Metoda Neville. Metode de interpolare cu funcţii spline cubice. Curbe Bézier. Algoritmul De Casteljau

Noțiuni teoretice

Metoda Neville

Se consideră o funcţie $f : [a, b] \to R$ cunoscută într-o mulţime finită de puncte $x_0, x_1, ..., x_n$ (numite suportul interpolării) prin valorile:

$$f(x_0), f(x_1), ..., f(x_n).$$

Metoda Neville este o metodă de interpolare care aproximează comportamentul funcţiei f în afara acestor $n + 1$ puncte. În cele ce urmează, folosim notaţia $P_{ij}(x)$ pentru a reprezenta polinomul de interpolare de grad $j - i$ care trece prin punctele $(x_l, f(x_l)), l = i, i + 1, ..., j$.

Polinomul de interpolare P_{ij} este dat de relaţia de recurență:

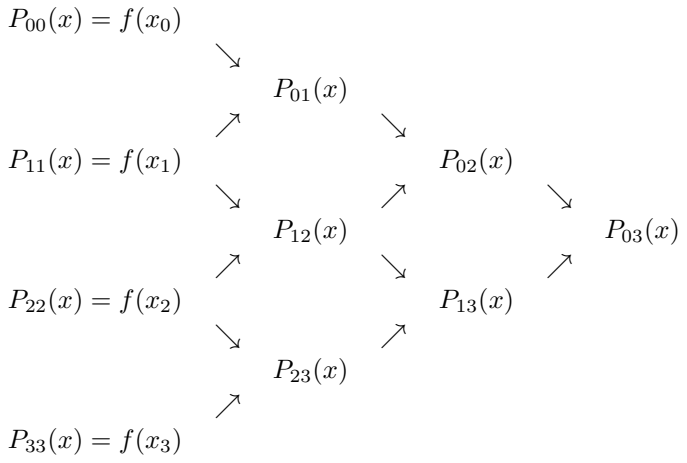
$$P_{ij}(x) = \frac{x - x_j}{x_i - x_j} P_{i,j-1}(x) + \frac{x_i - x}{x_i - x_j} P_{i+1,j}(x), \quad 0 \leq i < j \leq n$$

unde

$$P_{ii}(x) = f(x_i), \quad i = 0 : n.$$

În prima iteraţie, metoda Neville construieşte polinoame de interpolare de grad 0, reprezentatate prin $P_{ii}(x), i = 0 : n$. În următoarea iteraţie, oricare două polinoame de interpolare de grad 0, alăturate, P_{ii} şi $P_{i+1,i+1}, i = 0 : n - 1$, formează un polinom

de interpolare de grad 1 notat cu $P_{i,i+1}(x)$. Procesul continuă până când se obţine polinomul de interpolare $P_{0,n}$ care trece prin toate cele $n+1$ puncte $(x_l, f(x_l)), l = 0, ..., n$. De exemplu, procesul iterativ de obţinere al polinomului de interpolare $P_{03}(x)$, pentru $n = 3$, este arătat în schema următoare:



Metode de interpolare cu funcţii spline cubice

În continuare, considerăm că funcţia $f : [a, b] \to R$ este cunoscută atât într-o mulţime finită de puncte $x_0, x_1, ..., x_n$ prin valorile:

$$f(x_0), f(x_1), ..., f(x_n)$$

cât şi prin derivatele de ordin I ale funcţie în aceste puncte:

$$f^{'}(x_0), f'(x_1), ..., f'(x_n)$$

În cazul interpolării cu funcţii spline cubice, comportamentul funcţiei f se va studia local pe subintervalele:

$$[x_0, x_1], [x_1, x_2], ..., [x_i, x_{i+1}]..., [x_{n-1}, x_n]$$

Interpolare cu funcţii spline în clasa C^1:

Un polinom de gradul 3 se va considera pentru fiecare subinterval $[x_i, x_{i+1}]$ de forma:

$$s_i : [x_i, x_{i+1}] \to R, s_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, i = 0 : n - 1$$

Prin schimbarea de variabilă $t = \frac{x - x_i}{x_{i+1} - x_i}$ se obţine următoarea formă parametrică pentru o funcţie spline cubică:

$$s_i(t) = a_i + b_i h_i t + c_i h_i t^2 + d_i h_i t^3, i = 0 : n - 1$$

unde am notat $h_i = x_{i+1} - x_i$.

