Міністерство освіти і науки, молоді та спорту України

Національний технічний університет України

«Київський політехнічний інститут»

# Інтерполяція та згладжування

Методичні вказівки до виконання лабораторних робіт з
дисципліни „Числові методи”
для студентів спеціальності „Автоматизоване управління
технологічними процесами”
напряму „Автоматизація та комп'ютерно-інтегровані технології”

*Рекомендовано Вченою радою інженерно-хімічного факультету*

Київ
НТУУ «КПІ»
2014

Інтерполяція та згладжування: Метод. вказівки до виконання лабораторних робіт з дисципліни „Числові методи” для студентів спеціальності „Автоматизоване управління технологічними процесами” напряму „Автоматизація та комп'ютерно-інтегровані технологічні комплекси” / Уклад.: А.І. Ситніков”, 2014. - 60 с.

*Гриф надано Вченою радою ІХФ*

*(Протокол № від 2014р.)*

Навчальне видання

ІНТЕРПОЛЯЦІЯ ТА ЗГЛАДЖУВАННЯ
Методичні вказівки до виконання лабораторних робіт з дисципліни „Числові методи” для студентів спеціальності „Автоматизоване управління технологічними процесами” напряму „Автоматизація та комп'ютерно-інтегровані технології”

Укладачі: Ситніков Олексій Володимирович

Відповідальний редактор А.І.Жученко, д-р техн.наук, проф.

Рецензент : В.І. Сівецький, к.т.н., проф.

Авторська редакція

**Зміст**

# Вступ

В першому розділі були розглянуті основи алгоритмічної мови програмування *Pascal*. Розглянуто текстовий та графічний режим.

Другий розділ присвячений розв'язку задач інтерполяції (згладжування), пошуку розв'язків системи рівнянь, вирішення задач інтерполяції за допомогою кубічних сплайнів та дослідження роботи з кубічними сплайнами в цілому, пошук комплексних коренів полінома.

Методичні вказівки доповнені текстами демонстраційно-відлагоджувальних програм та місцями під внесення результатів роботи.

Основною задачею даного лабораторного курсу є комп'ютерна реалізація математичних алгоритмів дослідження поліномів з формуванням відповідних графічних зображень.

Перша робота присвячена розв'язанню системи лінійних алгебраїчних рівнянь методом Гауса, кількість рівнянь в системі обмежена параметром Nmax.

Друга та третя робота виконуються за допомогою однієї програми *AprPol*, де в одному пункті виконується процес інтерполяції за допомогою інтерполяційного полінома, а в іншому – процес згладжування, використовуючи метод найменших квадратів.

Наступним кроком дослідження – є інтерполяція за допомогою кубічних сплайнів.

Кінцевим етапом дослідження є визначення коренів полінома шляхом сканування ділянки комплексної площини.

*Лабораторна робота №1*

Метод Гауса розв'язання систем лінійних алгебраїчних рівнянь

Мета роботи : Дослідити спосіб розв'язання систем лінійних алгебраїчних рівнянь методом Гауса.

Теоретичні відомості

Система рівнянь має вигляд

$$\begin{cases} a_{1,1}x_1 + a_{1,2}x_2 + a_{1,3}x_3 + \ldots + a_{1,n}x_n = a_{1,n+1}, \\ a_{2,1}x_1 + a_{2,2}x_2 + a_{2,3}x_3 + \ldots + a_{2,n}x_n = a_{2,n+1}, \\ a_{3,1}x_1 + a_{3,2}x_2 + a_{3,3}x_3 + \ldots + a_{3,n}x_n = a_{3,n+1}, \\ \ldots \\ a_{n,1}x_1 + a_{n,2}x_2 + a_{n,3}x_3 + \ldots + a_{n,n}x_n = a_{n,n+1}. \end{cases} \quad (1)$$

Тут $x_1, x_2, x_3, \ldots x_n$ – невідомі,

$a_{z,s}$, де $1 \le z \le n$, $1 \le s \le n+1$ – коефіцієнти.

Розширена матриця коефіцієнтів, що відповідає системі (1), – має вигляд

| | $s$=1 | 2 | 3 | . . . | $n$ | $n$+1 |
|---|---|---|---|---|---|---|
| $z$=1 | $a_{1,1}$ | $a_{1,2}$ | $a_{1,3}$ | . . . | $a_{1,n}$ | $a_{1,n+1}$ |
| 2 | $a_{2,1}$ | $a_{2,2}$ | $a_{2,3}$ | . . . | $a_{2,n}$ | $a_{2,n+1}$ |
| *A*:*Matr* 3 | $a_{3,1}$ | $a_{3,2}$ | $a_{3,3}$ | . . . | $a_{3,n}$ | $a_{3,n+1}$ |
| . . . | . . . | . . . | . . . | . . . | . . . | . . . |
| $n$ | $a_{n,1}$ | $a_{n,2}$ | $a_{n,3}$ | . . . | $a_{n,n}$ | $a_{n,n+1}$ |
| | $x_1$ | $x_2$ | $x_3$ | . . . | $x_n$ | праві частини |

Під матрицею показано невідомі, коефіцієнтами при яких є елементи відповідного стовпчика матриці *A*.

Розв'язок системи – масив *X*: *Coef*, що має таку структуру

| | -1 | 0 | 1 | 2 | 3 | . . . | $n$ | . . . | 31 |
|---|---|---|---|---|---|---|---|---|---|
| *X*:*Coef* | $n$ | – | $x_1$ | $x_2$ | $x_3$ | . . . | $x_n$ | | |

Отже, в комірку *x*[-1] заноситься кількість знайдених (і "складованих" в масиві *x*) значень невідомих. Алгоритм розв'язання системи (1), що розглядається в даній роботі, побудований таким чином, що він приводить до визначення усіх невідомих даної системи, або не визначає жодного з них. В такому випадку в *x*[-1] записується n (порядок системи, а отже, і кількість знайдених невідомих в ній), у другому в *x*[-1] записується нуль (це означає, що дана система не має розв'язку або, принаймні, даним алгоритмом цей розв'язок не знайдено).

Алгоритм Гауса (з вибором головного елемента) реалізується в підпрограмі, заголовок якої виглядає так

```
procedure SystUr (n: integer; A: Matr; var X:Coef);
        const  Nmax =15;
Тут     type Matr = array [1. . Nmax , 1. . Nmax+1] of real;
```

Константу *Nmax* користувач при потребі може задавати на свій розсуд.

```
        type Coef = array [- 1. .31] of real;
```

*n* – порядок системи.

*A* – масив коефіцієнтів системи, розширена (за рахунок правих частин) матриця коефіцієнтів.

*x* – масив значень розв'язку систем.

Схема розміщення знайдених значень невідомих показана вище.

У підпрограмі *SystUr* ще задається *const Eps* =1*e* – 9;

Це значення модуля коефіцієнта (в тому числі в процесі перетворень матриці A за схемою Гауса), яке розглядається як потенційно рівне нулю (з урахуванням імовірних похибок як його задання, так і обчислення в процесі перетворень матриці). Якщо в процесі виконання прямого ходу методу Гауса якийсь діагональний елемент виявляється за модулем меншим *Eps* (і немає можливості замінити його "кращим"), то дана система вважається несумісною, отже, така що не має розв'язку (в *x*[-1] заноситься нуль – і процес пошуку розв'язку припиняється).

В програмі *OSystUr* прийнято спочатку вводити так званий “контрольний розв'язок” (масив *Xk*: *Coef*). Масив же *A* коефіцієнтів вводиться без правих частин відповідних рівнянь системи. Коефіцієнти масиву *A* вводяться рядок за рядком (рівняння за рівнянням). У процесі введення коефіцієнтів чергового (*z* -го) рядку, коефіцієнт *A*[*z*,*s*] відразу множиться на *Xk*[*s*], сума таких добутків накопичується, і після введення *A*[*z*,*n*] (з домноженням його на *Xk*[*n*] ) записується в якості *A*[*z*,*n*+1] (правої частини *z*-го рівняння).

Після спроби розв’язання створеної таким чином системи рівнянь (розширеної матриці *A*) з’являється можливість співставити “контрольний розв’язок” (масив *Xk*) з отриманим процедурою *SystUr* розв’язком (масив *x*).

Хід роботи.

1. Завантажити середовище *Pascal*.
2. Набрати текст програми *OSystUr*.

```
Program OSystUr;
 uses Crt,Serv,Graph;
 const Nmax=10;
 type Matr=array[1..Nmax,1..Nmax+1] of real;
 var Xk,X:coef; A:Matr; z,s,n:integer;
    sum:real; T20:string[20];
procedure SystUr(n:integer; A:Matr; var x:Coef);
 const Eps=1e-9;
 var zmax,z,s,i,j:integer;
    r:real;
    b:boolean;
 begin
  b:=true;
  if n=1
    then
      if abs(A[1,1])<Eps then b:=false
                  else X[1]:=a[1,2]/a[1,1]
```

```
else
  begin
    i:=1;
    while (i<=n-1) and b do
      begin
        zMax:=i;
        for z:=i+1 to n do
          if abs(A[z,i])>abs(A[zMax,i]) then zMax:=z;
        if abs(A[zMax,i])<Eps
          then b:=false
          else
            begin
              if zMax>i then
                for j:=i to n+1 do
                  begin
                    r:=a[i,j]; a[i,j]:=a[zMax,j];
                    a[zMax,j]:=r
                  end;
              for z:=i+1 to n do
                if abs(a[z,i])>Eps then
                  begin
                    r:=a[z,i]/a[i,i];
                    for s:=i+1 to n+1 do
                      a[z,s]:=a[z,s]-r*a[i,s]
                  end
            end;
          inc(i)
      end;
    if b then
      if abs(a[n,n])<Eps then b:=false;
```

```
        if b then
          begin
            x[n]:=a[n,n+1]/a[n,n];
            for i:=n-1 downto 1 do
              begin
                r:=0;
                for s:=i+1 to n do
                  r:=r+a[i,s]*x[s];
                x[i]:=(a[i,n+1]-r)/a[i,i]
              end
          end
    end;
  if b then x[-1]:=n else x[-1]:=0
end;
 begin
  A[1,1]:=5; A[1,2]:=2; n:=1;
  repeat
    PutA; Ou('Esc-exit,1-n,2-inpXk,3-inpA,4-outX');
    str(n,t10); Ts:='n='+t10;
    Info; j:=readkey;
    case j of
     '1': Oui('n=',n);
     '2': for s:=1 to n do
          begin
            str(s,t10);
            Our('Xk['+t10+']', Xk[s]);
          end;
     '3': for z:=1 to n do
          begin
            Sum:=0;Str(z,t10);
```

```
            for S:=1 to n do
              begin
                Str(S,Ts);
                Our('A['+t10+','+Ts+']',A[z,S]);
                Sum:=Sum+A[z,S]*Xk[s]
              end;
            A[z,n+1]:=Sum
          end;
    '4': begin
          SystUr(n,A,X); clear(0,15,getmaxx-1,getmaxy-15);
          if X[-1]=0
            then
              OutTextXY(50,20,'Solution is not found')
            else
              for S:=1 to n do
                begin
                  Str(S,t10);Str(Xk[S],Ts);Ts:='Xk['+t10+']='+Ts;
                  Str(X[S],t20);Ts:=Ts+'   '+'X['+t10+']='+t20;
                  OutTextXY(50,(S+2)*10,Ts)
                end
        end{'4'}
   end{case}
  until j=#27;
  CloseGraph
end.
```

3. Записати відповідні результати роботи

а) Контрольний вектор значень невідомих.

