Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informatiky a kvantitativních metod

Spline funkce a jejich aplikace

Bakalářská práce

Autor: David Pikal
Studijní obor: Aplikovaná informatika

Vedoucí práce: doc. RNDr. Pavel Pražák, Ph.D.

Hradec Králové srpen 2020

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 11.8.2020 David Pikal

Poděkování:

Tímto bych chtěl poděkovat svému vedoucímu bakalářské práce doc. RNDr. Pavlu Pražákovi, Ph.D. za metodické vedení, odborné připomínky a cenné rady při zpracovávání této práce.

Anotace

Cílem této bakalářské práce je z matematických definic odvodit výpočet všech typů spline funkcí, implementovat algoritmy jejich výpočtu v Matlabu a také uvést praktické využití těchto funkcí.

Práce může sloužit jako rozšiřující materiál předmětu Numerická matematika na navazujícím magisterském oboru Aplikovaná informatika. Také může být zdrojem pro ty, kteří by si rádi rozšířili znalosti v oblasti polynomické interpolace.

Annotation

Title: Spline functions and their application

The aim of this bachelor's thesis is to deduce the calculation of all types of spline functions from mathematical definitions, to implement algorithms and their calculations in Matlab, and to state the practical use of these functions.

This thesis may serve as an additional resource for the Numerical mathematics subject in the follow-up master´s degree studies of Applied informatics. It may also serve as a resource for those, who wish to further their knowledge in the field of polynomial interpolation.

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Existují funkce s jednoduchými předpisy, u nichž je snadné vypočítat funkční hodnoty „ručně“ nebo s pomocí kalkulačky. Co ale v případě, že funkční předpis je komplikovanější, se kterým se hůře manipuluje? Jak uvádí Pražák v [1]: „Při výpočtech nebo při některých úvahách o vlastnostech funkcí může být přímé použití funkčního předpisu obtížné“. Nebo kdyby funkce nebyla zadána žádným předpisem, pouze množinou bodů, získaných například nějakým měřením. Jako řešení se nabízí nahrazení složité funkce funkcí jednodušší, podobné původní. Podle Mathewse a Finka [2] lze tuto funkci získat metodou aproximace.

Aproximace je metoda nalezení přibližné hodnoty funkce v intervalu, ve kterém jsou známy některé hodnoty. Hledaná zjednodušená funkce může (např. spline funkce, Lagrangeův interpolační polynom) i nemusí (např. metoda nejmenších čtverců) procházet známými body. V případě, kdy neprochází, se k původní pouze blíží. Jestliže prochází nazývá se tento typ aproximace interpolace.

Jako jedna z jednoduchých možností se podle [2] jeví použít, jako náhradní funkci mnohočlen. Polynomy jsou funkce, se kterými se lehce počítá, zejména je lehké jejich derivování a integrování. Polynom stupně *n* je vyjádřený předpisem:

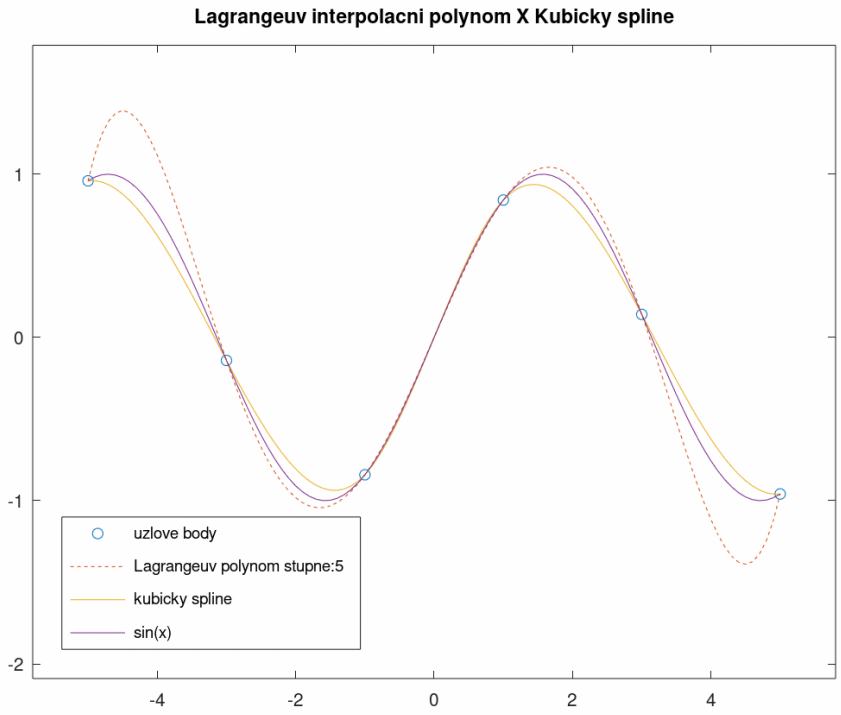
$$f(x) = s_0 + s_1 x + s_2 x^2 + \cdots + s_n x^n$$

Mnohočlen stupně *n* lze podle Kočandrlové a Černého [3] vypočítat Newtonovou nebo Lagrangovou metodou.

Pro vysvětlení, proč nepoužít interpolaci mnohočlenem stupně *n (n+1 bodů)* využijeme Lagrangeův polynom. Ten je podle [3] tvořen dílčími polynomy stejného stupně (${}^{i}l_n$). Při interpolování velkého množství bodů narazíme na první z nevýhod. Značně se zvýší počet operací sčítání a násobení, tím pádem i náročnost výpočtu. Další nevýhodou je, že při přidání dalšího bodu (tím pádem by se zvýšil i stupeň mnohočlenu o 1) bychom nemohli využít předchozí výsledky. Museli bychom znovu počítat dílčí polynomy.

Hlavním smyslem interpolace je, co nejvíce se přiblížit původní funkci. Jak uvádí Čermák a Hlavička [4], může dojít k tomu, že interpolační funkce bude příliš

oscilovat mezi uzlovými body (a tím se vzdalovat od původní funkce) – viz obrázek 1.



Obrázek 1: Srovnání Lagrangeova interpolačního polynomu a kubického splinu (vlastní zpracování)

Proto pro velký počet bodů je vhodnější využít jinou interpolační metodu – spline funkce.

2 Teorie spline funkcí

Cílem této kapitoly je popsat teorii spline funkcí. U každého typu splinu je nejprve uvedena definice, z té je následně odvozen výpočet, a nakonec je vyřešen příklad.

Při užití této interpolační metody neprokládáme podle [4] *n+1* bodů právě jedním polynomem *n-tého* stupně, ale po částech mnohočleny nižšího stupně.
Máme množinu bodů:

$$X_0 = [x_0, y_0], X_1 = [x_1, y_1], X_2 = [x_2, y_2], \dots, X_n[x_n, y_n]$$

splňující podmínku:

$$x_0 < x_1 < \cdots < x_n$$

Rozdělíme interval $\langle x_0, x_n \rangle$ na subintervaly $\langle x_k, x_{k+1} \rangle \; pro \; k \; \in \{0, \; 1, \; \dots, n-1\}$.

Po částech polynomický interpolant, který hledáme, označíme jako $S(x)$. Nazveme ho *interpolačním splinem*. Na každém intervalu $\langle x_k, x_{k+1} \rangle$ je $S(x)$ jiný polynom. Ke kterému intervalu patří využijeme index k ($S(x) = S_k(x)$ na intervalu $\langle x_k, x_{k+1} \rangle$).

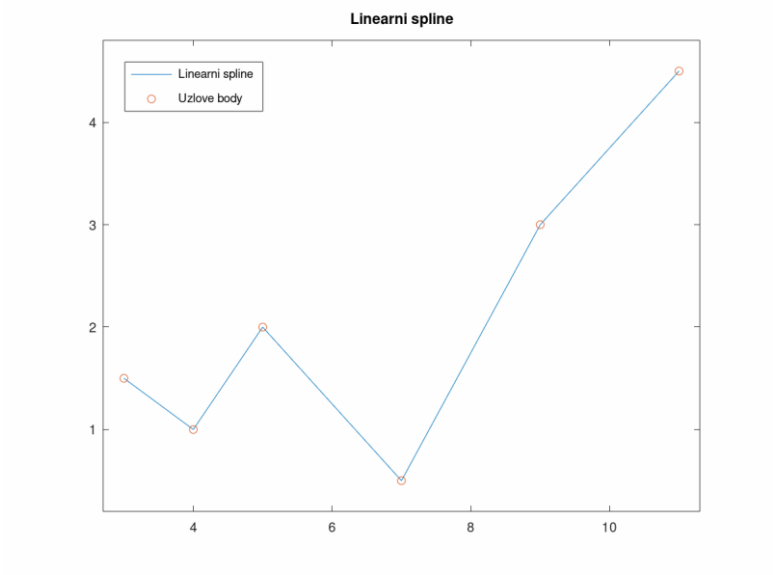
Spline funkce rozdělujeme podle stupně interpolačního mnohočlenu mezi uzlovými body:

1. stupně = lineární spline
2. stupně = kvadratický spline
3. stupně = kubický spline

Všechny typy splinů mají společnou podmínku $S_k(x_{k+1}) = S_{k+1}(x_{k+1})$. Výše uvedená rovnice říká, že interpolační spline je funkce spojitá. Je-li *n* stupeň interpolačního polynomu, pak je spojitá i jeho derivace *n-1*.

2.1 Lineární spline

Jak už název napovídá je tento typ spline funkce po částech lineární funkce [2]. Stupeň interpolačního mnohočlenu v tomto případě bude roven jedné, tedy nejnižší možný. Graficky bychom prostě pospojovali úsečkami sousední body (viz obrázek 2).



Obrázek 2: Lineární spline (vlastní zpracování)

Polynom stupně jedna má předpis:

$$f(x) = s_{k,0} + s_{k,1}(x - x_k) \tag{1}$$

Na intervalu $\langle x_k, x_{k+1}\rangle$ je $s_{k,0} = y_k,\ s_{k,1} = d_k$, kde d_k je směrnice přímky na daném intervalu. Vypočítaná podle:

$$d_k = \frac{y_{k+1} - y_k}{x_{k+1} - x_k} \tag{2}$$

Dosazením do (1) dostaneme předpis pro k-tý člen lineárního splinu:

$$S_k(x) = y_k + d_k(x - x_k), \text{pro } x \in \langle x_k, x_{k+1}\rangle \tag{3}$$

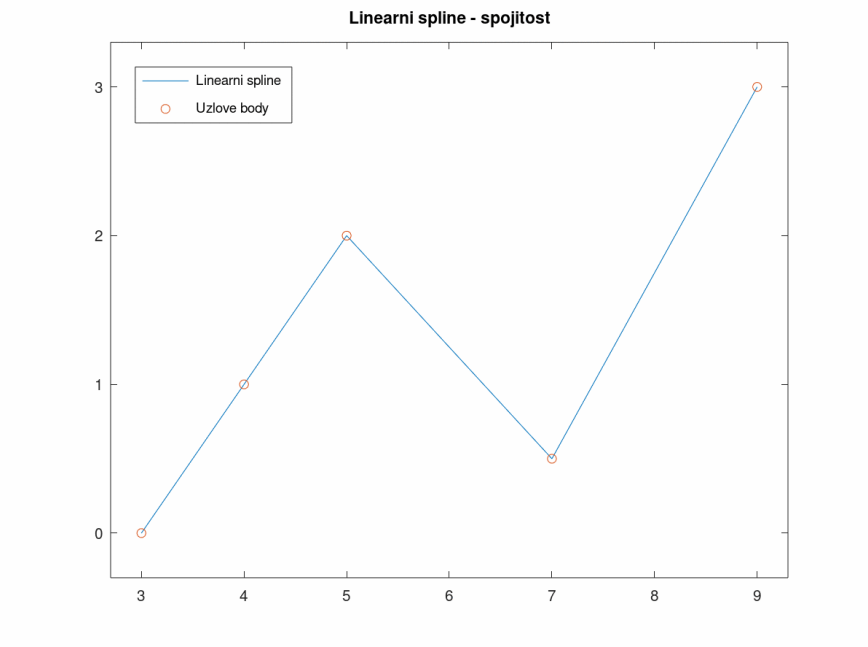
Lineární spline pak můžeme zapsat jako:

$$S(x) = \begin{cases} y_0 + d_0(x - x_0) & , x \in \langle x_0, x_1\rangle \\ y_1 + d_1(x - x_1) & , x \in \langle x_1, x_2\rangle \\ \vdots & \vdots \\ y_k + d_k(x - x_k) & , x \in \langle x_k, x_{k+1}\rangle \\ \vdots & \vdots \\ y_{n-1} + d_{n-1}(x - x_{n-1}) & , x \in \langle x_{n-1}, x_n\rangle \end{cases} \tag{4}$$

Rovnici (3) lze také zapsat podle [3] pomocí Lagrangeova interpolačního polynomu stupně jedna:

$$S_k(x) = y_k \frac{x - x_{k+1}}{x_k - x_{k+1}} + y_{k+1} \frac{x - x_k}{x_{k+1} - x_k} \tag{5}$$

Lineární spline je funkce spojitá, první derivace ve vnitřních uzlech spojitá obecně není. Ve většině případech tedy platí: $S'_k(x_{k+1}) \neq S'_{k+1}(x_{k+1})$. Rovnost by nastala, pokud $d_k = d_{k+1}$, což by znamenalo, že body x_k, x_{k+1}, x_{k+2} leží na jedné přímce (obrázek 3).



