МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

ІКНІ

Кафедра **ПЗ**

ЗВІТ

до лабораторної роботи № 2

на тему: “Програмування кривої Безьє”

з дисципліни: Комп’ютерна графіка

Лектор:

к.т.н., доцент

Левус Є.В.

Виконав:

ст. гр. ПЗ-26

Мінтус С. С.

Прийняла:

к.т.н, старший викладач

Ярема Н.П.

Львів – 2024

Тема: Програмування кривої Безьє.

Мета: Навчитися програмувати алгоритм побудови кривої Безьє.

Теоретичні відомості

Для виконання лабораторної роботи було обрано такі технології: HTML, CSS, JS. Для побудови двовимірних зображень було використано <canvas> - це HTML елемент, що використовується для малювання графіки засобами мови програмування JS. Вибрав саме його через простоту у використанні.

Розробив функції для побудови кривої Безьє параметричним методом та матричним. Написав допоміжні функції для додавання нових точок, редагування вже існуючих, множення матриці на матрицю. Реалізував перевірку даних на коректність вводу.

Формулювання завдання

Створити редактор кривої Безьє, який має такий функціонал:

- введення і редагування вершин характеристичної ламаної,
- побудова кривої за параметричною формулою,
- додавання нової точки для характеристичної ламаної,
- контроль коректності введених даних,
- виведення необхідних підказок, повідомлень,
- виконання індивідуального варіанту.

Індивідуальний варіант (Варіант 4)

Візуалізувати криву Безьє за матричною формулою; намалювати дотичні в опорних точках одним кольором, криву – іншим; обчислити координати точок певної кількості на заданому проміжку координати x (кількість, проміжок вводиться користувачем); вивести елементи матриці коефіцієнтів заданого користувачем рядка для матричного представлення кривої та суми елементів головної та побічної діагоналей.

Текст програми

index.html

```
<!DOCTYPE html>

<html lang="en">

<head>
```

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <link rel="stylesheet" href="style.css">
    <title>LAB 2</title>
</head>
<body>
    <main>
        <canvas class="coord" id="canvas"></canvas>
    </main>
    <footer>
        <h1>Lab 2 task</h1>
      <div class="table-container">
        <table>
          <tr>
            <td>
              <span>Point coords:</span>

            </td>
          </tr>
          <tr>
            <td>
                <span>x = </span>
                <input type="text" name="x", id="point-x-add">
              </td>
          </tr>
          <tr>
            <td>
                <span>y = </span>
```

```
                <input type="text" name="y", id="point-y-add">
              </td>
            </tr>
            <tr>
              <td><input type="color" id="curve-color" /></td>

              <td><input type="color" id="touchline-color" /></td>

              <td>
                <input type="button" name="add-point" id="btnAddPoint"
value="Add Point" onclick="addPoint()">
              </td>
            </tr>

            <tr>
              <td>
                <input type="checkbox" name="matrix-method"
id="checkboxMatrix">
                <span>Matrix Method</span>
              </td>
          </tr>
          </table>
          <table>
              <tr>
                <td>
                  <span>Change point coords:</span>
                </td>
              </tr>
              <tr>
```

```
                <td>
                    <span>Point number</span>
                    <select id="select-point">

                    </select>
                </td>
            </tr>
            <tr>
                <td>
                    <span>x = </span>
                    <input type="text" name="x", id="point-x-edit">
                </td>
            </tr>
            <tr>
                <td>
                    <span>y = </span>
                    <input type="text" name="y", id="point-y-edit">
                </td>
            </tr>
            <tr>
                <td>
                <input type="button" name="edit-point"
id="btnEditPoint" value="Edit Point" onclick="editPoint()">
                </td>
            </tr>
        </table>
        <table>
            <tr>
                <td>
```

```
                <span>Get points from interval:</span>
            </td>
        </tr>
        <tr>
            <td>
                <span>x1 = </span>
                <input type="text" name="x", id="point-x1">

