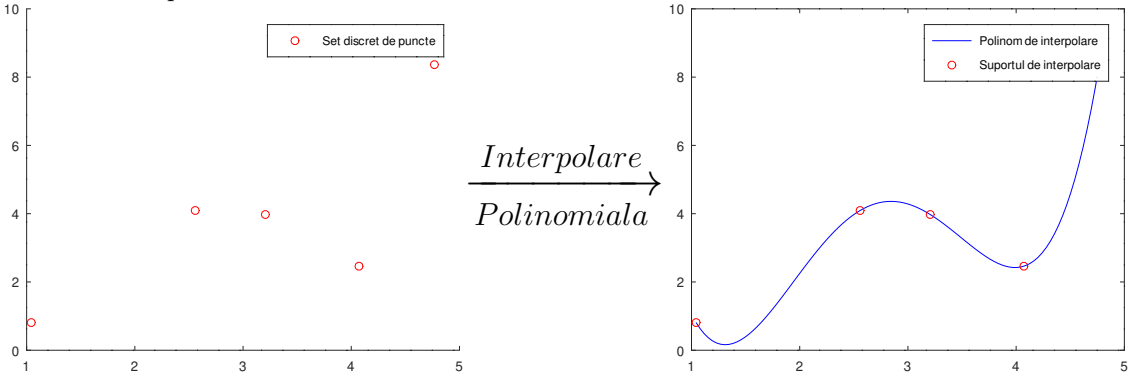
Interpolare

1 Introducere Interpolare Polinomiala

1.1 Notiuni generale

Fie o functie $f : [a, b] \to \mathbb{R}$. Cunoastem valorile functiei intr-un numar redus de puncte $(x_i, f(x_i))$, $i = \overline{0, n}$. Punctele $x_0, x_1, \ldots, x_n$ se numesc *noduri* si formeaza *suportul interpolarii*.

Pornind de la un set discret de puncte, dorim sa construim o functie (polinom) care va trece prin toate aceste puncte.



1.2 Motivatie

- In Computer Science se lucreaza in general cu date discrete.
- Functii complexe ca forma se pot simplifica, alegand cateva puncte $(x_i, f(x_i))$ si construind pe baza lor niste aproximari cu ajutorul unor polinoame.
- Polinoamele sunt usor de evaluat, derivat, integrat, deci interpolarea polinomiala are un avantaj.

1.3 Idee

Sa presupunem ca se cunoaste valoarea functiei $f(x)$ in $(n + 1)$ puncte. Asadar, suportul interpolarii va arata astfel: $S = [(x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_n, f(x_n))]$. Deci, $(n + 1)$ puncte in care stim valorile functiei pe care dorim sa o aproximam cu un polinom de grad (cel mult) n.

La modul general, un polinom de interpolare este o combinatie liniara de functii, de forma: $P_n(x) = a_0 \cdot u_0(x) + a_1 \cdot u_1(x) + \cdots + a_n \cdot u_n(x)$. Functiile $u_0(x), u_1(x), \ldots, u_n(x)$ sunt liniar independente si formeaza *baza interpolarii*. $a_0, a_1, \ldots, a_n$ reprezinta coeficientii care trebuie determinati.

1.4 Conditii de interpolare

Polinomul de interpolare construit $P_n(x)$ trebuie sa coincida cu functia initiala pe suportul interpolarii. Matematic, afirmatia devine: $P_n(x_i) = f(x_i), \forall\, i = \overline{0, n}$.

1.5 Teorema Weierstrass

Fie o functie continua $f : [a, b] \to \mathbb{R}$ si $\epsilon > 0$ (*toleranta*). Atunci $\exists\, P(x) \in \mathbb{R}^n$, a.î. $\boxed{|f(x) - P(x)| < \epsilon}$, $\forall\, x \in [a, b]$.

Asadar, teorema ne asigura ca pentru orice functie continua pe un interval $[a, b]$, exista un polinom cu care putem aproxima functia.

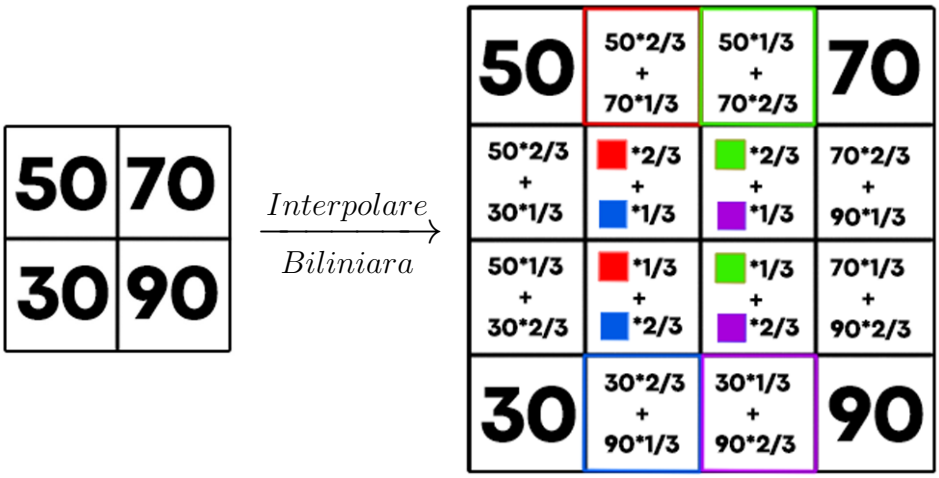
1.6 Tipuri de interpolari abordate

- Interpolare *polinomiala* de tip:
  - **Vandermonde**
  - **Lagrange**
  - **Newton (Diferente Divizate)**
- Interpolare folosind functii *spline*:
  - **Liniare**
  - **Cubice de clasa C^1**
  - **Cubice de clasa C^2**

1.7 Aplicatii din viata reala care folosesc interpolarea

- **Image scaling**

  Exemplu: Daca avem o *imagine* (cu un singur canal de culoare) de dimensiune 2×2 pixeli si dorim sa o marim de 2 ori, ne putem folosi de tehnica de interpolare.

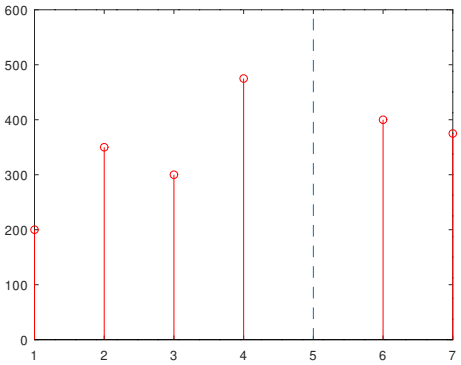


  Pentru acest exemplu s-a aplicat o interpolare biliniara [1] cu coeficientii $\frac{1}{3}$ si $\frac{2}{3}$.

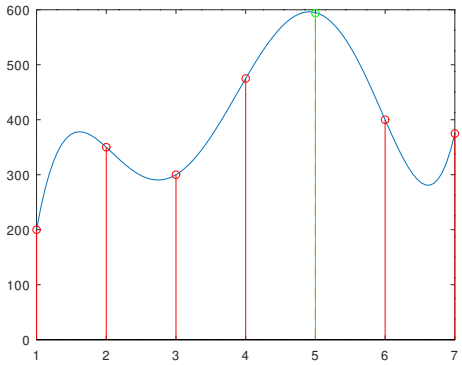
- **Filling missing values**

  Exemplu: O firma isi monitorizeaza vanzarile zilnic, timp de o saptamana, insa in ziua a 5-a a omis masuratoarea. Ce cantitate de produse (aproximativ) a vandut in ziua a 5-a?

  Sa presupunem ca graficul vanzarilor arata astfel:



  Folosind o interpolare polinomiala, gasim un polinom care trece prin cele 6 puncte cunoscute si astfel putem aproxima cantitatea de produse vanduta in ziua a 5-a.



1.8 Pregatirea terenului

Inainte de a discuta despre fiecare tip de interpolare, prezentam cadrul discutiei:

- Pornim de la o functie continua $f : [a, b] \to \mathbb{R}$ careia ii cunoastem valorile in $(n + 1)$ puncte.
- Asadar, suportul interpolarii va fi de forma: $S = [(x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_n, f(x_n))]$.
- Dorim sa gasim un polinom de grad (cel mult) n care sa aproximeze functia $f(x)$.

[1] https://en.wikipedia.org/wiki/Bilinear_interpolation

2 Interpolare Vandermonde

2.1 Baza de interpolare

Interpolarea Vandermonde foloseste ca baza de interpolare, baza polinoamelor: $\{1, x, x^2, \ldots, x^n\}$.

Deci, polinomul de interpolare Vandermonde arata astfel: $\boxed{P_n(x) = a_0 \cdot 1 + a_1 \cdot x + a_2 \cdot x^2 + \cdots + a_n \cdot x^n}$

Pentru a determina coeficientii $a_0, a_1, \ldots, a_n$, punem conditiile de interpolare: $\boxed{P_n(x_i) = f(x_i), \forall\, i = \overline{0, n}}$

$$\begin{cases} Pentru\ \ i = 0 : P_n(x_0) = f(x_0) \iff a_0 + a_1 \cdot x_0 + a_2 \cdot x_0^2 + \cdots + a_n \cdot x_0^n = f(x_0) \\ Pentru\ \ i = 1 : P_n(x_1) = f(x_1) \iff a_0 + a_1 \cdot x_1 + a_2 \cdot x_1^2 + \cdots + a_n \cdot x_1^n = f(x_1) \\ \ldots \\ Pentru\ \ i = n : P_n(x_n) = f(x_n) \iff a_0 + a_1 \cdot x_n + a_2 \cdot x_n^2 + \cdots + a_n \cdot x_n^n = f(x_n) \end{cases}$$

Sistemul de mai sus poate fi scris sub forma matriceala, astfel:

$$\underbrace{\begin{bmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ 1 & x_1 & x_1^2 & \ldots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{bmatrix}}_{A} \cdot \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix}}_{b}$$

Asadar, avem de rezolvat un sistem de forma $A \cdot x = b$.

Matricea A (de tip Vandermonde) este nesingulara [2], deci sistemul are solutie unica, ceea ce conduce la uniticitatea polinomul de interpolare [3].

2.2 Exemplu numeric

Sa consideram cunoscute urmatoarele puncte din plan: $\{(1,1), (2,8), (3,27)\}$.

x	1	2	3
$f(x)$	1	8	27

$\Rightarrow$ Avem urmatoarele noduri: $x_0 = 1;\ x_1 = 2;\ x_2 = 3$.

Suportul interpolarii este $S = [(1,1), (2,8), (3,27)]$.

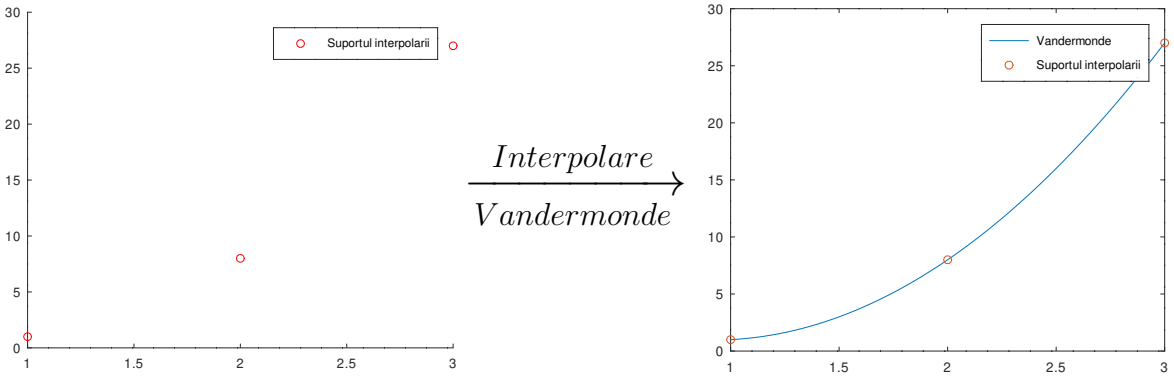
Asadar, avand 3 puncte in suportul interpolarii, cautam un polinom de interpolare de grad 2.

Deci, polinomul cautat are forma: $P_2(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2$.

