Звіт

з лабораторної роботи №4

з курсу алгоритми і методи обчислень.

(НТЕРПОЛЯЦІЯ ДАНИХ. ІНТЕРПОЛЯЦІЙНИЙ ПОЛІНОМ ЛАГРАНЖА.

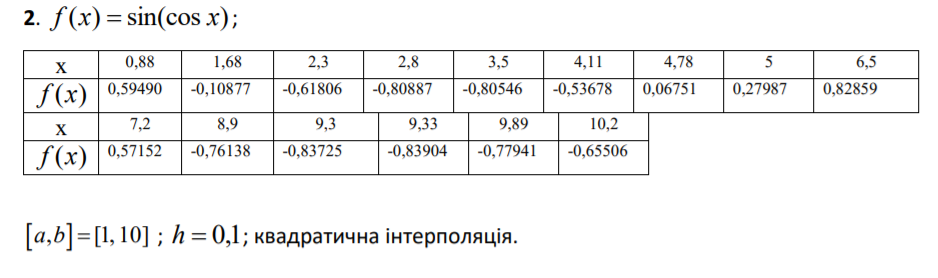
ЧИСЕЛЬНЕ ДИФЕРЕНЦІЮВАННЯ.)

Студента 1-го курсу,

Парфенюка Тимофія Павловича

2019

Завдання:



Код:

using System;

namespace Lab4\_Algoritmes

{

class Program

{

static double[] ArrX = { 0.88, 1.68, 2.3, 2.8, 3.5, 4.11, 4.78, 5, 6.5, 7.2, 8.9, 9.3, 9.33, 9.89, 10.2 };

static double[] ArrY = {0.59490, -0.10877, -0.61806, -0.80887, -0.80546, -0.53678, 0.06751, 0.27987,

0.82859, 0.57152, -0.76138, -0.83725, -0.83904, -0.77941, -0.65506 };

static double[] x\_Arr;

static double[] y\_Arr;

static void InitializeXArray(double a = 1, double b = 10, double h = 0.1)

{

int length = (int)((b - a) / h);

x\_Arr = new double[length];

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

{

x\_Arr[i] = a + h;

a += h;

}

}

static void InitializeYArray(double a = 1, double b = 10, double h = 0.1)

{

int length = (int)((b - a) / h);

y\_Arr = new double[length];

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

{

y\_Arr[i] = LocalInterpolation(x\_Arr[i]);

}

}

static double GlobalInterpolation(double x)

{

int num = 15;

double LFunc = 0;

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

{

double Lnum = 1, Lden = 1;

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

{

if (i != j)

{

Lnum \*= (x - ArrX[j]);

Lden \*= (ArrX[i] - ArrX[j]);

}

}

LFunc += (Lnum / Lden) \* ArrY[i];

}

return LFunc;

}

static int FindMinInArray(double x)

{

int i = 0;

int min = 0;

for (i = 0; i < ArrX.Length - 4; i++)

if ((Math.Abs(ArrX[min] - x) + Math.Abs(ArrX[min + 1] - x) + Math.Abs(ArrX[min + 2] - x) >

(Math.Abs(ArrX[i + 1] - x) + Math.Abs(ArrX[i + 2] - x) + Math.Abs(ArrX[i + 3] - x))))

min = i + 1;

return min;

}

static double LocalInterpolation(double x)

{

int min = FindMinInArray(x);

double Inter1 = (ArrY[min] \* (((x - ArrX[min + 1]) \* (x - ArrX[min + 2])) /

((ArrX[min] - ArrX[min + 1]) \* (ArrX[min] - ArrX[min + 2]))));

double Inter2 = (ArrY[min + 1] \* (((x - ArrX[min]) \* (x - ArrX[min + 2])) /

((ArrX[min + 1] - ArrX[min]) \* (ArrX[min + 1] - ArrX[min + 2]))));

double Inter3 = (ArrY[min + 2] \* (((x - ArrX[min]) \* (x - ArrX[min + 1])) /

((ArrX[min + 2] - ArrX[min]) \* (ArrX[min + 2] - ArrX[min + 1]))));

return Inter1 + Inter2 + Inter3;