Obrázek 3: Lineární spline – spojitost (vlastní zpracování)

Příklad 2.1

Máme za úkol vypočítat koeficienty lienárního splinu pro body zadané tabulkou 1.

k	0	1	2	3
x_k	3	4,5	7	9
$f(x_k)$	2,5	1	2,5	0,5

Tabulka 1: Uzlové body (vlastní zpracování)

Řešení:

Nejprve vypočítáme směrnice jednotlivých úseček podle rovnice (2):

$$d_0 = \frac{1 - 2{,}5}{4{,}5 - 3} = -1$$

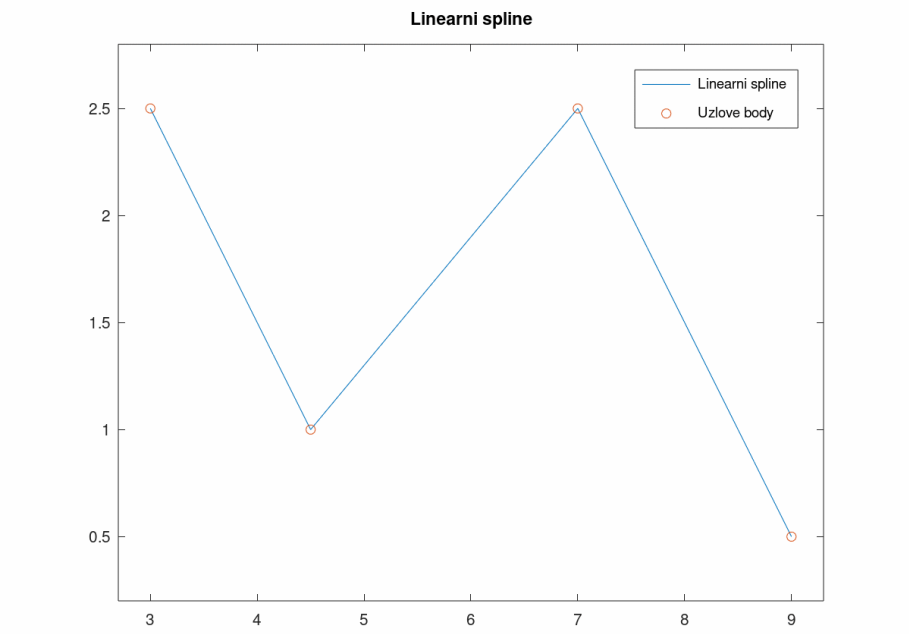
$$d_1 = \frac{2{,}5 - 1}{7 - 4{,}5} = \frac{3}{5}$$

$$d_2 = \frac{0{,}5 - 2{,}5}{9 - 7} = -1$$

Řešení můžeme zapsat takto:

$$S(x) = \begin{cases} 2{,}5 - (x - 3) & , x \in \langle 3; 4{,}5 \rangle \\ 1 + \frac{3}{5}(x - 4{,}5) & , x \in \langle 4{,}5; 7 \rangle \\ 2{,}5 - (x - 7) & , x \in \langle 7; 9 \rangle \end{cases}$$

Graf nalezeného lineárního splinu je na obrázku 4:



Obrázek 4: Lineární spline – příklad 2.1 (vlastní zpracování)

2.2 Kvadratický spline

Podle Chapry [5] je *kvadratický spline* definován takto:

Předpokládejme, že $\{(x_k, y_k)\}_{k=0}^{n}$ *je n + 1 bodů, které splňují podmínku:*

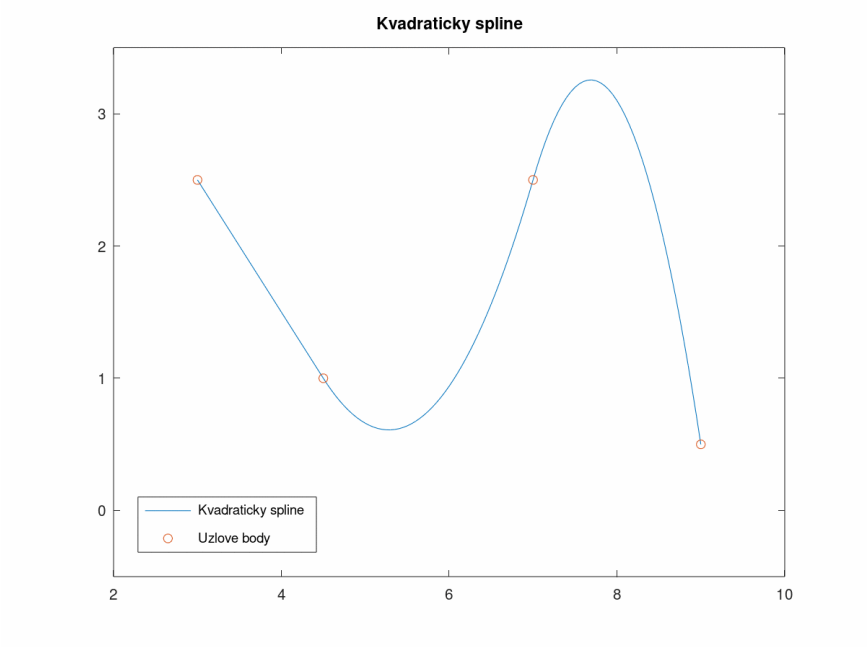
$$a = x_0 < x_1 < \cdots < x_n = b \qquad (6)$$

pak funkce S(x) se nazývá kvadratický spline, jestliže existuje n polynomů 2.stupně $S_k(x)$ *s koeficienty* $s_{k,0}, s_{k,1}, s_{k,2}$, *které splňují následující vlastnosti:*

1. $S(x) = S_k(x) = s_{k,0} + s_{k,1}(x - x_k) + s_{k,2}(x - x_k)^2 \; pro \; x \in \langle x_k, x_{k+1} \rangle, k = 0, 1, 2, \ldots, n-1.$
2. $S(x_k) = y_k$ pro $k = 0, 1, 2, \ldots, n.$
3. $S_k(x_{k+1}) = S_{k+1}(x_{k+1})$ pro $k = 0, 1, 2, \ldots, n-2.$
4. $S_k'(x_{k+1}) = S_{k+1}'(x_{k+1})$ pro $k = 0, 1, 2, \ldots, n-2.$

Podle definice tvoří *kvadratický spline* n polynomů 2. stupně (parabol). U každého mnohočlenu je zapotřebí vypočítat 3 koeficienty ($s_{k,0}$, $s_{k,1}$, $s_{k,2}$), 3n neznámých. Vlastnost 2 uvádí, že spline prochází uzlovými body (n + 1 podmínek). Oproti lineárnímu splinu, u kterého stačilo, aby platilo $S_k(x_{k+1}) = S_{k+1}(x_{k+1}), \text{pro } k = 0,1,\ldots,n-2$ (vlastnost 3, n–1 podmínek), musí mít tento typ splinu ještě jednu

podmínku, spojitou první derivaci $S_k^{'}(x_{k+1}) = S_{k+1}^{'}(x_{k+1})$. Spojitost první derivace zaručuje, že *interpolační spline* S(x) nebude obsahovat žádné ostré hrany (vlastnost 4, n–1 podmínek). Sečtením podmínek dostaneme: *n + 1 + n – 1 + n – 1 = 3n - 1*. To znamená, že jeden koeficient můžeme sami určit ($s_{0,2}$). V případě, že se bude rovnat nule, bude první dílčí mnohočlen úsečka – viz obrázek 5.



Obrázek 5: Kvadratický spline (vlastní zpracování)

Pro výpočet zbylých 3n – 1 neznámých využijeme definici. Do rovnice k-tého dílčího polynomu z první vlastnosti *kvadratického splinu* dosadíme x_k:

$$S_k(x_k) = s_{k,0} + s_{k,1}(x_k - x_k) + s_{k,2}(x_k - x_k)^2 \tag{7}$$

Dostaneme, že $S_k(x_k) = s_{k,0}$, to znamená podle druhé vlastnosti:

$$s_{k,0} = y_k \tag{8}$$

Tím se nám počet neznámých redukuje na 2n – 1 ($s_{k,1}, k \in \{0, 1, \dots, n-1\}, s_{k,2}, k \in \{1, \dots, n-1\}$).

Dále využijeme toho, že dílčí polynomy na sebe spojitě navazují ($S_k(x_{k+1}) = S_{k+1}(x_{k+1})$):

$$s_{k,0} + s_{k,1}(x_{k+1} - x_k) + s_{k,2}(x_{k+1} - x_k)^2 = s_{k+1,0} + s_{k+1,1}(x_{k+1} - x_{k+1}) + s_{k+1,2}(x_{k+1} - x_{k+1})^2 \tag{9}$$

Označíme-li délku intervalu $\langle x_k, x_{k+1} \rangle$ jako h_k a $\Delta y_k = y_{k+1} - y_k$ pak můžeme (9) zapsat takto:

$$s_{k,1}h_k + s_{k,2}{h_k}^2 = \Delta y_k, k = 0, 1, 2, \ldots, n-1 \quad (10)$$

Tímto jsme získali soustavu *n* rovnic o 2n – 1 neznámých (koeficient $s_{0,2}$ určíme sami). Tuto soustavu vyřešit neumíme, protože obsahuje příliš mnoho neznámých. Pro rozšíření soustavy využijeme poslední vlastnost *kvadratického splinu* – spojitost první derivace v uzlových bodech. Derivací polynomu druhého stupně dostaneme:

$$S_k^{'}(x) = s_{k,1} + 2s_{k,2}(x - x_k) \quad (11)$$

Dosadíme do čtvrté vlastnosti ($S_k^{'}(x_{k+1}) = S_{k+1}^{'}(x_{k+1})$):

$$s_{k,1} + 2s_{k,2}(x_{k+1} - x_k) = s_{k+1,1} + 2s_{k+1,2}(x_{k+1} - x_{k+1}) \quad (12)$$

Po úpravě:

$$s_{k,1} + 2s_{k,2}h_k - s_{k+1,1} = 0, k = 0, 1, 2, \ldots, n-2 \quad (13)$$

Dostáváme další soustavu rovnic, tentokrát n-1 rovnic o n + 2 neznámých. Spojené soustavy lineárních algebraických rovnic (10) a (13) můžeme zapsat pomocí matic:

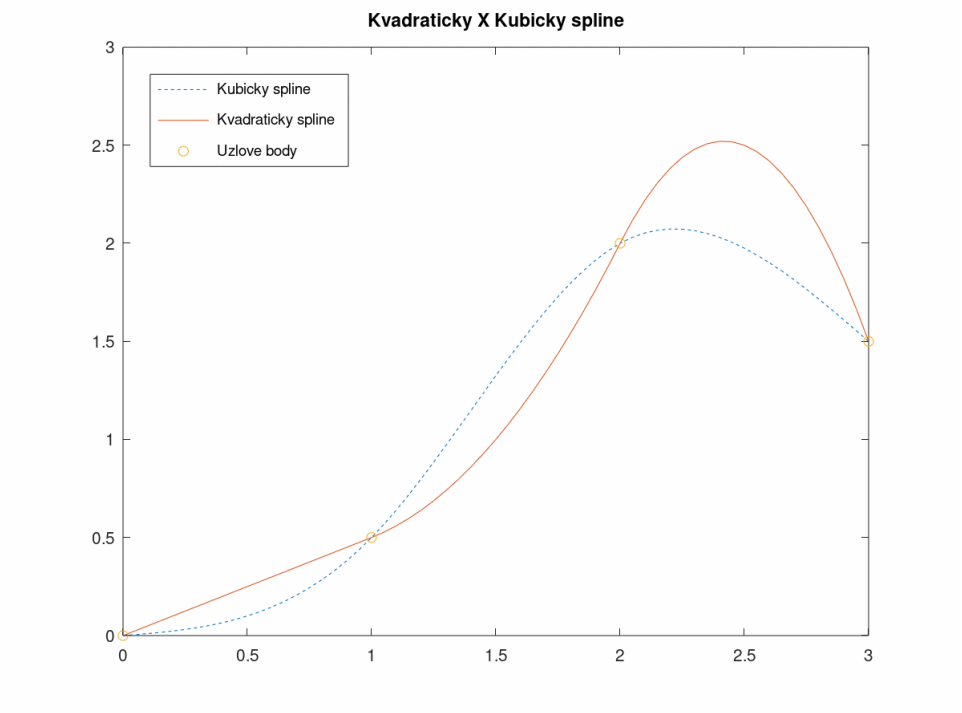
$$Hs = u \quad (14)$$

Hledané koeficienty *kvadratického splinu* tvoří vektor *s*, kde

$$\begin{pmatrix} h_0 & 0 & 0 & 0 & 0 & \cdots & \cdots & \cdots & 0 & 0 \\ 0 & h_1 & h_1^2 & 0 & 0 & \cdots & \cdots & \cdots & 0 & 0 \\ 0 & 0 & 0 & h_2 & h_2^2 & \cdots & \cdots & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & \cdots & \cdots & h_{n-1} & h_{n-1}^2 \\ 1 & -1 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 2h_1 & -1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2h_2 & -1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 1 & 2h_{n-1} & -1 \end{pmatrix} \begin{pmatrix} s_{0,1} \\ s_{1,1} \\ s_{1,2} \\ \vdots \\ \vdots \\ s_{k,1} \\ s_{k,2} \\ \vdots \\ \vdots \\ s_{n-1,1} \\ s_{n-1,2} \end{pmatrix} = \begin{pmatrix} \Delta y_0 - s_{0,2}h_0^2 \\ \Delta y_1 \\ \Delta y_2 \\ \vdots \\ \Delta y_{n-1} \\ -2s_{0,2}h_0 \\ 0 \\ \vdots \\ \vdots \\ 0 \\ 0 \end{pmatrix} \quad (15)$$

Výše uvedená soustava má právě jedno řešení.

Nevýhodou kvadratického splinu je to, že polynom 2. stupně není zas tak „tvárný“, interpolační křivka se může od původní v některých případech příliš vzdálit – viz obrázek 6. Je to dáno tím, že parabola může být pouze na jednom intervalu konvexní nebo konkávní. Tento problém řeší použití interpolačního splinu 3. řádu – kubický spline.



Obrázek 6: Kvadratický X Kubický spline (vlastní zpracování)

Příklad 2.2

Máme vypočítat koeficienty kvadratického splinu zadaného body z tabulky 1. Koeficient $s_{0,2} = 2$.