            </td>

        </tr>
        <tr>
            <td>
                <span>x2 = </span>
                <input type="text" name="x", id="point-x2">
            </td>
        </tr>
        <tr>
            <td>
                <span>How many points?</span>
                <input type="text" name="x", id="points-count">
            </td>
        </tr>
        <tr>
            <td>
                <input type="button" name="get-points"
id="btnGetPoints" value="Get Points" onclick="getPoints()">
            </td>
        </tr>
```

```
        </table>
        <table>
          <tr>
            <td>
              <span>Enter index of row to show:</span>
            </td>
          </tr>
          <tr>

              <td>
                <input type="text" name="num", id="matrix-row-num">
              </td>
          </tr>
          <tr>

              <td>
                <input type="button" name="get-matrix-data"
id="btnGetMatrixData" value="Get Matrix Data"
onclick="getMatrixData()">
              </td>
          </tr>

        </table>
      </div>
    </footer>
    <script src="script.js"></script>

</body>
</html>
```

style.css

```
* {
    padding: 0;
    margin: 0;
    font-family:'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande',
'Lucida Sans Unicode', Geneva, Verdana, sans-serif;
}

h1 {
    font-size: 24px;
    margin: 0 25px;
}

.table-container {
    display: flex;
}

table {
    padding: 0 50px;
    border-right: 2px black solid;
    border-top: 2px black solid;
}

tr {
    display: flex;
    margin: 5px;
}

td {
```

```
    padding: 0 10px;
}

input[type='text'] {
    width: 50px;
}

input[type='color'] {
    width: 20px;
}
```

script.js

```
let canvas = document.getElementById("canvas");
let context = canvas.getContext("2d");

let window_height = 800;
let window_width = window.innerWidth;

let Points = [];
let bezierPoints = [];
let step = 0.005;

let matrix = [];
let bezierPointsByMatrixMethod = [];

canvas.width = window_width;
canvas.height = window_height;

drawCoords();
```

```
// console.log(bezierPoints);


function determineMatrix(n) {

    matrix = [];

    for(let i = 0; i <= n; ++i) {

        matrix[i] = [];

        for(let j = 0; j <= n; ++j) {

            matrix[i][j] = 0;

            if(n - j - i >= 0)

            matrix[i][j] = binomialCoef(n, j) * binomialCoef(n - j, n -
j - i) * Math.pow(-1, n - i - j);

            else matrix[i][j] = 0;

        }

    }


    // console.log(matrix);

}


function matrixMethod() {

    determineMatrix(Points.length - 1);

    context.beginPath();

    context.moveTo(Points[0][0], Points[0][1]);

    let t = 0;

    let matrPx = [];

    let matrPy = [];


    for(let i = 0; i < Points.length; ++i) {

        matrPx[i] = Points[i][0];
```

```
        matrPy[i] = Points[i][1];
    }

    while(Math.abs(1 - t) >= 0.0001) {
        let matrT = [];

        for(let i = 0; i < matrix.length; ++i) {
            matrT[i] = Math.pow(t, matrix.length - 1 - i);
        }

bezierPointsByMatrixMethod.push([matrixMultiplication(matrixMultiplicat
ion(matrT, matrix), matrPx),
matrixMultiplication(matrixMultiplication(matrT, matrix), matrPy)]);

console.log(bezierPointsByMatrixMethod[bezierPointsByMatrixMethod.lengt
h - 1]);

        if(t != 0)

context.lineTo(bezierPointsByMatrixMethod[bezierPointsByMatrixMethod.le
ngth - 1][0],
bezierPointsByMatrixMethod[bezierPointsByMatrixMethod.length - 1][1]);
        t += step;
    }
    //
context.lineTo(bezierPointsByMatrixMethod[bezierPointsByMatrixMethod.le
ngth - 1][0],
bezierPointsByMatrixMethod[bezierPointsByMatrixMethod.length - 1][1]);

    context.strokeStyle = document.getElementById('curve-color').value;
    context.stroke();
```