Punand conditiile de interpolare, vom avea de rezolvat urmatorul SEL (sistem de ecuatii liniare):

$$\begin{bmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ f(x_2) \end{bmatrix} \iff \begin{bmatrix} 1 & 1 & 1^2 \\ 1 & 2 & 2^2 \\ 1 & 3 & 3^2 \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 8 \\ 27 \end{bmatrix} \xrightarrow{(\ldots)} \boxed{\begin{cases} a_0 = 6 \\ a_1 = -11 \\ a_2 = 6 \end{cases}}$$

$\Rightarrow P_2(x) = 6 - 11 \cdot x + 6 \cdot x^2 \Rightarrow \boxed{P_2(x) = 6 \cdot x^2 - 11 \cdot x + 6}$



[2] https://math.stackexchange.com/questions/426932/why-are-vandermonde-matrices-invertible

[3] https://en.wikipedia.org/wiki/Polynomial_interpolation#Uniqueness_of_the_interpolating_polynomial

2.3 Concluzii

Asadar, folosind baza de interpolare Vandermonde $\{1, x, x^2, \ldots, x^n\}$, obtinem sistemul:

$$\underbrace{\begin{bmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ 1 & x_1 & x_1^2 & \ldots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{bmatrix}}_{A} \cdot \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix}}_{b}$$

$\ominus$ Matricea Vandermonde A este o matrice rau conditionata.

$\ominus$ Pentru un numar mare de puncte (n mare), matricea A devine matrice plina mare, ceea ce conduce la numar de conditionare mare.

$\ominus$ Numarul de conditionare mare implica instabilitate numerica.

In concluzie, interpolarea Vandermonde nu este utilizata in practica $\rightarrow$ Cautam alte baze de interpolare.

3 Interpolare Lagrange

3.1 Baza de interpolare

Interpolarea Lagrange foloseste $(n+1)$ polinoame $L_k(x)$, $i = \overline{0,n}$. $L_k(x)$ sunt polinoame de grad (cel mult) n si se numesc *multiplicatori Lagrange*.

Matematic, multiplicatorul Lagrange se defineste astfel: $\boxed{L_k(x) = \prod\limits_{\substack{i=0 \\ i \neq k}}^{n} \frac{x-x_i}{x_k-x_i},\ k = \overline{0,n}}$

Deci, polinomul de interpolare Lagrange se poate scrie: $\boxed{P_n(x) = \sum\limits_{k=0}^{n} L_k(x) \cdot f(x_k)}$

Pentru a intelege mai bine, particularizam problema la 2 puncte in suportul interpolarii ($n = 1$): $\Rightarrow S = [(x_0, f(x_0)), (x_1, f(x_1))]$.

Astfel, multiplicatorii Lagrange vor fi: $L_0(x) = \frac{x-x_1}{x_0-x_1}$ si $L_1(x) = \frac{x-x_0}{x_1-x_0}$.

Tinand cont de forma generala a polinomului de interpolare Lagrange, putem particulariza pe exemplu nostru: $P_1(x) = L_0(x) \cdot f(x_0) + L_1(x) \cdot f(x_1) \iff P_1(x) = \frac{x-x_1}{x_0-x_1} \cdot f(x_0) + \frac{x-x_0}{x_1-x_0} \cdot f(x_1)$.

Observatie: Multiplicatorul Lagrange la forma generala $L_k(x_i)$ poate fi privit ca un intrerupator, deoarece pentru $i = k$, acesta este 0 (**OFF**), iar pentru $i \neq k$ este 1 (**ON**).

De exemplu:

- $L_0(x_i) \overset{\text{def}}{=} \prod\limits_{\substack{i=0 \\ i \neq 0}}^{n} \frac{x-x_i}{x_0-x_i} \iff L_0(x_i) = \frac{x-x_1}{x_0-x_1} \cdot \frac{x-x_2}{x_0-x_2} \ldots \frac{x-x_n}{x_0-x_n} \iff L_0(x_i) = \begin{cases} 0, & i = 0 \\ 1, & i \neq 0 \end{cases}$
- Procedand analog, obtinem generalizarea: $\boxed{L_k(x_i) = \begin{cases} 0, & i = k \\ 1, & i \neq k \end{cases}}$ **(*)**

Un alt aspect important este respectarea conditiilor de interpolare. Este usor de observat faptul ca polinomul de interpolare Lagrange respecta conditiile de interpolare:

Stim ca, la modul general, polinomul de interpolare Lagrange este: $P_n(x) \overset{\text{def}}{=} \sum\limits_{k=0}^{n} L_k(x) \cdot f(x_k)$

- Pentru $x = x_0$: $P_n(x_0) = L_0(x_0) \cdot f(x_0) + L_1(x_0) \cdot f(x_1) + \cdots + L_n(x_0) \cdot f(x_n)$
  Conform **(*)**: $\mathbf{L_0(x_0) = 1}$; $L_1(x_0) = 0$; $L_2(x_0) = 0$; $L_3(x_0) = 0$; $\ldots$; $L_n(x_0) = 0$
  $\Longrightarrow \boxed{P_n(x_0) = f(x_0)}$
- Pentru $x = x_1$: $P_n(x_1) = L_0(x_1) \cdot f(x_0) + L_1(x_1) \cdot f(x_1) + \cdots + L_n(x_1) \cdot f(x_n)$
  Conform **(*)**: $L_0(x_1) = 0$; $\mathbf{L_1(x_1) = 1}$; $L_2(x_1) = 0$; $L_3(x_1) = 0$; $\ldots$; $L_n(x_1) = 0$
  $\Longrightarrow \boxed{P_n(x_1) = f(x_1)}$

- ...
- Pentru $x = x_n$: $P_n(x_n) = L_0(x_n) \cdot f(x_0) + L_1(x_n) \cdot f(x_1) + \cdots + L_n(x_n) \cdot f(x_n)$
  Conform **(*)**: $L_0(x_n) = 0$; $L_1(x_n) = 0$; $L_2(x_n) = 0$; $L_3(x_n) = 0$; ...; $\mathbf{L_n(x_n) = 1}$
  $\implies \boxed{P_n(x_n) = f(x_n)}$

Asadar, $\boxed{P_n(x_i) = f(x_i), \forall i = \overline{0, n}} \implies$ Conditiile de interpolare se verifica.

3.2 Exemplu numeric

Sa consideram cunoscute urmatoarele puncte din plan: $\{(1, 1), (2, 8), (3, 27)\}$.

x	1	2	3
$f(x)$	1	8	27

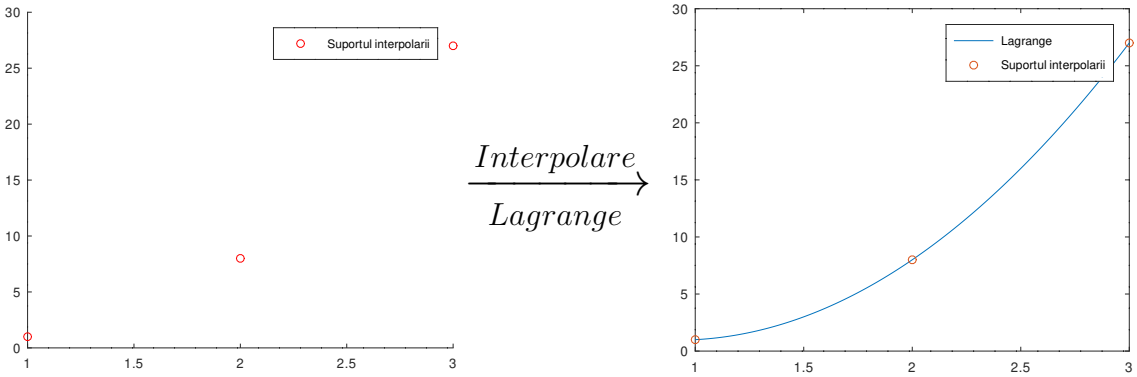
$\Rightarrow$ Avem urmatoarele noduri: $x_0 = 1$; $x_1 = 2$; $x_2 = 3$.
Suportul interpolarii este $S = [(1, 1), (2, 8), (3, 27)]$.
Asadar, avand 3 puncte in suportul interpolarii, cautam un polinom de interpolare de grad 2.

Tinand cont de forma generala a polinomului de interpolare Lagrange, putem particulariza pe exemplul nostru, astfel: $\boxed{P_2(x) = L_0(x) \cdot f(x_0) + L_1(x) \cdot f(x_1) + L_2(x) \cdot f(x_2)}$
Scriem desfasurat fiecare multiplicator Lagrange, tinand cont de valorile efective ale nodurilor:

- $L_0(x) \overset{\text{def}}{=} \frac{(x-x_1)\cdot(x-x_2)}{(x_0-x_1)\cdot(x_0-x_2)} = \frac{(x-2)\cdot(x-3)}{(1-2)\cdot(1-3)} = \frac{(x-2)\cdot(x-3)}{2}$
- $L_1(x) \overset{\text{def}}{=} \frac{(x-x_0)\cdot(x-x_2)}{(x_1-x_0)\cdot(x_1-x_2)} = \frac{(x-1)\cdot(x-3)}{(2-1)\cdot(2-3)} = -(x-1)\cdot(x-3)$
- $L_2(x) \overset{\text{def}}{=} \frac{(x-x_0)\cdot(x-x_1)}{(x_2-x_0)\cdot(x_2-x_1)} = \frac{(x-1)\cdot(x-2)}{(3-1)\cdot(3-2)} = \frac{(x-1)\cdot(x-2)}{2}$

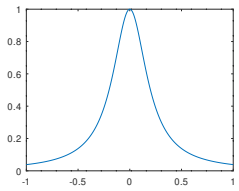
$\implies P_2(x) = 1 \cdot \frac{(x-2)\cdot(x-3)}{2} + 8 \cdot -(x-1)\cdot(x-3) + 27 \cdot \frac{(x-1)\cdot(x-2)}{2} \implies$

$\implies \boxed{P_2(x) = \frac{1}{2} \cdot (x-2)(x-3) - 8 \cdot (x-1)(x-3) + \frac{27}{2} \cdot (x-1)(x-2)}$



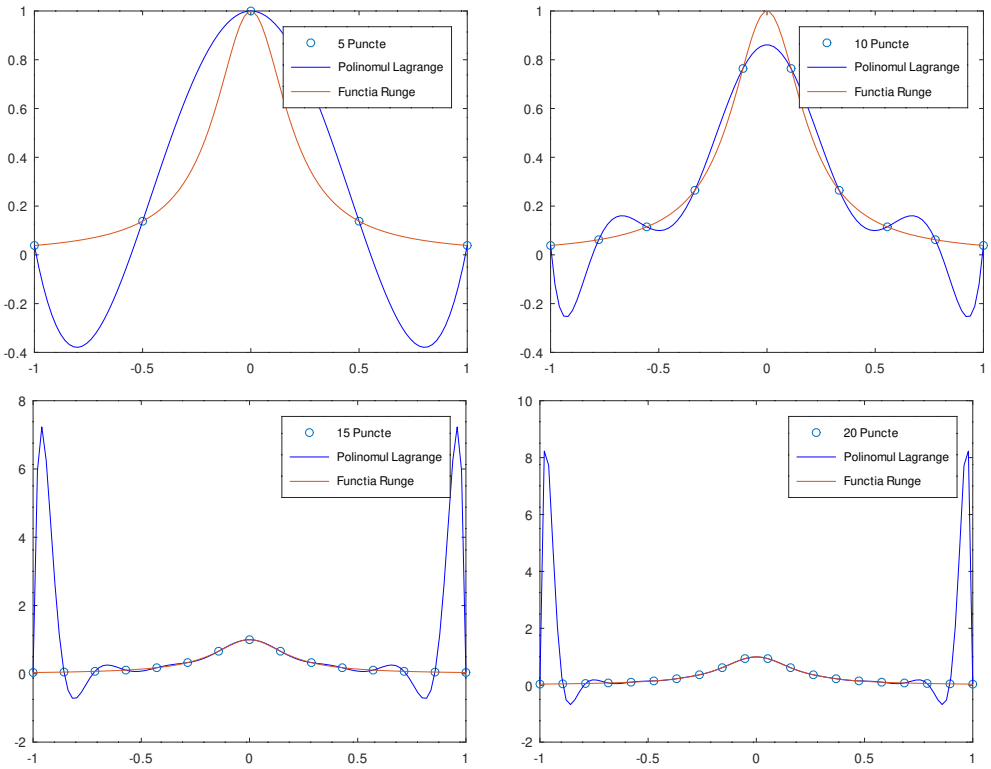
3.3 Fenomenul Runge

Functia Runge[4] se defineste astfel: $f : [-1, 1] \to \mathbb{R}$, $f(x) = \frac{1}{1+25\cdot x^2}$



Aplicand tehnica de interpolare Lagrange si considerand pe rand 5, 10, 15 si apoi 20 puncte (echidistante) in suportul de interpolare, obtinem urmatoarele polinoame de interpolare:

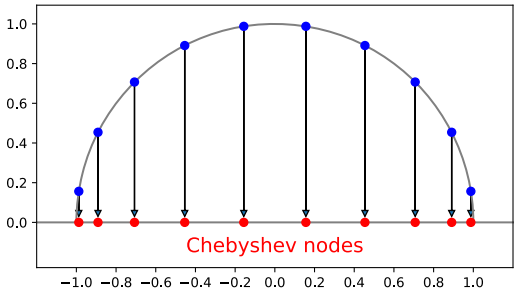
[4] https://en.wikipedia.org/wiki/Runge%27s_phenomenon



Asadar, cu cat numarul de puncte din suportul interpolarii creste ($n \uparrow$), gradul polinomului de interpolare creste si polinomul oscileaza in capete, adica eroarea de interpolare creste. Acest fenomen este cunoscut sub denumirea de *fenomen Runge*. Animatie

Observatie: Daca punctele din suportul interpolarii ar fi fost alese la intamplare (sa nu fie echidistante), polinomul ar fi oscilat si mai mult in capete.