Řešení

K naplnění matice H a vektoru u z rovnice (15) budeme muset vypočítat délky jednotlivých intervalů h_k a rozdíly funkčních hodnot v uzlových bodech y_k:

$$h_0 = 4{,}5 - 3 = 1{,}5$$
$$h_1 = 7 - 4{,}5 = 2{,}5$$
$$h_2 = 9 - 7 = 2$$
$$\Delta y_0 - {h_0}^2 s_{0,2} = 1 - 2{,}5 - 1{,}5 - 1{,}5^2 \cdot 2 = -6$$
$$\Delta y_1 = 2{,}5 - 1 = 1{,}5$$
$$\Delta y_2 = 0{,}5 - 2{,}5 = -2$$
$$-2 s_{0,2} h_0 = -2 \cdot 2 \cdot 1{,}5 = -6$$

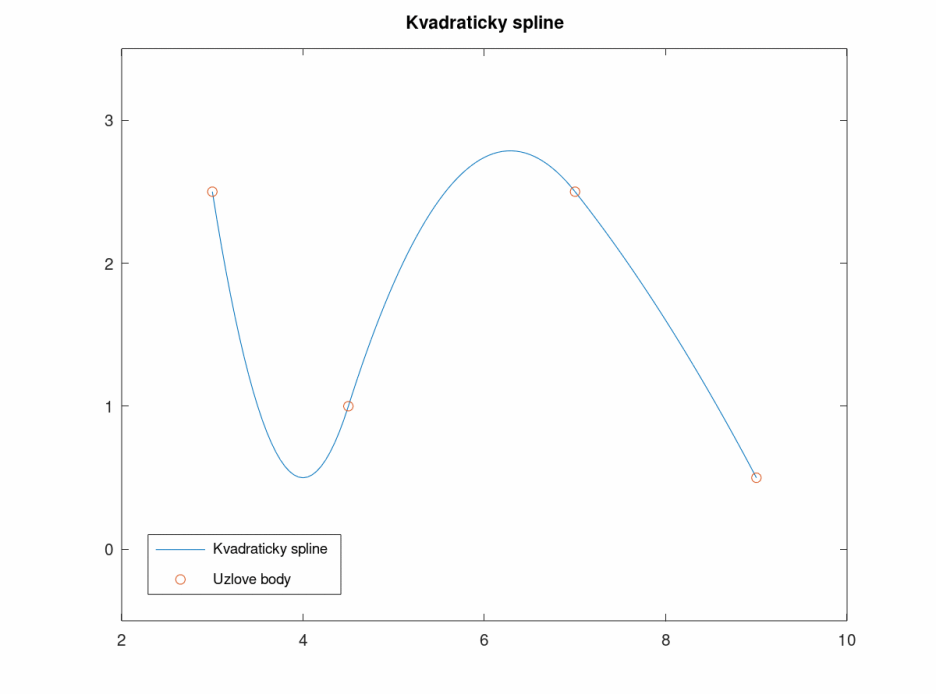
Dosadíme do (15):

$$\begin{pmatrix} 1{,}5 & 0 & 0 & 0 & 0 \\ 0 & 2{,}5 & 6{,}25 & 0 & 0 \\ 0 & 0 & 0 & 2 & 4 \\ 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & 5 & -1 & 0 \end{pmatrix} \begin{pmatrix} s_{0,1} \\ s_{1,1} \\ s_{1,2} \\ s_{2,1} \\ s_{2,2} \end{pmatrix} = \begin{pmatrix} -6 \\ 1{,}5 \\ -2 \\ -6 \\ 0 \end{pmatrix}$$

Vyřešením soustavy získáme koeficienty dílčích polynomů *kvadratického splinu.* Spline pak můžeme zapsat takto:

$$S(x) = \begin{cases} 2{,}5 - 4(x-3) + 2(x-3)^2 & , x \in \langle 3; 4{,}5\rangle \\ 1 + 2(x-4{,}5) - 0{,}56(x-4{,}5)^2 & , x \in \langle 4{,}5; 7\rangle \\ 2{,}5 - 0{,}8(x-7) - 0{,}1(x-7)^2 & , x \in \langle 7; 9\rangle \end{cases}$$

Grafické znázornění splinu:



Obrázek 7: Kvadratický spline – příklad 2.2 (vlastní zpracování)

2.3 Kubický spline

Definice kubického splinu [2]:

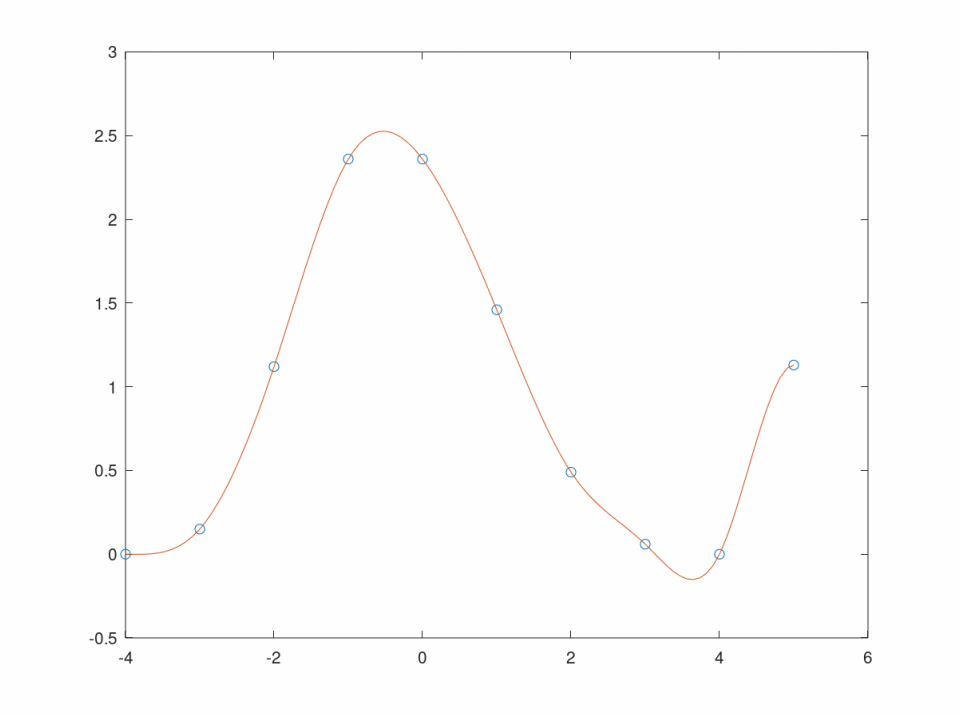
Předpokládejme, že $\{(x_k, y_k)\}_{k=0}^{n}$ *je n + 1 bodů, které splňují podmínku:*

$$a = x_0 < x_1 < \cdots < x_n = b \qquad (16)$$

pak funkce S(x) se nazývá kubický spline, jestliže existuje n polynomů 3.stupně $S_k(x)$ *s koeficienty* $s_{k,0}, s_{k,1}, s_{k,2}, s_{k,3}$*, které splňují následující vlastnosti:*

1. $S(x) = S_k(x) = s_{k,0} + s_{k,1}(x - x_k) + s_{k,2}(x - x_k)^2 + s_{k,3}(x - x_k)^3$, pro $x \in \langle x_k, x_{k+1}\rangle, k = 0, 1, 2, \dots, n-1.$
2. $S(x_k) = y_k$ pro $k = 0, 1, 2, \dots, n.$
3. $S_k(x_{k+1}) = S_{k+1}(x_{k+1})$ pro $k = 0, 1, 2, \dots, n-2.$
4. $S_k'(x_{k+1}) = S_{k+1}'(x_{k+1})$ pro $k = 0, 1, 2, \dots, n-2.$

5. $S_k''(x_{k+1}) = S_{k+1}''(x_{k+1})$ pro $k = 0, 1, 2, \dots, n-2$.



Obrázek 8: Kubický spline (vlastní zpracování)

Vlastnost 1 říká, že $S(x)$ je funkce skládající se po částech z polynomů 3. stupně. Vlastnost 2 uvádí, že po částech kubické polynomy interpolují zadané body $x_0, \dots, x_n$. Aby byla splněna vlastnost 3 musí být funkce $S(x)$ spojitá. Předposlední vlastnost uvádí, že kubické mnohočleny na sebe hladce navazují, že graf funkce $S(x)$ neobsahuje ostré hrany. Podle vlastnosti 5 je spojitá i druhá derivace mezi uzlovými body.

Abychom mohli zkonstruovat *kubický spline*, musíme určit čtyři koeficienty pro každý dílčí polynom. Mnohočlenů je *n*, to znamená *4n* koeficientů pro celý spline. Podle vlastnosti 2 musí být specifikováno *n+1* podmínek. Každá z vlastností 3, 4, 5 požaduje *n-1* podmínek. Když tyto podmínky sečteme: *n+1+3(n-1) = 4n-2*, zjistíme, že 2 podmínky můžeme sami určit. Jde buď o určení hodnoty první nebo druhé derivace v okrajových bodech x_0, x_n.

Jestliže *kubický spline* $S(x)$ tvoří po částech polynomy 3. stupně, pak druhá derivace $S''(x)$ je po částech lineární. Derivujeme-li dvakrát rovnici z vlastnosti 1, dostaneme rovnici přímky:

$$S''(x) = S_k''(x) = 2s_{k,2} + 6s_{k,3}(x - x_k) \qquad (17)$$

Podle [2] jde tato rovnice zapsat i pomocí Lagrangeova interpolačního polynomu stupně 1:

$$S_k^{''}(x) = S^{''}(x_k)\frac{x-x_{k+1}}{x_k-x_{k+1}} + S^{''}(x_{k+1})\frac{x-x_k}{x_{k+1}+x_k} \tag{18}$$

Označíme druhé derivace $S^{''}(x_k) = m_k$, $S^{''}(x_{k+1}) = m_{k+1}$ a délku intervalu $\langle x_k, x_{k+1}\rangle$ jako h_k, dosazením do rovnice (18) dostaneme:

$$S_k^{''}(x) = \frac{m_k}{h_k}(x_{k+1} - x) + \frac{m_{k+1}}{h_k}(x - x_k). \tag{19}$$

Abychom získali $S_k(x)$, dvakrát integrujeme $S_k^{''}(x)$.

$$S_k^{'}(x) = \int S_k^{''}(x)\,dx = -\frac{m_k}{2h_k}(x_{k+1} - x)^2 + \frac{m_{k+1}}{2h_k}(x - x_k)^2 + q_k + p_k \tag{20}$$

$$S_k(x) = \int S_k^{'}(x)\,dx = \frac{m_k}{6h_k}(x_{k+1} - x)^3 + \frac{m_{k+1}}{6h_k}(x - x_k)^3 + q_k(x_{k+1} - x) + p_k(x - x_k) \tag{21}$$

Po prvním integrováním dostaneme první derivaci k-tého polynomu, po druhé integraci samotný k-tý mnohočlen. Prozatím nám toto vyjádření moc platné není. Rovnice (21) obsahuje neznámé m_k, m_{k+1} (druhé derivace v uzlových bodech) a q_q, p_k (konstanty po integrování). Otázkou je, čím je nahradit nebo jak zjistit jejich hodnoty. K tomu využijeme, co již známe, funkční hodnoty v uzlových bodech ($y_k = S_k(x_k), y_{k+1} = S_k(x_{k+1})$ a dosadíme do rovnice (21):

$$y_k = \frac{m_k}{6}h_k^2 + p_k h_k \tag{22}$$

$$y_{k+1} = \frac{m_{k+1}}{6}h_k^2 + q_k h_k \tag{23}$$

Z každé rovnice vyjádříme příslušnou konstantu:

$$p_k = \frac{y_k}{h_k} - \frac{m_k}{6}h_k \tag{24}$$

$$q_k = \frac{y_{k+1}}{h_k} - \frac{m_{k+1}}{6}h_k \tag{25}$$

Dosadíme takto vyjádřené p_k a q_k do rovnice (21):

$$S_k(x) = \frac{m_k}{6h_k}(x_{k+1} - x)^3 + \frac{m_{k+1}}{6h_k}(x - x_k)^3 + \left(\frac{y_k}{h_k} - \frac{m_k}{6}h_k\right)(x_{k+1} - x) + \frac{y_{k+1}}{h_k} - (\frac{m_{k+1}}{6}h_k)(x - x_k). \tag{26}$$

Dosáhli jsme toho, že kubický polynom obsahuje už jenom neznámé druhé derivace (m_k, m_{k+1}). K nalezení těchto hodnot využijeme čtvrtou vlastnost *kubického splinu* ($S_{k-1}^{'}(x_k) = S_k^{'}(x_k)$).