În general, baza de interpolare Bernstein se foloseşte pentru a eficientiza procesul de calculare al coeficienţilor funcţiilor spline cubice:

$$(1 - t)^3, 3t(1 - t)^2, 3t^2(1 - t), t^3$$

Folosind baza de interpolare Bernstein, funcţia spline cubică devine:

$$s_i(t) = a_i^{'}(1 - t)^3 + 3b_i^{'}t(1 - t)^2 + 3c_i^{'}t^2(1 - t) + d_i^{'}t^3, i = 0 : n - 1$$

Pentru a calcula cei $4n$ coeficienţi, se impun următorele condiţii în cazul interpolării cu funcţii spline în clasă C^1:

- $2n + 2$ condiţii de interpolare de tip Hermite:

$$s_i(x_i) = f(x_i), \quad i = 0 : n - 1$$

$$s_{n-1}(x_n) = f(x_n)$$

$$s_i^{'}(x_i) = f^{'}(x_i), \quad i = 0 : n - 1$$

$$s_{n-1}^{'}(x_n) = f^{'}(x_n)$$

- $2n - 2$ condiţii de racordare ce asigură continuitatea atat a funcţiilor spline cubice cat şi a derivatei de ordinul I pentru funcţiile spline cubice:

$$s_i(x_{i+1}) = s_{i+1}(x_{i+1}), \quad i = 0 : n - 2$$

$$s_i^{'}(x_{i+1}) = s_{i+1}^{'}(x_{i+1}), \quad i = 0 : n - 2$$

Considerand pentru funcţiile spline cubice forma parametrică în baza Bernstein, se rezolva sistemul cu cele $4n$ ecuaţii si se obţin următoarele relaţii pentru coeficienţi:

$$a_i^{'} = f(x_i), \quad i = 0 : n-1$$

$$d_i^{'} = f(x_{i+1}), \quad i = 0 : n-1$$

$$b_i^{'} = f(x_i) + \frac{h_i}{3} f^{'}(x_i), \quad i = 0 : n-1$$

$$c_i^{'} = f(x_{i+1}) - \frac{h_i}{3} f^{'}(x_{i+1}), \quad i = 0 : n-1$$

În cele din urmă, forma în variabila x pentru fiecare funcţie spline cubică în clasă C^1 se obţine prin schimbarea de variabilă $t = \frac{x-x_i}{h_i}$.

Interpolare cu funcţii spline în clasă C^2:

Un polinom de gradul 3 se va considera pentru fiecare subinterval $[x_i, x_{i+1}]$ de forma:

$$s_i : [x_i, x_{i+1}] \to R, s_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, i = 0 : n-1$$

În cazul interpolării cu funcţii spline în clasă C^2, cei $4n$ coeficienţi se determină prin impunerea a:

- $n + 1$ condiţii de interpolare de tip Lagrange:

$$s_i(x_i) = f(x_i), \quad i = 0 : n-1$$

$$s_{n-1}(x_n) = f(x_n)$$

- $3n - 3$ condiţii de continuitate, derivabilitate şi curbură pentru funcţiile spline cubice vecine:

$$s_i(x_{i+1}) = s_{i+1}(x_{i+1}), \quad i = 0 : n-2$$

$$s_i^{'}(x_{i+1}) = s_{i+1}^{'}(x_{i+1}), \quad i = 0 : n-2$$

$$s_i^{''}(x_{i+1}) = s_{i+1}^{''}(x_{i+1}), \quad i = 0 : n-2$$

- următoarelor 2 condiţii pentru funcţiile spline naturale:

$$s_0^{''}(x_0) = 0$$

$$s_{n-1}^{''}(x_n) = 0$$

respectiv următoarelor 2 condiţi pentru funcţiile spline tensionate:

$$s_0{}^{'}(x_0) = f^{'}(x_0)$$

$$s_{n-1}^{'}(x_n) = f^{'}(x_n)$$

Coeficienţii funcţiilor spline cubice în clasă C^2 sunt daţi de relaţiile:

$$a_i = f(x_i), \quad i = 0 : n$$

$$d_i = \frac{c_{i+1} - c_i}{3h_i}, \quad i = 0 : n-1$$

$$b_i = \frac{a_{i+1} - a_i}{h_i} - \frac{h_i}{3}(2c_i + c_{i+1}), \quad i = 0 : n-1$$

iar coeficienţii $c_i, i = 0 : n-1$ se obţin prin rezolvarea unui sistem tridiagonal de forma:

- pentru funcţii spline naturale

$$\begin{bmatrix} 1 & 0 & 0 & & & 0 \\ h_0 & 2(h_0+h_1) & h_1 & & & 0 \\ & \ddots & \ddots & \ddots & & \\ & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} & \\ 0 & & & 0 & 1 & \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \\ c_n \end{bmatrix} =$$

$$\begin{bmatrix} 0 \\ \frac{3(a_2-a_1)}{h_1} - \frac{3(a_1-a_0)}{h_0} \\ \vdots \\ \frac{3(a_n-a_{n-1})}{h_{n-1}} - \frac{3(a_{n-1}-a_{n-2})}{h_{n-2}} \\ 0 \end{bmatrix}$$

- pentru funcţii spline tensionate

$$\begin{bmatrix} 2h_0 & h_0 & 0 & & & 0 \\ h_0 & 2(h_0+h_1) & h_1 & & & 0 \\ & \ddots & \ddots & \ddots & & \\ & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} & \\ 0 & & & h_{n-1} & 2h_{n-1} & \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \\ c_n \end{bmatrix} =$$

$$\begin{bmatrix} \frac{3(a_1-a_0)}{h_0} - 3f^{'}(x_0) \\ \frac{3(a_2-a_1)}{h_1} - \frac{3(a_1-a_0)}{h_0} \\ \vdots \\ \frac{3(a_n-a_{n-1})}{h_{n-1}} - \frac{3(a_{n-1}-a_{n-2})}{h_{n-2}} \\ 3f^{'}(x_n) - \frac{3(a_n-a_{n-1})}{h_{n-1}} \end{bmatrix}$$

Funcţia spline s_n a fost introdusă pentru a ajuta la calcularea funcţiilor spline $s_i, i = 0 : n-1$.

Curbe Bézier

Fiind dată o mulţime de $n+1$ puncte $P_0, P_1, ..., P_n$ (P_0 si P_n se numesc puncte de interpolare iar P_1 ... P_{n-1} se numesc puncte de control) în plan sau în spaţiu, curba Bézier de grad n determinată de aceste puncte are forma parametrică:

$$B(t) = \sum_{i=0}^{n} P_i B_{i,n}(t), \quad t \in [0,1]$$

unde $B_{i,n}(t)$ se numesc polinoame Bernstein de grad n definite prin relaţia:

$$B_{i,n}(t) = \binom{n}{i}(1-t)^{(n-i)}t^i, \quad i = 0 : n$$

Gradul polinomului care aproximează curba Bézier depinde de numărul punctelor de control. Fiecare punct de control determina forma curbei Bézier în mod particular. Dacă modificăm poziţia unui punct de control şi forma curbei Bézier va fi afectată.

Spre deosebire de funcţiile spline, curba Bézier nu trece prin toate punctele P_0,P_1,...,P_n ci doar prin punctele P_0 si P_n. Orice curbă Bézier începe în punctul P_0 şi se termină în punctul P_n, adică:

$$B(0) = P_0, B(1) = P_n$$

Mai mult, curba Bézier este tangentă segmentelor P_0P_1 si $P_{n-1}P_n$. Orice curbă Bézier este conţinută complet de înfăşurătoarea convexă pe care punctele $P_0, P_1, ..., P_n$ o definesc.

În mod uzual, se folosesc curbe Bézier cubice determinate de 4 puncte P_0, P_1, P_2, P_3, având forma parametrică:

$$B(t) = P_0B_{0,3}(t) + P_1B_{1,3}(t) + P_2B_{2,3}(t) + P_3B_{3,3}(t) = P_0(1-t)^3 + 3P_1t(1-t)^2 + 3P_2t^2(1-t) + P_3t^3 = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}, \quad t \in [0,1]$$

Algoritmul De Casteljau

Într-o abordare directă, calcularea unui punct aflat pe o curbă Bézier se obţine folosind ecuaţia parametrică $B(t) = \sum_{i=0}^{n} P_iB_i^n(t), t \in [0,1]$. Acestă metodă este ineficientă deoarece numerele mici ridicate la puteri mari genereaza erori mari.