б) Ліві частини системи рівнянь.

в) Отриманий вектор розв’язку.

Звіт повинен містити назву роботи, мету, короткі теоретичні відомості, заповнений п.3

Контрольні запитання

1. В чому суть метода Гауса.
2. Що відбувається :

```
begin
    r:=a[z,i]/a[i,i];
    for s:=i+1 to n+1 do
      a[z,s]:=a[z,s]-r*a[i,s]
 end;
```

3. Що відбувається :

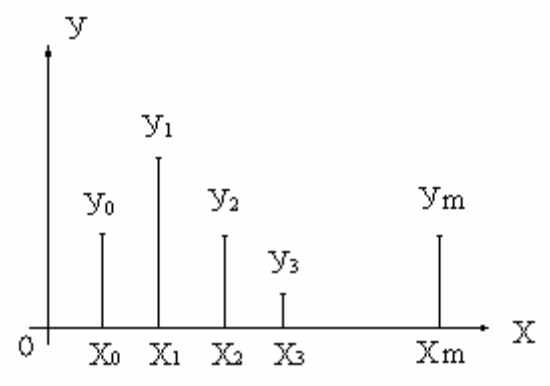
```
begin
    r:=a[i,j]; a[i,j]:=a[zMax,j];
    a[zMax,j]:=r
end;
```

*Лабораторна робота № 2*

Інтерполяційний поліном (у канонічній формі)

Мета роботи: Навчитись застосовувати інтерполяційний поліном, використовуючи масив коефіцієнтів.

Теоретичні відомості




Розглядається випадок, коли деяка функціональна залежність $y = f(x)$ задається рядом точок на площині $X0Y$ (нумерація точок загальному випадку може бути довільною), тобто маємо множину пар значень $Xs$, $Ys$ для $0 \le S \le m$.

Задача інтерполяції буде полягати в тому, щоб знайти якийсь спосіб (алгоритм) знаходження значень функції у проміжках між вузлами (вузли-це набір значень $Xs$, $0 \le S \le$ m). Мається на увазі ,що “справжня” функція $f(x)$ або невідома або ж з якихось міркувань незручна для використання дослідником.

В принципі, способів розв’язання задачі інтерполяції можна запропонувати безліч. Як правило, за умовчанням, приймається, що апроксимуюча функція має бути неперервною, бажано з неперервними нижчими похідними (чим більше-тим краще) і такою, щоб нею було просто і зручно користуватись. Значення апроксимуючої функції у вузлах інтерполяції, зрозуміло, має співпадати з відповідним значенням $Ys$, $0 \le S \le m$. Одним з найпопулярніших інструментів для розв’язання задач наближення (апроксимації) функції є поліном виду:

$$A(x) = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n.$$

Степінь полінома n та значення його коєфіціентів $a_0$, $a_1$, $a_2$,…, $a_n$ підбирається (визначається) дослідником так ,щоб задовольнити необхідні умови. В розглянутому випадку треба забезпечити проходження графіка полінома через задані точки ($Xs$,$Ys$) , $0 \le S \le m$ на площині $X0Y$ (значення

полінома у вузлах інтерполяції мають співпадати з відповідним значення *Ys* , $0 \le S \le m$) тобто:

$$\begin{cases} a_0 + a_1x_0 + a_2x_0^2 + \ldots + a_nx_0^n = y_0, \\ a_0 + a_1x_1 + a_2x_1^2 + \ldots + a_nx_1^n = y_1, \\ a_0 + a_1x_2 + a_2x_2^2 + \ldots + a_nx_2^n = y_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_0 + a_1x_m + a_2x_m^2 + \ldots + a_nx_m^n = y_m, \end{cases}$$

Виписана система умов може розглядатись як система рівнянь для визначення невідомих коефіцієнтів полінома $a_s$, $0 \le S \le m$. Щоб ця система мала єдиний розв'язок необхідно, щоб кількість рівнянь дорівнювала кількості невідомих в них, тобто щоб $m+1=n+1$ або ,що рівноцінно $n = m$.

Отже , степінь інтерполяційного полінома має (принаймні,формально) дорівнювати m. При n = m система рівнянь (лінійних-відносно невідомих коефіцієнтів-алгебраїчних) матиме розширену матрицю коефіцієнтів

| | s=1 | 2 | 3 | … | n+1 | n+2 |
|---|---|---|---|---|---|---|
| z=1 | 1 | X0 | $X0^2$ | … | $X0^n$ | Y0 |
| 2 | 1 | X1 | $X1^2$ | … | $X1^n$ | Y1 |
| 3 | 1 | X2 | $X2^2$ | … | $X2^n$ | Y2 |
| … | | | | | | |
| n+1 | 1 | Xn | $Xn^2$ | … | $Xn^n$ | Yn |
| | a0 | a1 | a2 | … | an | праві частини |

Під матрицею A показано значення невідомих, коефіцієнтами при яких є елементи матриці у відповідному стовпчику.

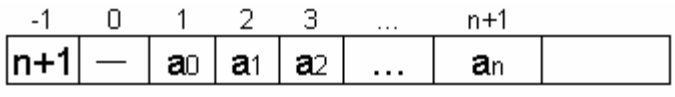
Тип матриці *A* можна задати так

```
const Nmax = 10;
type Matr = [1..Nmax+1, 1..Nmax+2] of real ;
```

Тут *Nmax* – максимальний степінь апроксимуючого полінома.

Формування матриці *A* і розв'язання відповідної системи алгебраїчних рівнянь реалізується у підпрограмі *IntPol*. Звертаємо увагу читача на ту обставину, що процедура *SystUr* повертає масив *A*:*Coef*

*A*:*Coef*




Щоб привести отриманий масив *A* до форми, прийнятої нами для розміщення коефіцієнтів полінома, треба виконати зсув елементів масиву на 1 позицію вліво і замінити вміст комірки *A*[-1] на n-степінь інтерполяційного полінома, що і робиться в кінці підпрограми *IntPol*.

В програмі AprPol для перевірки коректності розв'язання задач інтерполяції, а також для спрощення введення вузлів інтерполяції та відповідних їм значень функції використовується такий прийом (як один із можливих).

Вводиться "контрольний поліном" *Ak*: *Coef*, задається діапазон значень $Xn \le X \le Xk$, задається *m*. Даний діапазон розбивається m кроків і для кожного з кроків визначається відповідне *Xs* та *Ys* (останнє як значеня полінома *Ak*(*Xs*)). Сформовані таким чином масиви *X* та *Y* передаються в підпрограму *IntPol*, в результаті чого формується інтерполяційний поліном *A*(*x*). Користувач може співставити між собою як поліноми *Ak*(*x*) та *A*(*x*), так і їх графіки. На ці графіки накладаються зображені маленькими кружечками точки на площині *X*0*Y* які, з одного боку, є породження полінома *Ak*(*x*), а з іншого – через них має проходити графік полінома *A*(*x*).

Хід роботи.

1. Завантажити середовище *Pascal* .
2. Набрати текст програми *AprPol*.

```
Program AprPol;
  Uses Crt, Graph, Serv, Groms;
  Const n: integer=3;
        Nvr: integer=1;
        Nmax: integer=10;
        Alpha: real=1;
```

```
      Beta: real=2;
      K: real=1;
      M: integer=5;
  Type Matr=array[1..10, 1..11] of real;
  Var Mx, My: CoefR;
     Ak, A: Coef;
     Mo: CoefL;
     e:real;
{$i systur.pas}
procedure intpol(x,y:coefr;var a:coef);
  var z,s,n:integer;
  d:matr;
    begin
      n:=round(x[-1]);
      for z:=1 to n+1 do
        begin
          d[z,1]:=1; d[z,n+2]:=y[z-1];
          for s:=2 to n+1 do
          d[z,s]:=d[z,s-1]*x[z-1]
        end;
      systur (n+1,D,A);
      if A[-1]>0 then
        begin
         a[-1]:=n;
         for s:=0 to n do
         a[s]:=a[s+1]
        end;
    end;
  Function HorReal(a: Coef; x: Real): real;
    var n, s: integer;
```

```
    r: real;
  Begin
   n:= Round(A[-1]);    R := A[n];
   For s := n-1 downto 0 do
    R := R * x + A[s];
   HorReal := R;
  End;
Procedure InpPol(Id: string; var A: Coef);
   Var s, n: Integer;
   Begin
    Oui('n-step pol',n);
    A[-1] := n;
    For s := 0 to n do
     Begin
      Str(s, T10);     Our(Id+'['+T10+']',A[s])
     end
   End;
Procedure OutPol(Id: String; A: Coef; X: Integer);
  Var s: Integer;
  Begin
   For s := 0 to Round(A[-1]) do
    Begin
     Str(s, T10);
     Str(A[s], Ts);
     OutTextXY(x, (s + 2) * 10, Id + '[' + T10 + ']=' + Ts)
    end
  End;
Procedure FormMoPol(A: Coef);
  Var s: integer;
    Dx: Real;
```

```
  Begin
    Mo[-1] := L;    Dx := (Xk - Xn)/L;
    Mo[L+1] := Dx;
    For s := 0 to L do
      Mo[s] := HorReal(A, Xn + S * Dx)
  End;
Procedure FormPoints;
  Var s: integer;
      Dx1: real;
  Begin
    Dx1 := (Xk - Xn)/m;    Mx[-1] := m;
    My[-1] := m;
    Xmin := Xn;            Xmax := Xk;
    Ymin := 0;             Ymax := 0;
    For s := 0 to m do
      Begin
        X := Xn + s * Dx1;
        Mx[s] := X;
        Case Nvr of
          1: Y := HorReal(ak, x);
          2: Y := Sin(Alpha * x) + k * Cos(Beta * x)
        end;
        My[s] := y;
        if y < Ymin then Ymin := y;
        if y > Ymax then Ymax := y
      End;
    X0Y0(false);
    SetColor(15);
    ClearDevice;
    SystCoor;
```

```
    For s := 0 to m do
      Circle(x0 + Round(Mx[s]/Dx), y0 - Round(My[s]/Dy), 3)
  End;
Procedure Input;
   Begin
     repeat
       PutA;     Ou('0-exit,1-Ak,2-Alpha,3-Beta,4-K,5-Xn,6-Xk,7-M,8-Nvr');
       Str(Alpha:1:2, T10);     Ts := 'Alpha=' + T10;
       Str(Beta:1:2, T10);      Ts := Ts + ',Beta=' + T10;
       Str(k:1:2, T10);         Ts := Ts + ',K=' + T10;
       Str(Xn:1:2, T10);        Ts := Ts + ',Xn=' + T10;
       Str(Xk:1:3, T10);        Ts := Ts + ',Xk=' + T10;
       Str(m, T10);             Ts := Ts + ',M=' +T10;
       Str(Nvr, T10);           Ts := Ts + ',Nvr=' + T10;
       Info;           J1 := ReadKey;
       Case J1 of
         '1': InpPol('Ak', Ak);
         '2': Our('Alpha', Alpha);
         '3': Our('Beta', Beta);
         '4': Our('K', k);
         '5': Our('Xn', Xn);
         '6': Our('Xk', Xk);
         '7': Oui('M', m);
         '8': Oui('Nvr', Nvr);
       end;
     Until J1='0'
   End;
Begin
  Xn := -2;        Xk := 3;
  Ak[-1] := 2;     Ak[0] := 1;
```

```
Ak[1] := 4;      Ak[2] := 20;
L:=300; H:=240;
Repeat
  PutA;
  Ou('Esc-exit,1-Input,2-Points,3-C,4-Grf,5-GrInt,6-A(x),7-Serv');
  Str(c, T10);        Ts := 'C=' + T10;
  Str(n, T10);        Ts := Ts + ', n=' +T10;
  Info;  J:=ReadKey;
  case J of
    '1': Input;
    '2': FormPoints;
    '3': Oui('C', c);
    '4': Begin
           Case Nvr of
            1: FormMoPol(Ak);
            2: Begin
                 Mo[-1] := L;      {Dx1 := (Xk - Xn)/L;}
                 For s := 0 to L do
                   Begin
                     X := Xn + s * dx;
                     Mo[s] := sin(Alpha * x) + k * cos(Beta * x)
                   end
               End
           End;
           Graphic(Mo, c)
         End;
    '5': Begin
           IntPol(Mx, My, a);
           FormMoPol(a);
           Graphic(Mo, c)
```

