Національний технічний університет України «КПІ ім. Ігоря Сікорського»
Факультет Інформатики та Обчислювальної Техніки
Кафедра Обчислювальної Техніки

Лабораторна робота № 5

з дисципліни «Чисельні методи»

на тему

«**Інтерполяційні поліноми**»

Виконав:
студент гр. ІП-93
Домінський Валентин

Викладач:
доц. Рибачук Л.В.

Київ – 2021

Зміст

1 Постановка задачі

Створити програму, яка для заданої функції по заданим точкам будує інтерполяційний поліном Pn(x) у формі Лагранжа, а також здійснює інтерполяцію кубічними сплайнами.

Програма має розрахувавати значення похибки $\varepsilon = | P_n(x) - y(x) |$, для чого потрібно вивести на графік із кроком (графік можна будувати допоміжними засобами, наприклад, у Mathcad), меншим у 5-6 разів, ніж крок інтерполяції, відповідні значення поліному та точної функції. Якщо похибка дуже мала, застосувати масштабування.

Знайти кубічний інтерполяційний сплайн для заданої функції у Mathcad. Вивести графік результатів.

2 Розв'язок

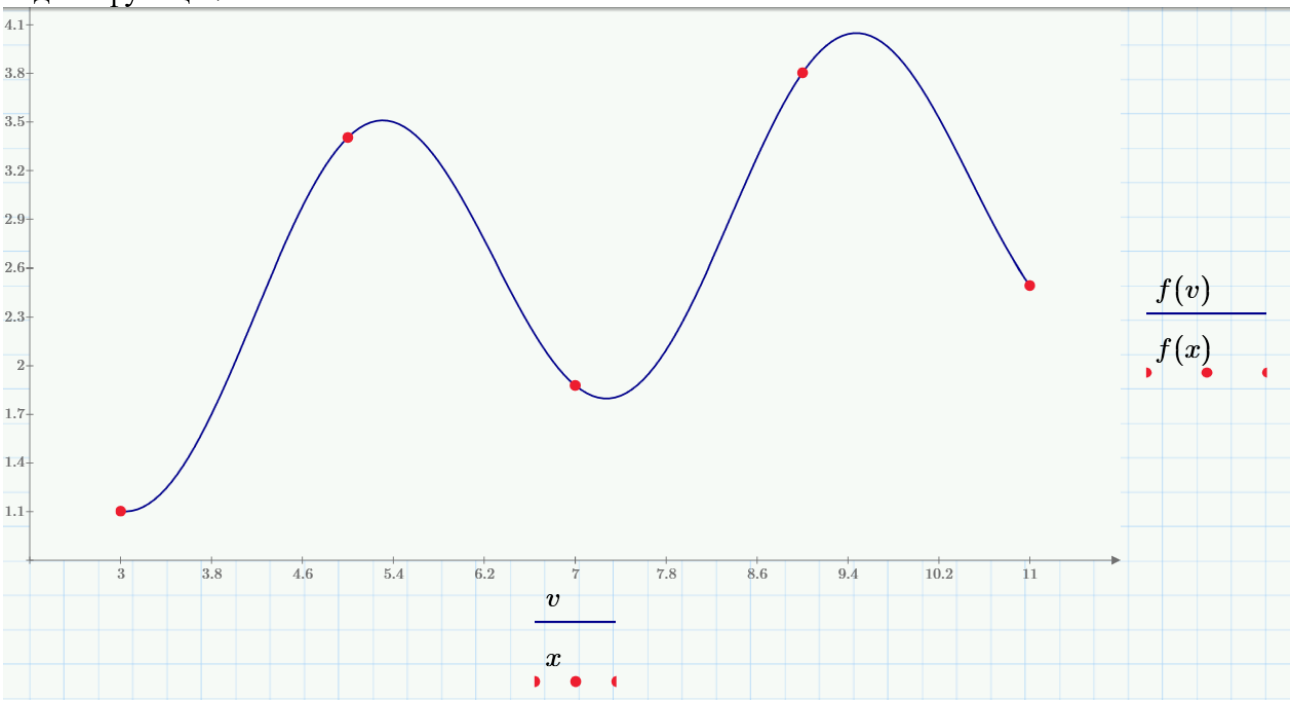
Номер варіанту = 9, отже метод Лагранжа

$$k := 9 - 1 \qquad \alpha := 3$$

$$x1 := -5 + k = 3$$
$$x2 := -3 + k = 5$$
$$x3 := -1 + k = 7$$
$$x4 := 1 + k = 9$$
$$x5 := 3 + k = 11$$

$$y(x) := \sin\left(\frac{\alpha}{2} \cdot x\right) + \sqrt[3]{x \cdot \alpha}$$

Задана функція:



Вигляд поліному Лагранжа:

Перевіряю поліном Лагранжа

$$temp1(z) := y1 \cdot \left(\frac{(z-5)}{(3-5)}\right) \cdot \left(\frac{(z-7)}{(3-7)}\right) \cdot \left(\frac{(z-9)}{(3-9)}\right) \cdot \left(\frac{(z-11)}{(3-11)}\right)$$

$$temp2(z) := y2 \cdot \left(\frac{(z-3)}{(5-3)}\right) \cdot \left(\frac{(z-7)}{(5-7)}\right) \cdot \left(\frac{(z-9)}{(5-9)}\right) \cdot \left(\frac{(z-11)}{(5-11)}\right)$$

$$temp3(z) := y3 \cdot \left(\frac{(z-3)}{(7-3)}\right) \cdot \left(\frac{(z-5)}{(7-5)}\right) \cdot \left(\frac{(z-9)}{(7-9)}\right) \cdot \left(\frac{(z-11)}{(7-11)}\right)$$

$$temp4(z) := y4 \cdot \left(\frac{(z-3)}{(9-3)}\right) \cdot \left(\frac{(z-5)}{(9-5)}\right) \cdot \left(\frac{(z-7)}{(9-7)}\right) \cdot \left(\frac{(z-11)}{(9-11)}\right)$$