}

static double LocalInterpolation\_v2(double x)

{

int min = FindMinInArray(x);

double L = 0;

for (int i = min; i <= min + 2; i++)

{

double Lnum = 1, Lden = 1;

for (int j = min; j <= min + 2; j++)

{

if (i != j)

{

Lnum \*= (x - ArrX[j]);

Lden \*= (ArrX[i] - ArrX[j]);

}

}

L += (Lnum / Lden) \* ArrY[i];

}

return L;

}

static double Function(double x)

{

return Math.Sin(Math.Cos(x));

}

static double FirstDer(double x)

{

return -Math.Sin(x) \* Math.Cos(Math.Cos(x));

}

static double SecondDer(double x)

{

return (-Math.Sin(Math.Cos(x)) \* (Math.Sin(x) \* Math.Sin(x))) - Math.Cos(Math.Cos(x)) \* Math.Cos(x);

}

static double Differentiation(double x)

{

double a = 1;

double b = 10;

double h = 0.1;

double dy = 0;

if (x == a)

{

dy = (LocalInterpolation(x + h) - LocalInterpolation(x)) / h;

}

else if (x == b)

{

dy = (LocalInterpolation(x) - LocalInterpolation(x - h)) / h;

}

else

{

dy = (LocalInterpolation(x + h) - LocalInterpolation(x - h)) / 2 \* h;

}

return dy;

}

static double SecondDifferentiation(double x)

{

double a = 1;

double b = 10;

double h = 0.1;

double ddy = 0;

if (x == a)

{

ddy = (LocalInterpolation(Differentiation(x + h)) - LocalInterpolation(Differentiation(x))) / h;

}

else if (x == b)

{

ddy = (LocalInterpolation(Differentiation(x)) - LocalInterpolation(Differentiation(x - h))) / h;

}

else

{

ddy = (LocalInterpolation(x + h) - LocalInterpolation(x - h) - 2 \* LocalInterpolation(x)) / 2 \* h;

}

return ddy;

}

static void InterpolationValuesPrint()

{

Console.WriteLine("X\t| Y Formuala\t| Y Global\t| Y Local\t|");

for (int i = 0; i < x\_Arr.Length; i++)

{

Console.WriteLine("{0:f1}\t| {1:f4}\t| {2:f4}\t| {3:f4}\t| ", x\_Arr[i], Function(x\_Arr[i]),

GlobalInterpolation(x\_Arr[i]), LocalInterpolation(x\_Arr[i]));

}

}

static void DifferentiationValuesPrint()

{

Console.WriteLine("X\t| Y' Formula\t| Y'' Formula\t| Y' \t\t|Y'' \t\t|");

for (int i = 0; i < x\_Arr.Length; i++)

{

Console.WriteLine("{0:f1}\t| {1:f4}\t| {2:f4}\t| {3:f4}\t| {4:f4}\t| ", x\_Arr[i], FirstDer(x\_Arr[i]),

SecondDer(x\_Arr[i]), Differentiation(x\_Arr[i]), SecondDifferentiation(x\_Arr[i]));

}

}

static void Main()

{

InitializeXArray();

InitializeYArray();

InterpolationValuesPrint();

Console.WriteLine();

DifferentiationValuesPrint();

}

}

}

***Висновок:***

Ця робта вчить застосуванню алгоритмів інтерполяції для побудови поліноміального наближення функції; вивчення методів числового диференціювання функції однієї змінної.