Algoritmului De Casteljau este o modalitate eficientă de calculare a unui punct aflat pe o curbă Bézier. Acest algoritm este puţin mai lent insa este numeric stabil şi cu ajutorul său putem obţine detalii despre curba Bézier:

- vectorul tangent într-un punct de pe curba Bézier folosind calculul derivatei;
- divizarea curbei Bézier. Uneori este necesar să separăm o curbă Bézier în alte curbe Bézier.

Algoritmul De Casteljau foloseşte relaţia de recurenţă următoare:

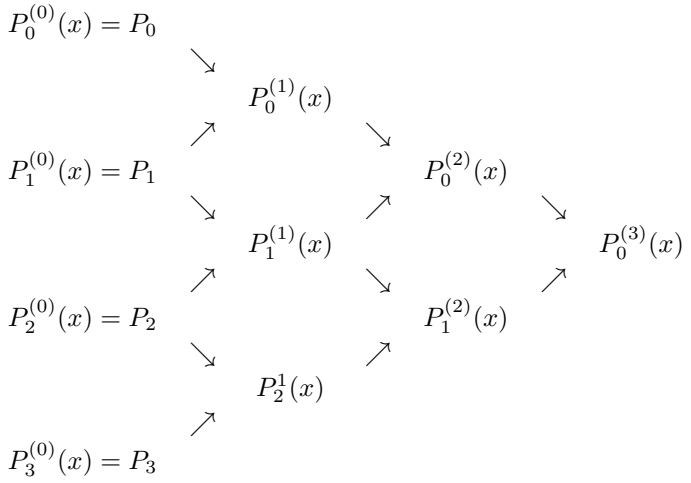
$$P_i^{(0)} = P_i, \quad i = 0 : n$$

$$P_i^{(j)} = P_i^{(j-1)}(1-t_0) + P_{i+1}^{(j-1)}t_0, \quad j = 1 : n, i = 0 : n-j,$$

Fie $P_i^{(0)}$ şi $P_{i+1}^{(0)}$ două puncte succesive şi $P_i^{(1)}$ un punct care împarte segmentul $P_i^{(0)}P_{i+1}^{(0)}$ în raportul $t/(1-t)$. $P_i^{(1)}$ este o combinatie liniara intre punctele $P_i^{(0)}$ şi $P_{i+1}^{(0)}$, adica :

$$P_i^{(1)} = (1-t)P_i^{(0)} + tP_{i+1}^{(0)}$$

Se formează în acest fel poligonul $P_0^{(1)}, P_1^{(1)}, ..., P_{n-1}^{(1)}$. Se aplică relaţia de recurenţă noului poligon obţinându-se poligonul $P_0^{(2)}, P_1^{(2)}, ..., P_{n-2}^{(2)}$. Repetând procesul de n ori, se obţine un singur punct $P_0^{(n)}$. Acest punct obţinut se află pe curba Bézier. Pentru $n = 3$, procesul iterativ de obţinere al punctului $P_0^{(3)}(x)$, este arătat în schema următoare:



Probleme rezolvate

Problema 1

Se consideră funcţia f cunoscută prin valorile:

x	**0.4**	**0.9**	**1.5**	**2.5**
f(x)	**0.9**	**0.5**	**0.2**	**0.1**

Determinaţi valoarea funcţiei în punctul $x = 4$.

Soluţie:

$$P_{00}(4) = f(x_0) = 0.9$$
$$P_{11}(4) = f(x_1) = 0.5$$
$$P_{22}(4) = f(x_2) = 0.2$$
$$P_{33}(4) = f(x_3) = 0.1$$
$$P_{01}(4) = \frac{(4 - x_1)P_{00}(4) + (x_0 - 4)P_{11}(4)}{x_0 - x_1} = \frac{(4 - 0.9)0.9 + (0.4 - 4)0.5}{0.4 - 0.9} = -1.98$$
$$P_{12}(4) = \frac{(4 - x_2)P_{11}(4) + (x_1 - 4)P_{22}(4)}{x_1 - x_2} = \frac{(4 - 1.5)0.5 + (0.9 - 4)0.2}{0.9 - 1.5} = -1.05$$

$$P_{23}(4) = \frac{(4-x_3)P_{22}(4) + (x_2-4)P_{33}(4)}{x_2 - x_3} = \frac{(4-2.5)0.2 + (1.5-4)0.1}{1.5-2.5} = -0.05$$