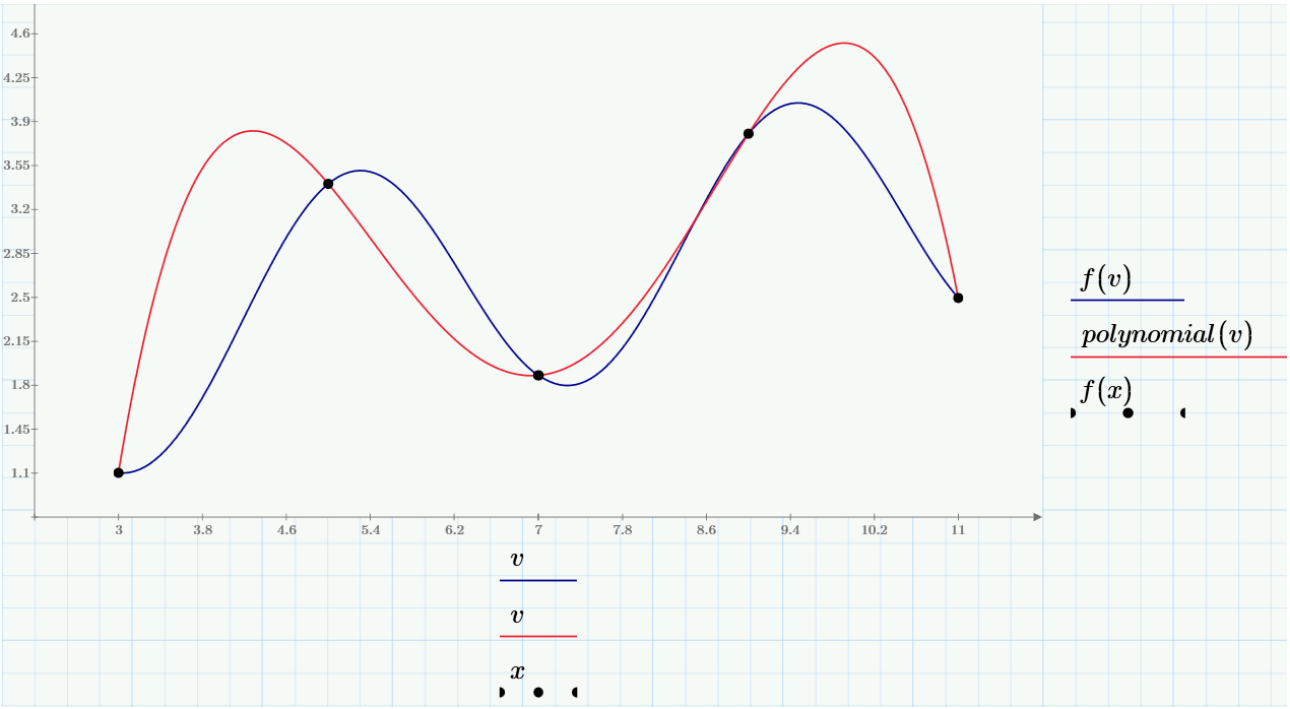
$$temp5(z) := y5 \cdot \left(\frac{(z-3)}{(11-3)}\right) \cdot \left(\frac{(z-5)}{(11-5)}\right) \cdot \left(\frac{(z-7)}{(11-7)}\right) \cdot \left(\frac{(z-9)}{(11-9)}\right)$$

$$polynomial(z) := temp1(z) + temp2(z) + temp3(z) + temp4(z) + temp5(z) \xrightarrow{simplify} -0.036349772706138297917 \cdot z^4 + 1.023981662140071 \cdot z^3 - 10.240190182907537 1082 \cdot z^2 + 42.784387367325635 6992 \cdot z - 59.792070039006980 8924$$

```
Lagrange Polynom:

1.102553705386807 * ((x - 5)/(3 - 5)) * ((x - 7)/(3 - 7)) * ((x - 9)/(3 - 9)) * ((x - 11)/(3 - 11)) + 3.4042120511052087
 * ((x - 3)/(5 - 3)) * ((x - 7)/(5 - 7)) * ((x - 9)/(5 - 9)) * ((x - 11)/(5 - 11)) + 1.8792284164094504 * ((x - 3)/(7 -
3)) * ((x - 5)/(7 - 5)) * ((x - 9)/(7 - 9)) * ((x - 11)/(7 - 11)) + 3.803784426551621 * ((x - 3)/(9 - 3)) * ((x - 5)/(9
- 5)) * ((x - 7)/(9 - 7)) * ((x - 11)/(9 - 11)) + 2.495748987626703 * ((x - 3)/(11 - 3)) * ((x - 5)/(11 - 5)) * ((x - 7)
/(11 - 7)) * ((x - 9)/(11 - 9))
```

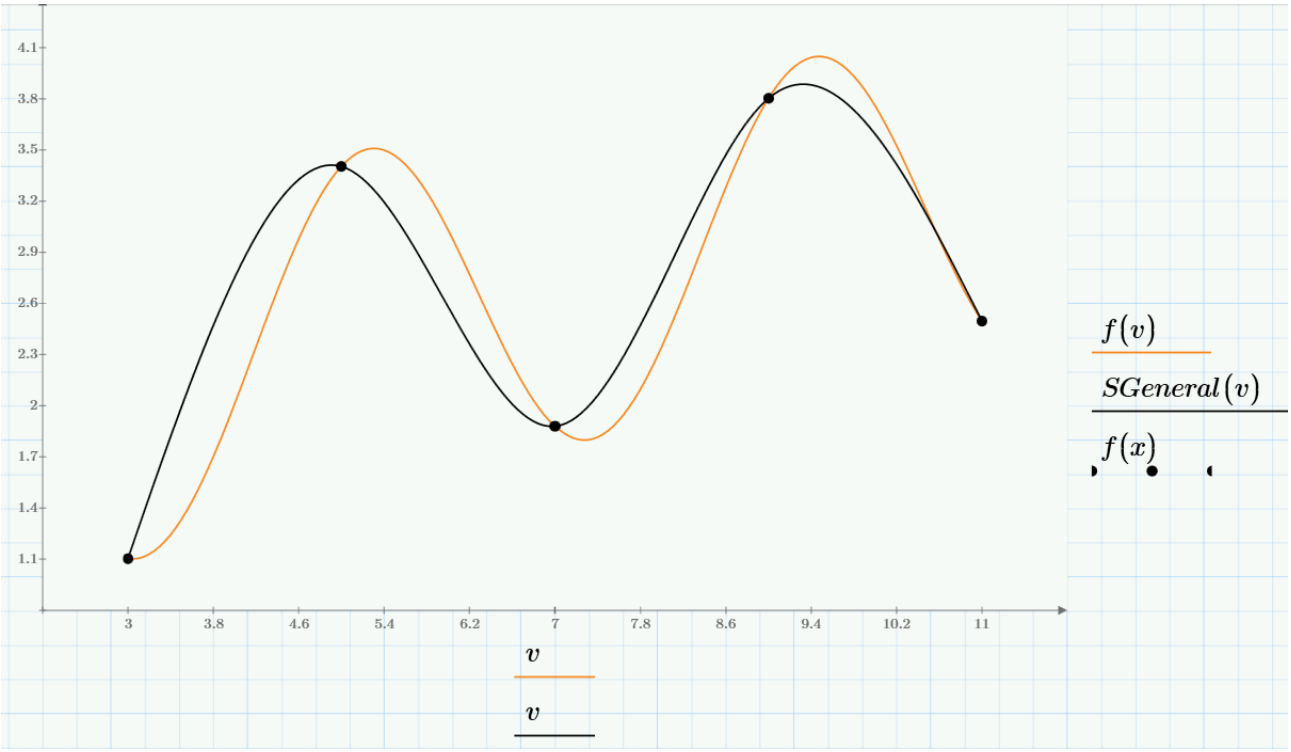
Порівняльний графік функції та інтерполяційного поліному:



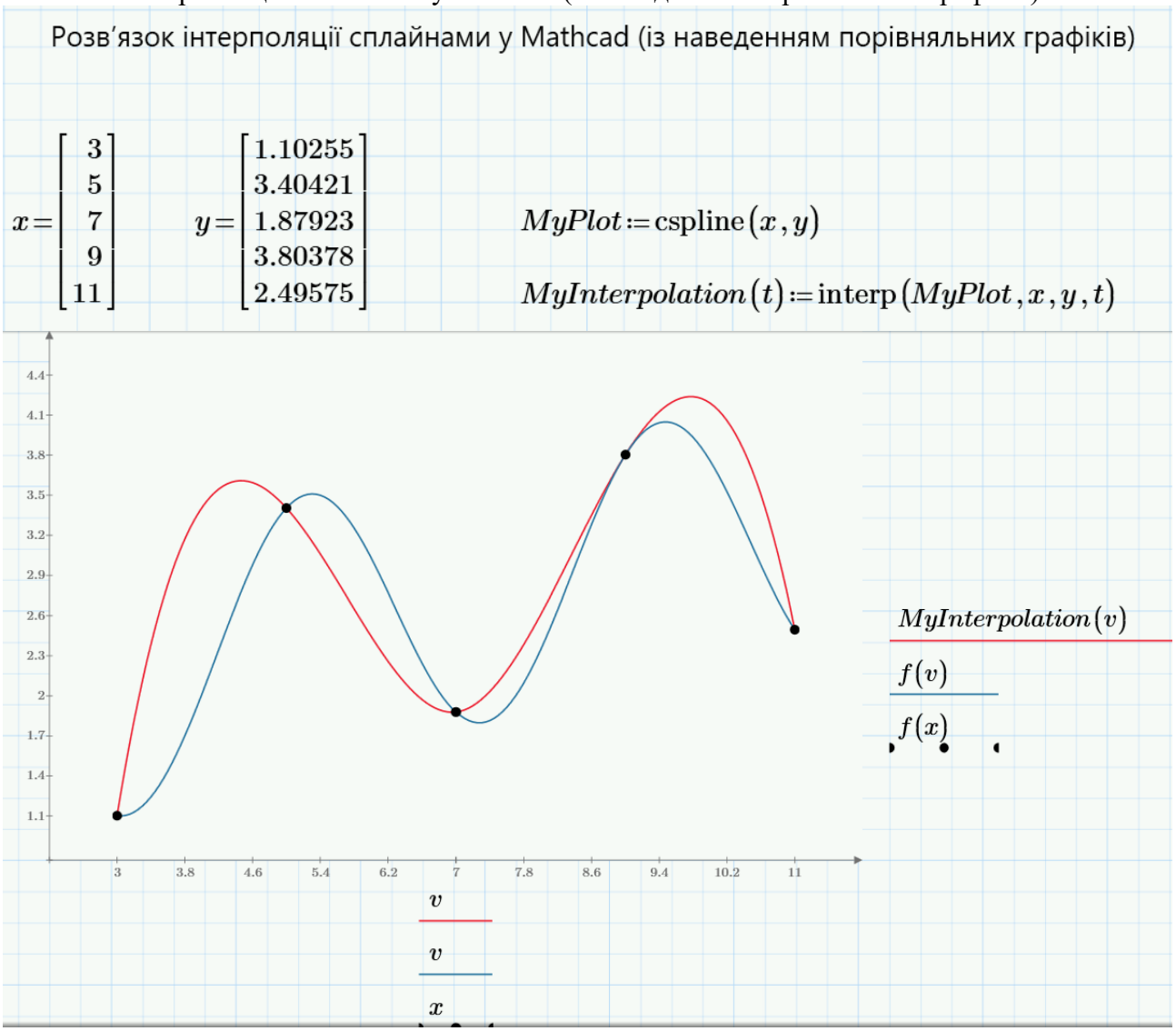
Сплайни (коефіцієнти сплайнових інтерполяційних поліномів):

```
 Coefficients of spline interpolation polynomial =
[ 1.81539 -0.17828  0.06276  0.52658 -0.      -0.99684  1.11736 -0.88545
 -0.16614  0.35237 -0.3338   0.14758]
```

Порівняльний графік функції та сплайн-інтерполяції:



Розв’язок інтерполяції сплайнами у Mathcad (із наведенням порівняльних графіків):



Увесь вивід:

```
 X =
[ 3  5  7  9 11]

 Y =
[1.10255 3.40421 1.87923 3.80378 2.49575]

Lagrange Polynom:

1.102553705386807 * ((x - 5)/(3 - 5)) * ((x - 7)/(3 - 7)) * ((x - 9)/(3 - 9)) * ((x - 11)/(3 - 11)) + 3.4042120511052087
 * ((x - 3)/(5 - 3)) * ((x - 7)/(5 - 7)) * ((x - 9)/(5 - 9)) * ((x - 11)/(5 - 11)) + 1.8792284164094504 * ((x - 3)/(7 -
3)) * ((x - 5)/(7 - 5)) * ((x - 9)/(7 - 9)) * ((x - 11)/(7 - 11)) + 3.803784426551621 * ((x - 3)/(9 - 3)) * ((x - 5)/(9
- 5)) * ((x - 7)/(9 - 7)) * ((x - 11)/(9 - 11)) + 2.495748987626703 * ((x - 3)/(11 - 3)) * ((x - 5)/(11 - 5)) * ((x - 7)
/(11 - 7)) * ((x - 9)/(11 - 9))

Coef of 3 element = 1.102553705386807
Fault of element 3 = 0.0
Coef of 5 element = 3.4042120511052087
Fault of element 5 = 0.0
Coef of 7 element = 1.8792284164094504
Fault of element 7 = 0.0
Coef of 9 element = 3.803784426551621
Fault of element 9 = 0.0
Coef of 11 element = 2.495748987626703
Fault of element 11 = 0.0

 Coefficients of spline interpolation polynomial =
[ 1.81539 -0.17828  0.06276  0.52658 -0.      -0.99684  1.11736 -0.88545
 -0.16614  0.35237 -0.3338   0.14758]
```

3 Розв’язок у Mathcad

Нижче наведено розв’язок системи у Mathcad

Задаю початкові значення

$k := 9 - 1$ $x1 := -5 + k = 3$ $x2 := -3 + k = 5$ $x3 := -1 + k = 7$ $x4 := 1 + k = 9$ $x5 := 3 + k = 11$ $\alpha := 3$