Dosadíme funkční hodnotu x_k do (20):

$$S_k'(x_k) = -\frac{m_k}{3}h_k - \frac{m_{k+1}}{6}h_k + d_k, \text{ kde } d_k = \frac{y_{k+1}-y_k}{h_k} \quad (27)$$

V rovnici (20) snížíme index k o jedna a dosadíme x_k:

$$S_{k-1}'(x_k) = \frac{m_k}{3}h_{k-1} + \frac{m_{k-1}}{6}h_{k-1} + d_{k-1} \quad (28)$$

Když se (27) a (28) rovnají dostáváme:

$$h_{k-1}m_{k-1} + 2m_k(h_{k-1} + h_k) + h_k m_{k+1} = 6(d_k - d_{k-1}) \quad (29)$$

Výraz na pravé straně nahradíme $u_k = 6(d_k - d_{k-1})$:

$$h_{k-1}m_{k-1} + 2m_k(h_{k-1} + h_k) + h_k m_{k+1} = u_k \quad (30)$$

Tento vztah samozřejmě nemůže platit pro $k = 0, 1, \dots, n$. Index *k* půjde od 1 do *n-1*. Dostali jsme tedy soustavu *n-1* lineárních algebraických rovnic o *n+1* neznámých. Aby řešení této soustavy bylo právě jedno musíme určit buďto první nebo druhou derivaci v krajních bodech (m_0, m_n). Možností, podle čeho tyto derivace určit, je více a budou popsány později. Pro větší přehlednost můžeme soustavu rovnic (30) zapsat pomocí matic:

$$Hm = u \quad (31)$$

Hledané druhé derivace tvoří vektor *m*, kde:

$$\begin{pmatrix} 2(h_0+h_1) & h_1 & 0 & 0 & \cdots & \cdots & 0 \\ h_1 & 2(h_1+h_2) & h_2 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & 0 & \ddots & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & h_{n-3} & 2(h_{n-3}+h_{n-2}) & h_{n-2} \\ 0 & 0 & 0 & 0 & 0 & h_{n-2} & 2(h_{n-2}+h_{n-1}) \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \\ \cdots \\ m_{n-2} \\ m_{n-1} \end{pmatrix} = \begin{pmatrix} u_1 - h_0 m_0 \\ u_2 \\ \cdots \\ u_{n-2} \\ u_{n-1} - h_{n-1} m_n \end{pmatrix} \quad (32)$$

Matice *H* je čtvercová, tridiagonální a má právě jedno řešení.

2.3.1 Koeficienty polynomů

Jsou-li vypočítány hodnoty druhých derivací v uzlových bodech $\{m_k\}$, pak už zbývá vypočítat koeficienty jednotlivých dílčích mnohočlenů. K odvození rovnic, ze kterých koeficienty vypočítáme, opět využijeme vlastnosti *kubického splinu* z definice. Do rovnice z vlastnosti 1 dosadíme x_k:

$$S_k(x_k) = s_{k,0} + s_{k,1}(x_k - x_k) + s_{k,2}(x_k - x_k)^2 + s_{k,3}(x_k - x_k)^3 \quad (33)$$

Vidíme, že $S_k(x_k) = s_{k,0}$. To podle druhé vlastnosti znamená:

$$s_{k,0} = y_k \quad (34)$$

Pro další výpočty využijeme rovnici (17). Vytvoříme soustavu 2 rovnic, první získáme dosazením x_k a druhou dosazením x_{k+1}:

$$m_k = 2s_{k,2} + 6s_{k,3}(x_k - x_k)$$
$$m_{k+1} = 2s_{k,2} + 6s_{k,3}(x_{k+1} - x_k) \quad (35)$$

Úpravou první rovnice soustavy dostáváme:

$$s_{k,2} = \frac{m_k}{2} \quad (36)$$

Odečtením první rovnice od druhé a následné úpravě získáme další koeficient:

$$s_{k,3} = \frac{m_{k+1} - m_k}{6h_k} \quad (37)$$

Nyní už zbývá poslední koeficient ($s_{k,1}$). K tomu použijeme - $S_k(x_{k+1}) = y_{k+1}$:

$$y_{k+1} = y_k + s_{k,1}(x_{k+1} - x_k) + \frac{m_k}{2}(x_{k+1} - x_k)^2 + \frac{m_{k+1} - m_k}{6(x_{k+1} - x_k)}(x_{k+1} - x_k)^3 \quad (38)$$

Úpravou rovnice a nahrazením $h_k = x_{k+1} - x_k, d_k = \frac{y_{k+1} - y_k}{h_k}$ dostaneme:

$$s_{k,1} = d_k - \frac{h_k}{6}(2m_k + m_{k+1}) \quad (39)$$

2.3.2 Typy kubických splinů

Z definice *kubického splinu víme*, že k jeho konstrukci musíme sami specifikovat hodnoty prvních nebo druhých derivací v krajních bodech. Možností, podle kterých derivace určit je, jak uvádí [2], pět.

2.3.2.1 Přirozený kubický spline

V případě tohoto splinu se hodnoty druhých derivací v krajních bodech rovnají nule ($m_0 = 0, m_n = 0$). Soustava lineárních algebraických rovnic z (30) potom bude vypadat takto:

$$\begin{aligned} 2m_1(h_0 + h_1) + h_1 m_2 &= u_1 \\ h_{k-1}m_{k-1} + 2m_k(h_{k-1} + h_k) + h_k m_{k+1} &= u_k, \text{ pro } k = 2, 3, \dots, n-2 \\ h_{n-2}m_{n-2} + 2m_{n-1}(h_{n-2} + h_{n-1}) &= u_{n-1} \end{aligned} \quad (40)$$

Příklad 2.3

Máme za úkol nalézt přirozený kubický spline procházející body zadané tabulkou 2.

k	0	1	2	3
x_k	5	6	7	9
$f(x_k)$	1,5	0,5	2	1,5

Tabulka 2: Uzlové body – kubický spline (vlastní zpracování)

Řešení

Nejprve vypočítáme délky jednotlivých intervalů h_k, dále d_k a u_k.

$$h_0 = 6 - 5 = 1$$
$$h_1 = 7 - 6 = 1$$
$$h_2 = 9 - 7 = 2$$
$$d_0 = \frac{y_1 - y_0}{h_0} = \frac{0{,}5 - 1{,}5}{1} = -1$$
$$d_1 = \frac{y_2 - y_1}{h_1} = \frac{2 - 0{,}5}{1} = \frac{3}{2}$$
$$d_2 = \frac{y_3 - y_2}{h_2} = \frac{1{,}5 - 2}{2} = -\frac{1}{4}$$
$$u_1 = 6(d_1 - d_0) = 6\big(1{,}5 - (-1)\big) = 15$$
$$u_2 = 6(d_2 - d_1) = 6(-0{,}25 - 1{,}5) = -10{,}5$$

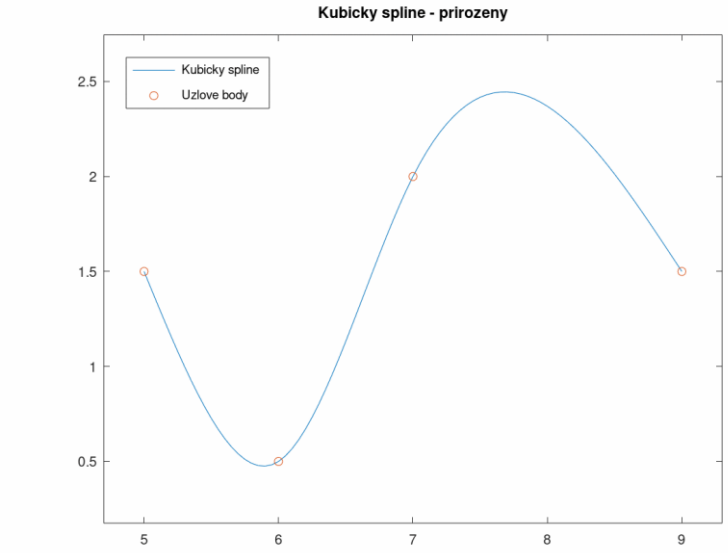
Dosadíme do soustavy (40):

$$2(1+1)m_1 + m_2 = 15$$
$$m_1 + 2(1+2)m_2 = -10{,}5$$

Řešení výše uvedené soustavy je - $m_1 = 4{,}37; m_2 = -2{,}48$. Koeficienty dílčích polynomů vypočítáme dosazením do (34), (36), (37) a (39). Po jejich výpočtu můžeme *přirozený kubický spline* zapsat takto:

$$S(x) = \begin{cases} 1{,}5 - 1{,}73(x-5) + 0{,}73(x-5)^3 & , x \in \langle 5; 6\rangle \\ 0{,}5 + 0{,}46(x-6) + 2{,}18(x-6)^2 - 1{,}14(x-6)^3 & , x \in \langle 6; 7\rangle \\ 2 + 1{,}4(x-7) - 1{,}24(x-7)^2 + 0{,}21(x-7)^3 & , x \in \langle 7; 9\rangle \end{cases}$$

Grafické řešení – obrázek 9:



Obrázek 9: Přirozený kubický spline – příklad 2.3 (vlastní zpracování)

2.3.2.2 Kubický spline s určenými prvními derivacemi

V anglické literatuře označován jako [5] *Clamped cubic spline* (upnutý kubický spline). Jak už z názvu podkapitoly vypovídá, jsou specifikovány hodnoty $S'(x_0)$, $S'(x_n)$. Abychom mohli spline zkonstruovat, potřebujeme pomocí prvních derivací v koncových bodech vypočítat derivace druhé. K výpočtu druhé derivace v počátečním bodě využijeme rovnici (27) a za *k* dosadíme nulu:

$$S'(x_0) = -\frac{m_0}{3}h_0 - \frac{m_1}{6}h_0 + d_0, \text{ kde } d_0 = \frac{y_1 - y_0}{h_0} \tag{41}$$

Z výše uvedené rovnice vyjádříme m_0:

$$m_0 = \frac{3}{h_0}\left(d_0 - S'(x_0)\right) - \frac{m_1}{2} \tag{42}$$

Pro výpočet druhé derivace v koncovém uzlovém bodě splinu použijeme rovnici (28). Za *k* dosadíme *n*:

$$S'(x_n) = \frac{m_n}{3}h_{n-1} + \frac{m_{n-1}}{6}h_{n-1} + d_{n-1}, \text{ kde } d_{n-1} = \frac{y_n - y_{n-1}}{h_{n-1}} \tag{43}$$

Upravíme a vyjádříme m_n:

$$m_n = \frac{3}{h_{n-1}}\left(S'(x_n) - d_{n-1}\right) - \frac{m_{n-1}}{2} \tag{44}$$

Dosazením m_0 a m_n do (30) dostaneme následující soustavu lineárních algebraických rovnic:

$$\begin{aligned} m_1\left(\frac{3}{2}h_0 + 2h_1\right) + h_1 m_2 &= u_1 - 3\left(d_0 - S'(x_0)\right) \\ h_{k-1}m_{k-1} + 2m_k(h_{k-1} + h_k) + h_k m_{k+1} &= u_k, \text{ pro } k = 2, 3, \dots, n-2 \\ h_{n-2}m_{n-2} + m_{n-1}\left(2h_{n-2} + \frac{3}{2}h_{n-1}\right) &= u_{n-1} - \left(S'(x_n) - d_{n-1}\right) \end{aligned} \tag{45}$$

Příklad 2.4

Máme vypočítat *kubický spline s určenými prvními derivacemi* ($S'(x_0) = 0{,}2;\ S'(x_3) = -1$) procházející body zadané tabulkou 2.

Řešení

Pro vyřešení využijeme již spočítané hodnoty h_k, d_k a u_k z předchozího příkladu. Dosadíme je do soustavy rovnic (45):

$$\left(\frac{3}{2} + 2\right)m_1 + m_2 = 15 - 3(-1 - 0{,}2) = 18{,}6$$

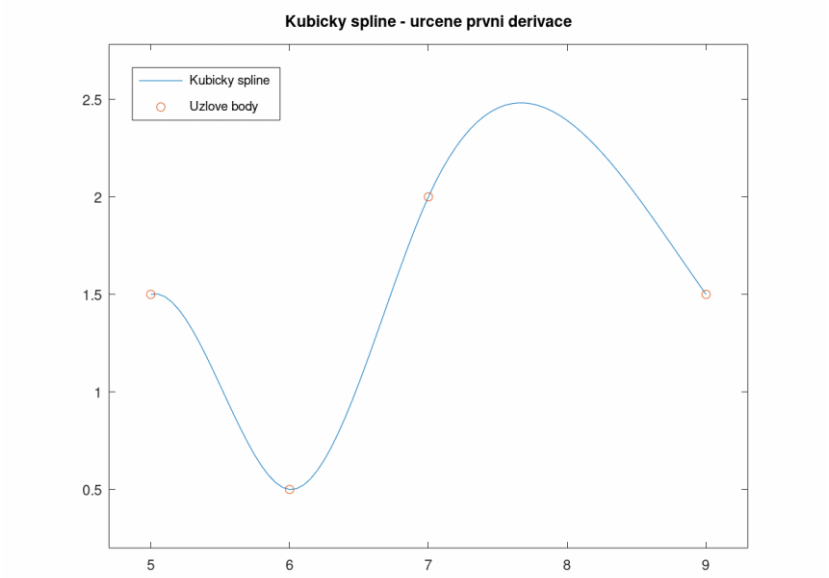
$$m_1 + \left(2 + 2 \cdot \frac{3}{2}\right)m_2 = -10{,}5 - 3\left(-1 - (-0{,}25)\right) = -9{,}75$$

Vyřešením soustavy dostaneme, že $m_1 = 6{,}14; m_2 = -2{,}88$. Hodnoty druhých derivací v krajních bodech získáme dosazením do (42) a (44) - $m_0 = -6{,}67; m_3 =$

0,31. Koeficienty splinu vypočítáme dosazením druhých derivací do (34), (36), (37) a (39). *Kubický spline s určenými prvními derivacemi* pak můžeme zapsat takto:

$$S(x) = \begin{cases} 1{,}5 + 0{,}2(x-5) - 3{,}33(x-5)^2 + 2{,}13(x-5)^3 & , x \in \langle 5; 6\rangle \\ 0{,}5 - 0{,}07(x-6) + 3{,}07(x-6)^2 - 1{,}5(x-6)^3 & , x \in \langle 6; 7\rangle \\ 2 + 1{,}56(x-7) - 1{,}44(x-7)^2 + 0{,}27(x-7)^3 & , x \in \langle 7; 9\rangle \end{cases}$$

Graf slinu je znázorněn na obrázku 10.



Obrázek 10: Kubický spline s určenými prvními derivacemi – příklad 2.4 (vlastní zpracování)

2.3.2.3 Kubický spline s konstantními druhými derivacemi v koncových intervalech

Tento typ splinu je podle [2] definován:

Existuje právě jeden kubický spline, pro který platí $S''(x) \equiv 0$ *na intervalu* $\langle x_0, x_1\rangle$ *a* $S''(x) \equiv 0$ *na intervalu* $\langle x_{n-1}, x_n\rangle$.