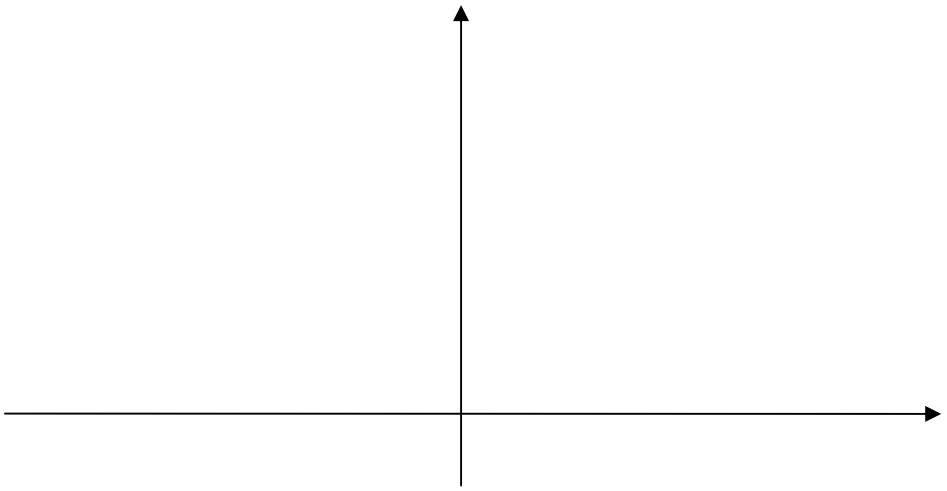
```
            End;
        '6': Begin
              Clear(0, 20, GetMaxX - 1, GetMaxY - 11);
              If Nvr=1 then
                OutPol('Ak', Ak, 20);
              OutPol('A', a, 320);
            End;
         '7':Service
      end
    Until j=#27;
    CloseGraph
  End.
```

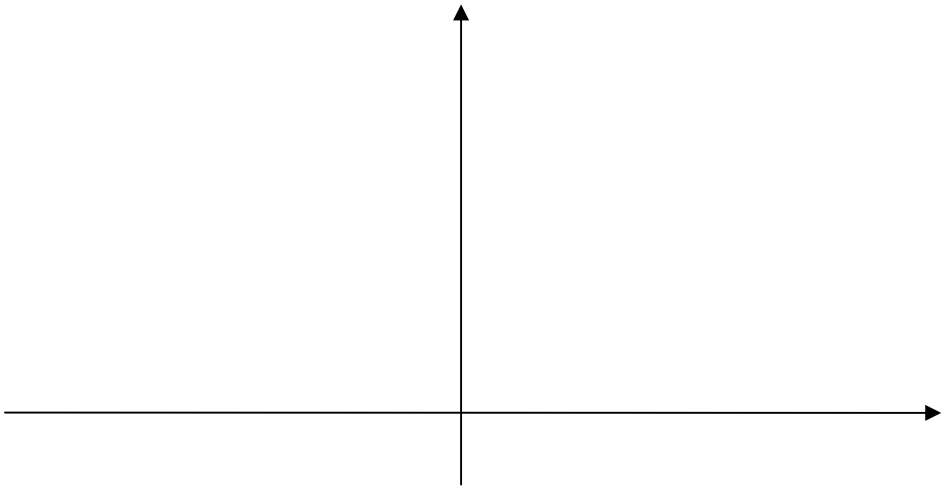
3.Результати роботи

а) Контрольний поліном

б) Інтерполяційний поліном

в) Графіки





Звіт повинен містити назву роботи, мету, короткі теоретичні відомості, заповнений п.3

Контрольні запитання

1. В чому суть інтерполяційного полінома.
2. Механізм роботи процедури *Intpol*.
3. Механізм роботи функції *HorReal*.
4. Механізм роботи процедури *InpPol*.
5. Механізм роботи процедури *OutPol*.
6. Механізм роботи процедури *FormMoPol*.
7. Механізм роботи процедури *FormPoints*.

*Лабораторна робота №3*

Згладжуючий поліном

Мета роботи : Дослідити апроксимацію полінома методом найменших квадратів.

Теоретичні відомості

Точки $(x_i, y_i)$, $0 \le i \le m$, що задають досліджувану функцію $y=f(x)$, можуть бути відомі з похибками, або ж апроксимуючу функцію бажано було б мати по можливості більш простою ніж інтерполяційна. В такому випадку замість умови проходження графіка апроксимуючої функції через усі задані точки ставиться більш "м'яка" умова, а саме, щоб показник якості апроксимації

$$E = \sum_{i=0}^{m} \left( a_0 + a_1 x^1 + a_2 x_i^{\,2} + \ldots + a_n x_i^{\,n} - y_i \right)^2 \to \min.$$

При такій постановці задачі $n$ і $m$ можуть задаватись довільно. Чим більше n тим, можна сподіватись, буде кращою якість апроксимації (менше значення $E$). Чим менше n – тим більш згладженою буде апроксимуюча функція (при $n$=0 – це буде константа, при $n$=1 – пряма лінія, при $n$=2 – квадратна парабола і т.д. ). Умови мінімізації $E$

$$\partial E/\partial a_r = 0, \qquad 0 \le r \le n$$

або в розгорнутому вигляді

$$2\sum_{\mathrm{i}=0}^{\mathrm{m}} (a_0 + a_1 x_i^1 + a_2 x_i^2 + \ldots + a_{\mathrm{n}} x_i^n - \mathrm{y_i}) x_i^r = 0, \;\; 0 \le \mathrm{r} \le \mathrm{n}.$$

Якщо в останній формулі виконати скорочення на 2, то розширена матриця коефіцієнтів отримана в системі рівнянь набуває вигляду ($z=r+1$)

| | *s* =1 | 2 | 3 | … | *n*+1 | *n*+2 |
|---|---|---|---|---|---|---|
| *z*=1 | $\sum_{i=0}^{m} x_i^0$ | $\sum_{i=0}^{m} x_i^1$ | $\sum_{i=0}^{m} x_i^2$ | … | $\sum_{i=0}^{m} x_i^n$ | $\sum_{i=0}^{m} y_i x_i^0$ |
| 2 | $\sum_{i=0}^{m} x_i^1$ | $\sum_{i=0}^{m} x_i^2$ | $\sum_{i=0}^{m} x_i^3$ | … | $\sum_{i=0}^{m} x_i^{n+1}$ | $\sum_{i=0}^{m} y_i x_i^1$ |
| *D* | $\sum_{i=0}^{m} x_i^2$ | $\sum_{i=0}^{m} x_i^3$ | $\sum_{i=0}^{m} x_i^n$ | … | $\sum_{i=0}^{m} x_i^{n+2}$ | $\sum_{i=0}^{m} y_i x_i^2$ |
| 3 | … | | | | | … |
| *n*+1 | $\sum_{i=0}^{m} x_i^n$ | $\sum_{i=0}^{m} x_i^{n+1}$ | $\sum_{i=0}^{m} x_i^{n+2}$ | … | $\sum_{i=0}^{m} x_i^{2n}$ | $\sum_{i=0}^{m} y_i x_i^n$ |
| | $a_0$ | $a_1$ | $a_2$ | … | $a_n$ | праві частини |

Аналіз структури матриці *D* показує, що при її формуванні, доцільно спочатку формувати масив *B, C: Coefr* такої структури

| | -1 | 0 | 1 | 2 | 3 | … | *m* | |
|---|---|---|---|---|---|---|---|---|
| *B* | | $X_0^R$ | $X_1^R$ | $X_2^R$ | $X_3^R$ | … | $X_m^R$ | |

де *R* послідовно приймає значення 0, 1, 2, … , 2*n*

| | -1 | 0 | 1 | 2 | 3 | … | 2*n* | |
|---|---|---|---|---|---|---|---|---|
| *C* | | $\sum_{i=0}^{m} x_i^0$ | $\sum_{i=0}^{m} x_i^1$ | $\sum_{i=0}^{m} x_i^2$ | $\sum_{i=0}^{m} x_i^3$ | … | $\sum_{i=0}^{m} x_i^{2n}$ | |

Формування масивів *B, C: Coefr*, з наступним “рознесенням” масиву *C* по комірках масиву *D*, розв’язання системи рівнянь, що відповідає *D*, приведення масиву коефіцієнтів згладжуючого полінома до стандартного вигляду – процедура *NaimKv*, заголовок якої має вигляд

```
procedure   NaimKv (X,Y: Coefr; n:integer; var
                    E: real; var A: Coef );
```

Тут *X,Y*: *Coefr* – масиви значень *x* та y для заданих точок згладжуваної функції. *n* – заданий (бажаний) степінь апроксимуючого (згладжуючого) полінома. *E* – значення показника якості апроксимації, що відповідає заданому *n*. *A*: *Coef* – масив з інформацією про згладжуючий поліном (формально – n-го степеня).