$f(x) := \sin\left(\frac{\alpha}{2} \cdot x\right) + \sqrt[3]{x \cdot \alpha}$ $y1 := f(x1) = 1.10255$ $y2 := f(x2) = 3.40421$ $y3 := f(x3) = 1.87923$

$y4 := f(x4) = 3.80378$ $y5 := f(x5) = 2.49575$

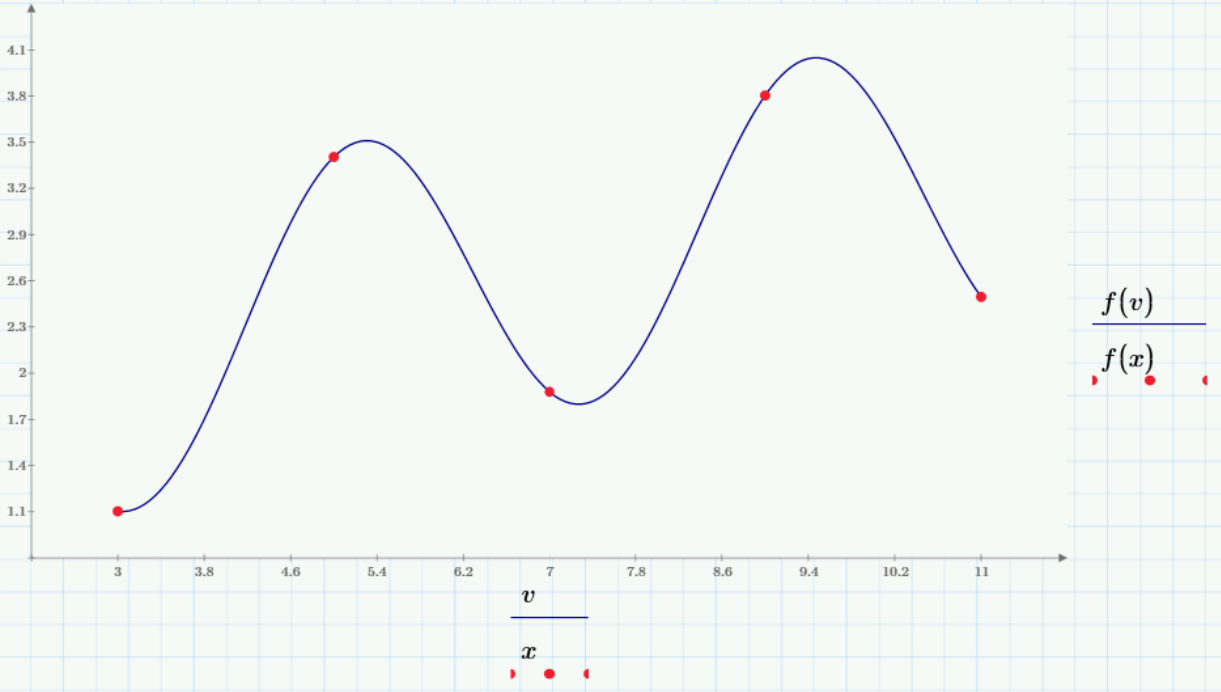
$ORIGIN := 1$

Задаю крок

$step := 0.05$ $v := x1, x1 + step .. x5$

$$x := \begin{bmatrix} x1 \\ x2 \\ x3 \\ x4 \\ x5 \end{bmatrix} \qquad y := \begin{bmatrix} y1 \\ y2 \\ y3 \\ y4 \\ y5 \end{bmatrix}$$

Виводжу графік функції



Перевіряю поліном Лагранжа

$$temp1(z) := y1 \cdot \left(\frac{(z-5)}{(3-5)}\right) \cdot \left(\frac{(z-7)}{(3-7)}\right) \cdot \left(\frac{(z-9)}{(3-9)}\right) \cdot \left(\frac{(z-11)}{(3-11)}\right)$$

$$temp2(z) := y2 \cdot \left(\frac{(z-3)}{(5-3)}\right) \cdot \left(\frac{(z-7)}{(5-7)}\right) \cdot \left(\frac{(z-9)}{(5-9)}\right) \cdot \left(\frac{(z-11)}{(5-11)}\right)$$

$$temp3(z) := y3 \cdot \left(\frac{(z-3)}{(7-3)}\right) \cdot \left(\frac{(z-5)}{(7-5)}\right) \cdot \left(\frac{(z-9)}{(7-9)}\right) \cdot \left(\frac{(z-11)}{(7-11)}\right)$$

$$temp4(z) := y4 \cdot \left(\frac{(z-3)}{(9-3)}\right) \cdot \left(\frac{(z-5)}{(9-5)}\right) \cdot \left(\frac{(z-7)}{(9-7)}\right) \cdot \left(\frac{(z-11)}{(9-11)}\right)$$

$$temp5(z) := y5 \cdot \left(\frac{(z-3)}{(11-3)}\right) \cdot \left(\frac{(z-5)}{(11-5)}\right) \cdot \left(\frac{(z-7)}{(11-7)}\right) \cdot \left(\frac{(z-9)}{(11-9)}\right)$$

$$polynomial(z) := temp1(z) + temp2(z) + temp3(z) + temp4(z) + temp5(z) \xrightarrow{simplify} -0.036349772706138297917 \cdot z^4 + 1.023981662140071 \cdot z^3 - 10.240190182907537 1082 \cdot z^2 + 42.784387367325635 6992 \cdot z - 59.792070039006980 8924$$

Перевіряю похибку без закруглення та з ним

$$|f(x1) - polynomial(x1)| = 0 \qquad |f(x2) - polynomial(x2)| = 0 \qquad |f(x3) - polynomial(x3)| = 0$$

$$|f(x4) - polynomial(x4)| = 0 \qquad |f(x5) - polynomial(x5)| = 0$$

$$polynomialWithRounding(z) := -0.03634 \cdot z^4 + 1.02398 \cdot z^3 - 10.24019 \cdot z^2 + 42.78438 \cdot z - 59.79207$$

$$|f(x1) - polynomialWithRounding(x1)| = 0.00073 \qquad |f(x2) - polynomialWithRounding(x2)| = 0.00587$$