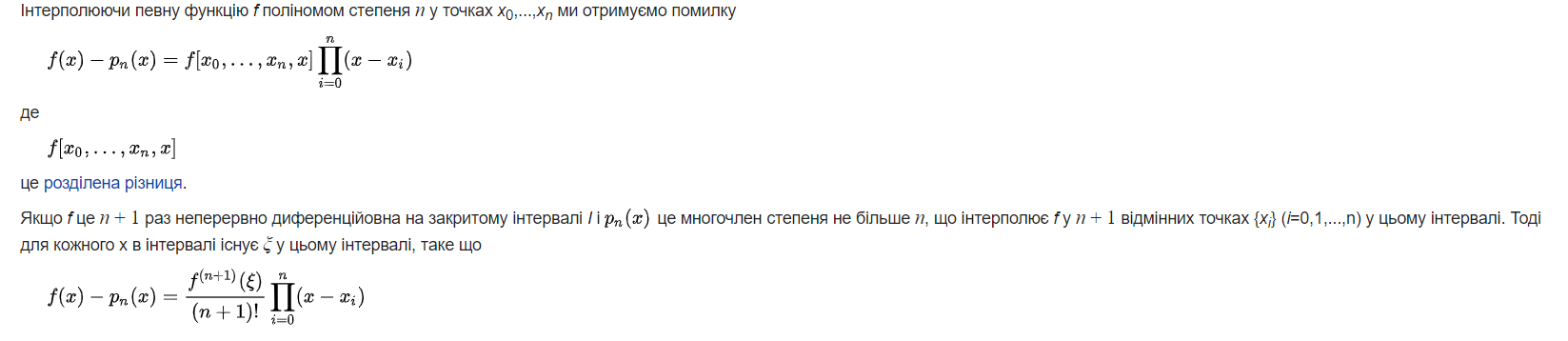
Для рызних статистичних моделей дуже важливо вміти інтерполювати функції. Тому цей алгоритм дуже важливий, особливо для матиматичної статистики.

1. **Що являє собою задача поліноміальної інтерполяції? Скільки розв’язків вона має?**

**Поліноміальна інтерполяція (Інтерполяція алгебраїчними многочленами) функції f(x) на відрізку [a, b]** - побудова многочлена Pn(x) степеня меншого або рівного n, що приймає у вузлах інтерполяції x0, x1, ..., xn значення f(xі):

Через фіксований набір точок можна провести нескінченне число кривих, тому задача інтерполяції не має єдиного розв'язку.