Definice říká, že m_0 a m_1, respektive m_{n-1} a m_n, mají zbytek po dělení stejným číslem rovný nule. To může nastat pouze v jediném případě. A to když $m_0 = m_1$ a $m_{n-1} = m_n$. Soustava lineárních algebraických rovnic (30) bude po dosazení vypadat takto:

$$\begin{array}{ccc} m_1(3h_0 + 2h_1) + h_1 m_2 & = & u_1 \\ h_{k-1}m_{k-1} + 2m_k(h_{k-1} + h_k) + h_k m_{k+1} & = & u_k, \text{ pro } k = 2, 3, \dots, n-2 \\ h_{n-2}m_{n-2} + m_{n-1}(2h_{n-2} + 3h_{n-1}) & = & u_{n-1} \end{array} \tag{46}$$

Příklad 2.5

Naším úkolem je najít *kubický spline s konstantními druhými derivacemi v koncových intervalech* procházející body zadané tabulkou 2.

Řešení

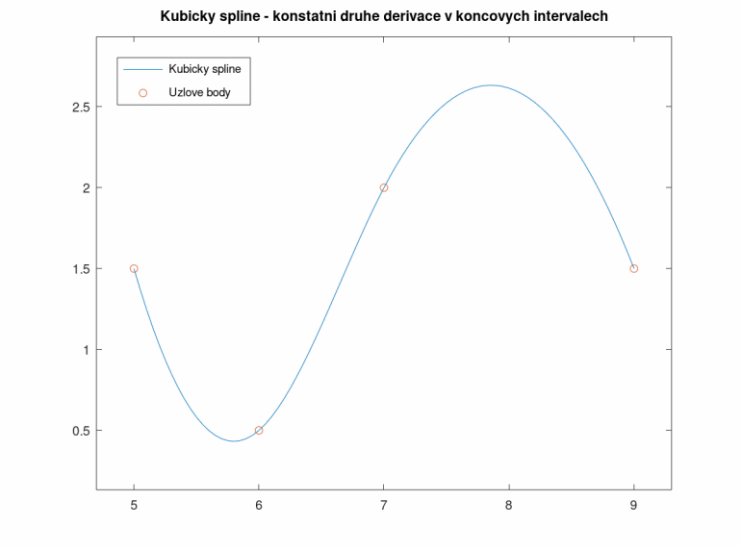
Protože spline prochází stejnými body jako v příkladu 2.3, využijeme již spočítané h_k, d_k a u_k. Druhé derivace ve vnitřních uzlových bodech vypočítáme dosazením do soustavy rovnic (46):

$$(3+2)m_1 + m_2 = 15$$
$$m_1 + (2 + 3 \cdot 2)m_2 = -10{,}5$$

Řešení soustavy je - $m_1 = 3{,}35\,; m_2 = -1{,}73$. Hodnota druhé derivace v prvním bodě se rovná druhé derivaci v bodě druhém (respektive v posledním bodě se rovná hodnotě v předposledním) - $m_0 = 3{,}35\,; m_3 = -1{,}73$. Koeficienty splinu pak vypočítáme dosazením do (34), (36), (37) a (39). Zápis splinu:

$$S(x) = \begin{cases} 1{,}5 - 2{,}67(x-5) + 1{,}67(x-5)^2 & , x \in \langle 5; 6\rangle \\ 0{,}5 + 0{,}67(x-6) + 1{,}67(x-6)^2 - 0{,}85(x-6)^3 & , x \in \langle 6; 7\rangle \\ 2 + 1{,}48(x-7) - 0{,}87(x-7)^2 & , x \in \langle 7; 9\rangle \end{cases}$$

Graf nalezeného splinu je na obrázku 11.



Obrázek 11: Kubický spline s konstantními druhými derivacemi v koncových intervalech – příklad 2.5 (vlastní zpracování)

2.3.2.4Kubický spline se specifikovanými druhými derivacemi

U tohoto typu kubického splinu přímo určujeme hodnoty druhých derivací v krajních bodech - $m_0 = S''(x_0), m_n = S''(x_n)$. Hledanou soustavu rovnic získáme dosazením do (30):

$$\begin{array}{rcl} 2m_1(h_0+h_1)+h_1m_2 & = & u_1 - h_0S''(x_0) \\ h_{k-1}m_{k-1}+2m_k(h_{k-1}+h_k)+h_km_{k+1} & = & u_k, \text{pro } k=2,3,\dots,n-2 \\ h_{n-2}m_{n-2}+2m_{n-1}(h_{n-2}+h_{n-1}) & = & u_{n-1} - h_{n-1}S''(x_n) \end{array} \quad (47)$$

Příklad 2.6

Máme nalézt *kubický spline se specifikovanými druhými derivacemi* $S''(x_0) = -0{,}3, S''(x_3) = 3{,}3.$ procházející body zadané tabulkou 2.

Řešení

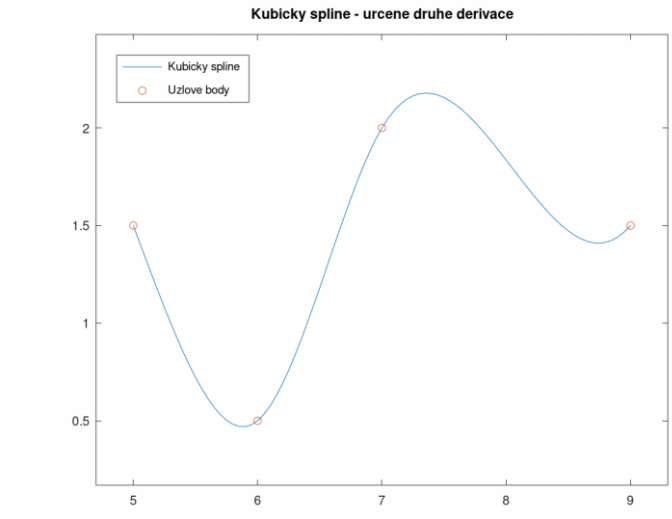
Spline prochází stejnými body jako v příkladě 2.3, takže využijeme už spočítané hodnoty h_k, d_k a u_k. Ty dosadíme do soustavy (47):

$$2(1+1)m_1 + m_2 = 15 - 1\cdot(-0{,}3) = 15{,}3$$
$$m_1 + 2(1+2)m_2 = -10{,}5 - 2\cdot 3{,}3 = -17{,}1$$

Vyřešením dostaneme, že $m_1 = 4{,}73\,; m_2 = -3{,}64$. Koeficienty splinu získáme dosazením do rovnic do (34), (36), (37) a (39). Spline pak můžeme zapsat takto:

$$S(x) = \begin{cases} 1{,}5 - 1{,}69(x-5) - 0{,}15(x-5)^2 + 0{,}84(x-5)^3 & , x \in \langle 5;6\rangle \\ 0{,}5 + 0{,}53(x-6) + 2{,}37(x-6)^2 - 1{,}4(x-6)^3 & , x \in \langle 6;7\rangle \\ 2 + 1{,}08(x-7) - 1{,}82(x-7)^2 + 0{,}58(x-7)^3 & , x \in \langle 7;9\rangle \end{cases}$$

Graf funkce je na obrázku 12.



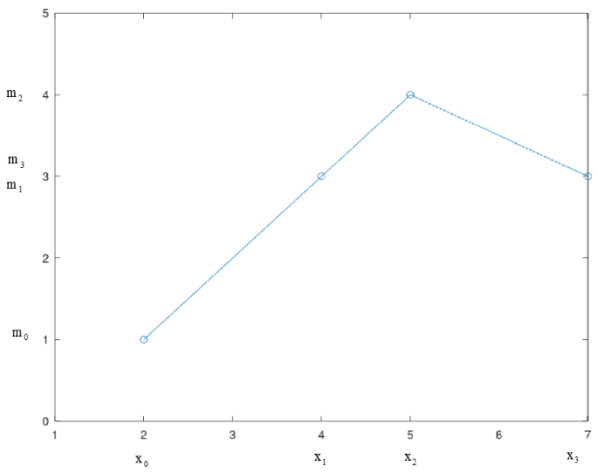
Obrázek 12: Kubický spline s určenými druhými derivacemi – příklad 2.6 (vlastní zpracování)

2.3.2.5 Extrapolovaný spline

Pro výpočet druhých derivací v krajních bodech je u toho splinu použita metoda extrapolace.

Extrapolace je podle [5] aproximační metoda, která odhaduje hodnoty funkce mimo interval známých hodnot. Jde vlastně o opak interpolace, ta oproti extrapolaci hledá hodnoty funkce v intervalu známých hodnot.

Jestliže je *kubický spline* po částech tvořen polynomy 3. stupně, které na sebe hladce navazují (spojitá první i druhá derivace), pak druhá derivace musí být po částech lineární. V případě extrapolovaného splinu se hodnota druhé derivace v prvním bodě splinu bude nacházet na polopřímce zadanou body x_2 a x_1, v posledním bodě na polopřímce $x_{n-2}x_{n-1}$. (viz obrázek 13).



Obrázek 13: Extrapolace (vlastní zpracování)

Jestliže body $[x_0, m_0], [x_1, m_1], [x_2, m_2]$ leží na jedné přímce, můžeme pro nelezení m_0 použít rovnici přímky: $y = kx + q$. Dosazením druhého a třetího bodu dostaneme soustavu rovnic:

$$\begin{aligned} m_1 &= kx_1 + q \\ m_2 &= kx_2 + q \end{aligned} \tag{48}$$

Z první rovnice vyjádříme q a k získáme odečtením první rovnice od druhé:

$$q = m_1 - kx_1, k = \frac{m_2 - m_1}{x_2 - x_1} \tag{49}$$

Hodnotu druhé derivace v bodě x_0 dostaneme, když do $m_0 = kx_0 + q$ dosadíme q a k z (49). Po úpravě by vyjádření vypadalo takto ($h_0 = x_1 - x_0, h_1 = x_2 - x_1$):

$$m_0 = m_1 + \frac{h_0(m_2 - m_1)}{h_1} \tag{50}$$

Stejným postupem, avšak s body $[x_{n-2}, m_{n-2}], [x_{n-1}, m_{n-1}], [x_n, m_n]$, vypočítáme hodnotu druhé derivace v posledním bodě splinu:

$$m_n = m_{n-1} + \frac{h_{n-1}(m_{n-1} - m_{n-2})}{h_{n-2}} \tag{51}$$

Soustava lineárních algebraických rovnic (30) bude po dosazení m_0 a m_n vypadat takto:

$$\begin{aligned} m_1\left(3h_0 + 2h_1 + \frac{h_0^2}{h_1}\right) + m_2\left(h_1 - \frac{h_0^2}{h_1}\right) &= u_1 \\ h_{k-1}m_{k-1} + 2m_k(h_{k-1} + h_k) + h_k m_{k+1} &= u_k, \text{pro } k = 2, 3, \dots, n-2 \\ m_{n-1}\left(3h_{n-1} + 2h_{n-2} + \frac{h_{n-1}^2}{h_{n-2}}\right) + m_{n-2}\left(h_{n-2} - \frac{h_{n-1}^2}{h_{n-2}}\right) &= u_{n-1} \end{aligned} \tag{52}$$

Příklad 2.7

Naší úkolem je nalézt *extrapolovaný kubický spline* procházející body z tabulky 2.

Řešení

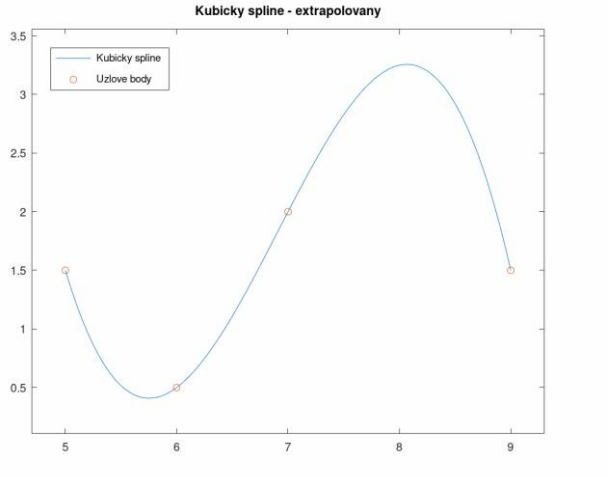
Stejně jako v předešlých příkladech, použijeme již vypočítané hodnoty z příkladu 2.3 - h_k, d_k a u_k. Hodnoty druhých derivací ve vnitřních bodech získáme vyřešením soustavy rovnic (52):

$$\begin{aligned} (3 + 2 + 1)m_1 + (1 - 1)m_2 &= 15 \\ (1 - 2^2)m_1 + (2 + 3 \cdot 2 + 2^2)m_2 &= -10{,}5 \end{aligned}$$

Řešení soustavy je $m_1 = 2{,}5\,; m_2 = -0{,}25$. Hodnoty druhých derivací v koncových bodech pak vypočítáme podle (50) a (51). *Extrapolovaný spline* pak můžeme zapsat:

$$S(x) = \begin{cases} 1{,}5 - 3{,}17(x-5) + 2{,}63(x-5)^2 - 0{,}46(x-5)^3 & , x \in \langle 5; 6\rangle \\ 0{,}5 + 0{,}71(x-6) + 1{,}25(x-6)^2 - 0{,}46(x-6)^3 & , x \in \langle 6; 7\rangle \\ 2 + 1{,}83(x-7) - 0{,}13(x-7)^2 - 0{,}46(x-7)^3 & , x \in \langle 7; 9\rangle \end{cases}$$

Graf splinu je na obrázku 14.



Obrázek 14: Extrapolovaný kubický spline – příklad 2.7 (vlastní zpracování)

3 Matlab – použité příkazy

Tato kapitola se zabývá příkazy, které byly použity při implementaci výpočtu spline funkcí. U každé funkce je nejprve uveden její název, dále co daná funkce dělá, a nakonec je uveden příklad. Při popisu příkazů bylo čerpáno z oficiální dokumentace Matlabu [6] a z [7], [8].