O solutie de atenuare a oscilatiilor este schimbarea modului de distribuire a nodurilor din suportul de interpolare. Un exemplu clasic este setul de noduri Chebyshev [5] pentru care eroarea maximala de aproximare a functiei Runge se diminueaza odata cu cresterea gradului polinomului de interpolare.



Discutia in amanunt a atenuarii oscilatiilor polinomului de interpolare Lagrange nu este de interes in acest moment.[6]

3.4 Concluzii

⊕ Metoda Lagrange este mai robusta decat Vandermonde.

⊕ Folosita in cadrul academic.

⊖ Instabila numeric pentru calculul polinomului.

⊖ Polinomul de interpolare oscileaza in capete.

[5] https://en.wikipedia.org/wiki/Chebyshev_nodes

[6] https://en.wikipedia.org/wiki/Runge%27s_phenomenon#Mitigations_to_the_problem

4 Interpolare Newton (Diferente Divizate)

4.1 Baza de interpolare

- La interpolarea Newton, scriem polinomul de interpolare in functie de $\boxed{(x - x_i)}$ in loc de x.
- **Deducerea formulelor:**
  - Interpolare polinomiala **liniara** → Suportul de interpolare este format din 2 puncte: $S = [(x_0, f(x_0)), (x_1, f(x_1))]$.
    Polinomul de interpolare va avea urmatoarea forma:
    $P_1(x) = a_0 + a_1 \cdot (x - x_0)$, unde a_0 si a_1 sunt cei 2 coeficienti care trebuie determinati.
    Punem conditiile de interpolare: $\begin{cases} P_1(x_0) = f(x_0) \\ P_1(x_1) = f(x_1) \end{cases} \iff \begin{cases} a_0 + a_1 \cdot (x_0 - x_0) = f(x_0) \\ a_0 + a_1 \cdot (x_1 - x_0) = f(x_1) \end{cases}$
    $\iff \begin{cases} a_0 = f(x_0) \overset{\text{not}}{=} F_0[x_0] \\ a_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0} \overset{\text{not}}{=} F_1[x_0, x_1] \end{cases} \implies \boxed{P_1(x) = F_0[x_0] + F_1[x_0, x_1] \cdot (x - x_0)}$
  - Interpolare polinomiala **patratica** → Suportul de interpolare este format din 3 puncte: $S = [(x_0, f(x_0)), (x_1, f(x_1)), (x_2, f(x_2))]$.
    Polinomul de interpolare va avea urmatoarea forma:
    $P_2(x) = a_0 + a_1 \cdot (x - x_0) + a_2 \cdot (x - x_0) \cdot (x - x_1)$, unde a_0, a_1 si a_2 sunt cei 3 coeficienti care trebuie determinati.
    Punem conditiile de interpolare: $\begin{cases} P_2(x_0) = f(x_0) \\ P_2(x_1) = f(x_1) \\ P_2(x_2) = f(x_2 \end{cases}$
    $\iff \begin{cases} a_0 + a_1 \cdot (x_0 - x_0) + a_2 \cdot (x_0 - x_0) \cdot (x_0 - x_1) = f(x_0) \\ a_0 + a_1 \cdot (x_1 - x_0) + a_2 \cdot (x_1 - x_0) \cdot (x_1 - x_1) = f(x_1) \\ a_0 + a_1 \cdot (x_2 - x_0) + a_2 \cdot (x_2 - x_0) \cdot (x_2 - x_1) = f(x_2) \end{cases}$
    $\iff \begin{cases} a_0 = f(x_0) \overset{\text{not}}{=} F_0[x_0] \\ a_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0} \overset{\text{not}}{=} F_1[x_0, x_1] \\ a_2 = \frac{\frac{f(x_2) - f(x_1)}{x_2 - x_1} - \frac{f(x_1) - f(x_0)}{x_1 - x_0}}{x_2 - x_0} = \frac{F_1[x_1, x_2] - F_1[x_0, x_1]}{x_2 - x_0} \overset{\text{not}}{=} F_2[x_0, x_1, x_2] \end{cases}$
    $\implies \boxed{P_2(x) = F_0[x_0] + F_1[x_0, x_1] \cdot (x - x_0) + F_2[x_0, x_1, x_2] \cdot (x - x_0) \cdot (x - x_1)}$
- **Generalizare:**
  - Baza de interpolare Newton:
    $\{1,\ (x - x_0),\ (x - x_0) \cdot (x - x_1), \ldots,\ (x - x_0) \cdot (x - x_1) \cdot \ \ldots \ \cdot (x - x_{n-1})\}$
  - Polinomul de interpolare Newton va fi de forma:
    $\boxed{P_n(x) = a_0 + a_1 \cdot (x - x_0) + a_2 \cdot (x - x_0) \cdot (x - x_1) + \ \ldots \ + a_n \cdot (x - x_0) \cdot (x - x_1) \ \ldots \ \cdot (x - x_{n-1})}$
  - **Diferentele divizate**[7] reprezinta un algoritm recursiv pentru a calcula coeficientii unui polinom de interpolare in forma Newton.

$$
\begin{array}{ccccc}
F_0[x_0] = f(x_0) & & & & \\
\searrow & & & & \\
 & F_1[x_0, x_1] = \frac{F_0[x_0] - F_0[x_1]}{x_0 - x_1} & & & \\
\nearrow & & \searrow & & \\
F_0[x_1] = f(x_1) & & F_2[x_0, x_1, x_2] = \frac{F_1[x_0, x_1] - F_1[x_1, x_2]}{x_0 - x_2} & & \\
\searrow & \nearrow & & \searrow & \\
 & F_1[x_1, x_2] = \frac{F_0[x_1] - F_0[x_2]}{x_1 - x_2} & & & F_3[x_0, x_1, x_2, x_3] = \frac{F_2[x_0, x_1, x_2] - F_2[x_1, x_2, x_3]}{x_0 - x_3} \\
\nearrow & & \searrow & \nearrow & \\
F_0[x_2] = f(x_2) & & F_2[x_1, x_2, x_3] = \frac{F_1[x_1, x_2] - F_1[x_2, x_3]}{x_1 - x_3} & & \\
\searrow & \nearrow & & & \\
 & F_1[x_2, x_3] = \frac{F_0[x_2] - F_0[x_3]}{x_2 - x_3} & & & \\
\nearrow & & & & \\
F_0[x_3] = f(x_3) & & & &
\end{array}
$$

[7] https://www.geeksforgeeks.org/newtons-divided-difference-interpolation-formula/

4.2 Exemplu numeric

Sa consideram cunoscute urmatoarele puncte din plan: $\{(1,1),(2,8),(3,27)\}$:

x	1	2	3
$f(x)$	1	8	27

$\Rightarrow$ Avem urmatoarele noduri: $x_0 = 1;\ x_1 = 2;\ x_2 = 3$.
Suportul interpolarii este $S = [(1,1),(2,8),(3,27)]$.
Asadar, avand 3 puncte in suportul interpolarii, cautam un polinom de interpolare de grad 2.

Tinand cont de forma generala a polinomului de interpolare Newton, putem particulariza pe exemplul nostru, astfel: $\boxed{P_2(x) = a_0 + a_1 \cdot (x - x_0) + a_2 \cdot (x - x_0) \cdot (x - x_1)}$, unde:

$$\begin{cases} a_0 = F_0[x_0] \\ a_1 = F_1[x_0, x_1] \\ a_2 = F_2[x_0, x_1, x_2] \end{cases} ; \begin{cases} x_0 = 1 \\ x_1 = 2 \\ x_2 = 3 \end{cases}$$

$$F_0[x_0] = f(x_0) = \boxed{1} \searrow$$

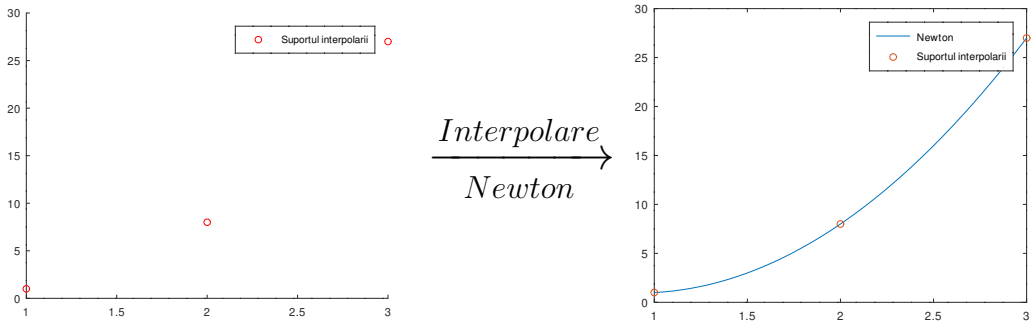
$$\nearrow \quad F_1[x_0, x_1] = \frac{F_0[x_0] - F_0[x_1]}{x_0 - x_1} = \frac{1-8}{1-2} = \boxed{7} \quad \searrow$$

$$F_0[x_1] = f(x_1) = 8 \qquad\qquad F_2[x_0, x_1, x_2] = \frac{F_1[x_0, x_1] - F_1[x_1, x_2]}{x_0 - x_2} = \frac{7-19}{1-3} = \boxed{6}$$

$$\searrow \quad F_1[x_1, x_2] = \frac{F_0[x_1] - F_0[x_2]}{x_0 - x_2} = \frac{8-27}{2-3} = 19 \quad \nearrow$$

$$F_0[x_2] = f(x_2) = 27 \nearrow$$

$$\Longrightarrow \boxed{P_2(x) = 1 + 7 \cdot (x-1) + 6 \cdot (x-1) \cdot (x-2)}$$



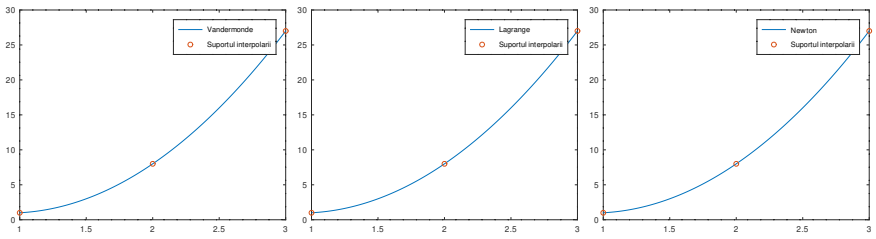
4.3 Concluzii

$\oplus$ Putem adauga incremental puncte noi in suportul interpolarii si avem de calculat doar un coeficient nou $\Rightarrow$ Foarte rapid.

$\ominus$ Echivalent cu Lagrange in rest.