1. **Що таке похибка інтерполяції?**

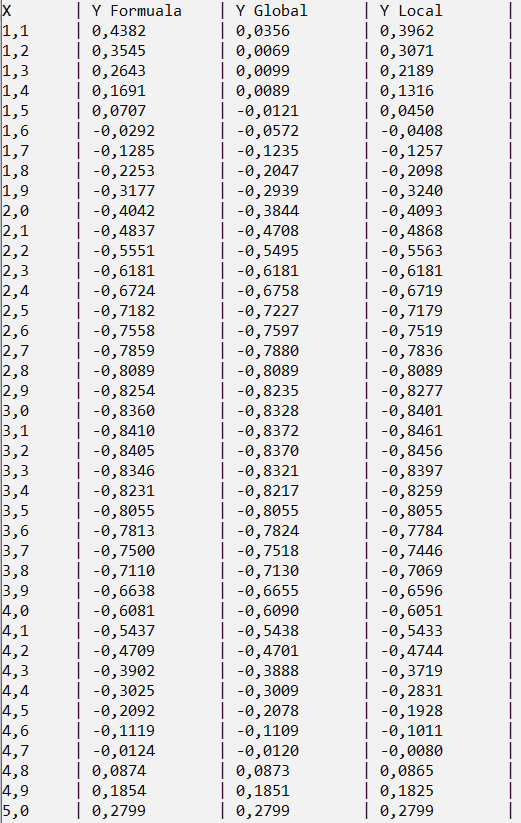


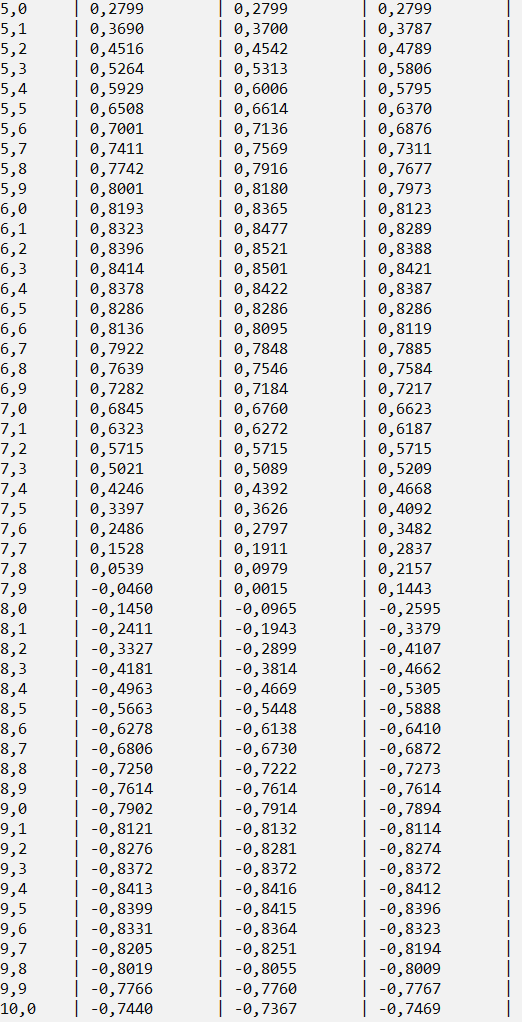
1. **Чим відрізняється глобальна інтерполяція від кускової?**

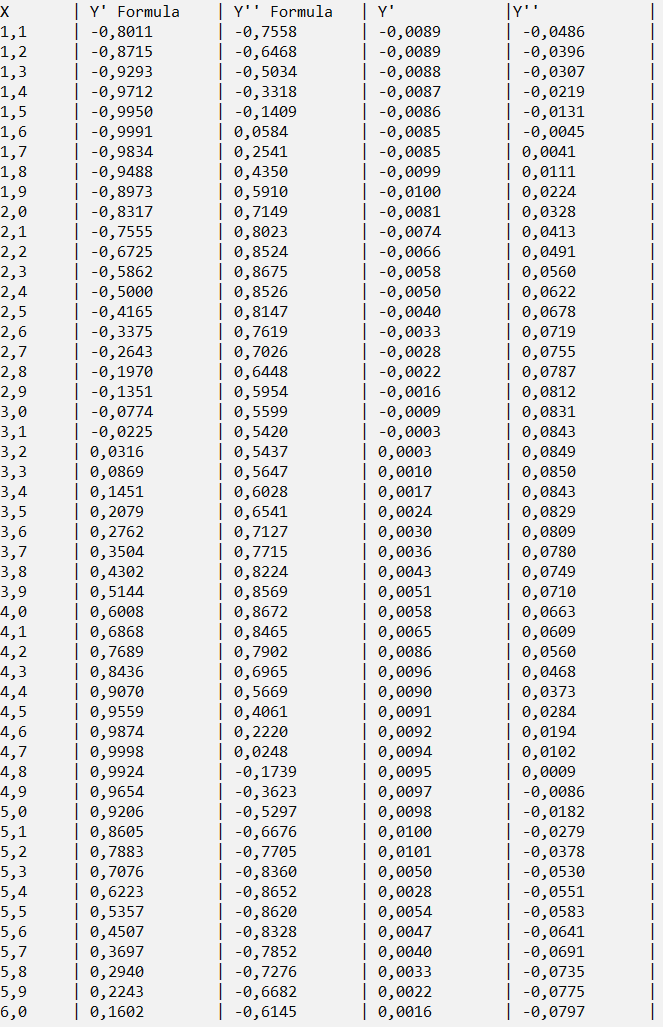
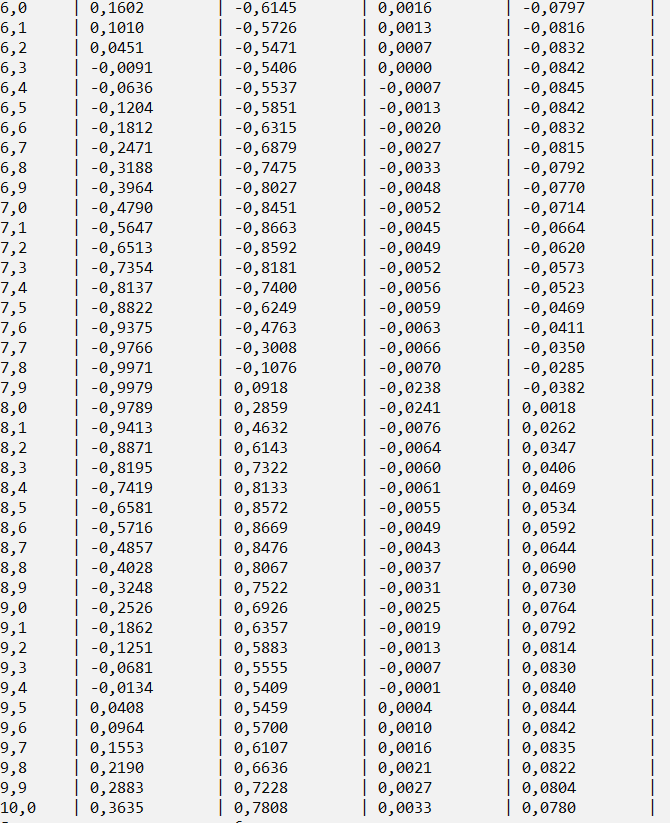
Глобальна апроксимує функцію по всьому проміжку апроксимації, а кускова, тільки по заданим, в залежності від заданої розмірності…