3.1 Příkazy pro grafický výstup

Tyto příkazy byli použity proto, aby ve výstupu výpočetních skritpů nebyly uvedeny jen koeficienty dílčích polynomů splinu, ale také graf interpolační funkce.

plot(X1, Y1, volby1, X2, Y2, volby2, ..., Xn, Yn, volbyN) – příkaz pro vykreslení 2D grafu. Na jednom grafu může být vykresleno 1 až N množin bodů. Každá sada je charakterizovaná trojicí parametrů (Xi, Yi, VolbyI). První dva, jsou povinné. Třetí není a udává, jak mají být daná data vykreslena (barva, styl, ...). Implicitně jsou body jen pospojovány modrými úsečkami. Pro reprezentaci hladké funkce je nezbytné, aby bodů bylo mnoho a byly blízko u sebe.

title(text) – přidá nadpis (parametr text) ke grafu.

ylabel(text) – přidá popis osy Y. Popis je nastaven na hodnotu parametru text.

xlabel(text) – příkaz pomocí, kterého můžeme popsat osu X. Co bude stát v popisu osy je určeno parametrem text.

legend(popis1, popis2, ..., popisN, volby) – přidá popis grafu. Vysvětlivky jsou popořadě přiřazeny k datové sadě, jak byly uvedeny v příkazu *plot()*. Poslední parametr *volby* udává, jak bude popis vypadat a kde bude umístěn.

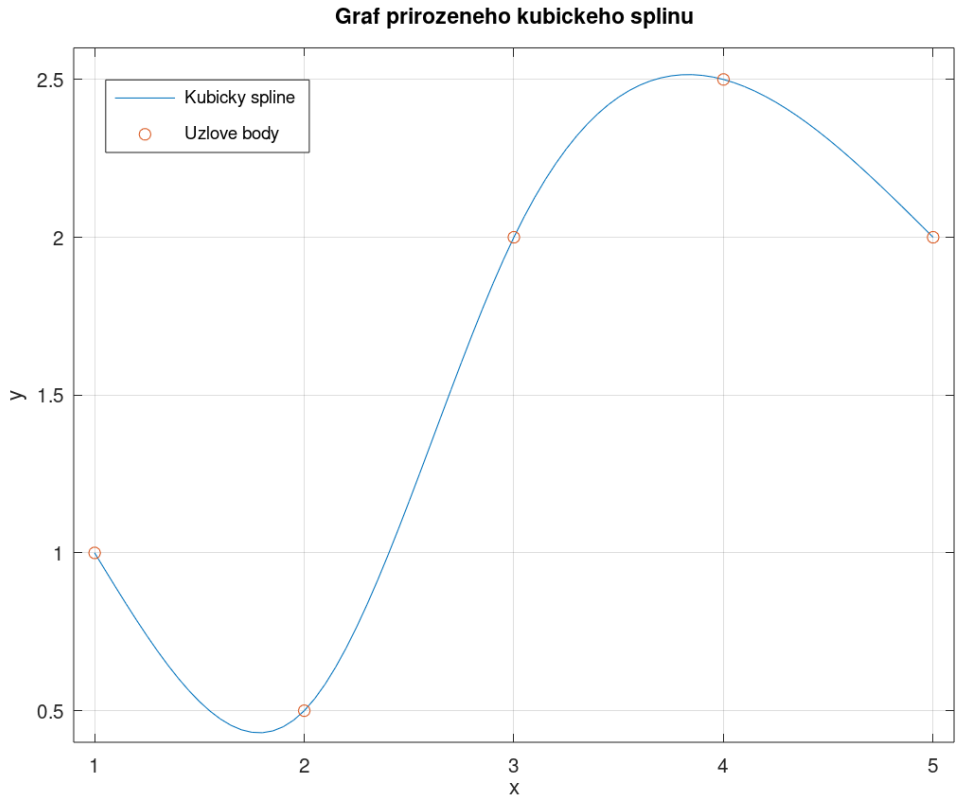
axis(minX, maxX, minY, maxY) – příkaz, který nastaví rozsah oblasti zobrazovaného grafu.

grid on – zobrazí souřadnicovou mřížku.

Příklad (obrázek 15):

```
plot(x, y, xP, yP, '-o', 'LineStyle', 'none')
title('Graf prirozeneho kubickeho splinu')
legend('Kubicky spline','Uzlove body','location','northwest')
axis([(min(xP) - 0.1) (max(xP) + 0.1) (min(yP) - 0.1) (max(yP) +
0.1)])
grid on
xlabel('x')
ylabel('y')
```

Vykreslí graf kubického splinu se zvýrazněnými uzlovými body. Součástí grafu je jeho titulek dále popis os, interpolační funkce a uzlových bodů. Zobrazovaná oblast je dána minimální a maximální hodnotou souřadnic uzlových bodu *x, y*.



Obrázek 15: Příklad vykreslení grafu (vlastní zpracování)

3.2 Příkazy pracující s řetězci

V praktické části jsou použity pro formátování výstupu. Aby neobsahoval pouze číselné hodnoty, ale také informaci, co dané číslo reprezentuje.

strcat(text1, text2, ..., textN) – funkce spojující textové řetězce text1, text2, ..., textN do jednoho.

Příklad: strcat('spline ', 'funkce') vrací 'spline funkce'.

sprintf(format, a1, a2, ..., aN) – formátuje data do textového řetězce nebo vektor znaků. Parametr format je text, do kterého jsou popořadě formátovány hodnoty a1, ... aN. Příklad: sprintf('Rozmery matice jsou: %dx%d', 3, 4) vrací 'Rozmery matice jsou: 3x4'.

num2str(x) – funkce převádějící číslo x na textový řetězec.
Příklad: num2str(9) vrací '9'.

3.3 Ostatní příkazy

Při výpočtech v praktické části bylo nezbytné znát počet uzlových bodů, ať při iteraci některých cyklů nebo při validaci vstupních proměnných. To zajistila funkce length(). Při kontrole vstupních parametrů byli také použity příkazy isnumeric() a isempty(). Potřebné matice a vektory vytvářejí funkce zeros() a cell().

length(x) – vrací délku parametru x. Parametr může být buďto objekt (například textový řetězec) anebo pole objektů.
Příklad: length([5 8 10]) vrací 3.

zeros(dimenze1, dimenze2, ..., dimenzeN) – vytvoří pole (matici) naplněné nulami o rozměrech dimenze1 X dimenze2 X … X dimenzeN.
Příklad: zeros(2, 3) vytvoří matici o 2 řádcích a 3 sloupcích.

cell(dimenze1, dimenze2, ..., dimenzeN) – vytvoří pole „buněk" o rozměrech dimenze1 X dimenze2 X … X dimenzeN. „Buňky" můžou obsahovat různé typy dat a ty nemusí být při deklaraci uvedeny. Toto je výhoda oproti funkci zeros, která vytváří matici obsahující pouze číselné datové typy.
Příklad: cell(2, 4) vytvoří dvourozměrné pole o 2 řádcích a 4 sloupcích.

display(x) – příkaz, který vypíše hodnotu proměnné x do Command Window ve vývojovém prostředí.
Příklad: display([10 12 16]) vypíše '10 12 16'.

isnumeric(x) – zkontroluje, zda vstupní parametr x obsahuje pouze číselné hodnoty. V případě, že ano vrací logickou 1 (true). Jestliže ne, návratová hodnota je 0 (false).
Příklad: isnumeric([1 5 'polynom']) vrací 0, protože ve vstupu se nachází i textový řetězec.

isempty(x) – funkce, která ověřuje, jestli je parametr x prázdný. Návratová hodnota je logická 1 (true) v případě, že ano. Pokud ne funkce vrací 0 (false).
Příklad: isempty([]) vrací true, protože vstup funkce tvoří prázdné pole.

4 Implementace algoritmů výpočtu spline funkcí v Matlabu

Tato kapitola se zabývá implementací algoritmů výpočtu jednotlivých spline funkcí. Kód každého algoritmu je po částech okomentován a u některých částí jsou pro větší přehlednost uvedeny rovnice, které daná část programu řeší.

Oproti teoretické části tu, avšak nesouhlasí indexy. Je to dáno tím, že Matlab neindexuje od 0, jako většina programovacích jazyků, ale od jedné. Pro nás to znamená to, že všechny indexy jsou posunuty o 1 nahoru.

4.1 Lineární spline

Deklarace funkce s názvem *lineární_spline* obsahuje dva vstupní parametry *x, y* – souřadnice uzlových bodů a jednu výstupní proměnnou *d* (směrnice úseček lineárního splinu).

```
function [d] = linearni_spline(x, y)
```

Hned na začátku skriptu jsou kontrolovány vstupní parametry. V případě, že test vrátí hodnotu 0 (false) je program ukončen. Funkce *test_uzlovych_bodu* je popsána v kapitole *Pomocné funkce*.

```
if(test_uzlovych_bodu(x, y) == 0)
    return;
endif
```

Dále je třeba inicializovat proměnnou *n,* reprezentující počet intervalů (kolik mnohočlenů stupně jedna tvoří spline), a vektor *d*, do kterého budeme zapisovat vypočítané směrnice jednotlivých úseček.

```
n = length(x) - 1;
d = zeros(n, 1);
```

Pro výpočet využijeme for-cyklus iterující od 1 do n. Při každém kroku se vypočte směrnice k-té úsečky podle vztahu:

$$d_k = \frac{y_{k+1} - y_k}{x_{k+1} - x_k} \tag{53}$$

Cyklus také vypíše výsledky. Aby index výstupu korespondoval s indexem z teoretické části, je zmenšen o jedna. Nakonec je vykreslen graf splinu funkcí *vykresleni_grafu.*

```
for k = 1 : n
  d(k) = [y(k + 1) - y(k)]/[x(k + 1) - x(k)];
  display(sprintf(strcat('d(%d): ', num2str(d(k))), k - 1));
endfor

[xP, yP] = vykresleni_grafu(d, x, y, 'Linearni');
```

4.2 Kvadratický spline

Funkce *kvadraticky_spline* přijímá tři vstupní proměnné souřadnice uzlových bodů *x, y* a koeficient $s_{0,2}$ – *s1*. Výstupem skriptu je vektor *s* (koeficienty kvadratického splinu).

```
function [s] = kvadraticky_spline(x,y,s1)
```

Před jakýmkoliv výpočtem proběhne kontrola stupních parametrů skripty *test_uzlovych_bodu* a *test_derivace*. Pokud alespoň u jednoho z nich je návratová hodnota rovna 0 (false) je zamezeno pokračování v hlavním algoritmu.

```
if(test_uzlovych_bodu(x, y) == 0)
    return;
  endif

if(test_derivace(s1, 'Kvadraticky') == 0)
    return;
endif
```

Naším cílem je teď vytvoření, naplnění a vyřešení následující maticové rovnice:

$$\begin{pmatrix} h_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & h_2 & h_2^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & h_3 & h_3^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & h_n & h_n^2 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2h_2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2h_3 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ddots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2h_n & -1 \end{pmatrix} \begin{pmatrix} s_{1,1} \\ s_{2,1} \\ s_{2,2} \\ \vdots \\ \vdots \\ s_{k,1} \\ s_{k,2} \\ \vdots \\ \vdots \\ s_{n,1} \\ s_{n,2} \end{pmatrix} = \begin{pmatrix} \Delta y_1 - s_{1,2}h_1^2 \\ \Delta y_2 \\ \Delta y_3 \\ \vdots \\ \Delta y_n \\ -2s_{1,2}h_1 \\ 0 \\ \vdots \\ \vdots \\ 0 \\ 0 \end{pmatrix} \quad (54)$$

Výše uvedená rovnice je totožná s (15), až na to, že jsme pro větší přehlednost zvýšili všechny indexy o jedna.