Для перевірки коректності розгляданого алгоритму в програмі *AprPol* (що використовується для формування як інтерполяційних так і згладжуючих поліномів) передбачено варіант, коли точки на площині *X*0*Y*, що служать базою для формування апроксимуючих поліномів, задаються як такі, що відповідають "контрольному поліному" *Ak*, і тоді апроксимуючий поліном може порівнюватись з "контрольним" як за коефіцієнтами, так і за графіками. Очевидно, що коли степінь n апроксимуючого полінома *A*(*x*) дорівнює або перевищує степінь *nk* "контрольного" полінома *Ak*(*x*), то коефіцієнти з індексами більшими *nk* у апроксимуючому поліномі мають бути рівними нулю (теоретично), а згладжуючий поліном перетворюється на інтерполяційний і цей останній має фактично співпадати з контрольним.

Хід роботи

1. Завантажити середовище *Pascal*.
2. Набрати текст програми *AprPol*.

```
Program AprPol;
  Uses Crt, Graph, Serv, Groms;
  Const n: integer=3;
        Nvr: integer=1;
        Nmax: integer=10;
        Alpha: real=1;
        Beta: real=2;
        K: real=1;
        M: integer=5;
  Type Matr=array[1..10, 1..11] of real;
  Var Mx, My: CoefR;
```

```
    Ak, A: Coef;
    Mo: CoefL;
    e:real;
Procedure SystUr(n:integer; A:Matr; var x:Coef);
 const Eps=1e-9;
 var zmax,z,s,i,j:integer;
    r:real; b:boolean;
 begin
  b:=true;
  if n=1
    then
     if abs(A[1,1])<Eps
       then b:=false else X[1]:=a[1,2]/a[1,1]
       else
         begin
           i:=1;
           while (i<=n-1) and b do
             begin
               zmax:=i;
               for z:=i+1 to n do
                 if abs(A[z,i])>abs(A[zmax,i])
                    then zmax:=z;
               if abs(A[zmax,i])<Eps
                 then b:=false
                 else
                   begin
                     if zmax>i then
                      for j:=i to n+1 do
                       begin
                        r:=a[i,j];
```

```
                    a[i,j]:=a[zmax,j];
                    a[zmax,j]:=r
                   end;
                  for z:=i+1 to n do
                   if abs(a[z,i])>Eps then
                    begin
                     r:=a[z,i]/a[i,i];
                     for s:=i+1 to n+1 do
                        a[z,s]:=a[z,s]-r*a[i,s]
                    end
                end;
              inc(i)
            end;
          if b then
            if abs(a[n,n])<Eps then b:=false;
          if b then
            begin
              x[n]:=a[n,n+1]/a[n,n];
              for i:=n-1 downto 1 do
                begin
                  r:=0;
                  for s:=i+1 to n do
                    r:=r+a[i,s]*x[s];
                  x[i]:=(a[i,n+1]-r)/a[i,i]
                end
            end
       end;
     if b then x[-1]:=n else x[-1]:=0
  end;
procedure intpol(x,y:coefr;var a:coef);
```

```
var z,s,n:integer;
d:matr;
  begin
    n:=round(x[-1]);
    for z:=1 to n+1 do
      begin
        d[z,1]:=1; d[z,n+2]:=y[z-1];
        for s:=2 to n+1 do
        d[z,s]:=d[z,s-1]*x[z-1]
      end;
    systur (n+1,D,A);
    if A[-1]>0 then
      begin
       a[-1]:=n;
       for s:=0 to n do
       a[s]:=a[s+1]
      end;
  end;
Function HorReal(a: Coef; x: Real): real;
  var n, s: integer;
      r: real;
  Begin
    n:= Round(A[-1]);    R := A[n];
    For s := n-1 downto 0 do
      R := R * x + A[s];
    HorReal := R;
  End;
procedure NaimKV(X,Y:Coefr; n:integer; var E:real; var A:Coef);
 var m,z,s: integer;
    B,C: Coefr;
```

```
    D: Matr;
    Sum: real;
begin
 m:=round(X[-1]);
 for s:= 0 to m do
  B[s]:=1;{if X[s]=0,then B[s]:=0}
  for z:=0 to 2*n do
   begin
    C[z]:=0; Sum:=0;
    for s:=0 to m do
     begin
      C[z]:=C[z]+B[s];
      if z<= n then
      Sum:= Sum+B[s]*Y[s];
      B[s]:=B[s]*X[s]
     end;
    if z<= n then D[z+1,n+2]:=Sum
   end;
  for z:=1 to n+1 do
   for s:=1 to n+1 do
    D[z,s]:=C[z+s-2];
  Systur(n+1,D,A);
  A[-1]:=n;E:=0;
  for s:=0 to n do
  A[s]:=A[s+1];
  for s:=0 to L do
  {E:=E+sqr(horreal(A,X[s])-Y[s])}
end;
Procedure InpPol(Id: string; var A: Coef);
   Var s, n: Integer;
```

```
  Begin
    Oui('n-step pol',n);
    A[-1] := n;
    For s := 0 to n do
      Begin
        Str(s, T10);     Our(Id+'['+T10+']',A[s])
      end
  End;
Procedure OutPol(Id: String; A: Coef; X: Integer);
  Var s: Integer;
  Begin
    For s := 0 to Round(A[-1]) do
      Begin
        Str(s, T10);
        Str(A[s], Ts);
        OutTextXY(x, (s + 2) * 10, Id + '[' + T10 + ']=' + Ts)
      end
  End;
Procedure FormMoPol(A: Coef);
  Var s: integer;
      Dx: Real;
  Begin
    Mo[-1] := L;    Dx := (Xk - Xn)/L;
    Mo[L+1] := Dx;
    For s := 0 to L do
      Mo[s] := HorReal(A, Xn + S * Dx)
  End;
Procedure FormPoints;
  Var s: integer;
      Dx1: real;
```

```
  Begin
    Dx1 := (Xk - Xn)/m;    Mx[-1] := m;
    My[-1] := m;
    Xmin := Xn;            Xmax := Xk;
    Ymin := 0;             Ymax := 0;
    For s := 0 to m do
     Begin
       X := Xn + s * Dx1;
       Mx[s] := X;
       Case Nvr of
         1: Y := HorReal(ak, x);
         2: Y := Sin(Alpha * x) + k * Cos(Beta * x)
       end;
       My[s] := y;
       if y < Ymin then Ymin := y;
       if y > Ymax then Ymax := y
     End;
    X0Y0(false);
    SetColor(15);
    ClearDevice;
    SystCoor;
    For s := 0 to m do
      Circle(x0 + Round(Mx[s]/Dx), y0 - Round(My[s]/Dy), 3)
  End;
Procedure Input;
   Begin
     repeat
       PutA;     Ou('0-exit,1-Ak,2-Alpha,3-Beta,4-K,5-Xn,6-Xk,7-M,8-Nvr');
       Str(Alpha:1:2, T10);     Ts := 'Alpha=' + T10;
       Str(Beta:1:2, T10);      Ts := Ts + ',Beta=' + T10;
```

```
      Str(k:1:2, T10);        Ts := Ts + ',K=' + T10;
      Str(Xn:1:2, T10);       Ts := Ts + ',Xn=' + T10;
      Str(Xk:1:3, T10);       Ts := Ts + ',Xk=' + T10;
      Str(m, T10);            Ts := Ts + ',M=' +T10;
      Str(Nvr, T10);          Ts := Ts + ',Nvr=' + T10;
      Info;          J1 := ReadKey;
      Case J1 of
        '1': InpPol('Ak', Ak);
        '2': Our('Alpha', Alpha);
        '3': Our('Beta', Beta);
        '4': Our('K', k);
        '5': Our('Xn', Xn);
        '6': Our('Xk', Xk);
        '7': Oui('M', m);
        '8': Oui('Nvr', Nvr);
      end;
    Until J1='0'
  End;
Begin
  Xn := -2;       Xk := 3;
  Ak[-1] := 2;    Ak[0] := 1;
  Ak[1] := 4;     Ak[2] := 20;
  L:=300; H:=240;
  Repeat
    PutA;
    Ou('Esc-exit,1-Input,2-Points,3-C,4-Grf,5-GrInt,6-A(x),7-N,8-GrMNK,9-
Serv');
    Str(c, T10);       Ts := 'C=' + T10;
    Str(n, T10);       Ts := Ts + ', n=' +T10;
    Info;  J:=ReadKey;
```

```
case J of
  '1': Input;
  '2': FormPoints;
  '3': Oui('C', c);
  '4': Begin
         Case Nvr of
          1: FormMoPol(Ak);
          2: Begin
               Mo[-1] := L;      {Dx1 := (Xk - Xn)/L;}
               For s := 0 to L do
                 Begin
                   X := Xn + s * dx;
                   Mo[s] := sin(Alpha * x) + k * cos(Beta * x)
                 end
             End
         End;
         Graphic(Mo, c)
       End;
  '5': Begin
         IntPol(Mx, My, a);
         FormMoPol(a);
         Graphic(Mo, c)
       End;
  '6': Begin
         Clear(0, 20, GetMaxX - 1, GetMaxY - 11);
         If Nvr=1 then
           OutPol('Ak', Ak, 20);
         OutPol('A', a, 320);
       End;
  '7': Oui('N', n);
```

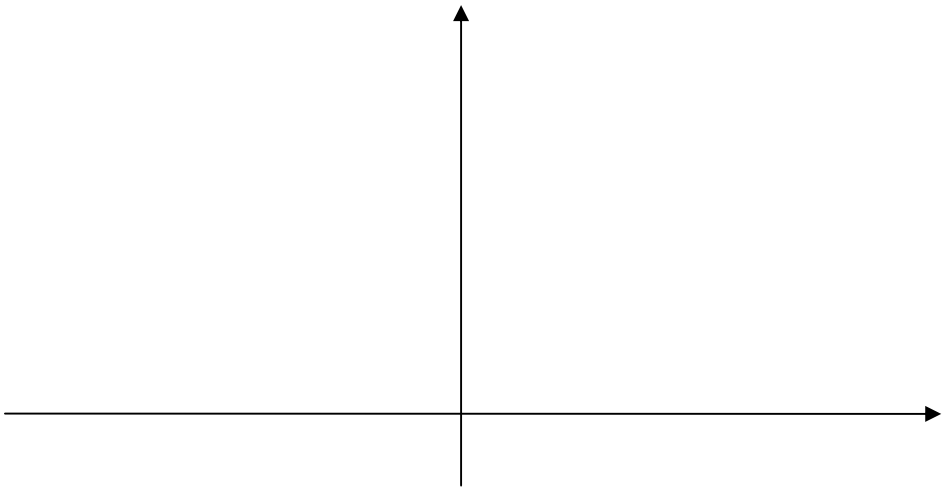
```
      '8': begin
            NaimKV(Mx,My,n,E,a);
            FormMoPol(a);
            Graphic(Mo,c)
           end;
       '9':Service
    end
  Until j=#27;
  CloseGraph
End.
```

3. Результати роботи

а) введення контрольного полінома

б) введення

в) графіки контрольного і згладжуючих поліномів при $n$=0,1,2,3,…



г) масиви коефіцієнтів $Ak(x)$, та $A(x)$ при $n$=0,1,2,3,… .