$$P_{02}(4) = \frac{(4-x_2)P_{01}(4) + (x_0-4)P_{12}(4)}{x_0 - x_2} =$$

$$\frac{(4-1.5)(-1.98) + (0.4-4)(-1.05)}{0.4-1.5} = 1.06$$

$$P_{13}(4) = \frac{(4-x_3)P_{12}(4) + (x_1-4)P_{23}(4)}{x_1 - x_3} =$$

$$\frac{(4-2.5)(-1.05) + (0.9-4)(-0.05)}{0.9-1.5} = 0.89$$

$$P_{03}(4) = \frac{(4-x_3)P_{02}(4) + (x_0-4)P_{13}(4)}{x_0 - x_3} = \frac{(4-2.5)1.06 + (0.4-4)0.88}{0.4-2.5} = 0.76$$

Prin urmare, am obţinut $f(4) \simeq P_{03}(4) = 0.76$

Problema 2

Scrieţi un program OCTAVE care calculează valoarea unei funcţii într-un punct folosind metoda Neville. Programul primeşte ca parametri de intrare: x - suportul interpolării, y - vectorul ordonatelor pentru valorile din suportul interpolării, xi - abscisa în care se calculează valoarea funcţiei. Rezultatul programului este valoarea funcţiei în xi.

Soluţie:

```
function yi = Neville(x, y, xi)
  n = length(x);

  for k = 1 : n-1
    for i = 1 : n-k
      raport = (xi-x(k+i))/(x(i)-x(k+i));
      y(i) = raport*y(i)+(1-raport)*y(i+1);
    endfor
  endfor

  yi = y(1);
endfunction
```

Listing 1: Metoda Neville de interpolare.

Probleme propuse

Problema 1

Scrieţi un program OCTAVE care calculează valoarea unei funcţii într-un punct, ca rezultat al interpolării obţinute folosind funcţii spline în clasă C^1. Programul primeşte ca parametrii de intrare: x - suportul interpolării, y - vectorul ordonatelor pentru valorile din suportul interpolării, dy - vectorul derivatelor pentru valorile din suportul interpolării, xi - abcisa în care se calculează valoarea funcţiei. Rezultatul programului este valoarea funcţiei în xi.

```
function yi = SplineC1(x, y, dy, xi)
```

Listing 2: Interpolare cu funcţii spline în clasă C^1.

Pentru testarea programului, puteţi folosi următoarea secvenţă:

```
x = 0:pi/3:7*pi;
y = sin(x)+cos(3*x);
dy = cos(x)-3*sin(3*x);
xi = 0:pi/15:7*pi;

for i = 1 : length(xi)
  yi(i) = feval('SplineC1', x, y, dy, xi(i));
endfor

plot(x, y, 'g-o', xi, yi, 'r-*');
legend ('data', 'spline C1');
axis([0 23 -2 2.7]);
```

Listing 3: Fişier de testare.

iar rezultatul este:

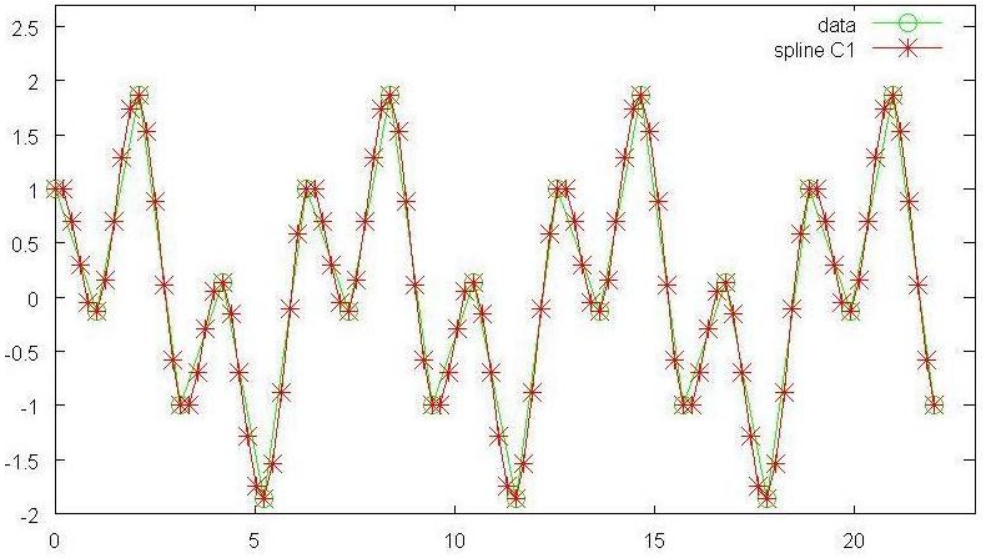


Figure 1: Graficul rezultat pentru fişierul de testare.