$$|f(x3) - polynomialWithRounding(x3)| = 0.02285$$

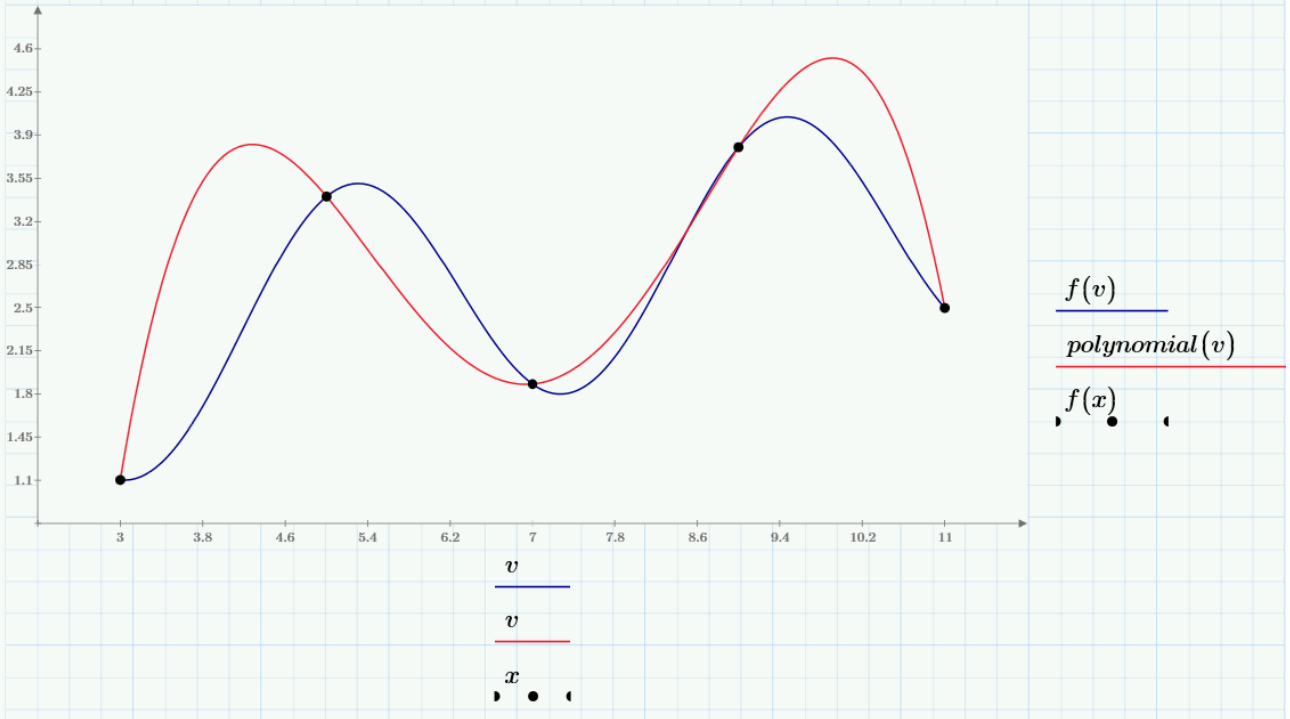
$$|f(x4) - polynomialWithRounding(x4)| = 0.06286 \qquad |f(x5) - polynomialWithRounding(x5)| = 0.14081$$

$$polynomial(x1) = 1.10255 \qquad polynomial(x3) = 1.87923$$

$$polynomial(x5) = 2.49575$$

$$polynomial(x2) = 3.40421 \qquad polynomial(x4) = 3.80378$$

Порівняльний графік функції та інтерполяційного поліному

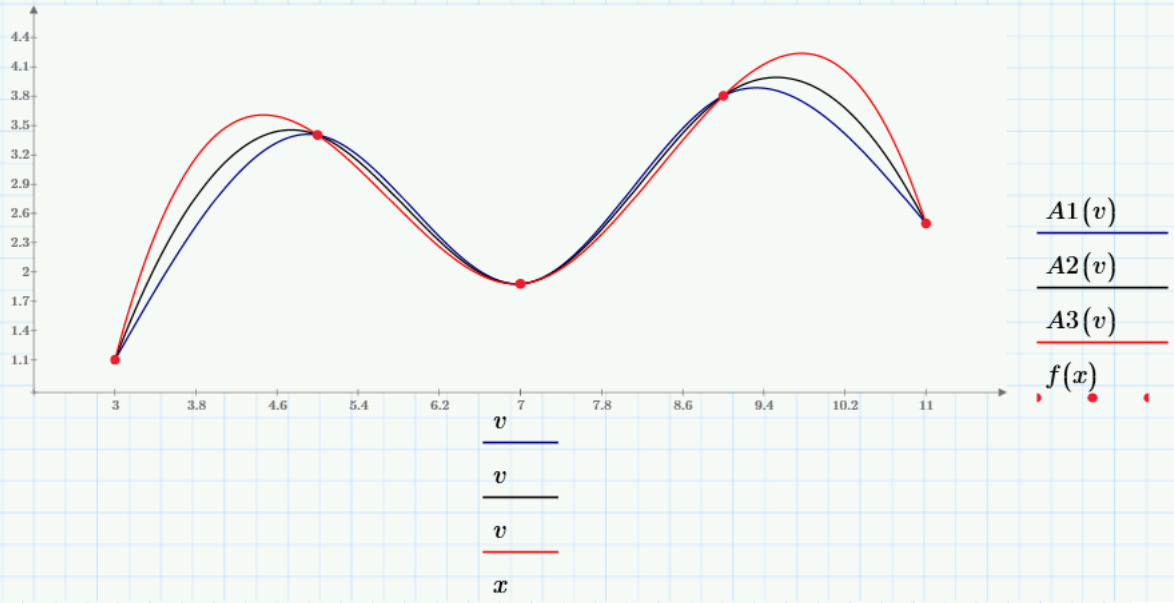


Приклад роботи різних функцій

$exampleS1 := \text{lspline}(x, y)$ $exampleS2 := \text{pspline}(x, y)$ $exampleS3 := \text{cspline}(x, y)$

$A1(v) := \text{interp}(exampleS1, x, y, v)$ $A2(v) := \text{interp}(exampleS2, x, y, v)$ $A3(v) := \text{interp}(exampleS3, x, y, v)$

$A1(3) = 1.10255$ $A2(3) = 1.10255$ $A3(3) = 1.10255$



Порівняльний графік функції та сплайн-інтерполяції з програми

$$x = \begin{bmatrix} 3 \\ 5 \\ 7 \\ 9 \\ 11 \end{bmatrix} \quad y = \begin{bmatrix} 1.10255 \\ 3.40421 \\ 1.87923 \\ 3.80378 \\ 2.49575 \end{bmatrix} \quad b := \begin{bmatrix} 1.81539 \\ -0.17828 \\ 0.06276 \\ 0.52658 \end{bmatrix} \quad c := \begin{bmatrix} 0 \\ -0.99684 \\ 1.11736 \\ -0.88545 \end{bmatrix} \quad d := \begin{bmatrix} -0.16614 \\ 0.35237 \\ -0.3338 \\ 0.14758 \end{bmatrix}$$