5 Concluzii Interpolarea Polinomiala

5.1 Comparatii intre tipurile de interpolare polinomiala



Asadar, cele 3 metode de interpolare exemplificate anterior (Vandermonde, Lagrange si Newton) produc acelasi polinom de interpolare, dar prin tehnici diferite.

5.2 Ce putem imbunatati?

- Asa cum am vazut in exemplele anterioare, gradul polinomului de interpolare creste odata cu cresterea numarului de puncte din suportul interpolarii.
- Acest lucru conduce la marirea erorii de aproximare si oscilatii ale polinomului de interpolare.
- Pentru a mentine gradul polinomului de interpolare cat mai mic, introducem in discutie *functiile spline* (functii de grad mic, definite pe subintervale).

6 Introducere Interpolare Cu Functii Spline

6.1 Notiuni generale

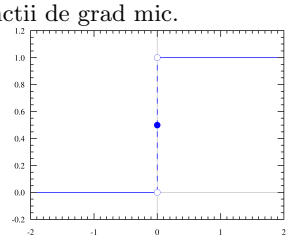
In continuare, consideram o functie $f : [a, b] \to \mathbb{R}$. Cunoastem **valorile functiei** si **valorile derivatei** intr-un numar redus de puncte $(x_i, f(x_i))$, $i = \overline{0, n}$. Dorim sa interpolam aceste puncte cu o functie spline.

Functiile spline sunt functii definite pe subintervale: $S(x) = \begin{cases} S_0(x), \ x \in [x_0, x_1] \\ S_1(x), \ x \in [x_1, x_2] \\ \dots \\ S_i(x), \ x \in [x_i, x_{i+1}] \\ \dots \\ S_{n-1}(x), \ x \in [x_{n-1}, x_n] \end{cases}$

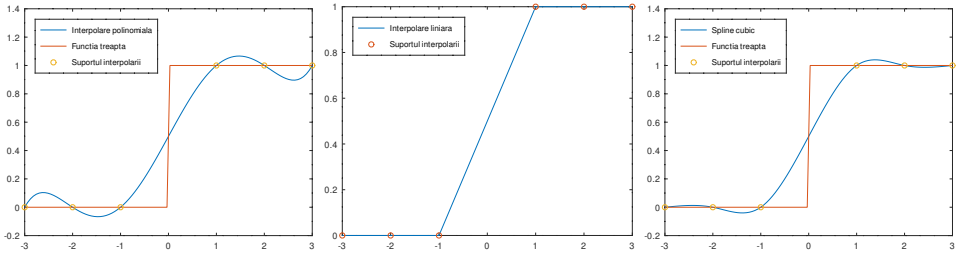
6.2 Motivatie

Dorim sa eliminam oscilatia polinomului de interpolare, folosind functii de grad mic.

Sa consideram ca exemplu functia Heaviside (functia treapta):
Vom interpola pe rand:



- *Polinomial*
- Folosind functii *spline liniare*
- Folosind functii *spline cubice*



Daca folosim spline liniar, vom avea 5 polinoame de gradul I, iar daca folosim spline cubic, vom avea 5 polinoame de gradul III (cate unul intre fiecare 2 puncte consecutive).

Asadar, putem folosi polinoame de grad I sau III pe subintervale, in loc sa interpolam cu un polinom de grad mare pe tot intervalul. Practic, impartim problema in probleme mai mici.

7 Interpolare cu functii spline liniare

7.1 Modul de determinare

Pornim de la forma generala a functiei spline $S(x) = \begin{cases} S_0(x), \ x \in [x_0, x_1] \\ \dots \\ S_i(x), \ x \in [x_i, x_{i+1}] \\ \dots \\ S_{n-1}(x), \ x \in [x_{n-1}, x_n] \end{cases}$ **(**)**

Alegem $S_i(x)$ de forma $a_i \cdot x + b_i$, deci o functie liniara.
Asadar, pornim de la forma $S_i(x) = a_i \cdot x + b_i, \ i = 0 : n-1$
Pentru a determina constantele a_i si b_i, avem nevoie de $(2 \cdot n)$ ecuatii:

Conditii de **interpolare**:
$S_i(x_i) = f(x_i), \ i = 0 : n-1$
$S_{n-1}(x_n) = f(x_n)$

Conditii de **racordare (de continuitate)**:
$S_i(x_{i+1}) = S_{i+1}(x_{i+1}), \ i = 0 : n-2$

$$\xRightarrow{(\dots)} \boxed{\begin{cases} a_i = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} \\ b_i = \frac{x_{i+1} \cdot f(x_i) - x_i \cdot f(x_{i+1})}{x_{i+1} - x_i} \end{cases}, \ i = 0 : n-1}$$

7.2 Exemplu numeric

Sa consideram cunoscute urmatoarele puncte din plan: $\{(1,1),(2,2),(3,0),(4,1)\}$:

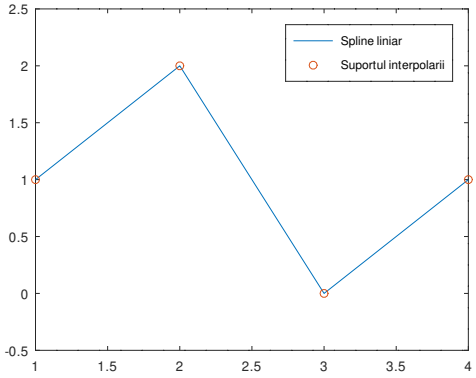
x	1	2	3	4
$f(x)$	1	2	0	1

$\Rightarrow$ Avem urmatoarele noduri: $x_0 = 1;\ x_1 = 2;\ x_2 = 3;\ x_3 = 4.$

Suportul interpolarii este $S = [(1,1),(2,2),(3,0),(4,1)]$.

Particularizand **(**)**, obtinem spline-ul: $S(x) = \begin{cases} S_0(x), & x \in [1,2] \\ S_1(x), & x \in [2,3] \\ S_2(x), & x \in [3,4] \end{cases}$, unde $S_i(x) = a_i \cdot x + b_i$

- Pentru $i = 0$: $\begin{cases} a_0 = \frac{f(x_1)-f(x_0)}{x_1-x_0} = \frac{2-1}{2-1} = 1 \\ b_0 = \frac{x_1 \cdot f(x_0) - x_0 \cdot f(x_1)}{x_1-x_0} = \frac{2\cdot 1 - 1\cdot 2}{2-1} = 0 \end{cases}$ $\Rightarrow S_0(x) = x$
- Pentru $i = 1$: $\begin{cases} a_1 = \frac{f(x_2)-f(x_1)}{x_2-x_1} = \frac{0-2}{3-2} = -2 \\ b_1 = \frac{x_2 \cdot f(x_1) - x_1 \cdot f(x_2)}{x_2-x_1} = \frac{3\cdot 2 - 2\cdot 0}{3-2} = 6 \end{cases}$ $\Rightarrow S_1(x) = 6x - 2 \Longrightarrow$ $\boxed{S(x) = \begin{cases} x, & x \in [1,2] \\ 6 \cdot x - 2, & x \in [2,3] \\ x - 3, & x \in [3,4] \end{cases}}$
- Pentru $i = 2$: $\begin{cases} a_2 = \frac{f(x_3)-f(x_2)}{x_3-x_2} = \frac{1-0}{4-3} = 1 \\ b_2 = \frac{x_3 \cdot f(x_2) - x_2 \cdot f(x_3)}{x_3-x_2} = \frac{4\cdot 0 - 3\cdot 1}{4-3} = -3 \end{cases}$ $\Rightarrow S_2(x) = x - 3$



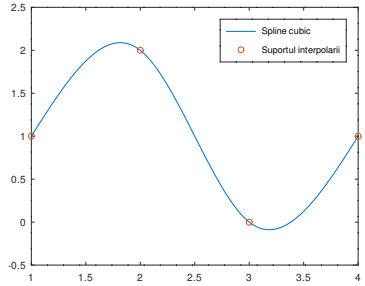
7.3 Concluzii

$\oplus$ Dispare fenomenul de oscilatie prin mentinerea gradului polinomului de interpolare mic.

$\ominus$ Derivata de ordin I nu este o functie continua $\Longleftrightarrow$ Nu se face o trecere neteda in capetele subintervalelor.

8 Interpolare cu functii spline cubice C1

8.1 De ce spline-uri cubice?



Tinand cont ca am discutat despre spline-uri *liniare* si acum discutia se indreapta catre spline-uri *cubice*, intrebarea fireasca este *"De ce cubice si nu patratice?"*. La nivel simplist, raspunsul este simplu: Folosim spline-uri cubice in locul celor patratice, deoarce:

- 3 este cel mai mic grad al unui polinom care permite o *inflexiune*.
- 3 nu este un grad foarte mare $\Rightarrow$ Polinomul de interpolare *nu oscileaza* in capete.

8.2 Modul de determinare

La modul general, o functie spline cubica scrisa in functie de $x - x_i$, arata astfel:

$$\boxed{S_i(x) = a_i + b_i \cdot (x - x_i) + c_i \cdot (x - x_i)^2 + d_i \cdot (x - x_i)^3,\ i = 0 : n-1}$$

Se pune problema determinarii coeficientilor a_i, b_i, c_i, d_i. Spline-urile de clasa C^1 necesita cunoasterea derivatei in suportul interpolarii. Asadar, pentru a determina coeficientii, punem conditii de *interpolare* (de tip Hermite) si conditii de *racordare*:

- Conditii de *interpolare* de tip Hermite:
  - Fiecare spline trece prin punctul sau de interpolare: $\boxed{\begin{cases} S_i(x) = f(x_i),\ i = 0 : n-1 \\ S_{n-1}(x_n) = f(x_n) \end{cases}}$

    $\Rightarrow (n+1)$ ecuatii
  - Fiecare spline are aceeasi panta cu functia pe care o aproximeaza: $\boxed{\begin{cases} S_i'(x) = f'(x_i),\ i = 0 : n-1 \\ S_{n-1}'(x_n) = f'(x_n) \end{cases}}$

    $\Rightarrow (n+1)$ ecuatii
- Conditii de *racordare*:
  - Racordam functiile (Unde se termina S_i, incepe S_{i+1}): $\boxed{S_i(x_{i+1}) = S_{i+1}(x_{i+1}),\ i = 0 : n-2}$

    $\Rightarrow (n-1)$ ecuatii
  - Spline-urile au aceeasi panta in punctele de contact: $\boxed{S_i'(x_{i+1}) = S_{i+1}'(x_{i+1}),\ i = 0 : n-2}$

    $\Rightarrow (n-1)$ ecuatii

Din conditiile de mai sus, obtinem $(4 \cdot n)$ ecuatii.
Astfel, putem sa determinam cele $(4 \cdot n)$ necunoscute $(a_i, b_i, c_i, d_i,\ i = 0 : n-1)$.

8.3 Forma parametrica

Pentru a ne apropia de o rezolvare *computationala*, introducem forma parametrica a functiei spline cubice.

Pornim de la forma generala a spline-ului cubic: $S_i(x) = a_i + b_i \cdot (x - x_i) + c_i \cdot (x - x_i)^2 + d_i \cdot (x - x_i)^3$ si notam $x - x_i = h_i \cdot t \iff t = \frac{x - x_i}{h_i}$, unde $h_i = x_{i+1} - x_i$ (lungimea intervalului $[x_i, x_{i+1}]$) si $t \in [0, 1]$.
$\Rightarrow S_i(t) = a_i + b_i \cdot h_i \cdot t + c_i \cdot h_i^2 \cdot t^2 + d_i \cdot h_i^3 \cdot t^3$.

Asadar, la modul general, forma parametrica a unei functii spline cubice, arata astfel:

$\boxed{S_i(t) = a_i + b_i \cdot h_i \cdot t + c_i \cdot h_i^2 \cdot t^2 + d_i \cdot h_i^3 \cdot t^3}$, unde: $\begin{cases} t \in [0, 1] \\ i = 0 : n-1 \\ h_i = x_{i+1} - x_i \text{ (lungimea intervalului } [x_i, x_{i+1}]) \end{cases}$

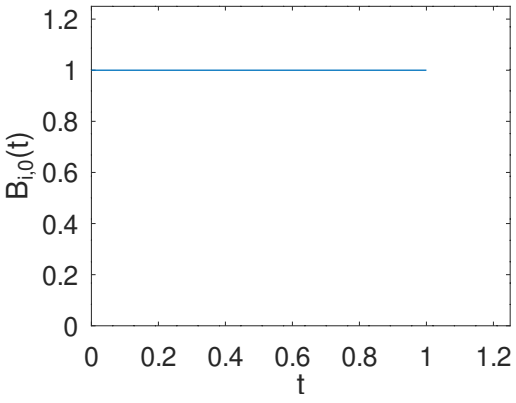
8.4 Polinoamele Bernstein

Introducem in cadrul discutiei **polinoamele Bernstein**[8] de grad n, deoarece sunt utilizate pentru a eficientiza procesul de calculare al coeficientilor functiilor spline cubice in forma parametrica.