```
    if(Points.length >= 3)
    drawTouchlines();

    context.closePath();




}

function matrixMultiplication(matr1, matr2) {
    if(typeof(matr2[0]) == 'number') {
        let res = 0;
        for(let i = 0; i < matr1.length; ++i) {
            res += matr1[i] * matr2[i];
        }
        console.log("NUMBER" + res);
        return res;
    }

    let resultMatrix = [];

    for(let i = 0; i < matr2.length; ++i) {
        resultMatrix[i] = 0;
        for(let j = 0; j < matr2.length; ++j) {
            resultMatrix[i] += matr1[j] * matr2[j][i];
        }
    }
```

```
    console.log("MATRIX" + resultMatrix);

    return resultMatrix;


}


function getPoints() {

    let x1 = document.getElementById('point-x1').value;

    let x2 = document.getElementById('point-x2').value;

    let count = document.getElementById('points-count').value;


    if(Math.abs(x1 * 10) > canvas.width / 2 || Math.abs(x2 * 10) >
canvas.width / 2) {

        alert('Non valid coordinates!');

        return false;

    }


    let points = [];


    for (let point of bezierPoints) {

        // console.log(point);

        if((point[0] > canvas.width / 2 + x1 * 10) && (point[0] <
canvas.width / 2 + x2 * 10))

            points.push([(point[0] - canvas.width / 2) / 10, (point[1]
- canvas.height / 2) / 10]);


        if(points.length == count) break;

    }


    let msg = "";
```

```
    for(let point of points) {
        msg += point;
        msg += '\n';
    }

    alert(msg);


}

function getMatrixData() {
    let row = document.getElementById('matrix-row-num').value;

    if(row > matrix.length - 1 || matrix == []) {
        alert('Uncorrect data!');
        return 0;
    }

    let msg = 'ROW ' + row + ':\n';

    for(let j = 0; j < matrix.length; ++j) {
        msg += matrix[row][j] + ' ';
    }
    msg += '\n\n';

    let sumMainDiag = 0;
    let sumAnotherDiag = 0;
```

```
    for(let i = 0; i < matrix.length; ++i) {
        sumMainDiag += matrix[i][i];
        sumAnotherDiag += matrix[i][matrix.length-i-1];
    }

    msg += 'Sum of main diagonal = ' + sumMainDiag + '\n';
    msg += 'Sum of another diagonal = ' + sumAnotherDiag + '\n';

    alert(msg);
}

function updateGraph() {
    bezierPoints = [];
    bezierPointsByMatrixMethod = [];
    context.clearRect(0, 0, canvas.width, canvas.height);
    context.beginPath();
    drawCoords();
    context.closePath();

    if(document.getElementById('checkboxMatrix').checked)
        matrixMethod();
    else
        bezierCurve(Points.length - 1);

    drawPoints();
}

function editPoint() {
```

```
    let num = document.getElementById('select-point').value - 1;
    let x = document.getElementById('point-x-edit').value;
    let y = document.getElementById('point-y-edit').value;

    if(Math.abs(x * 10) > canvas.width / 2 || Math.abs(y * 10) >
canvas.height / 2) {
        alert('Non valid coordinates!');
        return false;
    }

    Points[num][0] = canvas.width / 2 + x * 10;
    Points[num][1] = canvas.height / 2 - y * 10;

    updateGraph();
}

function addPoint() {
    let x = document.getElementById('point-x-add').value;
    let y = document.getElementById('point-y-add').value;

    if(Math.abs(x * 10) > canvas.width / 2 || Math.abs(y * 10) >
canvas.height / 2) {
        alert('Non valid coordinates!');
        return false;
    }

    Points.push(
        [canvas.width / 2 + x * 10, canvas.height / 2 - y * 10]
```

```
        );
    updateGraph();

    let select = document.getElementById('select-point');

    let option = document.createElement('option');
    option.innerHTML = Points.length;

    select.appendChild(option);