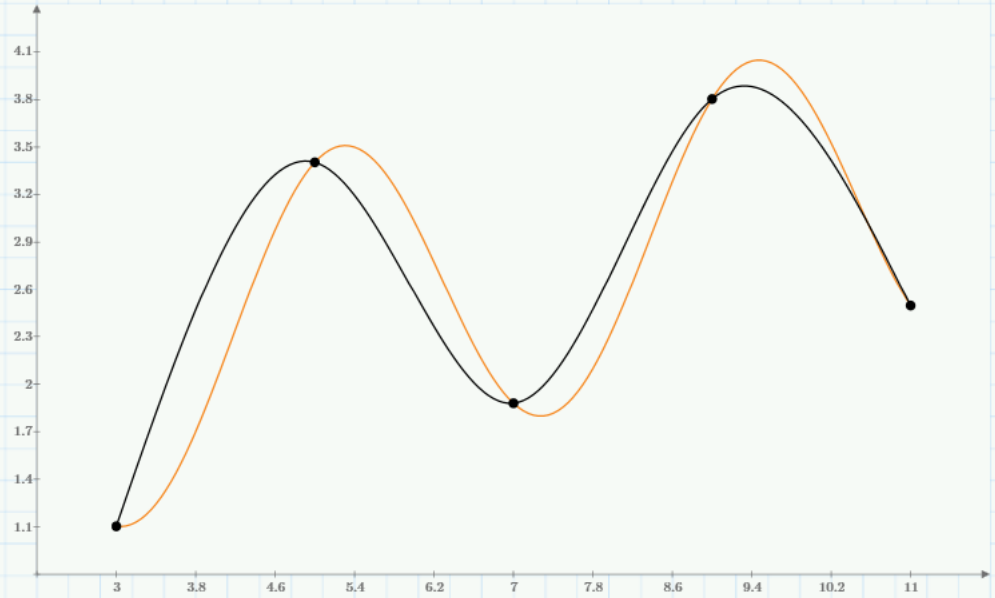
$$S1(t) := 1.10255 + 1.81539\,(t-3) + 0\,(t-3)^2 - 0.16614\,(t-3)^3$$

$$S2(t) := 3.40421 - 0.17828\,(t-5) - 0.99684\,(t-5)^2 + 0.35237\,(t-5)^3$$

$$S3(t) := 1.87923 + 0.06276\,(t-7) + 1.11736\,(t-7)^2 - 0.3338\,(t-7)^3$$

$$S4(t) := 3.80378 + 0.52658\,(t-9) - 0.88545\,(t-9)^2 + 0.14758\,(t-9)^3$$

$$SGeneral(x) := \text{if}(x<5, S1(x), \text{if}(x\geq 5 \wedge x<7, S2(x), \text{if}(x\geq 7 \wedge x<9, S3(x), \text{if}(x\geq 9 \wedge x, S4(x), 10))))$$

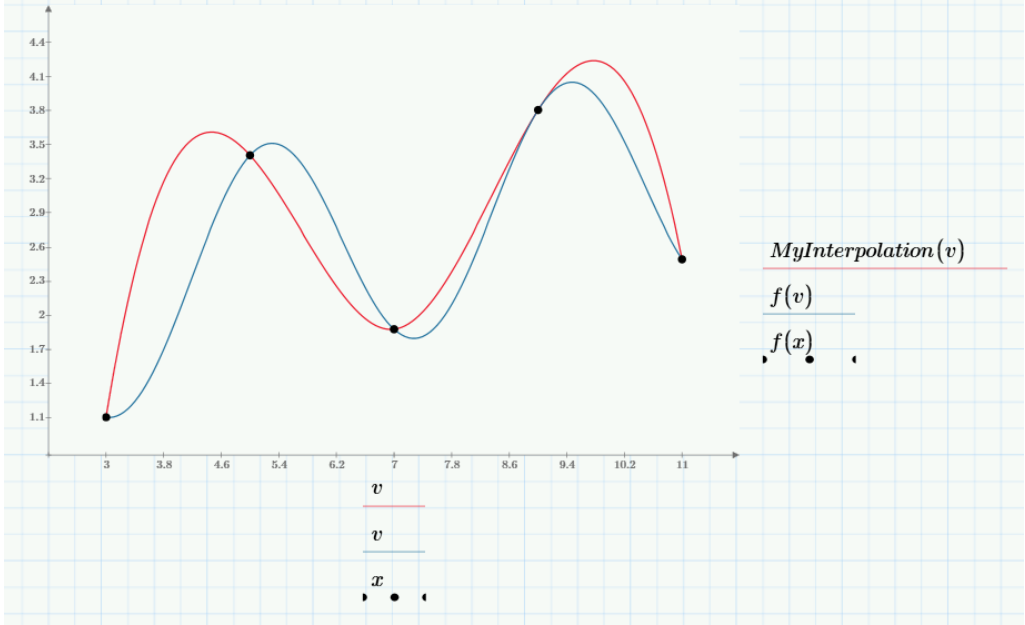


Розв'язок інтерполяції сплайнами у Mathcad (із наведенням порівняльних графіків)

$$x = \begin{bmatrix} 3 \\ 5 \\ 7 \\ 9 \\ 11 \end{bmatrix} \qquad y = \begin{bmatrix} 1.10255 \\ 3.40421 \\ 1.87923 \\ 3.80378 \\ 2.49575 \end{bmatrix}$$

$$MyPlot := \text{cspline}(x, y)$$

$$MyInterpolation(t) := \text{interp}(MyPlot, x, y, t)$$



$$|f(x1) - SGeneral(x1)| = 0 \qquad |f(x2) - SGeneral(x2)| = 0$$

$$|f(x3) - SGeneral(x3)| = 0 \qquad |f(x4) - SGeneral(x4)| = 0$$

$$|f(x5) - SGeneral(x5)| = 0.00003 \qquad |f(5.5) - SGeneral(5.5)| = 0.35852$$

$$|f(3.5) - SGeneral(3.5)| = 0.65865 \qquad |f(4.5) - SGeneral(4.5)| = 0.43377$$

$$|f(6.5) - SGeneral(6.5)| = 0.28894 \qquad |f(7.5) - SGeneral(7.5)| = 0.29292$$

$$|f(8.5) - SGeneral(8.5)| = 0.23487 \qquad |f(9.5) - SGeneral(9.5)| = 0.18404$$

$$|f(8.5) - SGeneral(10.5)| = 0.02651$$

У технічних розрахунках точність вимірювань характеризують відносною похибкою. Результат вважають гарним, якщо відносна похибка не перевищує 0,1 %. Отже Наш результат є гарним

