Національний технічний університет України «КПІ ім. Ігоря Сікорського»

Факультет Інформатики та Обчислювальної Техніки

Кафедра Автоматизованих Систем Обробки Інформації та Управління

Лабораторна робота № 5

з дисципліни «Чисельні методи»

на тему

«**Інтерполяційні поліноми**»

Виконала:

студентка гр. ІП-93

Узунлу Гамзенур

Викладач:

доц. Рибачук Л.В.

Київ – 2021

### Зміст

[Зміст 2](file:///C:\Users\vania\Downloads\Telegram%20Desktop\Zvit_example.doc#_Toc190845267)

[1 Постановка задачі 3](file:///C:\Users\vania\Downloads\Telegram%20Desktop\Zvit_example.doc#_Toc190845268)

[2 Розв’язок 4](file:///C:\Users\vania\Downloads\Telegram%20Desktop\Zvit_example.doc#_Toc190845269)

[3 Розв’язок у Mathcad 6](file:///C:\Users\vania\Downloads\Telegram%20Desktop\Zvit_example.doc#_Toc190845270)

[4 Лістинг програми 8](file:///C:\Users\vania\Downloads\Telegram%20Desktop\Zvit_example.doc#_Toc190845271)

### 1 Постановка задачі:

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

Програма має розрахувавати значення похибки | P (x) y(x) | ε = n − , 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.



a = 3

x = 6, 8, 10, 12, 14

### 2 Розв’язок

Результати виконання програми:

Поліном Лагранжа

1.4888697381153229 \*

(x - 8) \*

(x - 10) \*

(x - 12) \*

(x - 14) \*

1 / 384 +

4.271092364654717 \*

(x - 6) \*

1 / -96 +

7.574113448116863 \*

(x - 6) \*

(x - 8) \*

(x - 12) \*

(x - 14) \*

1 / 64 +

8.840312087141179 \*

(x - 6) \*

(x - 8) \*

(x - 10) \*

(x - 14) \*

1 / -96 +

12.79150332861507 \*

(x - 6) \*

(x - 8) \*

(x - 10) \*

(x - 12) \*

1 / 384 +

похибки:

6: 0

6.25: 0.07748513351308817

6.5: 0.09745359457590319

6.75: 0.08433280604681404

7: 0.05691816638401015

7.25: 0.02856476128582175

7.5: 0.007606829564789752

7.75: -0.0020210971009073653

8: 0

8.25: 0.011272117800220194

8.5: 0.027637155832837657

8.75: 0.044132776877698454

9: 0.05588777308994608

9.25: 0.05890126808143137

9.5: 0.05065730794358014

9.75: 0.030536628918661712

10: 0

10.25: -0.037468258893721185

10.5: -0.076653071403749

10.75: -0.1111195073331146

11: -0.1339100279158476

11.25: -0.13838472241972788

11.5: -0.119152294240223

11.75: -0.0730339122794863

12: 0

12.25: 0.09597828285327381

12.5: 0.20621580687156538

12.75: 0.3167828333390048

13: 0.4081899609302564

13.25: 0.45530824723745944

13.5: 0.4275294715831137

13.75: 0.28915657475333134

14: 0

Сплайн інтерполяція

f1

1.4888697381153229 + 1.0677730911636791(x - 6) + -0.23607159812979056(x - 6)^2 + 0.1988703545913998(x - 6)^3

f2

4.271092364654717 + 1.3639298620805242(x - 8) + 0.9571505294186081(x - 8)^2 + -0.4066800947966669(x - 8)^3

f3

7.574113448116863 + 1.780405283567354(x - 10) + -1.4829300393613933(x - 10)^2 + 0.45463852866689763(x - 10)^3

f4

8.840312087141179 + 0.3157274438836226(x - 12) + 1.2449011326399921(x - 12)^2 + -0.20748352210666535(x - 12)^3

похибки:

6: 0

6.25: -0.12476439211427848

6.5: -0.16910266801835117

6.75: -0.15088751736131467

7: -0.09184644815039711

7.25: -0.017370075110630268

7.5: 0.043907346189220764

7.75: 0.060729059088075577

8: 0

8.25: 0.12364728828944038

8.5: 0.15846150921962376

8.75: 0.12816821602948725

9: 0.05836129737530982

9.25: -0.022717820587867976

9.5: -0.08356558318337282

9.75: -0.08900453954164433

10: 0

10.25: -0.15418323176308135

10.5: -0.20549273283366176

10.75: -0.18533665516898346

11: -0.12282333611522134

11.25: -0.045601535010248995

11.5: 0.01920858072242204

11.75: 0.04405158419464428

12: 0

12.25: 0.11340348609521023

12.5: 0.15474755186812672

12.75: 0.14855096704512505

13: 0.11555004478515585

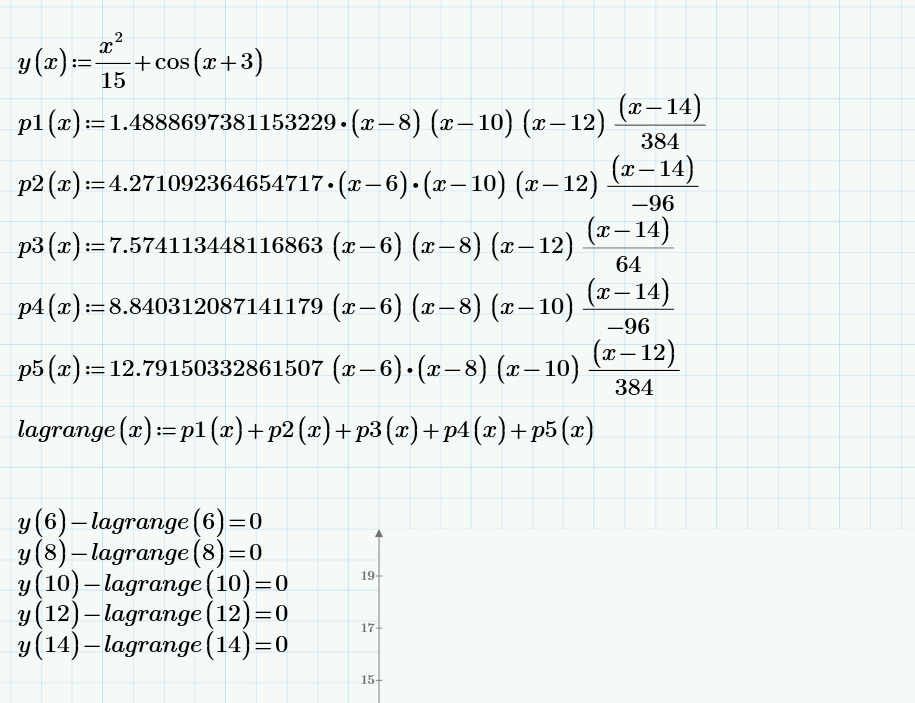
13.25: 0.07261876320671057

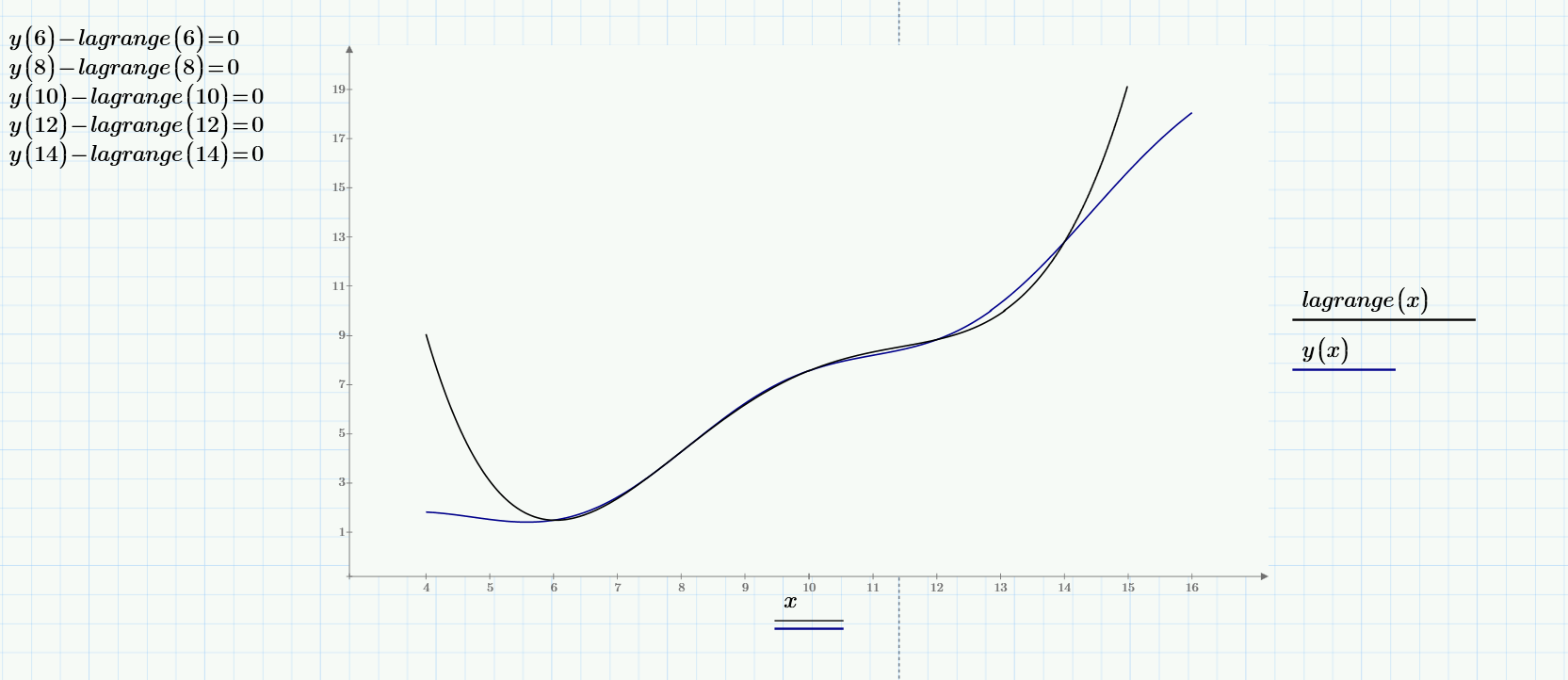
13.5: 0.03292902820068733

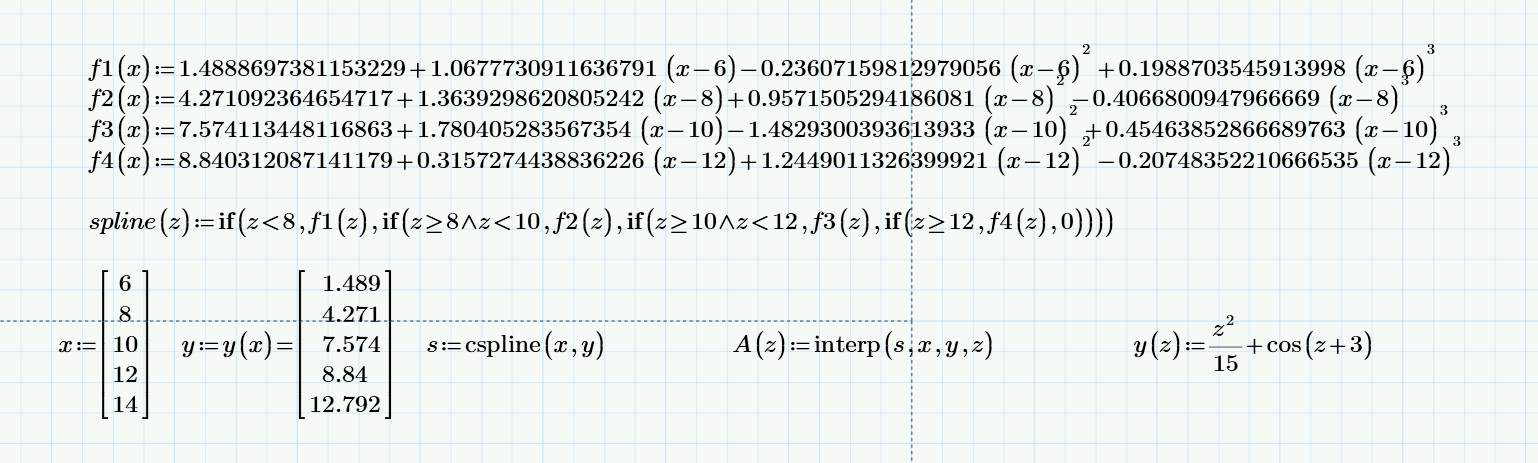
13.75: 0.006341113163653489

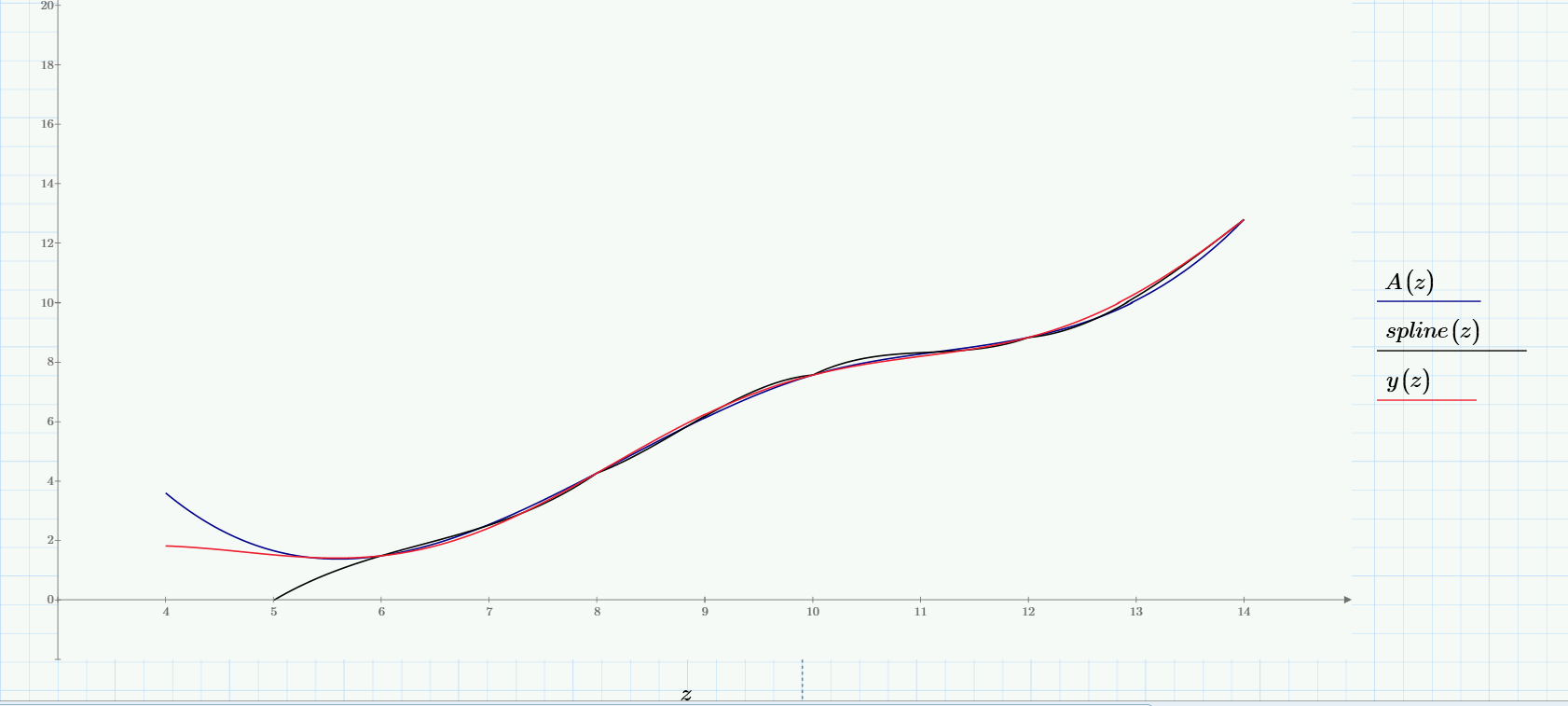
14: 1.7763568394002505e-15

### 3 Розв’язок у MathCad

****

****

****

****

### 4 Лістинг програми

const x = [6, 8, 10, 12, 14]  
const f = (x) => x \* x / 15 + Math.cos(x + 3)  
const y = x.map(v => f(v))  
  
//---------------- Поліном Лагранжа ------------------------  
console.log('Поліном Лагранжа')  
for (let k = 0; k < 5; k++) {  
 console.log(y[k] + ' \* ')  
 for (let i = 0; i < 5; i++) {  
 if (i === k) continue  
 console.log('(x - ' + x[i] + ') \* ')  
 }  
 let denum = 1.0  
 for (let i = 0; i < 5; i++) {  
 if (i === k) continue  
 denum \*= (x[k] - x[i])  
 }  
 console.log('1 / ' + denum + ' + ')  
}  
  
console.log('похибки:')  
const gottenF = (x) => 1.4888697381153229 \* (x - 8) \* (x - 10) \* (x - 12) \* (x - 14) / 384 +  
 4.271092364654717 \* (x - 6) \* (x - 10) \* (x - 12) \* (x - 14) / (-96) +  
 7.574113448116863 \* (x - 6) \* (x - 8) \* (x - 12) \* (x - 14) / 64 +  
 8.840312087141179 \* (x - 6) \* (x - 8) \* (x - 10) \* (x - 14) / (-96) +  
 12.79150332861507 \* (x - 6) \* (x - 8) \* (x - 10) \* (x - 12) / 384  
  
for (let i = 6; i <= 14; i += 0.25) {  
 console.log('' + i + ': ' + (f(i) - gottenF(i)))  
}  
  