Звіт повинен містити назву роботи, мету, короткі теоретичні відомості, заповнений п.3

Контрольні запитання

1. В чому суть метода найменших квадратів.
2. Механізм роботи процедури *NaimKv*.
3. Механізм заповнення масиву *D*.
4. Механізм заповнення масиву *B* та *C*.

*Лабораторна робота № 4*

Інтерполяційні кубічні сплайни

Мета роботи: Дослідити апроксимацію полінома інтерполяційними кубічними сплайнами.

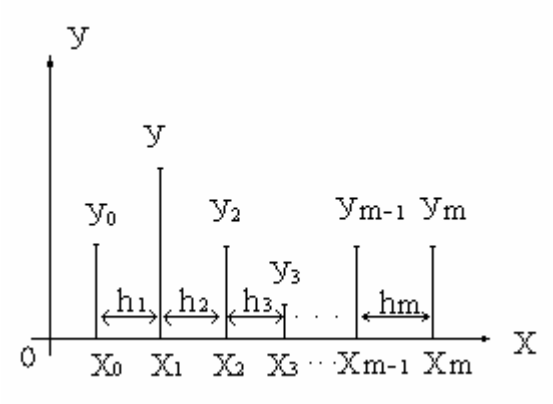
Теоретичні відомості

Нехай маємо упорядкований набір вузлів апроксимації

$$X_0<X_1<X_2<...<X_{m-1}<X_m$$

та відповідні їм значення функції , що підлягає апроксимації

$$Y_0,\ Y_1,\ Y_2,\ ...\ ,Y_{m-1},\ Y_m.$$




Розрбиваємо інтервал ($X_0 \le X \le X_m$ ) на підінтервали $X_{s-1} \le X \le X_s$ , $1 \le S \le m$ . Ширина $S$-го підінтервалу $h = X_s - X_{s-1}$. На s-ому інтервалі апроксимуючу функцію $\Psi_s(x)$ шукаємо у вигляді кубічного полінома у формі

$$\Psi_s(x) = a_{0,s} + a_{1,s}(X-X_{s-1}) + a_{2,s}(X-X_{s-1})^2 + a_{3,s}(X-X_{s-1})^3 ,\ 1 \le S \le m \quad (1).$$

Перша та друга похідна за $X$ від (1) мають вигляд :

$$\Psi_s'(X) = a_{1,s} + 2a_{2,s}(X-X_{s-1}) + 3a_{3,s}(X-X_{s-1})^2 ,\ 1 \le S \le m \quad (2).$$

$$\Psi_s''(X) = 2a_{2,s} + 6a_{3,s}(X-X_{s-1}) ,\ 1 \le S \le m \quad (3).$$

Структура (1) буде виконувати функції інтерполяційного сплайна , якщо її коєфіцієнти підібрати таким чином ,щоб :

а) значення функції (1) у вузлах апроксимації співпадали б з відповідними значеннями функції $Y_s$ , $0 \le S \le m$ , що апроксимується ;

б) на стиках інтервалів мають бути неперервними значення першої та другої похідних структури (1).

Отже , умова а) – умова неперервності кубічного сплайна полягає в тому ,що значення (1) на початку (точка $X_{s-1}$) і в кінці (точка $X_s$) $s$-го інтервалу мають бути рівними $Y_{s-1}$ та $Y_s$ відповідно :

$$\Psi_s(X_{s-1}) = Y_{s-1} ,\ 1 \le S \le m \quad (4)$$

$$\Psi_s(X_s) = Y_s \ , \ 1 \le S \le m \qquad (5)$$

Підставляючи $X_{s-1}$ та $X_s$ замість $X$ в (1) отримуємо з (4) та (5)

$$a_{0,s} = Y_{s-1} \ , \ 1 \le S \le \mathrm{m} \qquad (6)$$

$$a_{0,s} + a_{1,s} h_s + a_{2,s} h_s^2 + a_{3,s} h_s^3 = Y_s, \ 1 \le S \le m \qquad (7)$$

де $h_s = X_s - X_{s-1}$.

Умова неперервності 1-ої та 2-ої похідних у точці $X_s$ :

$$a_{1,s} + 2a_{2,s} h_s + 3\mathrm{a}_{3,s} h_s^3 = a_{1,s+1} \ , \ 1 \le S \le m\text{-}1 \qquad (8)$$

$$\mathrm{a}_{2,s} + 3\mathrm{a}_{3,s} \mathrm{h}_s^3 = \mathrm{a}_{2,s+1} \ , \ 1 \le S \le m\text{-}1 \qquad (9)$$

Система (6) є уже фактично розв'язаною відносно $a_{0,s}$ . Рівняння (7),(8),(9) утворюють систему відносно решти невідомих коєфіцієнтів структури (1) , яка може бути остаточно розв'язана за умов розширення її за рахунок двох додакових рівнянь , а саме рівнянь граничних умов на лівому (точка $X = X_0$) та правому (точка $X = X_m$) кінцях сплайна .

Сам термін "сплайн" означає "гнучка лінійка", оскільки в механіці показується , що форма гнучкої балки (лінійки) , описується саме рівнянням типу (1) ,за умови , що вузлами апроксимації є шарнірні опори. Існує ряд варіантів задання граничних умов , найбільш часто використовуваними з яких є граничні умови 1-го та 2-го роду.

Гранична умова 1-го роду задає значення першої похідної , а 2-го роду – значеня другої похідної на відповідному кінці сплайна.

$$Ngl = 1, \quad \Psi_1'(X_0) = Y_0' \qquad (10)$$

$$Ngl = 2, \quad \Psi_1''(X_0) = Y_0'' \qquad (11)$$

$$Ngr = 1, \quad \Psi_m'(X_m) = Y_m' \qquad (12)$$

$$Ngr = 2, \quad \Psi_m''(X_m) = Y_m'' \qquad (13)$$

Система (7)-(9) шляхом виключення $a_{1,s}$ та $a_{3,s}$ зводиться до системи відносно $a_{2,s}$ виду :

$$h_s a_{2,s} + e_s a_{2,s+1} + h_{s+1} a_{2,s+2} = r_s \ , \qquad (14)$$

де $e_s = 2(h_s - h_{s-1})$, $1 \le S \le m\text{-}1$,

$$r_s = 3((Y_s + 1 - Y_s)/h_{s+1} - (Y_s - Y_{s-1})/h_s).$$

Якщо систему (14) доповнити двома рівняннями граничних умов , то вона може бути розв'язана методом прогонки , після чого можуть бути визначені коєфіцінтами $a_{1,s}$ та $a_{3,s}$. Шукані коєфіцієнти $a_{0,s}$, $a_{1,s}$, $a_{2,s}$, $a_{3,s}$ розміщуються у масивах *A*0, *A*1, *A*2, *A*3 : *Coefr*, де $1 \le S \le m$.

Формування масивів *A*0, *A*1, *A*2, *A*3 : *Coefr* коєфіцієнтів структури (1) (інтерполяційного кубічного сплайна) реалізує підпрограма *SplineN*. Саму ж задачу інтерполяції виконує підпрограма – функція SplintN.

Тут у підпрограмі *SplineN* масиви *X,Y*:*Coefr* задають точки на площині *X*0*Y*, через які має пройти графік інтерполяційного сплайна.

*Ngl*, *Ngr* – номер граничної умови на лівому (*Ngl*) та правому (*Ngr*) кінцях сплайна .

*D*0 – значеннння першої чи відповідної другої похідної (в залежності від значення *Ngl*) від сплайна на лівому його кінці.

*Dm* – аналогічно для правого кінця сплайна .

У підпрограмі *SplintN* спочатку визначається , в який інтервал попадає поточне значення X, а потім за схемою Хорнера обчисляється значення полінома (1).

У демонстраційно-відлагоджувальній програмі *DemSpl* набір точок для апроксимації формується за допомогою підпрограми *InpD* , яка реалізує таку залежність

$$y = B1\ sin\ (w1*x) + B2\ cos\ (w2*x).$$

Коєфіцієнти *B*1, *B*2,*w*1,*w*2, а також діапазон *Xmin* , *Xmax* та кількість кроків (інтервалів) *m* задаються користувачем на його розсуд .

Підпрограма *InpD* не лише формує масиви *Mx*, *My* : *Coefr* координат точок, але і формує відповідну систему координат та зображує дані точки у цій системі координат.

Хід роботи.