    // determineMatrix(Points.length - 1);

}

function bezierCurve(n) {
    context.beginPath();
    context.moveTo(Points[0][0], Points[0][1]);
    let t = 0;
    while(Math.abs(1 - t) >= 0.0001) {
        sumB(n, t);
        if(t != 0)
            context.lineTo(bezierPoints[bezierPoints.length - 1][0],
bezierPoints[bezierPoints.length - 1][1]);
        t += step;
    }
    sumB(n, t);
    context.lineTo(bezierPoints[bezierPoints.length - 1][0],
bezierPoints[bezierPoints.length - 1][1]);
```

```
    console.log(bezierPoints[bezierPoints.length - 1]);


    context.strokeStyle = document.getElementById('curve-color').value;
    context.stroke();
    if(Points.length >= 3)
        drawTouchlines();
    context.closePath();


}


function drawTouchlines() {
    context.beginPath();
    context.strokeStyle =
document.getElementById('touchline-color').value;
    context.moveTo(Points[0][0], Points[0][1]);
    context.lineTo(Points[1][0], Points[1][1]);
    context.stroke();
    context.closePath();


    context.beginPath();
    context.moveTo(Points[Points.length - 2][0], Points[Points.length -
2][1]);
    context.lineTo(Points[Points.length - 1][0], Points[Points.length -
1][1]);
    context.closePath();
    context.stroke();
    context.strokeStyle = 'black';



}
```

```
function sumB(n, t) {

    bezierPoints.push([0, 0]);

    for(let i = 0; i <= n; ++i) {

        bezierPoints[bezierPoints.length - 1][0] +=
Points[i][0]*polynomBernstein(n, i, t);

        bezierPoints[bezierPoints.length - 1][1] +=
Points[i][1]*polynomBernstein(n, i, t);

    }

}



function binomialCoef(n, i) {

    return factorial(n) / (factorial(i) * factorial (n - i));

}



function polynomBernstein(n, i, t){

    return factorial(n) / (factorial(i)*(factorial(n-i))) * Math.pow(t,
i) * Math.pow(1 - t, n - i);

}



function factorial(num) {

    if(num <= 1) return 1;

    return num * factorial(num-1);

}



function drawPoints() {

    for(let i = 0; i < Points.length; ++i) {

        context.beginPath();

        context.arc(Points[i][0], Points[i][1], 3, 0, 2*Math.PI);

        context.fill();
```

```
        context.font = "12px serif";
        context.fillText(i + 1, Points[i][0] + 12, Points[i][1] - 15);
        context.closePath();
    }
}

function drawCoords() {
    context.moveTo(window_width / 2, 0);
context.lineTo(window_width / 2, window_height);

context.moveTo(0, window_height / 2);
context.lineTo(window_width, window_height / 2);
context.stroke();

context.font = "12px serif";
context.fillText(0, window_width / 2 - 12, window_height / 2 + 15);

// context.textAlign = "start";
for (let i = 0; i < window_width / 2; i += 10) {
  context.moveTo(window_width / 2 + i, window_height / 2 - 3);
  context.lineTo(window_width / 2 + i, window_height / 2 + 3);
  context.moveTo(window_width / 2 - i, window_height / 2 - 3);
  context.lineTo(window_width / 2 - i, window_height / 2 + 3);

  context.stroke();

  if (i % 50 == 0 && i != 0) {
    context.fillText(i / 10, window_width / 2 - 5 + i, window_height /
2 + 14);
```