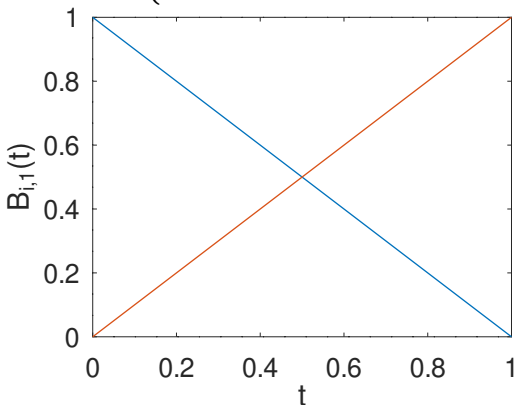
$\boxed{B_{i,n}(t) \overset{\text{def}}{=} C_n^i \cdot (1-t)^{n-i} \cdot t^i}$, $t \in [0, 1],\ i = \overline{0, n}$.

Pentru a intelege mai bine, exemplificam primele polinoame:

Grad 0: $B_{0,0}(t) = C_0^0 \cdot (1-t)^{0-0} \cdot t^0 = 1$



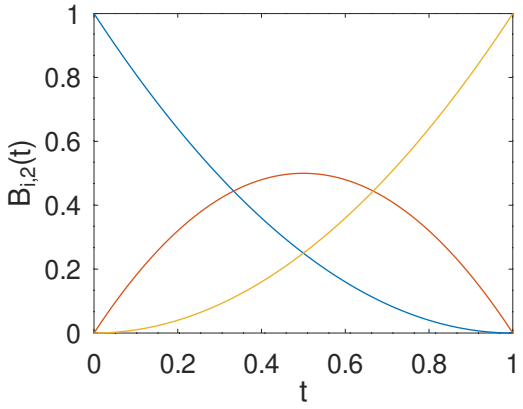
Grad 1: $\begin{cases} B_{0,1}(t) = C_1^0 \cdot (1-t)^{1-0} \cdot t^0 = 1 - t \\ B_{1,1}(t) = C_1^1 \cdot (1-t)^{1-1} \cdot t^1 = t \end{cases}$

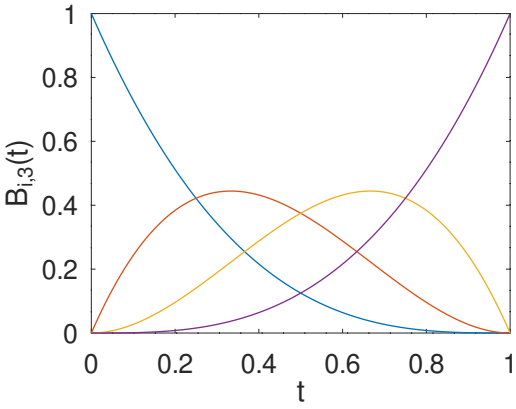


[8] https://mathworld.wolfram.com/BernsteinPolynomial.html

Grad 2: $\begin{cases} B_{0,2}(t) = C_2^0 \cdot (1-t)^{2-0} \cdot t^0 = (1-t)^2 \\ B_{1,2}(t) = C_2^1 \cdot (1-t)^{2-1} \cdot t^1 = 2 \cdot (1-t) \cdot t \\ B_{2,2}(t) = C_2^2 \cdot (1-t)^{2-2} \cdot t^2 = t^2 \end{cases}$

Grad 3: $\begin{cases} B_{0,3}(t) = C_3^0 \cdot (1-t)^{3-0} \cdot t^0 = (1-t)^3 \\ B_{1,3}(t) = C_3^1 \cdot (1-t)^{3-1} \cdot t^1 = 3 \cdot (1-t)^2 \cdot t \\ B_{2,3}(t) = C_3^2 \cdot (1-t)^{3-2} \cdot t^2 = 3 \cdot (1-t) \cdot t^2 \\ B_{3,3}(t) = C_3^3 \cdot (1-t)^{3-3} \cdot t^3 = t^3 \end{cases}$





Folosind polinoamele Bernstein de grad $n = 3$, obtinem forma parametrica a spline-ului cu care vom lucra de acum inainte:

$$\boxed{S_i(t) = a_i' \cdot (1-t)^3 + b_i' \cdot 3 \cdot t \cdot (1-t)^2 + c_i' \cdot 3 \cdot t^2 \cdot (1-t) + d_i' \cdot t^3,\ i = 0 : n-1,\ t \in [0,1]}$$

$$\xrightarrow[\text{prezentate anterior}]{\text{Se impun conditiile}} \boxed{\begin{cases} a_i' = f(x_i),\ i = 0 : n-1 \\ d_i' = f(x_{i+1}),\ i = 0 : n-1 \\ c_i' = f(x_{i+1}) - \frac{h_i}{3} \cdot f'(x_{i+1}),\ i = 0 : n-1 \\ b_i' = f(x_i) + \frac{h_i}{3} \cdot f'(x_i),\ i = 0 : n-1 \end{cases}}$$

In cele din urma, forma in variabila x pentru fiecare functie spline cubica de clasa C^1 se obtine prin *schimbarea de variabila* $\boxed{t = \frac{x - x_i}{h_i}}$

8.5 Exemplu numeric

Sa consideram cunoscute urmatoarele puncte din plan:

x	1	2	4
$f(x)$	3	4	6
$f'(x)$	0	2	5

$\Rightarrow$ Avem urmatoarele noduri: $x_0 = 1;\ x_1 = 2;\ x_2 = 4.$

Tinand cont de forma unei functii spline, putem particulariza pe exemplul nostru: $S(x) = \begin{cases} S_0(x),\ x \in [1,2] \\ S_1(x),\ x \in [2,4] \end{cases}$

8.5.1 Metoda 1 - Utilizand forma generala

Avem $n = 3$ puncte in suportul interpolarii, deci vom avea 2 subintervale.
Pornim de la forma generala a functiei spline si derivam expresia:

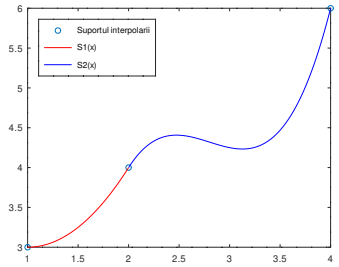
$$\boxed{S_i(x) = a_i + b_i \cdot (x - x_i) + c_i \cdot (x - x_i)^2 + d_i \cdot (x - x_i)^3,\ i = 0 : 1} \ |()'$$

$$S_i'(x) = b_i + 2 \cdot c_i \cdot (x - x_i) + 3 \cdot d_i \cdot (x - x_i)^2,\ i = 0 : 1$$

- Punem conditiile de *interpolare* de tip Hermite:
  - Fiecare spline trece prin punctul sau de interpolare: $\begin{cases} S_0(x_0) = f(x_0) \iff S_0(1) = f(1) = 3 \\ S_1(x_1) = f(x_1) \iff S_1(2) = f(2) = 4 \\ S_1(x_2) = f(x_2) \iff S_1(4) = f(4) = 6 \end{cases}$
  - Fiecare spline are aceeasi panta cu functia pe care o aproximeaza: $\begin{cases} S_0'(x_0) = f'(x_0) \iff S_0'(1) = f'(1) = 0 \\ S_1'(x_1) = f'(x_1) \iff S_1'(2) = f'(2) = 2 \\ S_1'(x_2) = f'(x_2) \iff S_1'(4) = f'(4) = 5 \end{cases}$
- Punem conditiile de *racordare*:
  - Racordam functiile (Unde se termina S_0, incepe S_1): $S_0(x_1) = S_1(x_1) \iff S_0(2) = S_1(2) = 4$
  - Spline-urile au aceeasi panta in punctele de contact: $S_0'(x_1) = S_1'(x_1) \iff S_0'(2) = S_1'(2) = 2$

Se rezolva sistemul de 8 ecuatii cu 8 necunoscute $\Longrightarrow$ Coeficientii: $\begin{cases} a_0 = 3 \\ b_0 = 0 \\ c_0 = 1 \\ d_0 = 0 \end{cases}$; $\begin{cases} a_1 = 4 \\ b_1 = 2 \\ c_1 = -3 \\ d_1 = \frac{5}{4} \end{cases}$

$$\Longrightarrow \boxed{S(x) = \begin{cases} 3 + 0 \cdot (x-1) + 1 \cdot (x-1)^2 + 0 \cdot (x-1)^3, \ x \in [1,2] \\ 4 + 2 \cdot (x-2) - 3 \cdot (x-2)^2 + \frac{5}{4} \cdot (x-2)^3, \ x \in [2,4] \end{cases}}$$



8.5.2 Metoda 2 - Utilizand forma parametrica

Pornim de la forma parametrica dedusa cu ajutorul polinoamelor Bernstein de grad 3:

$$S_i(t) = a_i' \cdot (1-t)^3 + b_i' \cdot 3 \cdot t \cdot (1-t)^2 + c_i' \cdot 3 \cdot t^2 \cdot (1-t) + d_i' \cdot t^3, \ i = 0 : n-1, \ t \in [0,1]$$

De data aceasta, pentru determinarea coeficientilor, folosim formulele: $\begin{cases} a_i' = f(x_i), \ i = 0 : n-1 \\ d_i' = f(x_{i+1}), \ i = 0 : n-1 \\ c_i' = f(x_{i+1}) - \frac{h_i}{3} \cdot f'(x_{i+1}), \ i = 0 : n-1 \\ b_i' = f(x_i) + \frac{h_i}{3} \cdot f'(x_i), \ i = 0 : n-1 \end{cases}$

Tinem cont de faptul ca $\begin{cases} h_0 = x_1 - x_0 = 2 - 1 = 1 \\ h_1 = x_2 - x_1 = 4 - 2 = 2 \end{cases}$

Aplicam formulele mentionate anterior si obtinem coeficientii:

$$\begin{cases} a_0' = f(x_0) = f(1) = 3 \\ a_1' = f(x_1) = f(2) = 4 \end{cases} \qquad \begin{cases} d_0' = f(x_1) = f(2) = 4 \\ d_1' = f(x_2) = f(4) = 6 \end{cases}$$

$$\begin{cases} c_0' = f(x_1) - \frac{h_0}{3} \cdot f'(x_1) = 4 - \frac{1}{3} \cdot 2 = \frac{10}{3} \\ c_1' = f(x_2) - \frac{h_1}{3} \cdot f'(x_2) = 6 - \frac{2}{3} \cdot 5 = \frac{8}{3} \end{cases} \qquad \begin{cases} b_0' = f(x_0) + \frac{h_0}{3} \cdot f'(x_0) = 3 + \frac{1}{3} \cdot 0 = 3 \\ b_1' = f(x_1) + \frac{h_1}{3} \cdot f'(x_1) = 4 + \frac{2}{3} \cdot 2 = \frac{16}{3} \end{cases}$$

Deci, pentru primul spline: $S_0(t)$, avem coeficientii: $a_0' = 3, \ b_0' = 3, \ c_0' = \frac{10}{3}, \ d_0' = 4$.
Facand schimbarea de variabila: $t = \frac{x - x_0}{h_0} \iff t = \frac{x-1}{1} \iff \boxed{t = x - 1}$

$$\Longrightarrow \boxed{S_0(x) = 3 \cdot (2-x)^3 + 9 \cdot (x-1) \cdot (2-x)^2 + 10 \cdot (x-1)^2 \cdot (2-x) + 4 \cdot (x-1)^3, \ x \in [1,2]}$$

Procedand in mod analog, obtinem:

$$\Longrightarrow \boxed{S_1(x) = 4 \cdot (\tfrac{4-x}{3})^3 + 16 \cdot (\tfrac{x-2}{2}) \cdot (\tfrac{4-x}{2})^2 + 8 \cdot (\tfrac{x-1}{2})^2 \cdot (\tfrac{4-x}{2}) + 6 \cdot (\tfrac{x-2}{2})^3, \ x \in [2,4]}$$

In forma finala, functia spline cubica de clasa C^1 care interpoleaza cele 3 puncte din exemplul nostru, arata astfel:

$$\boxed{S(x) = \begin{cases} S_0(x) = 3 \cdot (2-x)^3 + 9 \cdot (x-1) \cdot (2-x)^2 + 10 \cdot (x-1)^2 \cdot (2-x) + 4 \cdot (x-1)^3, \ x \in [1,2] \\ S_1(x) = 4 \cdot (\frac{4-x}{3})^3 + 16 \cdot (\frac{x-2}{2}) \cdot (\frac{4-x}{2})^2 + 8 \cdot (\frac{x-1}{2})^2 \cdot (\frac{4-x}{2}) + 6 \cdot (\frac{x-2}{2})^3, \ x \in [2,4] \end{cases}}$$

Asa cum ne-am fi asteptat, in mod evident, cele 2 metode converg la aceeasi functie spline cubica, diferenta fiind facuta de modul in care se ajunge la rezultat.