4 Лістинг програми

Lab5.py

```
# region Starting Values
import numpy as np
np.set_printoptions(suppress=True)
from math import sin

rounding = 5
N = 5
numberOfUnknown = 12
lengthOfRowForMatrix = numberOfUnknown + 1

differenceBetweenTwoPoints = 2

firstIndex = 1
secondIndex = 3
thirdIndex = 7
lastIndex = 12

Xarray = [3, 5, 7, 9, 11]

# region xValues for Faults

XarrayForFault1 = [3, 3.25, 3.5, 3.75, 4]
XarrayForFault2 = [4.25, 4.5, 4.75, 5, 5.25]
XarrayForFault3 = [5.5, 5.75, 6, 6.25, 6.5]
XarrayForFault4 = [6.75, 7, 7.25, 7.5, 7.75]
XarrayForFault5 = [8, 8.25, 8.5, 8.75, 9]
XarrayForFault6 = [9.25, 9.5, 9.75, 10, 10.25]
XarrayForFault7 = [10.5, 10.75, 11]

# endregion xValues for Faults

# My Sin Function
def MySinFun(x) -> float:
    element = sin(3 / 2 * x) + (x * 3) ** (1 / 3)
    return element

Yarray = [MySinFun(Xarray[0]), MySinFun(Xarray[1]), MySinFun(Xarray[2]), MySinFun(Xarray[3]),
MySinFun(Xarray[4])]

# region yValues Faults

YarrayForFault1 = [MySinFun(XarrayForFault1[0]), MySinFun(XarrayForFault1[1]),
MySinFun(XarrayForFault1[2]), MySinFun(XarrayForFault1[3]), MySinFun(XarrayForFault1[4])]
YarrayForFault2 = [MySinFun(XarrayForFault2[0]), MySinFun(XarrayForFault2[1]),
MySinFun(XarrayForFault2[2]), MySinFun(XarrayForFault2[3]), MySinFun(XarrayForFault2[4])]
YarrayForFault3 = [MySinFun(XarrayForFault3[0]), MySinFun(XarrayForFault3[1]),
MySinFun(XarrayForFault3[2]), MySinFun(XarrayForFault3[3]), MySinFun(XarrayForFault3[4])]
YarrayForFault4 = [MySinFun(XarrayForFault4[0]), MySinFun(XarrayForFault4[1]),
MySinFun(XarrayForFault4[2]), MySinFun(XarrayForFault4[3]), MySinFun(XarrayForFault4[4])]
YarrayForFault5 = [MySinFun(XarrayForFault5[0]), MySinFun(XarrayForFault5[1]),
MySinFun(XarrayForFault5[2]), MySinFun(XarrayForFault5[3]), MySinFun(XarrayForFault5[4])]
YarrayForFault6 = [MySinFun(XarrayForFault6[0]), MySinFun(XarrayForFault6[1]),
MySinFun(XarrayForFault6[2]), MySinFun(XarrayForFault6[3]), MySinFun(XarrayForFault6[4])]
YarrayForFault7 = [MySinFun(XarrayForFault7[0]), MySinFun(XarrayForFault7[1]),
MySinFun(XarrayForFault7[2])]

# endregion xValues Faults

# endregion Starting Values
```

```
# region Prints

# Print matrix
def PrintMatrixAsNp(matrixName,matrix):
    print("\n", matrixName,"=")
    npMatrix = np.array(matrix)
    print(npMatrix.round(rounding))

# Print vector
def PrintVectorAsNp(vectorName, vector):
    print("\n", vectorName,"=")
    npMatrix = np.array(vector)
    print(npMatrix.round(rounding))

# Print Lagrange
def PrintLagrange(Xarray, Yarray):
    PrintVectorAsNp("X", Xarray)
    PrintVectorAsNp("Y", Yarray)
    print("\nLagrange Polynom:")
    firstPart = f"\n{Yarray[0]} * ((x - {Xarray[1]})/({Xarray[0]} - {Xarray[1]})) * ((x - {Xarray[2]})/({Xarray[0]} - {Xarray[2]})) * ((x - {Xarray[3]})/({Xarray[0]} - {Xarray[3]})) * ((x - {Xarray[4]})/({Xarray[0]} - {Xarray[4]})) +"
    secondPart = f"{Yarray[1]} * ((x - {Xarray[0]})/({Xarray[1]} - {Xarray[0]})) * ((x - {Xarray[2]})/({Xarray[1]} - {Xarray[2]})) * ((x - {Xarray[3]})/({Xarray[1]} - {Xarray[3]})) * ((x - {Xarray[4]})/({Xarray[1]} - {Xarray[4]})) +"
    thirdPart = f"{Yarray[2]} * ((x - {Xarray[0]})/({Xarray[2]} - {Xarray[0]})) * ((x - {Xarray[1]})/({Xarray[2]} - {Xarray[1]})) * ((x - {Xarray[3]})/({Xarray[2]} - {Xarray[3]})) * ((x - {Xarray[4]})/({Xarray[2]} - {Xarray[4]})) +"
    fourthPart = f"{Yarray[3]} * ((x - {Xarray[0]})/({Xarray[3]} - {Xarray[0]})) * ((x - {Xarray[1]})/({Xarray[3]} - {Xarray[1]})) * ((x - {Xarray[2]})/({Xarray[3]} - {Xarray[2]})) * ((x - {Xarray[4]})/({Xarray[3]} - {Xarray[4]})) +"
    fifthPart = f"{Yarray[4]} * ((x - {Xarray[0]})/({Xarray[4]} - {Xarray[0]})) * ((x - {Xarray[1]})/({Xarray[4]} - {Xarray[1]})) * ((x - {Xarray[2]})/({Xarray[4]} - {Xarray[2]})) * ((x - {Xarray[3]})/({Xarray[4]} - {Xarray[3]}))"
    print(firstPart, secondPart, thirdPart, fourthPart, fifthPart, "\n")

# endregion Prints

# Implementing Lagrange Interpolation
def Lagrange(X_array, Y_array, pointToShow, show) -> float:
    # Create some prerequisites
    resultAsYpoint = 0
    for k in range(len(Y_array)):
        # Create some prerequisites
        tempPoint1 = 0
        tempPoint2 = 0

        tempPointArray = [tempPoint1, tempPoint2]

        tempPointArray[0] = 1
        tempPointArray[1] = 1

        for m in range(len(X_array)):
            if m == k:
                for z in range(2):
                    tempPointArray[z] = tempPointArray[z] * 1

            else:
                tempPointArray[1] = tempPointArray[1] * (X_array[k] - X_array[m])
                tempPointArray[0] = tempPointArray[0] * (pointToShow - X_array[m])

        resultAsYpoint = resultAsYpoint + Y_array[k] * tempPointArray[0] / tempPointArray[1]

    if show:
        print("Coef of",pointToShow,"element =",resultAsYpoint)

    return resultAsYpoint
```

```
def CreateMatrixForCramer(Xarray, Yarray) -> [list, list]:
    # Create some prerequisites
    matrixForCramer = list()
    rightPartForCramer = [0] * (lengthOfRowForMatrix - 1)

    # Call Our functions to create matrix
    FirstPartOfEquation(Xarray, Yarray, matrixForCramer)

    SecondPartOfEquation(Xarray, Yarray, matrixForCramer)

    ThirdPartOfEquation(Xarray, Yarray, matrixForCramer)