Pro další postup budeme potřebovat proměnnou, která se bude rovnat počtu intervalů (polynomů druhého stupně), v tomto případě *n*. Délky intervalů budou zapisovány do vektoru *h* a pravou stranu rovnice (54) označíme jako vektor *u*.

```
  n = length(x) - 1;
  h = zeros(n, 1);
  u = zeros(2 * n - 1, 1);
```

Pro naplnění vektorů *h* a *u* použijeme for-cyklus jdoucí od 1 do *n.*

```
for k = 1 : n
  h(k) = x(k + 1) - x(k);
  u(k) = y(k + 1) - y(k);
endfor
```

Vektor *u* ale ještě zdaleka není úplný. Cyklus vypočítal rozdíly funkčních hodnot v uzlových bodech Δy_k. K úplnosti chybí výpočet hodnot na pozici *1* a *n + 1*. Dále vytvoříme matici z rovnice (54) a označíme ji jako *H*, ta je čtvercová o rozměrech *2n-1*. Rovnou přiřadíme hodnoty v prvním a *n + 1* řádku matice.

```
  u(1) = u(1) - s1 * [h(1)]^2;
  u(n + 1) = -2 * s1 * h(1);

  H = zeros(2 * n - 1, 2 * n - 1);
  H(1, 1) = h(1);
  H(n + 1, 1) = 1;
  H(n + 1, 2) = -1;
```

K naplnění zbylých *2n - 3* řádků použijeme cyklus. Ten bude iterovat od 2 do *n*. Pro první polovinu matice nám postačí index samotného cyklu *k*, u druhé poloviny využijeme *n*.

```
for k = 2 : n
    H(k, 2 * k - 2) = h(k);
    H(k, 2 * k - 1) = (h(k))^2;

    if(k < n)
      H(k + n, 2 * k - 2) = 1;
      H(k + n, 2 * k - 1) = 2 * h(k);
      H(k + n, 2 * k) = -1;
    endif
endfor
```

Následuje vyřešení rovnice (54). To zajistí v Matlabu operátor zpětného lomítka.

```
s = H\u;
```

Vektor s v tuto chvíli neobsahuje koeficient $s_{0,2}$. Proto vektor rozšíříme a zařadíme ho na příslušnou pozici. Posledním krokem je vykreslení grafu splinu.

```
tempS = s;
s(1) = tempS(1);
s(2) = s1;
s(3 : 2 * n) = tempS(2 : 2 * n - 1);

[xP, yP] = vykresleni_grafu(s, x, y, 'Kvadraticky');
```

4.3 Kubický spline

U všech typů kubického splinu je nejpodstatnější vyřešení maticové rovnice (32). Aby vektor druhých derivací obsahoval i hodnoty v krajních bodech, rozšíříme rovnici o jeden řádek a jeden sloupec na začátku a na konci:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ h_1 & 2(h_1+h_2) & h_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & h_2 & 2(h_2+h_3) & h_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & h_{n-1} & 2(h_{n-1}+h_n) & h_n \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \\ \vdots \\ \vdots \\ \vdots \\ m_n \\ m_{n+1} \end{pmatrix} = \begin{pmatrix} 0 \\ u_2 \\ u_3 \\ \vdots \\ u_{n-1} \\ u_n \\ 0 \end{pmatrix} \quad (55)$$

4.3.1 Přirozený kubický spline

Vstupními proměnnými funkce *kubicky_prirozeny_spline* jsou souřadnice *x, y* uzlových bodů. Skript pak vrací koeficienty dílčích mnohočlenů v matici *S*.

```
function [S] = kubicky_prirozeny_spline(x, y)
```

Před jakýmkoliv výpočtem se zkontrolují vstupní parametry. Jestliže funkce *test_uzlovych_bodu* vrátí hodnotu 0 (false) je hlavní skript přerušen.

```
if(test_uzlovych_bodu(x, y) == 0)
    return;
endif
```

U každého typu kubického splinu budeme řešit modifikaci (55). Aby bylo zabráněno redundanci, jsou potřebné matice, vektory a proměnné vytvořeny a částečně naplněny pomocným skriptem *kubicky_zaklad.*

```
[h, d, u, H, n] = kubicky_zaklad(x, y);
```

K výpočtu hodnot v matici *H* využijeme cyklus. Protože do prvního a posledního řádku matice už nepotřebujeme zasahovat, postačí, že cyklus bude iterovat od 2 do *n*. K-tá hodnota vektoru *u*, pravé strany rovnice (55), se vypočte podle vztahu:

$$u_k = 6(d_k - d_{k-1}) \tag{56}$$

```
for k = 2 : n
    H(k, k) = 2*(h(k - 1)+h(k));
    H(k, k - 1) = h(k - 1);
    H(k, k + 1) = h(k);
    u(k) = 6 * [d(k) - d(k - 1)];
endfor
```

Poslední částí skriptu je vyřešení maticové rovnice (55) *Hm=u*, vypočítání koeficientů dílčích polynomů funkcí *koeficienty* a vykreslení grafu splinu.

```
m = H\u;

S = koeficienty(x, y, h, m);

[xP, yP] = vykresleni_grafu(S, x, y, 'Prirozeny');
```

4.3.2 Kubický spline s určenými prvními derivacemi

Kromě souřadnic uzlových bodů *x, y* přijímá funkce *kubicky_urcene_prvni* první derivace v koncových bodech splinu *d0* a *dn*. Návratovou hodnotou je matice koeficientů dílčích mnohočlenů *S*.

```
function [S] = kubicky_urcene_prvni(x, y, d0, dn)
```

Aby nenastal nečekaný problém ve výpočtu jsou zde kromě souřadnic testovány i derivace v krajních bodech.

```
if(test_uzlovych_bodu(x, y) == 0)
    return;
endif

if(test_derivace([d0, dn], 'Urcene prvni') == 0)
    return;
endif
```

Inicializujeme potřebné proměnné skriptem *kubicky_zaklad*.

```
[h, d, u, H, n] = kubicky_zaklad(x, y);
```

Změna nastává ve druhém a předposledním řádku soustavy rovnic (55). Ta budou vypadat následovně:

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & \frac{3}{2}h_1 + 2h_2 & h_2 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & h_{n-1} & \frac{3}{2}h_n + 2h_{n-1} & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \\ \vdots \\ m_n \\ m_{n+1} \end{pmatrix} = \begin{pmatrix} 0 \\ u_2 - 3(d_1 - S'(x_1)) \\ \vdots \\ u_n - (S'(x_{n+1}) - d_n) \\ 0 \end{pmatrix} \quad (57)$$

```
H(2,2) = 1.5 * h(1) + 2 * h(2);
H(2,3) = h(2);
u(2) = 6 * [d(2) - d(1)] - 3 * [d(1) - d0];

H(n, n) = 1.5 * h(n) + 2 * h(n - 1);
H(n, n - 1) = h(n - 1);
u(n) = 6 * [d(n) - d(n - 1)] - 3 * [dn - d(n)];
```

Ve zbylých řádcích matice *H* se nic nemění.

```
for k = 3 : n - 1
    H(k, k)=2 * (h(k - 1)+h(k));
    H(k, k - 1)=h(k - 1);
    H(k, k + 1)=h(k);
    u(k) = 6 * [d(k) - d(k - 1)];
endfor
```

Dále je vyřešena rovnice (57).

```
m=H\u;
```

Před výpočtem koeficientů už stačí jen dopočítat druhé derivace v krajních bodech podle:

$$\begin{aligned} m_1 &= \frac{3}{h_1}(d_1 - S'(x_1)) - \frac{m_2}{2} \\ m_{n+1} &= \frac{3}{h_n}(S'(x_{n+1}) - d_n) - \frac{m_n}{2} \end{aligned} \quad (58)$$

```
m(1) = 3/h(1) * (d(1) - d0) - m(2)/2;
m(n + 1) = 3/h(n) * (dn - d(n)) - m(n)/2;
```

Předposledním krokem algoritmu je výpočet koeficientů splinu funkcí *koeficienty*. Skript je zakončen vykreslením grafu.

```
S = koeficienty(x, y, h, m);

[xP, yP] = vykresleni_grafu(S, x, y, ' Urcene prvni derivace');
```

4.3.3 Kubický spline s konstantními druhými derivacemi v koncových intervalech

Skript, který vypočítá koeficienty tohoto splinu se jmenuje *kubicky_konstatni_derivace* a jeho parametrem jsou jen souřadnice uzlových bodů *x, y*. Funkce vrací matici *S* – koeficienty mnohočlenů.

```
function [S] = kubicky_konstantni_derivace(x, y)
```

Následující kód je stejný jako u přirozeného kubického splinu. Nejprve otestujeme vstupní proměnné a vytvoříme proměnné pro další výpočet funkcí *kubicky_zaklad*.

```
if(test_uzlovych_bodu(x, y) == 0)
  return;
endif

[h, d, u, H, n] = kubicky_zaklad(x, y);
```

Opět modifikujeme (55), k tomu využijeme soustavu rovnic (46). Výsledkem je následujíc vztah:

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 3h_1 + 2h_2 & h_2 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & h_{n-1} & 3h_n + 2h_{n-1} & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \\ \vdots \\ m_n \\ m_{n+1} \end{pmatrix} = \begin{pmatrix} 0 \\ u_2 \\ \vdots \\ u_n \\ 0 \end{pmatrix} \tag{59}$$

A znovu se (59) od (55) odlišuje ve druhé a předposledním řádku. Hodnoty v nich vypočítáme následovně:

```
H(2,2) = 3*h(1) + 2*h(2);
H(2,3) = h(2);
u(2) = 6 * [d(2) - d(1)];

H(n, n) = 3*h(n) + 2*h(n - 1);
H(n, n - 1) = h(n - 1);
u(n) = 6 * [d(n) - d(n - 1)];
```

Zbylých *n - 4* řádků naplníme for-cyklem.

```
for k = 3 : n - 1
    H(k, k)=2*(h(k - 1)+h(k));
    H(k, k - 1)=h(k - 1);
    H(k, k + 1)=h(k);
    u(k) = 6 * [d(k) - d(k - 1)];
endfor
```

Hodnoty druhých derivací m_2 až m_n získáme vyřešením (59). Před výpočtem koeficientů splinu funkcí *koeficienty* přiřadíme hodnoty druhých derivací v krajních bodech. Posledním krokem je vykreslení grafu.

```
m = H\u;

m(1) = m(2);
m(n + 1) = m(n);

S = koeficienty(x, y, h, m);

[xP, yP] = vykresleni_grafu(S, x, y, 'Druhe derivace konstatni v
koncovych bodech');
```

4.3.4 Kubický spline se specifikovanými druhými derivacemi

Metoda pro výpočet koeficientů mnohočlenů tohoto splinu kromě souřadnic uzlových bodů přijímá jako vstupní parametry také druhé derivace v koncových bodech.

```
function [S] = kubicky_urcene_druhe(x, y, m0, mN)
```

Hned na začátku skriptu jsou otestovány vstupní proměnné. Jestliže nejsou validní, je zamezeno dalšímu pokračování v programu. Pokud parametry testy projdou, jsou vytvořeny potřebné proměnné pro další výpočet pomocnou funkcí *kubicky_zaklad*.

```
if(test_uzlovych_bodu(x, y) == 0)
    return;
endif

if(test_derivace([m0, mN], 'Urcene druhe') == 0)
    return;
endif

[h, d, u, H, n] = kubicky_zaklad(x, y);
```

Podstatné je upravení maticové rovnice (55) podle soustavy (47).

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 2(h_1+h_2) & h_2 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & h_{n-1} & 2(h_n+h_{n-1}) & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \\ \vdots \\ m_n \\ m_{n+1} \end{pmatrix} = \begin{pmatrix} 0 \\ u_2 - h_1 S''(x_1) \\ \vdots \\ u_n - h_n S''(x_{n+1}) \\ 0 \end{pmatrix} \quad (60)$$

Nejprve jsou vypočítány hodnoty ve druhém a předposledním řádku. Zbytek matice *H* je naplněn cyklem.

```
H(2,2) = 2* [h(1) + h(2)];
H(2,3) = h(2);
u(2) = 6 * [d(2) - d(1)] - h(1) * m0;

H(n, n) = 2 * [h(n) + h(n - 1)];
H(n, n - 1) = h(n - 1);
u(n) = 6 * [d(n) - d(n - 1)] - h(n) * mN;

for k = 3 : n - 1
    H(k, k) = 2 * (h(k - 1) + h(k));
    H(k, k - 1) = h(k - 1);
    H(k, k + 1) = h(k);
    u(k) = 6 * [d(k) - d(k - 1)];
endfor
```

Ke druhým derivacím v uzlových bodech se dostaneme vyřešením (60). A následně jsou přiřazeny hodnoty derivací i v krajních bodech.

```
m = H\u;

m(1) = m0;
m(n + 1) = mN;
```

Koeficienty dílčích polynomů tohoto splinu jsou vypočítány funkcí *koeficienty*. Nakonec je vykreslen graf splinu skriptem *vykresleni_grafu*.

```
S = koeficienty(x, y, h, m);

[xP, yP] = vykresleni_grafu(S, x, y, 'Urcene druhe derivace');
```

4.3.5 Extrapolovaný spline

Funkce pro výpočet extrapolovaného splinu se nazývá *kubicky_extrapolovany* a jako vstupní proměnné má pouze souřadnice uzlových bodů *x, y*.

```
function [S] = kubicky_extrapolovany(x, y)
```

Stejně jako u všech ostatních typů splinů jsou nejprve zkontrolovány vstupní parametry. Pak se vytvoří a spočítají základní proměnné a vektory skriptem *kubicky_zaklad*.

```
if(test_uzlovych_bodu(x, y) == 0)
    return;
endif

[h, d, u, H, n] = kubicky_zaklad(x, y);
```

Klíčové je upravit (55) podle soustavy rovnic (52). Maticová rovnice bude vypadat následovně:

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 3h_1 + 2h_2 + \frac{h_1^2}{h_2} & h_2 - \frac{h_1^2}{h_2} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & h_{n-1} - \frac{h_n^2}{h_{n-1}} & 3h_n + 2h_{n-1} + \frac{h_n^2}{h_{n-1}} & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \\ \vdots \\ m_n \\ m_{n+1} \end{pmatrix} = \begin{pmatrix} 0 \\ u_2 \\ \vdots \\ u_n \\ 0 \end{pmatrix} \quad (61)$$

Nejprve vypočítáme hodnoty na druhém a předposledním řádku. A zbytek cyklem.

```
H(2,2) = 3*h(1) + 2*h(2) + [h(1)]^2/h(2);
H(2,3) = h(2) - [h(1)]^2/h(2);
u(2) = 6 * [d(2) - d(1)];

H(n, n) = 3*h(n) + 2*h(n - 1) + [h(n)]^2/h(n - 1);
H(n, n - 1) = h(n - 1) - [h(n)]^2/h(n - 1);
u(n) = 6 * [d(n) - d(n - 1)];

for k = 3 : n - 1
    H(k, k) = 2 * (h(k - 1) + h(k));
    H(k, k - 1) = h(k - 1);
    H(k, k + 1) = h(k);
    u(k) = 6 * [d(k) - d(k - 1)];
endfor
```

Druhé derivace m_2 až m_n získáme vyřešením rovnice (61). Před výpočtem koeficientů dílčích polynomů ještě vypočítáme derivace v krajních bodech podle:

$$m_1 = m_2 + \frac{h_1(m_3 - m_2)}{h_2}$$
$$m_{n+1} = m_n + \frac{h_n(m_n - m_{n-1})}{h_{n-1}} \tag{62}$$

```
m = H\u;

m(1) = m(2) - [h(1) * (m(3) - m(2))]/h(2);
m(n + 1) = m(n) + [h(n) * (m(n) - m(n - 1))]/h(n - 1);

S = koeficienty(x, y, h, m);

[x1, y1] = vykresleni_grafu(S, x, y, ' Extrapolovany');
```

4.4 Pomocné funkce

V této kapitole jsou popsány pomocné skripty. Byly napsány z důvodu, aby se předešlo opakování určitých částí kódu (výpočet koeficientů, vykreslení grafu funkce, validace vstupních proměnných).