```
    context.fillText(
      -i / 10,
      window_width / 2 - 10 - i,
      window_height / 2 + 14
    );
  }
}

for (let i = 0; i < window_height / 2; i += 10) {
  context.moveTo(window_width / 2 - 3, window_height / 2 + i);
  context.lineTo(window_width / 2 + 3, window_height / 2 + i);
  context.moveTo(window_width / 2 - 3, window_height / 2 - i);
  context.lineTo(window_width / 2 + 3, window_height / 2 - i);

  context.stroke();
  if (i % 50 == 0 && i != 0) {
    context.fillText(-i / 10, window_width / 2 - 20, window_height / 2
+ i + 5);
    context.fillText(i / 10, window_width / 2 - 20, window_height / 2 -
i + 5);
  }
}

context.font = "20px serif";

context.moveTo(window_width, window_height / 2);
context.lineTo(window_width - 5, window_height / 2 + 5);
context.moveTo(window_width, window_height / 2);
context.lineTo(window_width - 5, window_height / 2 - 5);
```

```
context.fillText("x", window_width - 15, window_height / 2 + 30);


context.moveTo(window_width / 2, 0);

context.lineTo(window_width / 2 - 5, 5);

context.moveTo(window_width / 2, 0);

context.lineTo(window_width / 2 + 5, 5);

context.fillText("y", window_width / 2 - 20, 15);


context.stroke();