    FourthPartOfEquation(Xarray, Yarray, matrixForCramer)

    FifthPartOfEquation(Xarray, Yarray, matrixForCramer)

    # Create Right Part
    for i in range(len(matrixForCramer)):
        rightPartForCramer[i] = matrixForCramer[i][-1]

    # Clean matrix
    matrixForCramer = np.delete(matrixForCramer, np.s_[-1:], axis=1)

    return rightPartForCramer, matrixForCramer

#region PartsOfEquation

def FirstPartOfEquation(Xarray, Yarray, matrixForCramer):
    for i in range(1, len(Xarray)):
        queue = [0] * lengthOfRowForMatrix

        queue[i-firstIndex] = differenceBetweenTwoPoints; queue[i+secondIndex] = differenceBetweenTwoPoints
** 2
        queue[i+thirdIndex] = differenceBetweenTwoPoints ** 3; queue[lastIndex] = Yarray[i] - Yarray[i - 1]

        matrixForCramer.append(queue)

def SecondPartOfEquation(Xarray, Yarray, matrixForCramer):
    for i in range(1, len(Xarray) - 1):
        queue = [0] * lengthOfRowForMatrix

        queue[i] = 1; queue[i-firstIndex] = -1; queue[lastIndex] = 0
        queue[i+secondIndex] = -2 * differenceBetweenTwoPoints; queue[i+thirdIndex] = -3 *
differenceBetweenTwoPoints ** 2

        matrixForCramer.append(queue)

def ThirdPartOfEquation(Xarray, Yarray, matrixForCramer):
    for i in range(1, len(Xarray) - 1):
        queue = [0] * lengthOfRowForMatrix

        queue[i+secondIndex+1] = 1; queue[i+secondIndex] = -1; queue[i+thirdIndex] = -3 *
differenceBetweenTwoPoints; queue[lastIndex] = 0

        matrixForCramer.append(queue)

def FourthPartOfEquation(Xarray, Yarray, matrixForCramer):
    queue = [0] * lengthOfRowForMatrix

    queue[thirdIndex] = 1; queue[lastIndex-1] = 3 * differenceBetweenTwoPoints; queue[lastIndex] = 0

    matrixForCramer.append(queue)
```

```
def FifthPartOfEquation(Xarray, Yarray, matrixForCramer):
    queue = [0] * lengthOfRowForMatrix

    queue[secondIndex+1] = 1; queue[lastIndex] = 0

    matrixForCramer.append(queue)

#endregion PartsOfEquation

# Do Cramer Method
def Cramer(matrixForComputations, matrixForCramer, rightPartForCramer) -> list:
    # Create some prerequisites
    SIPcoeffs = list()

    for k in range(0, len(rightPartForCramer)):
        for h in range(0, len(rightPartForCramer)):
            # Get values for right part
            matrixForComputations[h][k] = rightPartForCramer[h]

            if k > 0:
                matrixForComputations[h][k - 1] = matrixForCramer[h][k - 1]

        # Add value to the vector
        SIPcoeffs.append(np.linalg.det(matrixForComputations) / np.linalg.det(matrixForCramer))

        # Rounding values
        SIPcoeffs[k] = round(SIPcoeffs[k], rounding)

    return SIPcoeffs

# Printing Lagrange Equation
PrintLagrange(Xarray, Yarray)

# region Faults
for i in range(N):
    Lagrange(Xarray, Yarray, Xarray[i], True)
    print("Fault of element", Xarray[i], "=", abs(MySinFun(Xarray[i]) - Lagrange(Xarray, Yarray, Xarray[i],
False)))

print("\n")

for i in range(N):
    Lagrange(XarrayForFault1, YarrayForFault1, XarrayForFault1[i], True)
    print("Fault of element", XarrayForFault1[i], "=", abs(MySinFun(XarrayForFault1[i]) -
Lagrange(XarrayForFault1, YarrayForFault1, XarrayForFault1[i], False)))

for i in range(N):
    Lagrange(XarrayForFault2, YarrayForFault2, XarrayForFault2[i], True)
    print("Fault of element", XarrayForFault2[i], "=", abs(MySinFun(XarrayForFault2[i]) -
Lagrange(XarrayForFault2, YarrayForFault2, XarrayForFault2[i], False)))

for i in range(N):
    Lagrange(XarrayForFault3, YarrayForFault3, XarrayForFault3[i], True)
    print("Fault of element", XarrayForFault3[i], "=", abs(MySinFun(XarrayForFault3[i]) -
Lagrange(XarrayForFault3, YarrayForFault3, XarrayForFault3[i], False)))

for i in range(N):
    Lagrange(XarrayForFault4, YarrayForFault4, XarrayForFault4[i], True)
    print("Fault of element", XarrayForFault4[i], "=", abs(MySinFun(XarrayForFault4[i]) -
Lagrange(XarrayForFault4, YarrayForFault4, XarrayForFault4[i], False)))

for i in range(N):
```

```
    Lagrange(XarrayForFault5, YarrayForFault5, XarrayForFault5[i], True)
    print("Fault of element", XarrayForFault5[i], "=", abs(MySinFun(XarrayForFault5[i]) -
Lagrange(XarrayForFault5, YarrayForFault5, XarrayForFault5[i], False)))

for i in range(N):
    Lagrange(XarrayForFault6, YarrayForFault6, XarrayForFault6[i], True)
    print("Fault of element", XarrayForFault6[i], "=", abs(MySinFun(XarrayForFault6[i]) -
Lagrange(XarrayForFault6, YarrayForFault6, XarrayForFault6[i], False)))

for i in range(len(XarrayForFault7)):
    Lagrange(XarrayForFault7, YarrayForFault7, XarrayForFault7[i], True)
    print("Fault of element", XarrayForFault7[i], "=", abs(MySinFun(XarrayForFault7[i]) -
Lagrange(XarrayForFault7, YarrayForFault7, XarrayForFault7[i], False)))

# endregion Faults

rightPartForCramer, matrixForCramer = CreateMatrixForCramer(Xarray.copy(), Yarray.copy())
matrixForComputations = list(map(list, matrixForCramer))

# Coefficients of spline interpolation polynomials
SIPcoeffs = Cramer(matrixForComputations, matrixForCramer, rightPartForCramer)
PrintVectorAsNp("Coefficients of spline interpolation polynomial", SIPcoeffs)
```

Висновок:

Я навчився використовувати різні методи інтерполяції, визначати сплайн коефіцієнти. Покращив навички роботи з графіками у маткад.