8.6 Concluzii

$\oplus$ Dispare fenomenul de oscilatie prin mentinerea gradului polinomului de interpolare mic.

$\oplus$ De data aceasta, in capetele intervalelor se face o trecere neteda, deoarce se impun niste conditii de racordare mai puternice.

$\ominus$ Este necesar sa se cunoasca valorile derivatei I in punctele din suportul interpolarii.

9 Interpolare cu functii spline cubice C2

9.1 Modul de determinare

Pornim de la forma generala: $\boxed{S_i(x) = a_i + b_i \cdot (x - x_i) + c_i \cdot (x - x_i)^2 + d_i \cdot (x - x_i)^3,\ i = 0 : n-1}$

Comparativ cu cele de clasa C^1, functiile spline de clasa C^2 au proprietatea ca sunt derivabile de 2 ori si au derivatele continue.

Vom deduce in cele ce urmeaza conditiile care trebuie puse pentru a determina coeficientii a_i, b_i, c_i, d_i:

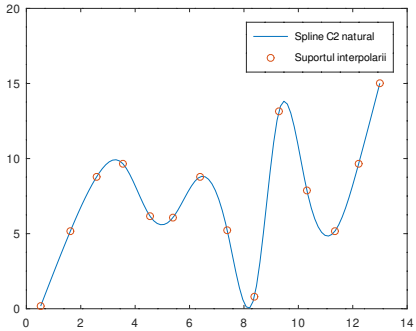
- Conditii de *interpolare* de tip Lagrange:
  - Fiecare spline trece prin punctul sau de interpolare: $\boxed{\begin{cases} S_i(x_i) = f(x_i), i = 0 : n-1 \\ S_{n-1}(x_n) = f(x_n) \end{cases}}$

    $\Rightarrow (n+1)$ ecuatii
- Conditii de *racordare*:
  - Racordam functiile (Asiguram continuitatea C^0): $\boxed{S_i(x_{i+1}) = S_{i+1}(x_{i+1}), i = 0 : n-2}$

    $\Rightarrow (n-1)$ ecuatii
  - Functiile au aceeasi *panta* in punctele de contact (Asiguram C^1): $\boxed{S_i'(x_{i+1}) = S_{i+1}'(x_{i+1}), i = 0 : n-2}$

    $\Rightarrow (n-1)$ ecuatii
  - Functiile au aceeasi *convexitate* in punctele de contact (Asiguram C^2): $\boxed{S_i''(x_{i+1}) = S_{i+1}''(x_{i+1}), i = 0 : n-2}$

    $\Rightarrow (n-1)$ ecuatii
- Pana acum avem $(n+1) + 3 \cdot (n-1) = 4n - 2$ ecuatii, pentru $4n$ necunoscute. Pentru ultimele 2 ecuatii, putem pune urmatoarele conditii in *capete*, in functie de tipul de spline ales:
  - Spline **natural**: $\boxed{\begin{cases} S_0''(x_0) = 0 \\ S_{n-1}''(x_n) = 0 \end{cases}}$
  - Spline **tensionat**: $\boxed{\begin{cases} S_0'(x_0) = f'(x_0) \\ S_{n-1}'(x_n) = f'(x_n) \end{cases}}$



Astfel, am cumulat $(4n-2) + 2 = 4n$ ecuatii pentru $4n$ necunoscute.

Observatie: Spline-ul *tensionat* ofera o aproximare mai buna, insa necesita cunoasterea derivatei I in capetele suportului de interpolare.

Din punct de vedere al automatizarii, conditiile care trebuie puse nu sunt de folos. Astfel, prelucrand ecuatiile obtinute prin punerea conditiilor, obtinem urmatoarele relatii:

Coeficientii functiilor spline cubice de clasa C^2 sunt dati de relatiile:

$\boxed{a_i = f(x_i),\ i = 0 : n}$

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

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

Coeficientii $\boxed{c_i,\ i = 0 : n-1}$ se obtin prin rezolvarea unui sistem *tridiagonal* de forma:

- Pentru spline $\mathbf{C^2}$ **natural**:

$$\begin{bmatrix} \mathbf{1} & \mathbf{0} & 0 & \dots & \dots & 0 \\ \mathbf{h_0} & \mathbf{2(h_0 + h_1)} & \mathbf{h1} & \dots & \dots & \vdots \\ 0 & \mathbf{h_1} & \mathbf{2(h_1 + h_2)} & \mathbf{h_2} & \dots & 0 \\ & \dots & \dots & \dots & \dots & \dots \\ \vdots & \dots & \dots & \mathbf{h_{n-2}} & \mathbf{2(h_{n-2} + h_{n-1})} & \mathbf{h_{n-1}} \\ 0 & \dots & \dots & 0 & \mathbf{0} & \mathbf{1} \end{bmatrix} \cdot \begin{bmatrix} 0 \\ \frac{3}{h_1}(a_2 - a_1) - \frac{3}{h_0}(a_1 - a_0) \\ \vdots \\ \frac{3}{h_{n-1}}(a_n - a_{n-1}) - \frac{3}{h_{n-2}}(a_{n-1} - a_{n-2}) \\ 0 \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix}$$

- Pentru spline $\mathbf{C^2}$ **tensionat**:

$$\begin{bmatrix} \mathbf{2h_0} & \mathbf{h_0} & 0 & \dots & \dots & 0 \\ \mathbf{h_0} & \mathbf{2(h_0+h_1)} & \mathbf{h1} & \dots & \dots & \vdots \\ 0 & \mathbf{h_1} & \mathbf{2(h_1+h_2)} & \mathbf{h_2} & \dots & 0 \\ & \dots & \dots & \dots & \dots & \dots \\ \vdots & \dots & \dots & \mathbf{h_{n-2}} & \mathbf{2(h_{n-2}+h_{n-1})} & \mathbf{h_{n-1}} \\ 0 & \dots & \dots & 0 & \mathbf{h_{n-1}} & \mathbf{2h_{n-1}} \end{bmatrix} \cdot \begin{bmatrix} \frac{3}{h_0}(a_1-a_0)-3f'(a) \\ \frac{3}{h_1}(a_2-a_1)-\frac{3}{h_0}(a_1-a_0) \\ \vdots \\ \frac{3}{h_{n-1}}(a_n-a_{n-1})-\frac{3}{h_{n-2}}(a_{n-1}-a_{n-2}) \\ 3f'(b)-\frac{3}{h_{n-1}}(a_n-a_{n-1}) \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix}$$

Observatie: Functia spline S_n a fost introdusa pentru a ajuta la calcuarea functiilor spline $S_i,\ i = 0 : n-1$.

9.2 Exemplu numeric

Sa consideram cunoscute urmatoarele puncte din plan:

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

$\Rightarrow$ Avem urmatoarele noduri: $x_0 = -1;\ x_1 = 0;\ x_2 = 1;\ x_3 = 2$.

Tinand cont de forma unei functii spline, putem particulariza pe exemplul nostru: $S(x) = \begin{cases} S_0(x),\ x \in [-1, 0] \\ S_1(x),\ x \in [0, 1] \\ S_2(x),\ x \in [1, 2] \end{cases}$

9.2.1 Metoda 1 - Punand conditiile

Pornim de la forma generala a functiei spline si derivam expresia de 2 ori:

$$\boxed{S_i(x) = a_i + b_i \cdot (x - x_i) + c_i \cdot (x - x_i)^2 + d_i \cdot (x - x_i)^3,\ i = 0 : 2} \ |()'$$

$$S_i'(x) = b_i + 2 \cdot c_i \cdot (x - x_i) + 3 \cdot d_i \cdot (x - x_i)^2,\ i = 0 : 2 \ |()'$$

$$S_i''(x) = 2 \cdot c_i + 6 \cdot d_i \cdot (x - x_i),\ i = 0 : 2$$

- Conditii de *interpolare* de tip Lagrange:
  - Fiecare spline trece prin punctul sau de interpolare: $\boxed{\begin{cases} S_0(x_0) = f(x_0) \iff S_0(-1) = f(-1) = -2 \\ S_1(x_1) = f(x_1) \iff S_1(0) = f(0) = -1 \\ S_2(x_2) = f(x_2) \iff S_2(1) = f(1) = 2 \\ S_2(x_3) = f(x_3) \iff S_2(2) = f(2) = 4 \end{cases}}$
- Conditii de *racordare*:
  - Racordam functiile (Asiguram continuitatea C^0): $\boxed{\begin{cases} S_0(x_1) = S_1(x_1) \iff S_0(0) = S_1(0) \\ S_1(x_2) = S_2(x_2) \iff S_1(1) = S_2(1) \end{cases}}$
  - Functiile au aceeasi *panta* in punctele de contact (C^1): $\boxed{\begin{cases} S_0'(x_1) = S_1'(x_1) \iff S_0'(0) = S_1'(0) \\ S_1'(x_2) = S_2'(x_2) \iff S_1'(1) = S_2'(1) \end{cases}}$
  - Functiile au aceeasi *convexitate* in punctele de contact (C^2): $\boxed{\begin{cases} S_0''(x_1) = S_1''(x_1) \iff S_0''(0) = S_1''(0) \\ S_1''(x_2) = S_2''(x_2) \iff S_1''(1) = S_2''(1) \end{cases}}$
- Pentru ultimele 2 ecuatii, punem urmatoarele conditii in *capete*, in functie de tipul de spline:
  - Spline **natural**: $\boxed{\begin{cases} S_0''(x_0) = 0 \iff S_0''(-1) = 0 \\ S_2''(x_3) = 0 \iff S_2''(2) = 0 \end{cases}}$
  - Spline **tensionat**: $\boxed{\begin{cases} S_0'(x_0) = f'(x_0) \iff S_0'(-1) = f'(-1) = 3 \\ S_2'(x_3) = f'(x_3) \iff S_2'(2) = f'(2) = 2 \end{cases}}$

Rezolvam sistemul de 12 ecuatii cu 12 necunoscute si astfel determinam coeficientii celor 3 spline-uri $S_0,\ S_1$ si S_2.

Observatie: Pentru spline-ul tensionat s-a tinut cont de valorile derivatei I in capete suportului de interpolare.

9.2.2 Metoda 2 - Rezolvand sistemul tridiagonal

In continuare, vom determina coeficientii spline-urilor, rezolvand sistemul tridiagonal. Prezentam rezolvarea pentru spline C^2 natural (analog se procedeaza si pentru spline C^2 tensionat).