1. **Для чого використовується апроксимація функції при чисельному відшуканні її похідної?**Бо для знаходження похідної нам потрібне значення функції у заданій точці.
2. **Що називається порядком похибки формули чисельного диференціювання відносно кроку** **диференціювання?**

Показник степеня k у виразі похибки формул числового диференціювання називається порядком похибки апроксимації похідної відносно кроку h (або просто порядком апроксимації похідної). • При цьому припускається: h <1

***Вхідні та вихідні данні:***





Сплайн

Завдання: апроксимувати функцію кубічним сплайном.

Код:

using System;

namespace Splines2

{

class CubicSpline

{

SplineTuple[] splines;

struct SplineTuple

{

public double a, b, c, d, x;

}

public void BuildSpline(double[] x, double[] y)

{

int n = x.GetLength(0);

splines = new SplineTuple[n];

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

{

splines[i].x = x[i];

splines[i].a = y[i];

}

splines[0].c = splines[n - 1].c = 0.0;

double[] alpha = new double[n - 1];

double[] beta = new double[n - 1];

alpha[0] = beta[0] = 0.0;

for (int i = 1; i < n - 1; ++i)

{

double h\_i = x[i] - x[i - 1], h\_i1 = x[i + 1] - x[i];

double A = h\_i;

double C = 2.0 \* (h\_i + h\_i1);

double B = h\_i1;

double F = 3.0 \* ((y[i + 1] - y[i]) / h\_i1 - (y[i] - y[i - 1]) / h\_i);

double z = (A \* alpha[i - 1] + C);

alpha[i] = -B / z;

beta[i] = (F - A \* beta[i - 1]) / z;

}

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

splines[i].c = alpha[i] \* splines[i + 1].c + beta[i];

for (int i = n - 1; i > 0; --i)

{

double h\_i = x[i] - x[i - 1];

splines[i].d = (splines[i].c - splines[i - 1].c) / h\_i;

splines[i].b = h\_i \* (2.0 \* splines[i].c + splines[i - 1].c) / 6.0 + (y[i] - y[i - 1]) / h\_i;

}

}

public double Func(double x)

{

int n = splines.Length;

SplineTuple s;

if (x <= splines[0].x)

s = splines[1];

else if (x >= splines[n - 1].x)

s = splines[n - 1];

else

{

int i = 0, j = n - 1;

while (i + 1 < j)

{

int k = i + (j - i) / 2;

if (x <= splines[k].x)

j = k;

else

i = k;

}

s = splines[j];

}

double dx = (x - s.x);

return s.a + (s.b + (s.c / 2.0 + s.d \* dx / 6.0) \* dx) \* dx;

}

public double Der(double x)

{

int n = splines.Length;