1. Завантажити середовище *Pascal*.
2. Набрати текст програми *DemSpline*.

```
program DemSpline;
```

```
uses Crt,Graph,Serv;
const c1:real=2;
      w1:real=1.5;
      c2:real=0.5;
      w2:real=5;
      x1:real=-2;
      x2:real=3;
      m:integer=10;
      Ngl:integer=1;
      Ngr:integer=2;
      D0:real=0;
      Dm:real=0;
var   Mx,My,A0,A1,A2,A3:Coefr;Mo:coefl;  Dx1,x,g:real;i:integer;
  Function HorReal(a: Coef; x: Real): real;
    var n, s: integer;
        r: real;
    Begin
      n:= Round(A[-1]);    R := A[n];
      For s := n-1 downto 0 do
        R := R * x + A[s];
      HorReal := R;
    End;
function SplintN(A0,A1,A2,A3:coefr;x:real):real;
  var
   m,s:integer;
  begin
   s:=1;
   while Mx[s-1]<x do inc(s);
   if s>1 then dec(s);
   x:=x-Mx[s-1];
```

```
  SplintN:=((A3[s]*x+A2[s])*x+a1[s])*x+a0[s];
 end;
procedure SplineN (X,Y:coefr;Ngl,Ngr:integer; D0,Dm:real);
  var
   m,s:integer;
   h,h1,e,v,z,r:real;
   a,b:coefr;
  begin
   m:=round(x[-1]);
   case Ngl of
    1: begin
        h:=x[1]-x[0]; A[0]:=-0.5;
        B[0]:=1.5*((y[1]-y[0])/h-d0)/h;
       end;
    2: begin
        A[0]:=0; b[0]:=d0/2;
       end;
   end;
   for s:=1 to m-1 do
    begin
     h:=x[s]-x[s-1]; h1:=x[s+1]-x[s];
     e:=2*(h+h1);
     r:=3*((y[s+1]-y[s])/h1-(y[s]-y[s-1])/h);
     z:=e+a[s-1]*h; a[s]:=-h1/z;
     b[s]:=(r-b[s-1]*h)/z;
    end;
   case Ngr of
    1: begin
        h:=x[m]-x[m-1];
        a2[m+1]:=(3*(dm-(y[m]-y[m-1])/h)-b[m-1]*h)/((a[m-1]+2)*h);
```

```
      end;
    2: a2[m+1]:=dm/2;
   end;
   for s:= m-1 downto 0 do
    a2[s+1]:=a[s]*a2[s+2]+b[s];
   for s:=1 to m do
    begin
     h:=x[s]-x[s-1];
     a1[s]:=(y[s]-y[s-1])/h-2/3*a2[s]*h-1/3*a2[s+1]*h;
     a3[s]:=(a2[s+1]-a2[s])/(3*h);
     a0[s]:=y[s-1];
    end;
   a0[-1]:=m;
 end;
procedure InpD;
 begin
  repeat
   putA;
   Ou('0-exit,1-C1,2-w1,3-C2,4-w2,5-x1,6-x2,7-m,8-SystCoor,9-points');
   str(c1:1:3,T10); Ts:='C1='+T10;
   str(w1:1:3,T10); Ts:=Ts+', w1='+T10;
   str(c2:1:3,T10); Ts:=Ts+', C2='+T10;
   str(w2:1:3,T10); Ts:=Ts+', w2='+T10;
   str(x1:1:3,T10); Ts:=Ts+', x1='+T10;
   str(x2:1:3,T10); Ts:=Ts+', x2='+T10;
   str(m:1,T10);    Ts:=Ts+', m='+T10;
   Info; j1:=readKey;
   case j1 of
    '1': Our('C1',C1);
    '2': Our('w1',w1);
```

```
    '3': Our('C2',C2);
    '4': Our('w2',w2);
    '5': Our('x1',x1);
    '6': Our('x2',x2);
    '7': Oui('m',m);
    '8': begin
         xMax:=x2; xMin:=x1;
         yMax:=0;  yMin:=0;
         Mx[-1]:=m; My[-1]:=m;
         Dx:=(x2-x1)/m;
         for s:=0 to m do
          begin
           x:=x1+s*Dx; Mx[s]:=x;
           y:=C1*cos(w1*x)+C2*sin(w2*x);
           My[s]:=y;
           if y<yMin then yMin:=y;
           if y>yMax then yMax:=y;
          end;
         X0Y0(false); cleardevice;
         SystCoor;
        end;
    '9': for s:=0 to m do
         circle(X0+round(Mx[s]/Dx),Y0-round(My[s]/Dy),3);
   end;
  until j1='0';
end;
begin
 c:=15;
 repeat
  PutA;
```

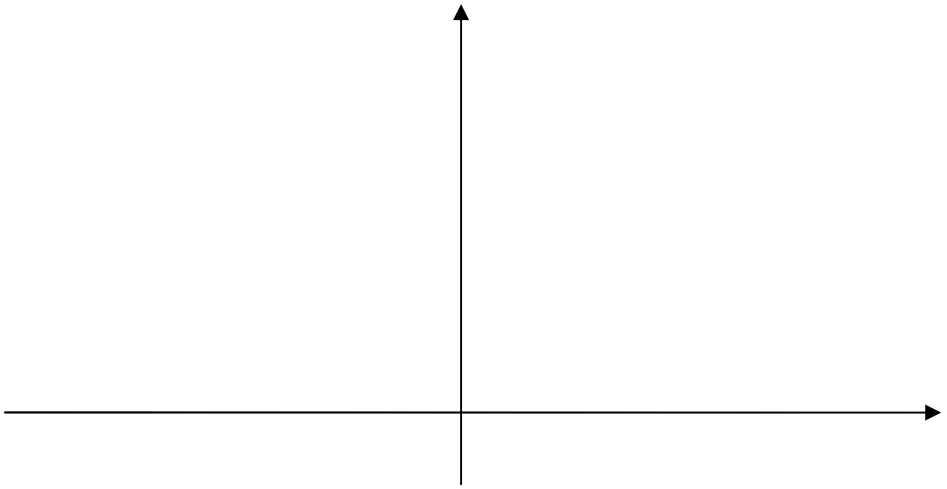
```
Ou('Esc-exit,1-InpD,2-c,3-Grf,4-Ngl,5-Ngr,6-D0,7-Dm,8-GrSpl');
str(c,T10);   Ts:='c='+T10;
str(Ngl,T10); Ts:=Ts+', Ngl='+T10;
str(Ngr,T10); Ts:=Ts+', Ngr='+T10;
str(D0:1:3,T10);  Ts:=Ts+', D0='+T10;
str(Dm:1:3,T10);  Ts:=Ts+', Dm='+T10;
Info;
j:=ReadKey;
case j of
 '1': InpD;
 '2': Oui('c',c);
 '3': begin
       SetColor(c);
       Mo[-1]:=L;
       for s:=0 to L do
        begin
         x:=x1+s*Dx;
         Mo[s]:=c1*cos(w1*x)+c2*sin(w2*x);
        end; {X0Y0(false); SystCoor;}
       Graphic(Mo,c);
      end;
 '4': Oui('Ngl',Ngl);
 '5': Oui('Ngr',Ngr);
 '6': Our('D0',D0);
 '7': Our('Dm',Dm);
 '8': begin
       Dx1:=(x2-x1)/m;Mx[-1]:=m;
       for s:=0 to m do
        begin
         Mx[s]:=x1+s*Dx1;
```

```
          My[s]:=c1*cos(w1*Mx[s])+c2*sin(w2*Mx[s]);
         end;
        SplineN(Mx,My,Ngl,Ngr,D0,Dm);
        Mo[-1]:=L; Dx:=(x2-x1)/L;
        for i:=0 to L do
         begin
          x:=x1+i*Dx;
          Mo[i]:=splintn(A0,A1,A2,A3,x);
         end;
       graphic(mo,c);
       end;
  end;
 until j=#27;
end.
```

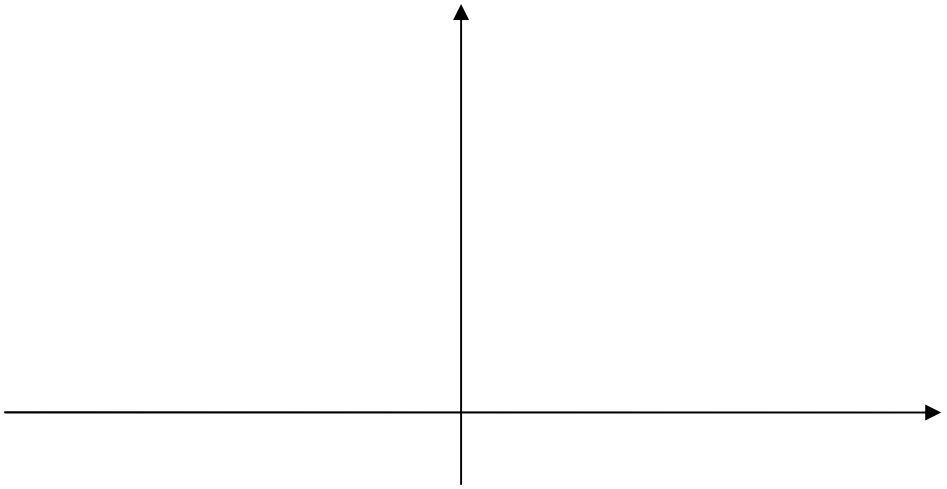
3. Результати роботи

а) Задати параметри функції

б) Сформувати систему координат з “точками”



в) Сформувати графік базової функції та графіки інтерполяційних сплайнів при різних варіантах граничних умов.



Звіт повинен містити назву роботи, мету, короткі теоретичні відомості, заповнений п.3

Контрольні запитання

1. В чому суть інтерполяції кубічними  сплайнами
2. Механізм роботи процедури *SplineN*.
3. Механізм роботи функції *SplintN*.
4. Що таке гранична умова?

*Лабораторна робота №5*

Інтерполяційні В-сплайни

Мета роботи : Дослідити характер поведінки В-сплайна та його похідних.

Теоретичні відомості

Нехай вузли апроксимації утворюють арифметичну прогресію

$$x_S = x_0 + sh,\ \ 1 \le s \le m, \qquad (1)$$

де $h = const$. У відповідність кожному $x_S$ поставлено значення $y_S$ функції, що підлягає апроксимації.

Кубічний В-сплайн (базовий сплайн) для інтервалу $x_{S-2} \le x \le x_{S+2}$ визначається так:

$$\mathrm{B_S}(x) = \begin{cases} \frac{1}{6}(u_S + 2)^3, x \in [x_{S-2}, x_{S-1}], \\ \frac{2}{3} - \frac{1}{2}\left(u_S{}^3 + 2u_S{}^2\right), x \in [x_{S-1}, x_S], \\ \frac{2}{3} + \frac{1}{2}\left(u_S{}^3 - 2u_S{}^2\right), x \in [x_S, x_{S+1}], \\ \frac{1}{6}(2 - u_S)^3, x \in [x_{S+1}, x_{S+2}], \\ 0 \ \text{для усіх інших x}, \end{cases}, \qquad (2)$$

де $u_s = \dfrac{x - x_s}{h}$.

Перша, друга та третя похідні від В-сплайна визначаються відповідно

$$\mathrm{B}_s'(x)=\begin{cases}\dfrac{1}{2\mathrm{h}}(u_s+2)^2, x\in[x_{s-2},x_{s-1}],\\ -\dfrac{1}{2h}(3u_s{}^2+4u_s), x\in[x_{s-1},x_s],\\ \dfrac{1}{2h}(3u_s{}^2-4u_s), x\in[x_s,x_{s+1}],\\ -\dfrac{1}{2\mathrm{h}}(2-u_s)^2, x\in[x_{s+1},x_{s+2}],\\ 0 \text{ для усіх інших x .}\end{cases}, \qquad (3)$$

$$\mathrm{B}_s''(x)=\begin{cases}\dfrac{1}{\mathrm{h}^2}(u_s+2), x\in[x_{s-2},x_{s-1}],\\ -\dfrac{1}{h^2}(3u_s+2), x\in[x_{s-1},x_s],\\ \dfrac{1}{h^2}(3u_s-2), x\in[x_s,x_{s+1}],\\ \dfrac{1}{\mathrm{h}^2}(2-u_s), x\in[x_{s+1},x_{s+2}],\\ 0 \text{ для усіх інших x ,}\end{cases}, \qquad (4)$$

$$\mathrm{B}_s'''(x)=\begin{cases}\dfrac{1}{\mathrm{h}^3}, x\in[x_{s-2},x_{s-1}],\\ -\dfrac{1}{3h^3}, x\in[x_{s-1},x_s],\\ \dfrac{1}{3h^3}, x\in[x_s,x_{s+1}],\\ -\dfrac{1}{\mathrm{h}^3}, x\in[x_{s+1},x_{s+2}],\\ 0 \text{ для усіх інших x .}\end{cases}, \qquad (5)$$

Уявлення про форму В-сплайна та його першої, другої та третьої похідних можна отримати з використанням програми *BSplGrs*, яка формує графік самого В-сплайна (його нульової похідної *Np*=0) та відповідних похідних (*Np* – порядок похідної).