Coeficientii functiilor spline cubice de clasa C^2 sunt dati de relatiile:

$$a_i = f(x_i),\ i = 0:3 \iff \begin{cases} a_0 = f(x_0) \iff a_0 = f(-1) = -2 \\ a_1 = f(x_1) \iff a_1 = f(0) = -1 \\ a_2 = f(x_2) \iff a_2 = f(1) = 2 \\ a_3 = f(x_3) \iff a_3 = f(2) = 4 \end{cases} \implies \boxed{\begin{cases} a_0 = -2 \\ a_1 = -1 \\ a_2 = 2 \\ a_3 = 4 \end{cases}}$$

In continuare, avem nevoie de lungimile subintervalelor: $\begin{cases} h_0 = x_1 - x_0 \iff h_0 = 0 - (-1) = 1 \\ h_1 = x_2 - x_1 \iff h_1 = 1 - 0 = 1 \\ h_2 = x_3 - x_2 \iff h_2 = 2 - 1 = 1 \end{cases}$

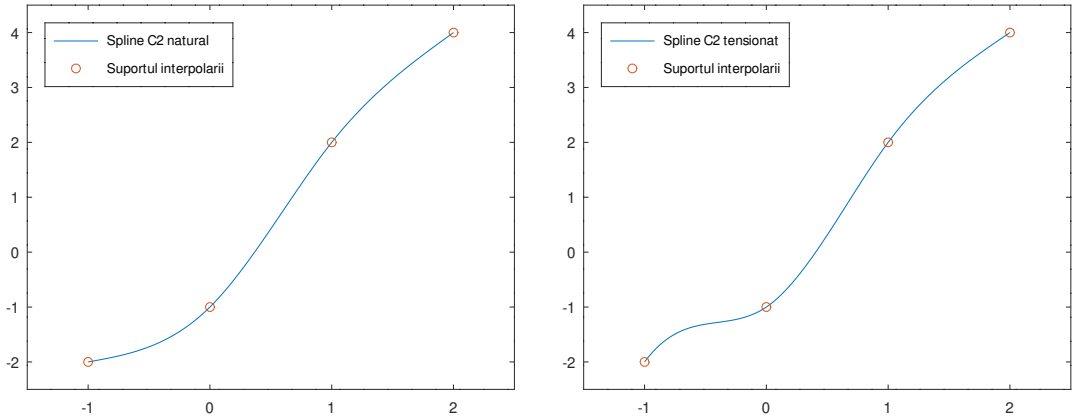
$$\begin{bmatrix} \mathbf{1} & \mathbf{0} & 0 & 0 \\ \mathbf{h_0} & \mathbf{2(h_0 + h_1)} & \mathbf{h_1} & 0 \\ 0 & \mathbf{h_1} & \mathbf{2(h_1 + h_2)} & \mathbf{h_2} \\ 0 & 0 & 0 & \mathbf{1} \end{bmatrix} \cdot \begin{bmatrix} 0 \\ \frac{3}{h_1}(a_2 - a - 1) - \frac{3}{h_0}(a_1 - a_0) \\ \frac{3}{h_2}(a_3 - a_2) - \frac{3}{h_1}(a_2 - a_1) \\ 0 \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \iff$$

$$\begin{bmatrix} \mathbf{1} & \mathbf{0} & 0 & 0 \\ \mathbf{1} & \mathbf{4} & \mathbf{1} & 0 \\ 0 & \mathbf{1} & \mathbf{4} & \mathbf{1} \\ 0 & 0 & \mathbf{0} & \mathbf{1} \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 6 \\ -3 \\ 0 \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \implies \boxed{\begin{cases} c_0 = 0 \\ c_1 = \frac{9}{5} \\ c_2 = -\frac{6}{5} \\ c_3 = 0 \end{cases}}$$

$$d_i = \frac{c_{i+1} - c_i}{3 \cdot h_i},\ i = 0:2 \implies \begin{cases} d_0 = \frac{c_1 - c_0}{3h_0} \iff d_0 = \frac{\frac{9}{5} - 0}{3 \cdot 1} = \frac{3}{5} \\ d_1 = \frac{c_2 - c_1}{3h_1} \iff d_1 = \frac{-\frac{6}{5} - \frac{9}{5}}{3 \cdot 1} = -1 \\ d_2 = \frac{c_3 - c_2}{3h_2} \iff d_2 = \frac{0 + \frac{6}{5}}{3 \cdot 1} = \frac{2}{5} \end{cases} \implies \boxed{\begin{cases} d_0 = \frac{3}{5} \\ d_1 = -1 \\ d_2 = \frac{2}{5} \end{cases}}$$

$$b_i = \frac{a_{i+1} - a_i}{h_i} - \frac{h_i}{3} \cdot (2 \cdot c_i + c_{i+1}),\ i = 0:2 \implies \begin{cases} b_0 = \frac{a_1 - a_0}{h_0} - \frac{h_0}{3}(2c_0 + c_1) = \frac{-1+2}{1} - \frac{1}{3}(2 \cdot 0 + \frac{9}{5}) \\ b_1 = \frac{a_2 - a_1}{h_1} - \frac{h_1}{3}(2c_1 + c_2) = \frac{2+1}{1} - \frac{1}{3}(2 \cdot \frac{9}{5} - \frac{6}{5}) \\ b_2 = \frac{a_3 - a_2}{h_2} - \frac{h_2}{3}(2c_2 + c_3) = \frac{4-2}{1} - \frac{1}{3}(2 \cdot \frac{-6}{5} + 0) \end{cases} \implies \boxed{\begin{cases} b_0 = \frac{2}{5} \\ b_1 = \frac{11}{5} \\ b_2 = \frac{14}{5} \end{cases}}$$

Astfel, am determinat coeficientii celor 3 spline-uri C^2 naturale. Evident, ambele metode conduc la acelasi rezultat (acelasi spline de interpolare), cu precizarea ca metoda 2 poate fi aplicata intr-un algoritm.



9.3 Concluzii

$\oplus$ Dispare fenomenul de oscilatie (la fel ca la C^1).

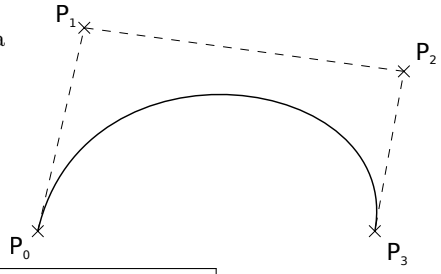
$\oplus$ Se impun conditii de racordare mai puternice decat la C^1, astfel spline-ul va reda mai bine realitatea.

$\oplus$ Este necesar sa se cunoasca valorile derivatei I in capetele suportului de interpolare (**doar** pentru spline-urile tensionate).

10 Curbe Bézier

10.1 Notiuni generale

O curba *Bézier* este o curba *parametrica* cu importante aplicatii in grafica pe calculator. Primeste ca input niste *puncte in plan* $(P_0, P_1, \ldots, P_{n-1}, P_n)$:



- Punctele P_0 si P_n se numesc *puncte de interpolare (anchor points)*, deoarece curba Bézier trece prin ele (incepe in P_0 si se termina in P_1).
- Punctele $P_1, \ldots, P_{n-1}$ se numesc *puncte de control (handle points)*, deoarece ele influenteaza path-ul curbei Bézier.

La modul general, curba Bézier de grad n este definita astfel: $\boxed{B_n(t) = \sum_{i=0}^{n} P_i \cdot B_{i,n}(t),\ t \in [0,1]}$, unde:

$\begin{cases} P_i \text{ sunt punctele de intrare} \\ B_{i,n}(t) \text{ sunt polinoamele } \textit{Bernstein} \text{ de grad } n\text{: } \boxed{B_{i,n}(t) \overset{\text{def}}{=} C_n^i \cdot (1-t)^{n-i} \cdot t^i} \end{cases}$

10.2 Modul de determinare

Particularizam forma generala a curbei Bézier pentru $n = 1$, $n = 2$ si $n = 3$:

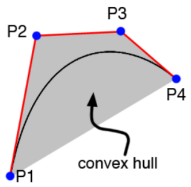
- Curba Bézier Liniara $(n = 1)$ Animatie
  Este similara cu interpolarea liniara sau parametrizarea segmentului.
  $B_1(t) = P_0 \cdot B_{0,1}(t) + P_1 \cdot B_{1,1}(t) \Rightarrow \boxed{B_1(t) = (1-t) \cdot P_0 + P_1,\ t \in [0,1]}$

- Curba Bézier Patratica $(n = 2)$ Animatie
  $B_2(t) = P_0 \cdot B_{0,2}(t) + P_1 \cdot B_{1,2}(t) + P_2 \cdot B_{2,2}(t) \Rightarrow \boxed{B_2(t) = (1-t)^2 \cdot P_0 + 2 \cdot t \cdot (1-t) \cdot P_1 + t^2 \cdot P_2,\ t \in [0,1]}$

- Curba Bézier Cubica $(n = 3)$ Animatie
  $B_3(t) = P_0 \cdot B_{0,3}(t) + P_1 \cdot B_{1,3}(t) + P_2 \cdot B_{2,3}(t) + P_3 \cdot B_{3,3}(t)$
  $\Rightarrow \boxed{B_3(t) = (1-t)^3 \cdot P_0 + 3 \cdot t \cdot (1-t)^2 \cdot P_1 + 3 \cdot t^2 \cdot (1-t) \cdot P_2 + t^3 \cdot P_3,\ t \in [0,1]}$

Pentru explicatii suplimentare referitoare la modul de constructie al curbelor Bézier:[9]

Orice curba Bézier este continuta complet in poligonul convex definit de punctele $P_0, P_1, \ldots, P_n$ (are proprietatea de *convex hull*)



In mod uzual, se folosesc curbe Bézier *cubice* determinate de 4 puncte P_0, P_1, P_2, P_3, avand forma parametrica determinata anterior:

$B_3(t) = (1-t)^3 \cdot P_0 + 3 \cdot t \cdot (1-t)^2 \cdot P_1 + 3 \cdot t^2 \cdot (1-t) \cdot P_2 + t^3 \cdot P_3,\ t \in [0,1]$

$$\Longrightarrow \boxed{B_3(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix},\ t \in [0,1]}$$

Observati cum se schimba path-ul curbei Bézier odata cu modificarea pozitiei punctelor de control [10].

Observatie: Pentru puncte in spatiu, notiunea de curba Bézier se extinde la *suprafata Bézier* [11].
Asadar, pentru a determina curba Bézier de grad n, ne folosim de polinoamele Bernstein de grad n.
Asa cum am vazut, polinoamele Bernstein se pot calcula si *recurent*.
Pentru a facilita calculul computerizat al curbei Bézier, introducem algoritmul **De Casteljau**.

[9] https://www.youtube.com/watch?v=pnYccz1Ha34
[10] https://www.desmos.com/calculator/gptceium9j
[11] https://bit.ly/2W6OonD

11 Algoritmul De Casteljau

11.1 Notiuni generale

Intr-o abordare directa, calcularea unui punct aflat pe o curba Bézier se obtine folosind ecuatia parametrica $B_n(t) = \sum_{i=0}^{n} P_i \cdot B_{i,n}(t),\ t \in [0,1]$.

Aceasta metoda este ineficienta, deoarece este instabila numeric (numerele mici ridicate la puteri mari, genereaza erori).

Algoritmul De Casteljau este o metoda recursiva de calcul a polinoamelor Bernstein folosite in determinarea curbelor Bézier.

- Date de *intrare*: Punctele $P_0, P_1, \ldots, P_n$.
- Se foloseste relatia de *recurenta*:

  $P_i^{(0)} = P_i,\ i = 0 : n$
  for $j = 1 : n$
  for $i = 0 : n - j$
  $P_i^{(j)} = P_i^{(j-1)} \cdot (1-t) + P_{i+1}^{(j-1)} \cdot t$
- Date de *iesire*: $P_0^{(n)}$

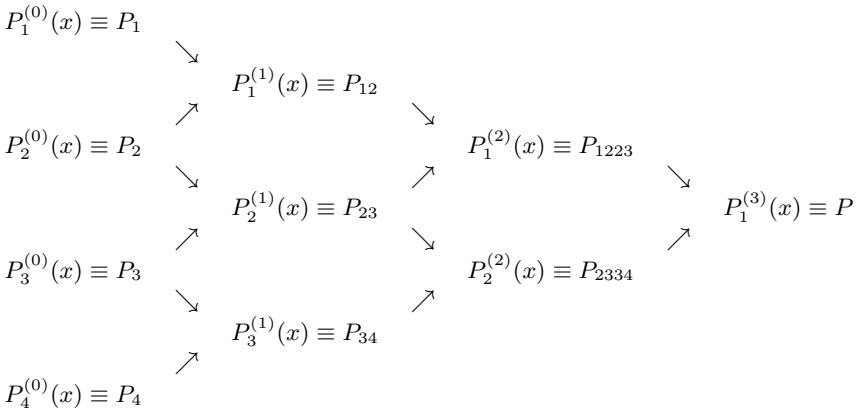
Observatie: $P_i^{(j)}$ reprezinta punctul P_i la pasul j.