}
```

Результати виконання програми

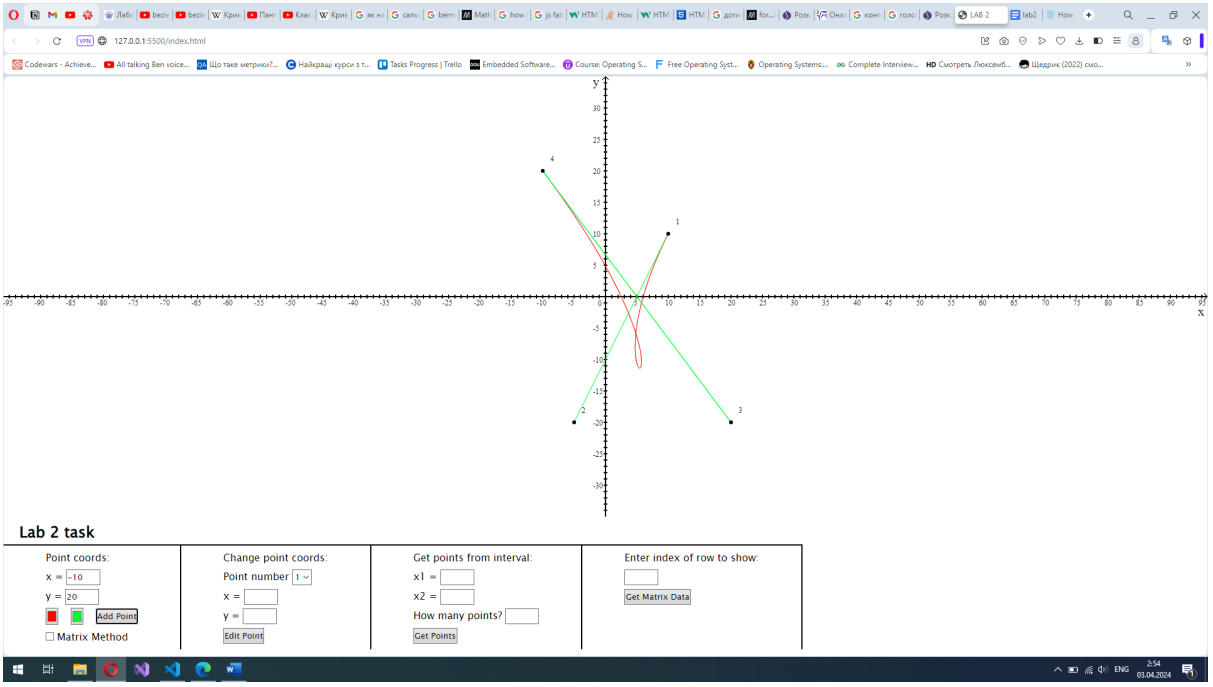

Рис. 1. Інтерфейс програми та побудована крива параметричним методом.

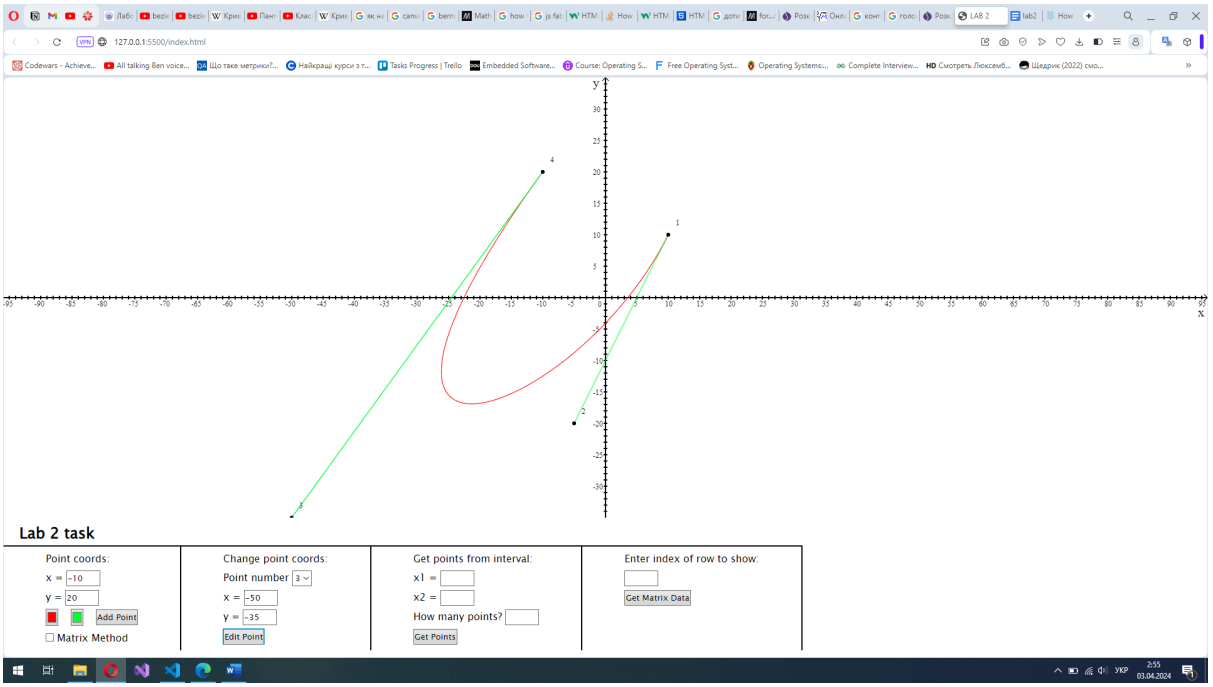

Рис. 2. Крива після зміни координат точки 3.

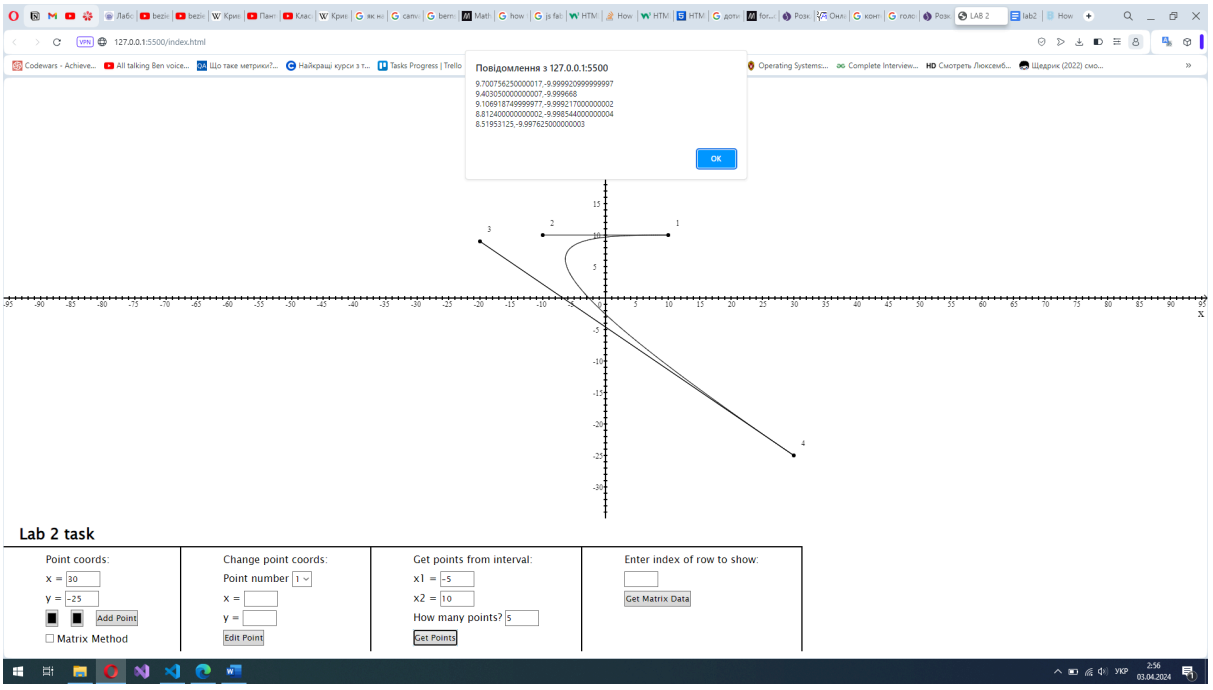

Рис. 3. 5 точок, що лежать на кривій в області [-5; 10].

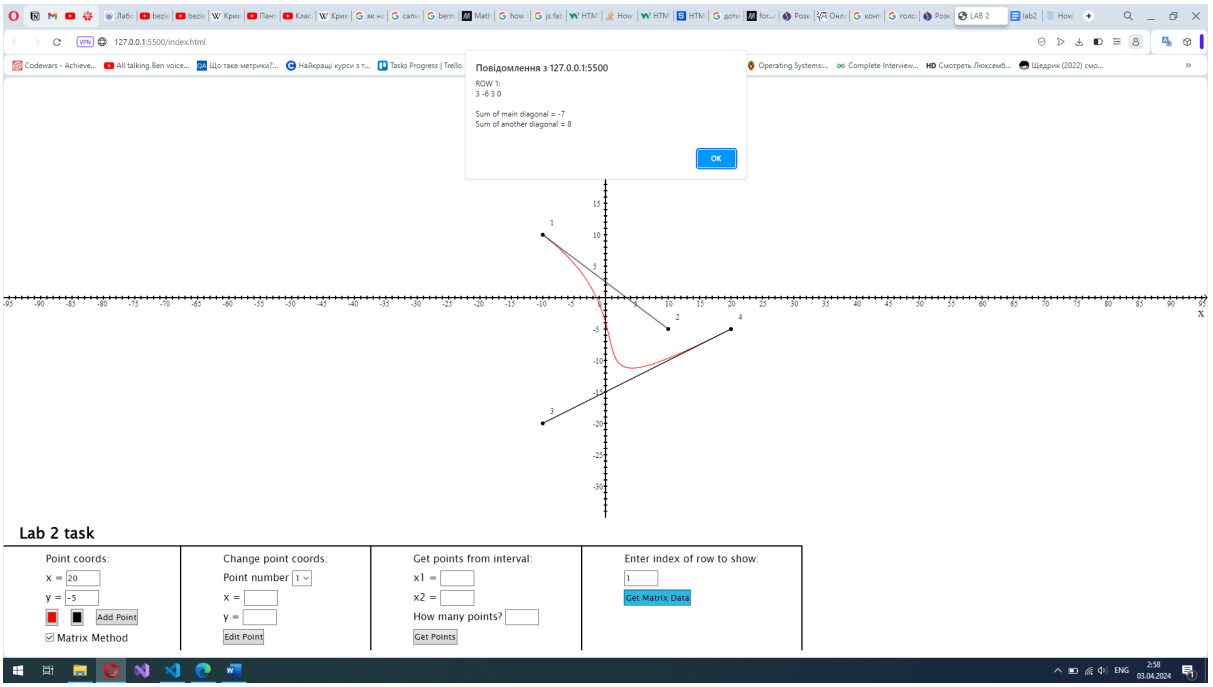