Хід роботи.

1. Завантажити середовище *Pascal*.
2. Набрати текст програми *BSplGrs*.

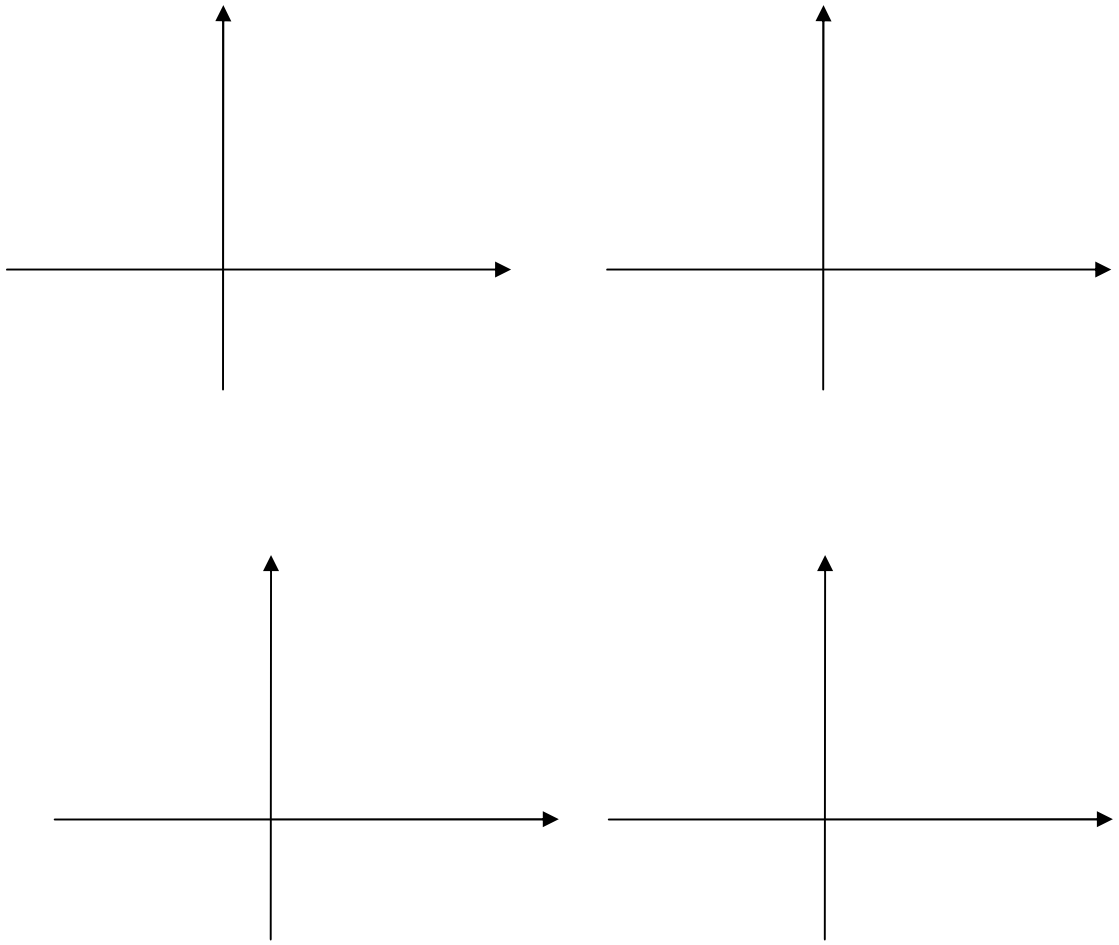
```
program BSplGrs;
uses Crt,Graph, Serv, Gromk;
var Mo:Coefl; Np:integer;
function F(Np:integer; x:real):real;
begin
  if (x<=-2) or (x>=2) then f:=0;
  if(x>-2) and (x<=-1) then
    begin
      y:=x+2;
      case Np of
        0: f:=y*sqr(y)/6;
        1: f:=sqr(y)/2;
        2: f:=y;
        3: f:=1
      end
    end;
  if (x>-1) and (x<=0) then
    case Np of
      0: f:=2/3-0.5*sqr(x)*(x+2);
      1: f:=-0.5*x*(3*x+4);
      2: f:=-3*x-2;
      3: f:=-3
    end;
```

```
 if (x>0) and (X<=1) then
   case Np of
     0: f:=2/3+0.5*sqr(x)*(x-2);
     1: f:=0.5*x*(3*x-4);
     2: f:=3*x-2;
     3: f:=3
   end;
 if (x>1) and (x<=2) then
   begin
     y:=2-x;
     case Np of
       0: f:=y*sqr(y)/6;
       1: f:=-sqr(y)/2;
       2: f:=y;
       3: f:=-1
     end
   end
end;
begin
  Np:=0; c:=10;
  repeat
    puta;
Ou('Esc-exit, 1-L, 2-H, 3-Np, 4-C, 5-Graph, 6-Mark, 7-Net, 8-Num, 9-Inscr');
str(L,t10); ts:='L='+t10;
str(h,t10); ts:=ts+', H='+t10;
str(Np,t10); ts:=ts+', Np='+t10;
str(c,t10); ts:=ts+', C='+t10;
info; J:=ReadKey;
case J of
  '1':Oui('L',L);
```

```
  '2':Oui('H',H);
  '3':Oui('Np',Np);
  '4':Oui('C',C);
  '5': begin
         Ymin:=0; Ymax:=0;
         Xmin:=-2; Xmax:=2;
         dx:=(Xmax-Xmin)/L; Mo[-1]:=L;
         for s:=0 to L do
          begin
            x:=Xmin+s*dx; y:=f(Np,x);
            if (s=L) and (Np=3) then y:=0;
            Mo[s]:=y;
            if y<Ymin then Ymin:=y;
            if y>Ymax then Ymax:=y
          end;
         X0Y0(false); ClearDevice; SystCoor;
         Graphic(Mo,C)
       end;
  '6': Mark;
  '7': Net;
  '8': Num;
  '9':Inscr
 end
 until J=#27;
 CloseGraph
end.
```

3. Результати роботи

Сформувати на екрані та відобразити в протоколі графіки B-сплайнів при різних значеннях *Np*



Звіт повинен містити назву роботи, мету, короткі теоретичні відомості, заповнений п.3

Контрольні запитання

1. Основна різниця B-спалйнів і інтерполяційних кубічних сплайнів.
2. Механізм роботи функції *F*.

*Лабораторна робота №6*

Інтерполяційні В-сплайни

Мета роботи : Дослідити апроксимацію полінома за допомогою В-сплайнів.

Теоретичні відомості

Ідея використання В-сплайнів для апроксимації, зокрема для інтерполяції полягає в тому, що до кожного із вузлів (нумерація від 1 до $m$+1, вузли – рівновіддалені, тобто утворюють арифметичну прогресію) «прив'язується» (з коефіцієнтом $b_s$) В-сплайн. Сума усіх прив'язаних таким чином В-сплайнів і утворюють арифметичну функцію.

Якщо врахувати, що дія окремого В-сплайна розраховується на 2 кроки вліво та вправо відносно точки його «прив'язкиується», апроксимуючу функцію можна представити так

$$\Psi(x) = \sum_{s=-1}^{m+1} b_s B_s(x) \,, \tag{6}$$

А якщо врахувати, що В-сплайн в точку на крок лівіше та крок правіше від центрального вузла (точки «прив'язки») дорівнює 1/6, а в центральній точці він дорівнює 2/3=4/6, то формула (6) набуває вигляду

$$\Psi(x)\ b_{s-1}B_{S-1}(x_S) + b_S B_S(x_S) + b_{S+1}B_{S+1}(x_S) = y_S, 0 \le s \le m. \tag{7}$$

де $s$-номер інтервалу, в якій попадає значення $x$. Якщо система (7) доповнити двома рівняннями граничних умов (лівому $s$=0 та правому$s$=$m$ кінцях інтервалу апроксимації), то отримана система лінійних алгебраїчних рівнянь може бути розв'язана відносно «коефіцієнтів прив'язки» $b_s$, $-1 \le s \le m+1$. Формування масиву $B$:*Coefr* для коефіцієнтів $b_s$ забезпечує підпрограма Bspline.

Задачу інтерполяції (визначення значень апроксимуючої функції у проміжках між вузлами) вирішує підпрограма-функція *Bsplint*.

Можна показати, що апроксимуюча функція, отримана (для рівновіддалених вузлів!) з використанням звичайного кубічного

інтерполяційного сплайна є тотожною інтерполяційній функції (6) на базі B-сплайнів (при однакових граничних умовах). Справді – сума поліномів (B-сплайн – кубічний поліном на із під інтервалів) є кубічний поліном. А набір кубічних сплайнів (кубічних поліномів) при заданому наборі граничних умов є унікальним (реалізується в єдиному варіанті). Отже, сумарний кубічний поліном, сформований на базі B-сплайнів на розгляданому під інтервалі повинен бути тотожним відповідному кубічному поліному у складі кубічного сплайна.

В програмі передбачено формування інтерполяційного графіка як на базі звичайного кубічного сплайна, так і на базі B-сплайннів.

Хід роботи.