Problema 2

Scrieţi un program OCTAVE care calculează valoarea unei funcţii într-un punct, ca rezultat al interpolării obţinute folosind funcţii spline naturale în clasă C^2. Programul primeşte ca parametri de intrare: x - suportul interpolării, y - vectorul ordonatelor pentru valorile din suportul interpolării, xi - abscisa în care se calculează valoarea funcţiei. Rezultatul programului este valoarea funcţiei în xi.

```
function yi = SplineC2natural(x, y, xi)
```

Listing 4: Interpolare cu funcţii spline naturale în clasă C^2.

Pentru testarea programului puteţi folosi următoarea secvenţă:

```
x = 0:pi/3:7*pi;
y = sin(x)+cos(3*x);
xi = 0:pi/15:7*pi;

```

```
5   for i = 1 : length(xi)
6     yi(i) = feval('SplineC2natural', x, y, xi(i));
7   endfor
8
9   plot(x, y, 'g-o', xi, yi, 'r-*');
10  legend ('data', 'spline C2 natural');
11  axis([0 23 -2 2.7]);
```

Listing 5: Fişier de testare.

iar rezultatul este:

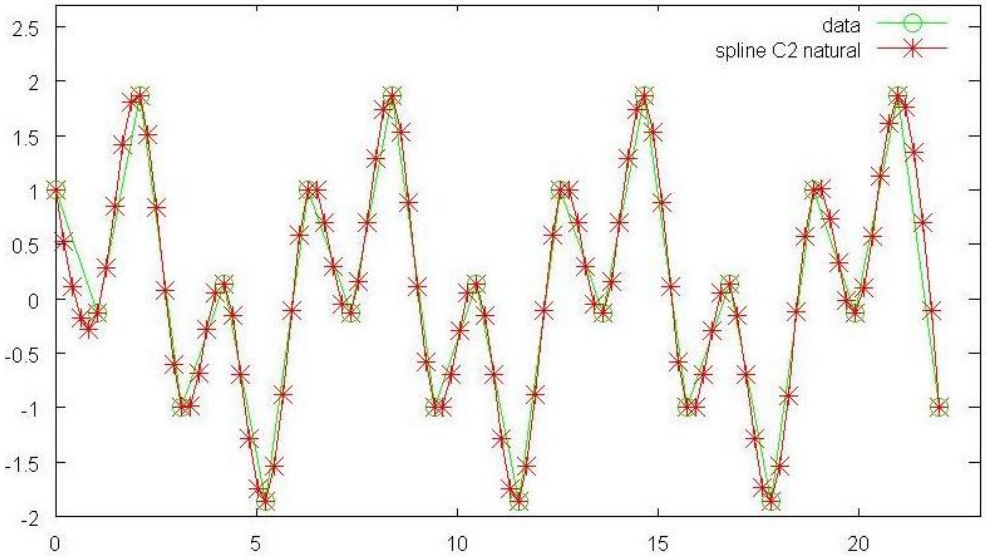


Figure 2: Graficul rezultat pentru fişierul de testare.

Problema 3

Scrieţi un program OCTAVE care calculează valoarea unei funcţii într-un punct ca rezultat al interpolării obţinute folosind funcţii spline tensionate în clasă C^2. Programul primeşte ca parametri de intrare:

x - suportul interpolării,

y - vectorul ordonatelor pentru valorile din suportul interpolării,

$dy1$ - derivata funcţiei în primul punct din suportul interpolării,

dyn - derivata funcţiei în ultimul punct din suportul interpolării,

xi - abscisa în care se calculează valoarea funcţiei.

Rezultatul programului este valoarea funcţiei în xi.