//------------------- Сплайн інтерполяція --------------------  
  
console.log('\nСплайн інтерполяція')  
const A =[[8, 4, 0, 0],  
 [4, 8, 4, 0],  
 [0, 4, 8, 4],  
 [0, 0, 4, 8]]  
const b = [0, 0, 0, 0, 0]  
  
for (let i = 0; i < 4; i++) {  
 let y0 = i === 0 ? 0 : y[i - 1]  
 let y1 = y[i]  
 let y2 = y[i + 1]  
 b[i] = 3 \* (y0 + y2 - 2 \* y1)  
}  
const c = gaussMethod(A, b)  
for (let i = 0; i < 4; i++) {  
 const di = ((c[i + 1] || 0) - c[i]) / 2  
 const bi = (y[i + 1] - y[i]) / 2 - 2 / 3 \* di - c[i]  
 console.log('f' + (i+1))  
 console.log(`${y[i]} + ${bi}(x - ${x[i]}) + ${c[i]/2}(x - ${x[i]})^2 + ${di/6}(x - ${x[i]})^3`)  
}  
  
console.log('похибки:')  
const f1 = (x) => 1.4888697381153229  
 + 1.0677730911636791 \* (x - 6)  
 - 0.23607159812979056 \* (x - 6) \* (x - 6)  
 + 0.1988703545913998 \* (x - 6) \* (x - 6) \* (x - 6)  
const f2 = (x) => 4.271092364654717  
 + 1.3639298620805242 \* (x - 8)  
 + 0.9571505294186081 \* (x - 8) \* (x - 8)  
 + -0.4066800947966669 \* (x - 8) \* (x - 8) \* (x - 8)  
const f3 = (x) => 7.574113448116863  
 + 1.780405283567354 \* (x - 10)  
 - 1.4829300393613933 \* (x - 10) \* (x - 10)  
 + 0.45463852866689763 \* (x - 10) \* (x - 10) \* (x - 10)  
const f4 = (x) => 8.840312087141179  
 + 0.3157274438836226 \* (x - 12)  
 + 1.2449011326399921 \* (x - 12) \* (x - 12)  
 - 0.20748352210666535 \* (x - 12) \* (x - 12) \* (x - 12)  
  
for (let i = 6; i < 8; i += 0.25) {  
 console.log('' + i + ': ' + (f(i) - f1(i)))  
}  
for (let i = 8; i < 10; i += 0.25) {  
 console.log('' + i + ': ' + (f(i) - f2(i)))  
}  
for (let i = 10; i < 12; i += 0.25) {  
 console.log('' + i + ': ' + (f(i) - f3(i)))  
}  
for (let i = 12; i <= 14; i += 0.25) {  
 console.log('' + i + ': ' + (f(i) - f4(i)))  
}  
  
function gaussMethod(A, b) {  
 let y = [0, 0, 0, 0]  
 let x = [0, 0, 0, 0]  
 const t = [[0, 0, 0, 0],  
 [0, 0, 0, 0],  
 [0, 0, 0, 0],  
 [0, 0, 0, 0]]  
 const n = A.length - 1  
  
 for (let i = 0; i < A.length; i++) {  
 for (let j = i; j < A.length; j++) {  
 if (i === j) {  
 let kt = 0  
 for (let k = 0; k < i; k++) {  
 kt += t[k][i] \* t[k][i]  
 }  
 t[i][i] = Math.sqrt(A[i][i] - kt)  
 } else {  
 let kt = 0  
 for (let k = 0; k < i; k++)  
 kt += t[k][i] \* t[k][j]  
 t[i][j] = (A[i][j] - kt) / t[i][i]  
 }  
 }  
 }  
 y[0] = b[0] / t[0][0]  
 for (let i = 1; i < A.length; i++) {  
 let kt = 0  
 for (let k = 0; k < i; k++)  
 kt += t[k][i] \* y[k]  
 y[i] = (b[i] - kt) / t[i][i]  
 }  
 for (let i = n; i >= 0; i--) {  
 let kt = 0  
 for (let k = i + 1; k < A.length; k++)  
 kt += t[i][k] \* x[k]  
 x[i] = (y[i] - kt) / t[i][i]  
 }  
 return x  
}