.size());

for (size\_t i = 0; i < system2.A.size(); i++)

{

for (size\_t j = 0; j < system2.B.size(); j++)

{

A2(i, j) = system2.A.at(i).at(j);

}

B2(i) = system2.B.at(i);

}

Eigen::VectorXd solutions2 = obj2.solve\_linear\_system(A2, B2);

vector<double> coefficients2;

for (double i : solutions2)

{

coefficients2.push\_back(i);

}

obj2.show\_coeff(system2);

obj2.show\_polinom(coefficients2);

cout << std::fixed << "\nFluff of calculating: " << obj2.get\_fluff(coefficients2);

}

CApproximation::CApproximation(vector<double> x\_vector, vector<double> y\_vector)

{

for (double i : x\_vector)

m\_X.push\_back(i);

for (double i : y\_vector)

m\_Y.push\_back(i);

}

CApproximation::~CApproximation()

{

m\_X.clear();

m\_Y.clear();

}

void CApproximation::show\_polinom(std::vector<double> A) const

{

cout << " \nPolynomial:\n";

size\_t i = 0;

for (const double data : A)

{

cout << " (" << data << ")x^" << i++ << " + ";

}

cout << "\b\b\b ";

}

double CApproximation::get\_fluff(std::vector<double> A) const

{

double fluff = 0;

for (size\_t i = 0; i < m\_X.size(); i++)

{

double polinom = 0;

for (size\_t j = 0; j < A.size(); j++)

{

polinom += A.at(j) \* pow(m\_X.at(i), j);

}

double polinomMinusYPow = (polinom - m\_Y.at(i)) \* (polinom - m\_Y.at(i));

fluff += polinomMinusYPow;

}

return sqrt(1.0 / static\_cast<double>(m\_X.size()+1) \* fluff);

}

NormalSystem CApproximation::get\_system(size\_t rang)

{

if (m\_X.size() != m\_Y.size())

{

cout << "Number of X values does not match the number of Y value." << endl;

exit(-1);

}

NormalSystem system;

system.A.resize(rang, vector<double>(rang, 0));

for (size\_t k = 0; k < rang; k++)

{

for (size\_t j = 0; j < rang; j++)

{

double buffer{};

for (size\_t i = 0; i < m\_X.size(); i++)

{

buffer += pow(m\_X.at(i), j + k);

}

system.A.at(k).at(j) = buffer;

}

}

for (size\_t k = 0; k < rang; k++)

{

double buffer = 0;

for(size\_t i = 0; i < m\_X.size(); i++)

{

buffer += m\_Y.at(i) \* pow(m\_X.at(i), k);

}

system.B.push\_back(buffer);

}

return system;

}

Eigen::VectorXd CApproximation::solve\_linear\_system(Eigen::MatrixXd& A, Eigen::VectorXd& b)

{

return A.fullPivLu().solve(b);

}

void CApproximation::show\_coeff(NormalSystem& coeff)

{

cout << std::setprecision(5) << "A = " << endl;

for (size\_t i = 0; i < coeff.A.size(); i++)

{

cout << "|| ";

for (size\_t j = 0; j < coeff.A.at(i).size(); j++)

{

cout << std::setw(4) << coeff.A.at(i).at(j) << " ";

}

cout << "||" << endl;

}

cout << "\nB = ";

cout << "|| ";

for (size\_t j = 0; j < coeff.B.size(); j++)

{

cout << coeff.B.at(j) << " ";

}

cout << "||" << endl;

}