Problema 4

Scrieţi o funcţie OCTAVE care să calculeze un punct aflat pe o curbă Bézier folosind algoritmul De Casteljau. Funcţia primeşte ca parametri de intrare: x - vectorul absciselor punctelor de control, y - vectorul ordonatelor punctelor de control, t - parametru. Rezultatul funcţiei este un punct P aflat pe curba Bézier.

```
function P = Casteljau(x, y, t)

  n = length(x);

  plot(x, y, 'k-o');
  legend ('puncte de control');
  hold on;
  axis([0.5, 4.5, 0.2, 2.1]);

  TODO
endfunction
```

Listing 6: Algoritmului De Casteljau.

Pentru testarea funcţiei anterioare puteţi folosi următoarea secvenţă, care trasează o curbă Bézier folosind algoritmul De Casteljau:

```
  x = [1 0.7 2.7 3.7];
  y = [0.5 2 2 0.5];

  i = 1;
  B_x = zeros();
  B_y = zeros();
  hold off;
```

```
8
9   for t = 0 : 0.05 : 1
10    P = Casteljau(x, y, t);
11    B_x(i) = P(1);
12    B_y(i) = P(2);
13    plot(B_x, B_y, 'b-s');
14    legend ('puncte calculate', 'puncte aflate pe curba Bezier'
      );
15    i++;
16
17    pause(0.5);
18    hold off;
19  endfor
```

Listing 7: Fişier de testare.

iar rezultatul este:

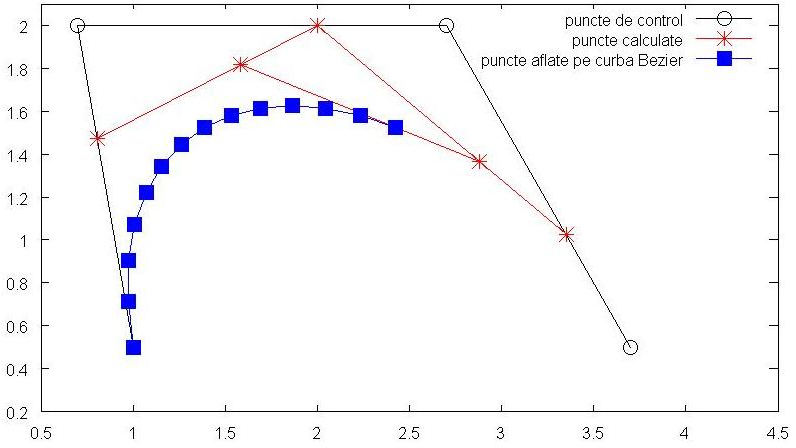


Figure 3: Graficul intermediar rezultat pentru fişierul de testare.

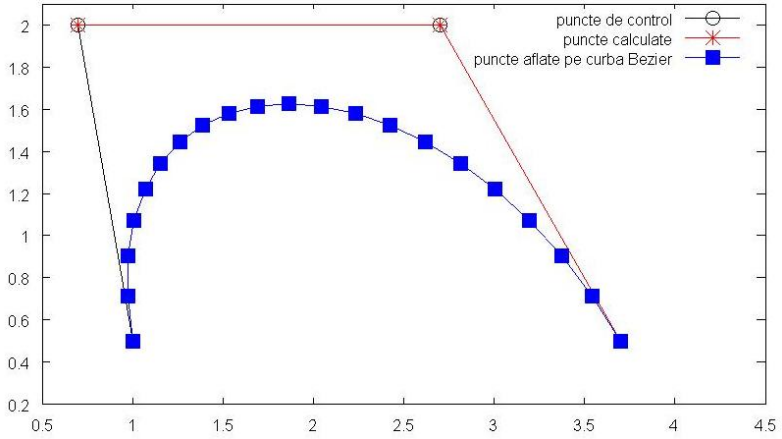


Figure 4: Graficul final rezultat pentru fişierul de testare.

Problema 5

Se consideră funcţia f cunoscută prin valorile:

x	**-3**	**1**	**2**
f(x)	**0**	**3**	**4**
f'(x)	**1**	**2**	**3**

Determinaţi funcţiile spline în clasa C^1 pentru interpolarea lui f.

Problema 6

Se consideră funcţia f cunoscută prin valorile:

x	**-2**	**2**	**3**	**5**
f(x)	**-3**	**1**	**3**	**7**

Determinaţi valoarea funcţiei în punctul $x = 1$.