Рис. 4. Крива побудована матричним методом та виведені дані з матриці за індивідуальним завданням.

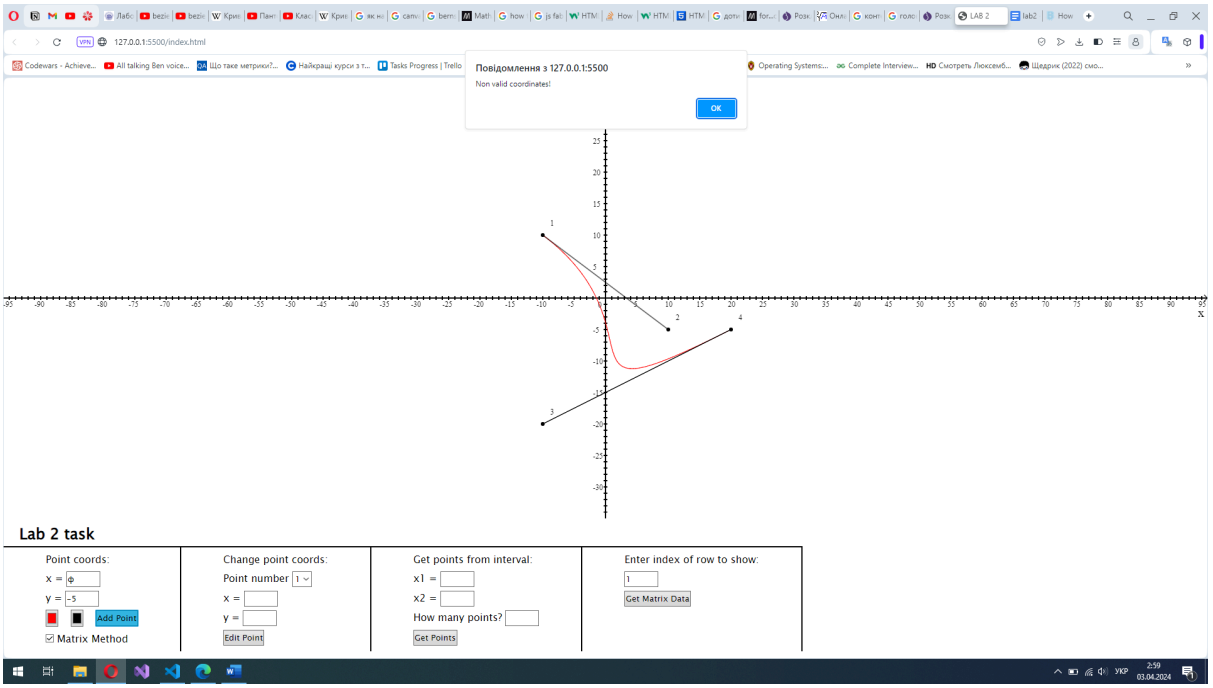

Рис. 5. Перевірка на коректність даних.

Висновки

Завдання лабораторної роботи виконано повністю. Під час його виконання я ознайомився теоретично з кривою Безьє, всі знання закріпив практикою, написавши програму для побудови кривої за введеними координатами точок.

Найбільшу складність для мене представив матричний метод побудови кривої, оскільки при обрахунку матриці дуже легко допуститися

помилки, яка стане критичною. Відстежити помилку виявилось не важко, було достатньо перевірити кожен етап побудови кривої - від обрахунків допоміжних матриць, закінчуючи послідовністю малювання ліній.