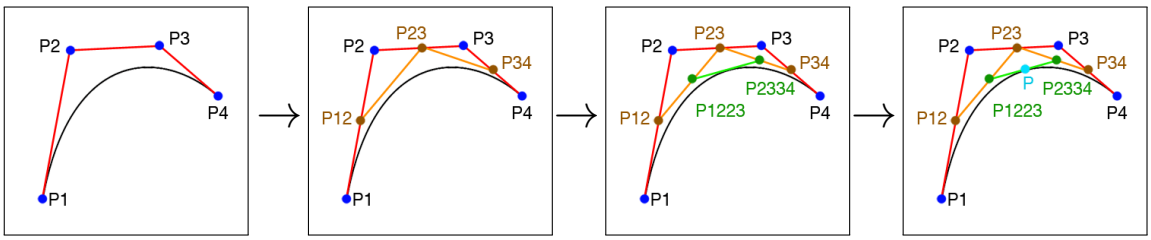
Fie $P_i^{(0)}$ si $P_{i+1}^{(0)}$ doua puncte succesive si $P_i^{(1)}$ un punct care imparte segmentul $P_i^{(0)}P_{i+1}^{(0)}$ in raportul $\frac{t}{1-t}$. Atunci, $P_i^{(1)}$ este o combinatie liniara intre punctele $P_i^{(0)}$ si $P_{i+1}^{(0)}$, adica: $P_i^{(1)} = (1-t) \cdot P_i^{(0)} + t \cdot P_{i+1}^{(0)}$

Se formeaza in acest fel poligonul $P_0^{(1)}, P_1^{(1)}, \ldots, P_{n-1}^{(1)}$. Se aplica relatia de recurenta noului poligon, obtinandu-se poligonul $P_0^{(2)}, P_1^{(2)}, \ldots, P_{n-2}^{(2)}$. Repetand procesul de n ori, se obtine un singur punct $P_0^{(n)}$. Acest punct obtinut se afla pe curba Bézier.

Pentru a intelege mai bine cum functioneaza algoritmul, vom particulariza $n = 3$, adica vom construi o curba Bézier de grad 3. Ideea din spatele algoritmului este prezentata grafic aici [12].

Pentru $n = 3$, procesul de obtinere al punctului $P_1^{(3)}(x)$, este prezentat mai jos (indexarea din exemplu este facuta de la 1):





11.2 Concluzii

⊕ Algoritm stabil numeric.

⊖ Mai lent decat abordarea directa (non-recursiva).

[12] https://www.youtube.com/watch?v=YATikPP2q70

12 Probleme Propuse

12.1 Problema 1

Scrieti un program Octave care interpoleaza un set de date, folosind interpolarea **Vandermonde**.
Descarca scriptul:

```
function [P] = vandermonde(x, y)
  % Input:
  %        Coordonatele punctelor din suportul de interpolare (x, y)
  % Output:
  %        Coeficientii polinomului de interpolare Vandermonde

  % Asigura dimensiunea (n, 1) pentru vectorii input
  [n m] = size(x);
  if (n == 1) % Vector linie
    x = x';
    n = m;
  endif

  [p q] = size(y);
  if (p == 1) % Vector linie
    y = y';
    p = q;
  endif

  P = [];
  % Constructie matrice A (matrice Vandermonde)
  % TODO

  % Constructie vector coeficienti
  % TODO

  % Rezolvam sistemul A * P_coef = y
  % TODO
  % Hint: Output-ul este reprezentat de vectorul de coeficienti P (P_coef)


endfunction
```

Pentru testarea programului, puteti folosi urmatoarea functie:
Descarca scriptul:

```
function [] = test_vandermonde()

  % Suportul de interpolare
  x = [-3 0 2 5 7 11 14];
  y = [5 -3 1 3 8 4 -10];

  % Polinomul de interpolare
  P = vandermonde(x, y);

  % Grafic puncte + polinom de interpolare obtinut
  xx = linspace(min(x), max(x)); % Generare 100 de puncte intre min(x)si max(x)
  yy = polyval(P, xx); % Evaluare polinom de interpolare in cele 100 de puncte

  % Grafic polinom
  plot(xx, yy);
  hold on;

  % Grafic punctele din suportul interpolarii
  plot(x, y, 'o');
  hold off;

  legend('Vandermonde', 'Suportul interpolarii', 'location', 'northwest');

endfunction
```

Pentru verificare, puteti compara rezultatul vostru cu graficul de aici:
Extra: Puteti rezolva problema si pe hartie si astfel determinati coeficientii polinomului de interpolare.

12.2 Problema 2

Scrieti un program Octave care interpoleaza un set de date, folosind interpolarea **Lagrange**.
Descarca scriptul:

```
function y0 = lagrange(x, y, x0)
  % Input:
  %        Coordonatele punctelor din suportul de interpolare (x, y)
  %        Punctul (x0) in care dorim sa aflam valoarea dupa interpolare
  % Output:
  %        Valoarea (y0) in punctul dorit (x0)

  y0 = 0; % Valoarea polinomului Lagrange in punctul dorit x0
  L = ones(length(x), 1); % Multiplicator Lagrange

  % Calculare polinom de interpolare Lagrange
  for i = 1 : length(x)

    % Calculare multiplicator Lagrange
    % TODO


    % Actualizare valoare polinom Lagrange
    % TODO
    % Hint: y0 += ...

  endfor

endfunction
```

Pentru testarea programului, puteti folosi urmatoarea functie:
Descarca scriptul:

```
function [] = test_lagrange()

  % Suportul de interpolare
  x = [-3 0 2 5 7 11 14];
  y = [5 -3 1 3 8 4 -10];

  % Grafic puncte + polinom de interpolare obtinut
  xx = linspace(min(x), max(x)); % Generare 100 de puncte intre min(x)si max(x)
  for i = 1 : 100
    yy(i) = lagrange(x, y, xx(i)); % Calculare valoare in punctele xx
  endfor

  % Grafic polinom
  plot(xx, yy);
  hold on;

  % Grafic punctele din suportul interpolarii
  plot(x, y, 'o');
  hold off;

  legend('Lagrange', 'Suportul interpolarii', 'location', 'northwest');

endfunction
```

Pentru verificare, puteti compara rezultatul vostru cu graficul de aici:
Extra: Puteti rezolva problema si pe hartie si astfel determinati coeficientii polinomului de interpolare.

12.3 Problema 3

Scrieti un program Octave care interpoleaza un set de date, folosind interpolarea **Newton**.
Descarca scriptul:

```
function [P] = newton(x, y)
  % Input:
  %        Coordonatele punctelor din suportul de interpolare (x, y)
  % Output:
  %        Polinomul de interpolare Newton (P)

  % Asigura dimensiunea (n, 1) pentru vectorii input
  [n m] = size(x);
  if (n == 1) % Vector linie
    x = x';
    n = m;
  endif

  [p q] = size(y);
  if (p == 1) % Vector linie
    y = y';
    p = q;
  endif

  % Calculam recurent diferentele divizate (D)
  D = zeros(n, n);
  D(:, 1) = y;
  % TODO

  % Constructie polinom de interpolare Newton (P)
  P = [];
  P = D(n, n);
  % TODO
  for k = n-1 : -1 : 1
    % Hints:
      % poly(x(k)) - creeaza un polinom de gr 1 a carui radacina este x(k)
      % conv(P, poly(x(k))) - inmulteste cele 2 polinoame

  endfor

endfunction
```

Pentru testarea programului, puteti folosi urmatoarea functie:
Descarca scriptul:

```
function [] = test_newton()

  % Suportul de interpolare
  x = [-3 0 2 5 7 11 14];
  y = [5 -3 1 3 8 4 -10];

  % Polinomul de interpolare
  P = newton(x, y);

  % Grafic puncte + polinom de interpolare obtinut
  xx = linspace(min(x), max(x)); % Generare 100 de puncte intre min(x)si max(x)
  yy = polyval(P, xx); % Evaluare polinom de interpolare in cele 100 de puncte

  % Grafic polinom
  plot(xx, yy);
  hold on;

  % Grafic punctele din suportul interpolarii
  plot(x, y, 'o');
  hold off;

  legend('Newton', 'Suportul interpolarii', 'location', 'northwest');

endfunction
```

Pentru verificare, puteti compara rezultatul vostru cu graficul de aici:
Extra: Puteti rezolva problema si pe hartie si astfel determinati coeficientii polinomului de interpolare.

12.4 Problema 4

Scrieti un program Octave care interpoleaza un set de date, folosind functii **Spline C^2 Natural**.
Descarca scriptul:

```
function y0 = splineC2_natural(x, y, x0)
  % Input:
  %        Coordonatele punctelor din suportul de interpolare (x, y)
  %        Punctul x0 in care se doreste valoarea dupa interpolare
  % Output:
  %        Valoarea dorita (y0) in punctul x0

  % Asigura forma de vector coloana pentru vectorii de input
  [n m] = size(x);
  if (n == 1) % Vector linie
    x = x';
    n = m;
  endif

  [p q] = size(y);
  if (p == 1) % Vector linie
    y = y';
    p = q;
  endif
  y0 = 0;

  % Avem de rezolvat un sistem de forma A * x = b, unde A - 3diag
  % Setare lungime intervale: h(i) = x(i+1) - x(i)
  % TODO

  % Setare coeficienti a(i) (pentru spline-uri)
  % TODO

  % Setare coeficienti pentru vectorul de termeni liberi (g)
  for i = 2 : n - 1
    % TODO
  endfor

  % Setare cele 3 diagonale din matricea A
  % dd - Subdiagonala
  % TODO

  % aa - Diagonala principala
  for i = 2 : n-1
    % TODO
  endfor

  % cc - Supradiagonala
  % TODO

  % Rezolvare sistem A * x = b (A - 3diag)
  % TODO

  % =>  Coeficientii c(i) pentru fiecare spline

  % Calculam coeficientii ramasi (b(i), d(i)) pentru fiecare spline
  for i = 1 : n-1
    % Pentru 'd' - folosim relatia: c(j+1) = c(j) + 3*d(j)*h(j)
    % TODO

    % Pentru 'b' - folosim relatia: b(j+1) = b(j) + 2*c(j)*h(j) + 3*d(j)*h(j)^2
    % TODO
  endfor

  % Determinare subinterval [x(i), x(i+1)]
  % in care se afla x0 (puctul dorit)
  for i = 1 : n-1
    if((x(i) <= x0) && (x0 <= x(i+1)))
      % TODO
      % Hint: Pur si simplu, aplica formula de la forma generala
      % a spline-ului cubic cu a(i), b(i), c(i), d(i) determinati anterior

      break;
    endif
  endfor

endfunction
```

Pentru testarea programului, puteti folosi urmatoarea functie:
Descarca scriptul:

```
function [] = test_splineC2_natural()

  % Suportul de interpolare
  x = [-3 0 2 5 7 11 14];
  y = [5 -3 1 3 8 4 -10];

  % Suport de interpolare - Logo MN
  % x = [0.526 1.617 2.583 3.550 4.547 5.389 6.387 7.384 8.382 9.286 10.315 11.344
    12.217 12.996];
  % y = [0.177 5.165 8.782 9.654 6.163 6.069 8.782 5.227 0.800 13.146 7.877 5.165
    9.654 15.017];

  % Grafic puncte + polinom de interpolare obtinut
  xx = linspace(min(x), max(x)); % Generare 100 de puncte intre min(x)si max(x)
  for i = 1 : 100
    yy(i) = splineC2_natural(x, y, xx(i)); % Calculare valoare in punctele xx
  endfor

  % Grafic spline C2
  plot(xx, yy);
  hold on;

  % Grafic punctele din suportul interpolarii
  plot(x, y, 'o');
  hold off;

  legend('Spline C2 natural', 'Suportul interpolarii', 'location', 'southwest');

endfunction
```

Pentru verificare, puteti compara rezultatul vostru graficul de aici:
Extra: Puteti rezolva problema si pe hartie si astfel determinati coeficientii spline-urilor.

12.5 Extra

In incheiere, puteti lectura un material referitor la utilizarea interpolarii intr-o aplicatie din viata reala (**Heat Transfer**):