SplineTuple s;

if (x <= splines[0].x)

s = splines[1];

else if (x >= splines[n - 1].x)

s = splines[n - 1];

else

{

int i = 0, j = n - 1;

while (i + 1 < j)

{

int k = i + (j - i) / 2;

if (x <= splines[k].x)

j = k;

else

i = k;

}

s = splines[j];

}

double dx = (x - s.x);

return s.b + s.c \* (dx) + s.d \* dx \* dx / 2; ;

}

public double SecondDer(double x)

{

int n = splines.Length;

SplineTuple s;

if (x <= splines[0].x)

return 0;

else if (x >= splines[n - 1].x)

return 0;

if (x == splines[0].x || (x == splines[n - 1].x)) return 0;

else

{

int i = 0, j = n - 1;

while (i + 1 < j)

{

int k = i + (j - i) / 2;

if (x <= splines[k].x)

j = k;

else

i = k;

}

s = splines[j];

}

double dx = (x - s.x);

return s.c + s.d \* dx;

}

}

class Program

{

static double[] x\_Arr;

static double[] y\_Arr;

static double Func1(double x)

{

return Math.Sin(Math.Cos(x));

}

static void CreatePointsArray(double a = 1, double b = 10, double h = 0.1)

{

int length = (int)((b - a) / h);

x\_Arr = new double[length];

y\_Arr = new double[length];

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

{

x\_Arr[i] = a + h;

y\_Arr[i] = Func1(x\_Arr[i]);

a += h;

}

}

static double[] ArrX = { 0.88, 1.68, 2.3, 2.8, 3.5, 4.11, 4.78, 5, 6.5, 7.2, 8.9, 9.3, 9.33, 9.89, 10.2 };

static double[] ArrY = {0.59490, -0.10877, -0.61806, -0.80887, -0.80546, -0.53678, 0.06751, 0.27987,

0.82859, 0.57152, -0.76138, -0.83725, -0.83904, -0.77941, -0.65506 };

static double L\_SecondDerivative(int i, double h = 0.01)

{

if (i == 0) return 0;

CubicSpline cb = new CubicSpline();

cb.BuildSpline(ArrX, ArrY);

return (cb.Func(x\_Arr[i]) + (cb.Func(x\_Arr[i-1])) - (2 \* (cb.Func(x\_Arr[i])))) / (h \* h);

}

static double L\_Derivative(int i, double h = 0.01)

{

if (i == 0||i == x\_Arr.Length -1) return 0;

CubicSpline cb = new CubicSpline();

cb.BuildSpline(ArrX, ArrY);

return ((cb.Func(x\_Arr[i+1])) - cb.Func(x\_Arr[i - 1]) )/ (h \* 2);

}

static void Main(string[] args)

{

CreatePointsArray();

CubicSpline cb = new CubicSpline();

cb.BuildSpline(ArrX, ArrY);

Console.WriteLine("X\t| Y Formuala\t| Y Spline\t| Y' Spline\t| Y'' Spline\t| ");

Console.WriteLine("{0:f1}\t| {1:f4}\t| {2:f4}\t| {3:f4}\t| {4:f5}\t|", x\_Arr[0], Func1(x\_Arr[0]),

cb.Func(x\_Arr[0]), cb.Der(x\_Arr[0]), 0);

for (int i = 0; i < x\_Arr.Length; i++)

{

Console.WriteLine("{0:f1}\t| {1:f4}\t| {2:f4}\t| {3:f4}\t| {4:f5}\t|", x\_Arr[i], Func1(x\_Arr[i]),

cb.Func(x\_Arr[i]), cb.Der(x\_Arr[i]), cb.SecondDer(x\_Arr[i]));

}

Console.WriteLine("{0:f1}\t| {1:f4}\t| {2:f4}\t| {3:f4}\t| {4:f5}\t|", x\_Arr[x\_Arr.Length - 1], Func1(x\_Arr[x\_Arr.Length - 1]),

cb.Func(x\_Arr[x\_Arr.Length - 1]), cb.Der(x\_Arr[x\_Arr.Length - 1]), 0);

}

}

}

***Вхідні та вихідні данні:***