1. Завантажити середовище *Pascal*.
2. Набрати текст програми *DemSpl.*

```
program DemSpl;
 uses Crt,Graph,Serv,Groms;
 const
  c1:real=2;
  w1:real=1.5;
  c2:real=0.5;
  w2:real=5;
  x1:real=-2;
  x2:real=3;
  m:integer=10;
  Ngl:integer=1;
  Ngr:integer=2;
  D0:real=0;
  Dm:real=0;
 var
  Mx,My,A0,A1,A2,A3,B:Coefr;
  Mo,Ma:coefl;
```

```
 Dx1:real;
 i:integer;
function SplintN(A0,A1,A2,A3:coefr;x:real):real;
 var
  m,s:integer;
 begin
  s:=1;
  while Mx[s-1]<x do inc(s);
  if s>1 then dec(s);
  x:=x-Mx[s-1];
  SplintN:=((A3[s]*x+A2[s])*x+a1[s])*x+a0[s];
 end;
procedure SplineN (X,Y:coefr;Ngl,Ngr:integer; D0,Dm:real);
  var
   m,s:integer;
   h,h1,e,v,z,r:real;
   a,b:coefr;
  begin
   m:=round(x[-1]);
   case Ngl of
    1: begin
        h:=x[1]-x[0]; A[0]:=-0.5;
        B[0]:=1.5*((y[1]-y[0])/h-d0)/h;
       end;
    2: begin
        A[0]:=0; b[0]:=d0/2;
       end;
   end;
   for s:=1 to m-1 do
    begin
```

```
    h:=x[s]-x[s-1]; h1:=x[s+1]-x[s];
    e:=2*(h+h1);
    r:=3*((y[s+1]-y[s])/h1-(y[s]-y[s-1])/h);
    z:=e+a[s-1]*h; a[s]:=-h1/z;
    b[s]:=(r-b[s-1]*h)/z;
   end;
  case Ngr of
   1: begin
       h:=x[m]-x[m-1];
       a2[m+1]:=(3*(dm-(y[m]-y[m-1])/h)-b[m-1]*h)/((a[m-1]+2)*h);
      end;
   2: a2[m+1]:=dm/2;
  end;
  for s:= m-1 downto 0 do
   a2[s+1]:=a[s]*a2[s+2]+b[s];
  for s:=1 to m do
   begin
    h:=x[s]-x[s-1];
    a1[s]:=(y[s]-y[s-1])/h-2/3*a2[s]*h-1/3*a2[s+1]*h;
    a3[s]:=(a2[s+1]-a2[s])/(3*h);
    a0[s]:=y[s-1];
   end;
  a0[-1]:=m;
 end;
procedure InpD;
 begin
  repeat
   putA;
   Ou('0-exit,1-C1,2-w1,3-C2,4-w2,5-x1,6-x2,7-m,8-SystCoor,9-points');
   str(c1:1:3,T10); Ts:='C1='+T10;
```

```
str(w1:1:3,T10); Ts:=Ts+', w1='+T10;
str(c2:1:3,T10); Ts:=Ts+', C2='+T10;
str(w2:1:3,T10); Ts:=Ts+', w2='+T10;
str(x1:1:3,T10); Ts:=Ts+', x1='+T10;
str(x2:1:3,T10); Ts:=Ts+', x2='+T10;
str(m:1,T10);    Ts:=Ts+', m='+T10;
Info; j1:=readKey;
case j1 of
 '1': Our('C1',C1);
 '2': Our('w1',w1);
 '3': Our('C2',C2);
 '4': Our('w2',w2);
 '5': Our('x1',x1);
 '6': Our('x2',x2);
 '7': Oui('m',m);
 '8': begin
      xMax:=x2; xMin:=x1;
      yMax:=0;  yMin:=0;
      Mx[-1]:=m; My[-1]:=m;
      Dx:=(x2-x1)/m;
      for s:=0 to m do
       begin
        x:=x1+s*Dx; Mx[s]:=x;
        y:=C1*cos(w1*x)+C2*sin(w2*x);
        My[s]:=y;
        if y<yMin then yMin:=y;
        if y>yMax then yMax:=y;
       end;
      X0Y0(false); cleardevice;
      SystCoor;
```

```
        end;
    '9': for s:=0 to m do
         circle(X0+round(Mx[s]/Dx),Y0-round(My[s]/Dy),3);
   end;
  until j1='0';
end;
procedure BSpline (Y:Coefr; Ngl,Ngr:integer;D0,Dm:real;
                                                  var m:integer; var B:Coefr);
        var     s:integer;
                z,h:real;
                A,C:Coefr;
        begin
          m:=round(Y[-1]); h:=1;
        case Ngl of
                1: begin A[0]:=-0.5; C[0]:=(3*Y[0]+h*D0)/2 end;
                2:begin A[0]:=0; C[0]:=Y[0]-D0*sqr(h)/6 end
        end;
        for s:=1 to m-1 do
        begin
                z:=A[s-1]+4;
                A[s]:=-1/z; C[s]:=(6*Y[s]-C[s-1])/z
        end;
        case Ngr of
                1: B[m]:=(-C[m-1]+3*Y[m]-h*Dm)/(2+A[m-1]);
                2: B[m]:=Y[m]-Dm*sqr(h)/6
        end;
        for s:=m-1 downto 0 do
                B[s]:=A[s]*B[s+1]+C[s];
                B[-1]:=6*Y[0]-B[1]-4*B[0];
                B[m+1]:=6*Y[m]-B[m-1]-4*B[m]
end;
function BSplint(x:real):real;
        var s:integer;
        begin
          s:=trunc(x); x:=frac(x);
        if s>m then  BSplint:=(B[m-1]+4*B[m]+B[m+1])/6
        else
        BSplint:= B[s-1]*(1-x)*sqr(1-x)/6+B[s]*(2/3+0.5*sqr(x)*(x-2))+
                                B[s+1]*(2/3-0.5*sqr(x-1)*(x+1))+B[s+2]*x*sqr(x)/6

        end;
```

```
begin
 L:=300;H:=240;
 c:=15;
 repeat
  PutA;
  Ou('Esc-exit,1-InpD,2-c,3-Grf,4-Ngl,5-Ngr,6-D0,7-Dm,8-AprSpl,9-AprBSpl');
  str(c,T10);       Ts:='c='+T10;
  str(Ngl,T10);     Ts:=Ts+', Ngl='+T10;
  str(Ngr,T10);     Ts:=Ts+', Ngr='+T10;
  str(D0:1:3,T10);  Ts:=Ts+', D0='+T10;
  str(Dm:1:3,T10);  Ts:=Ts+', Dm='+T10;
  Info; j:=ReadKey;
  case j of
   '1': InpD;
   '2': Oui('c',c);
   '3': begin
         SetColor(c); Mo[-1]:=L;
         for s:=0 to L do
          begin
           x:=x1+s*Dx;
           Mo[s]:=c1*cos(w1*x)+c2*sin(w2*x);
          end;
         Graphic(Mo,c);
        end;
   '4': Oui('Ngl',Ngl);
   '5': Oui('Ngr',Ngr);
   '6': Our('D0',D0);
   '7': Our('Dm',Dm);
   '8': begin
         SplineN(Mx,My,Ngl,Ngr,D0,Dm);
```

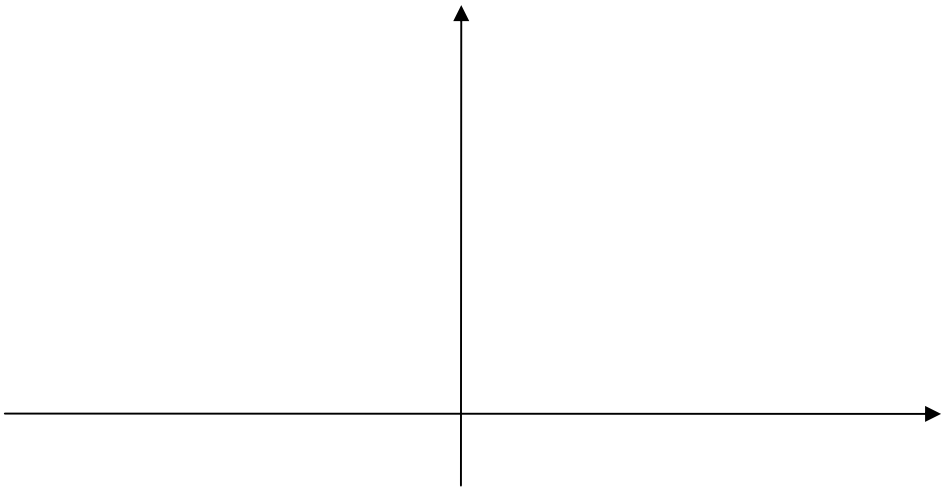
```
        Mo[-1]:=L;
        for s:=0 to L do
          Ma[s]:=splintn(A0,A1,A2,A3,Mx[s]);
       Graphic(Ma,c);
      end;
    '9':begin
        BSpline(My,Ngl,Ngr,D0,Dm,m,B);
        Mo[-1]:=L;
        for s:=0 to L do
          Ma[s]:=BSplintn(x);
       Graphic(Ma,c);
      end;
       end;
  end;
 until j=#27;
end.
```

3. Результати роботи

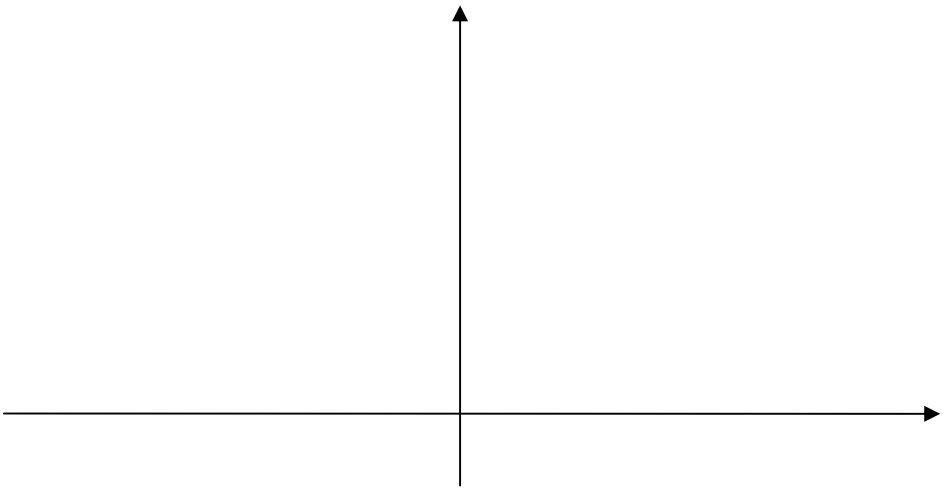
а) точковий графік, що підлягає апроксимації за заданими значеннями

$x_n$ = _____ ; $x_k$ = _____ ; m = _____ ;

B1= _____ ; B2= _____ ; w1= _____ ; w2 = _____ .



б) При різних комбінаціях значень граничних умов Ngl, Ngr та D0, Dm, сформулювати графіки інтерполяційного пучкового сплайна та апроксимацію за допомогою B-сплайна.



Звіт повинен містити назву роботи, мету, короткі теоретичні відомості, заповнений п.3

Контрольні запитання

1. Принцип апроксимації B-спалйнами.
2. Механізм роботи процедури *InpD*.

Список рекомендованої літератури.

1. Аверіна Т.В., Кубрак Н.А. Динаміка елементів систем : Навч. посібник – К.: ІЗМН, 1998 – 224 с.

2. Карачун В.В., Кваско М.З., Кубрак Н.А. Прикладний аналіз і візуалізація характеристик динамічних систем : Навч. посібник – К.: ІЗМН, 1999 – 138 с.

3. Кубрак Н.А. Хвильові процеси в гнучких ланках автоматичних систем : Навч. посібник – К.: НМЦ ВО, 2000 – 160 с.

4. Калиткин Н.Н. Численные методы – М.: Наука, 1978 – 512 с.

5. Кошляков Н.С., Глинер Э.Б., Смирнов М.М. Основные дифференциальные уравнения математической физики – М.: Физмат, 1962 – 767 с.

6. Кубрак А.І. Ідентифікація динамічних характеристик елементів систем керування. Част. 1. Математичні методи : Навч. посібник – К.: ІСДО, 1995 – 208 с.