4.4.1 Výpočet koeficientů kubického splinu

Funkce pro výpočet koeficientů má jako vstupní parametry souřadnice uzlových bodů *x, y*, délky intervalů *h* a druhé derivace *m*. Koeficienty budeme zapisovat do matice *S*. Ty se vypočítají podle následujících vztahů.

$$\begin{aligned} s_{k,1} &= y_k \\ s_{k,2} &= d_k - \frac{h_k}{6}(2m_k + m_{k+1}) \\ s_{k,3} &= \frac{m_k}{2} \\ s_{k,4} &= \frac{m_{k+1} - m_k}{6h_k} \end{aligned} \tag{63}$$

```
function [S] = koeficienty(x, y, h, m);

n = length(h);
S = zeros(n, 4);

for k=1:n
    S(k, 1)=y(k);
    S(k, 2)=(y(k+1)-y(k))/h(k)-h(k)/2*m(k)-h(k)/6*(m(k+1)-m(k));
    S(k, 3)=m(k)/2;
    S(k, 4)=(m(k+1)-m(k))/(6*h(k));
endfor
endfunction
```

4.4.2 Testování vstupních dat – uzlové body

Funkce *test_uzlovych_bodu* vrací boolean *t*, který udává, zda jsou vstupní parametry *x, y* validní.

```
function [t] = test_uzlovych_bodu(x, y)

t = true;
```

Nejprve se kontroluje, zda nebyli souřadnice zadány nebo jestli jsou pouze prázdnými poli.

```
if(isempty(x) || isempty(y))
   display('Vstup neobsahuje zadne hodnoty!');
   t = false;
endif
```

Dále se testuje, jestli se délky vektorů obsahující souřadnice sobě rovnají.

```
if(length(x) != length(y))
   display('Pocet souradnic X se neshoduje s poctem souradnic Y!');
   t = false;
endif
```

Pokud parametry *x* a *y* obsahují nějaký textový řetězec je opět vrácena záporná hodnota booleanu *t*.

```
if(isnumeric(x) != 1 || isnumeric(y) != 1)
   display('Vstup neobsahuje pouze numericke hodnoty!');
   t = false;
endif
```

V definici každého spline je uvedeno, že souřadnice *x* musí být seřazeny vzestupně. To zkontrolujeme cyklem procházející celý vektor *x*. Nakonec je vrácen boolean *t*.

```
n = length(x) - 1;
for k = 1 : n
   if(x(k) > x(k + 1))
     display('Souradnice X nejsou serazeny vzestupne!');
     t = false;
   endif
endfor

return;
```

4.4.3 Testování vstupních dat – derivace

Kromě kontroly souřadnic interpolovaných bodů jsou testovány i derivace skriptem *test_derivace*. Ta přijímá dva parametry. Parametr *s1* reprezentuje derivace (u kubického splinu), respektive derivaci u kvadratického splinu. Druhou proměnnou je typu splinu *type*. Návratovou hodnotou skritpu je boolean *t*. Vrací true, pokud jsou vstupní parametry validní.

```
function [t] = test_derivace(s1, type)

t = true;
```

Nejprve se zkontroluje, jestli je vstup tvořen pouze číselnými datovými typy.

```
if(isnumeric(s1) != 1)
     display('Parametr derivaci neni cislo!');
     t = false;
endif
```

Protože u kvadratického splinu je pouze jedna derivace a u kubického dvě. Použijeme pro rozdělení switch-case. Na konci skriptu je vrácen boolean *t*.

```
switch type
    case 'Kvadraticky'

      if(length(s1) != 1)
        display('Parametr s1 musi byt prave jedno cislo');
        t = false;
      endif

    otherwise

      if(length(s1) != 2)
        display('Parametr musi obsahovat prave dve cisla');
        t = false;
      endif

endswitch

return;
```

4.4.4 Vykreslení grafu

Pro vykreslení grafu potřebujeme koeficienty dílčích polynomů *S*, souřadnice uzlových bodů *x, y* a typ splinu *type.*

```
function [xP, yP] = vykresleni_grafu(S, x, y, type)
```

Vykreslované body splinu jsou *xP, yP.* Aby příkaz plot vykreslil hladkou křivku musí být bodů hodně a musí být blízko u sebe. Pro výpočet budeme také potřebovat pomocnou proměnou *temp* pro indexování.

```
n = length(x) - 1;

xP = zeros(20 * (x(end) - x(1)), 1);
yP = zeros(length(xP), 1);

temp = 1;
```

Pro rozdělení podle typ splinu je použit switch-case.

```
switch type
    case 'Linearni'
…

    case 'Kvadraticky'
…
    Otherwise
…
endswitch
```

Výpočet zařídí dva for-cykly. Ve vnějším, jdoucí od 1 do *n* (počet polynomů), jsou přiřazeny do proměnných *k1* a *k2* hodnoty x_k, x_{k+1}. Mezi nimi pak iteruje vnitřní cyklus, ve kterém jsou spočítány funkční hodnoty *yP* podle:

$$S(x) = s_{k,0} + s_{k,1}(x - x_k) \tag{64}$$

```
case 'Linearni'
    for k = 1 : n
      k1 = x(k);
      k2 = x(k + 1);

      for j = k1 : 0.05 : k2
        xP(temp) = j;
        yP(temp) = y(k) + S(k,1) * (j - x(k));
        temp ++;
      endfor
    endfor

titleString = strcat(type, ' spline');
```

Stejný postup je užit u *kvadratického splinu*, akorát že je potřebná další proměnná pro indexování *tempX* a funkční hodnoty *yP* jsou počítány podle vztahu:

$$S(x) = s_{k,0} + s_{k,1}(x - x_k) + s_{k,2}(x - x_k)^2 \tag{65}$$

```
case 'Kvadraticky'
        tempX = tempI = 1;

        CoeficientsString = cell(3, n);

        for k = 1 : n
          k1 = x(k);
          k2 = x(k + 1);

          for j = k1 : 0.05 : k2
            xP(tempX) = j;

            yP(tempX) = y(k) + S(tempI,1) * (j - x(k)) + S(tempI +
1, 1) * (j - x(k))^2;
            tempX ++;
          endfor

            CoeficientsString(1, k) = sprintf(strcat('s(%d,0): ',
num2str(y(k))), k - 1);
            CoeficientsString(2, k) = sprintf(strcat('s(%d,1): ',
num2str(S(tempI))), k - 1);
            CoeficientsString(3, k) = sprintf(strcat('s(%d,2): ',
num2str(S(tempI + 1))), k - 1);

            tempI += 2;

        endfor
        display(CoeficientsString);
        titleString = strcat(type, ' spline');
```

Kubický spline je vypočítán podle:

$$S(x) = s_{k,0} + s_{k,1}(x - x_k) + s_{k,2}(x - x_k)^2 + s_{k,3}(x - x_k)^3 \tag{66}$$

```
CoeficientsString = cell(4, n);

for k = 1 : n
    k1 = x(k);
    k2 = x(k + 1);
    for j = k1 : 0.05 : k2
      xP(temp) = j;
      yP(temp) = S(k,1) + S(k,2) * [j - x(k)] + S(k,3) * [j -
x(k)]^2 + S(k,4) * [j - x(k)]^3;
       temp ++;
    endfor

    CoeficientsString(1, k) = sprintf(strcat('s(%d,0): ',
num2str(S(k, 1))), k - 1);
    CoeficientsString(2, k) = sprintf(strcat('s(%d,1): ',
num2str(S(k, 2))), k - 1);
    CoeficientsString(3, k) = sprintf(strcat('s(%d,2): ',
num2str(S(k, 3))), k - 1);
```

```
    CoeficientsString(4, k) = sprintf(strcat('s(%d,3): ',
num2str(S(k, 4))), k);
endfor

display(CoeficientsString);
titleString = strcat('Kubicky spline - ',type);
```

Nakonec je vykreslen graf dané spline funkce, ve kterém jsou zvýrazněny uzlové body.

```
plot(xP, yP, x, y, '-o','LineStyle','none');
title(titleString);
legend(titleString,'Uzlove body','location','northwest');
axis([(min(x) - .8) (max(x) + .8) (min(y) - .8) (max(y) + .8)]);
grid on;
xlabel('X');
ylabel('Y');
```

4.4.5 Inicializace proměnných u kubického splinu

Existuje 5 typů kubického splinu a u každého vychází výpočet druhých derivací z maticové rovnice (55). U všech typů musíme vytvořit a spočítat vektory h_k,d_k podle:

$$h_k = x_{k+1} - x_k$$
$$d_k = \frac{y_{k+1} - y_k}{x_{k+1} - x_k} \tag{67}$$

Dále musíme vytvořit matici H a vektor u. Aby bylo zabráněno redundancím v kódu napíšeme pomocný skript, který toto všechno zařídí – *kubicky_zaklad*.

```
function [h, d ,u, H, n] = kubicky_zaklad(x, y)

    n = length(x) - 1;
    h = zeros(n, 1);
    d = zeros(n, 1);

    for k=1:n
     h(k) = x(k+1) - x(k);
     d(k) = [y(k+1) - y(k)]/h(k);
    endfor

    H = zeros(n + 1,n + 1);
    H(1,1) = 1;
    H(n + 1,n + 1) = 1;

    u = zeros(n + 1, 1);

endfunction
```

5 Aplikace spline funkcí

Tato kapitola má za cíl uvést využití spline funkcí. V praxi se první dva typy splinů, lineární a kvadratický, příliš nevyužívají. Jak uvádí Mathews a Fink v [2] jsou tyto funkce nevhodné, protože nejsou dostatečně hladké (kvadratický spline), respektive vůbec (lineární spline). Nejvíce se tedy používá spline kubický. Níže jsou uvedeny dvě úlohy využívající tuto interpolační metodu.

Úloha č.1 – Teplotní stratifikace

Tato aplikace byla převzata z [5].

Nástin problematiky:

V mírném podnebném pásu se během léta voda v jezerech rozděluje na 2 vrstvy podle teploty, *Epilimnion* a *Hypolimnion. Epilimnion* je teplejší, má vyšší koncentraci kyslíku a ve většině případech i vyšší pH. Absorbuje také nejvíce slunečního záření. *Hypolimnion* je oproti tomu chladnější a má větší hustotu. U dna některých jezer dochází k výraznému snížení saturace kyslíkem. Přechod mezi těmito dvěma vrstvami se nazývá *Termoklina* (v angličtině *thermocline*). Ta je za příznivých podmínek viditelná i pouhým okem.

Takovéto rozdělení má velký význam pro přírodovědce, protože některé organismy jsou citlivé na množství kyslíku ve vodě nebo na teplotu. Více informací o teplotní stratifikaci lze získat například z [9].

Zadání úlohy:

Naším úkolem je určit, v jaké hloubce se nachází *termoklina* v Platte Lake ve Spojených státech amerických z naměřených hodnot uvedených v tabulce 3.

***h* [m]**	0	2,3	4,9	9,1	13,7	18,3	22,9	27,2
***t* [°C]**	22,8	22,8	22,8	20,6	13,9	11,7	11,1	11,1

Tabulka 3: Teplota vody v závislosti na hloubce – zdroj [5]

Řešení:

Pozice *termokliny* je definovaná, jako inflexní bod funkce, jejíž definičním oborem je hloubka ***h*** a oborem hodnot naměřená teplota ***t***. To podle [1] znamená, že hledáme takový bod, který je na svém levém okolí ryze konkávní (respektive konvexní) a na pravém okolí ryze konvexní (respektive konkávní). Pak se takovém bodě druhá derivace rovná nule:

$$\frac{d^2t}{dh^2} = 0 \tag{68}$$

Funkce je na intervalu ryze konvexní, když v každém jeho vnitřním bodě je druhá derivace větší než 0. Ryze konkávní je v opačném případě, tedy že $f^{''} < 0$.

Abychom našli takový bod potřebujeme nejprve proložit body z tabulky 3 funkcí. A tou bude kubický přirozený spline. Po výpočtu délek intervalů h_k a diferencí d_k, u_k bude maticová rovnice (32) vypadat následovně:

$$\begin{pmatrix} 9{,}8 & 2{,}6 & 0 & 0 & 0 & 0 \\ 2{,}6 & 13{,}6 & 4{,}2 & 0 & 0 & 0 \\ 0 & 4{,}2 & 17{,}6 & 4{,}6 & 0 & 0 \\ 0 & 0 & 4{,}6 & 18{,}4 & 4{,}6 & 0 \\ 0 & 0 & 0 & 4{,}6 & 18{,}4 & 4{,}6 \\ 0 & 0 & 0 & 0 & 4{,}6 & 17{,}8 \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ m_5 \\ m_6 \end{pmatrix} = \begin{pmatrix} 0 \\ -3{,}14 \\ -5{,}62 \\ 5{,}88 \\ 2{,}1 \\ 0{,}78 \end{pmatrix} \tag{69}$$

Řešením je:

$$\begin{aligned} m_1 &= 0{,}03 \\ m_2 &= -0{,}11 \\ m_3 &= -0{,}4 \\ m_4 &= 0{,}42 \\ m_5 &= -0{,}003 \\ m_6 &= 0{,}05 \end{aligned} \tag{70}$$

Abychom našli inflexní bod (*termoklinu*) nemusíme vypočítávat koeficienty všech dílčích polynomů, stačí pouze jednoho. Z tabulky 3 vidíme, že propad teploty nastává mezi čtvrtým a pátým bodem. Koeficienty čtvrtého mnohočlenu vypočítáme podle (34), (36), (37), (39). Samotný mnohočlen pak má předpis:

$$t(h) = 20{,}6 - 1{,}16(h - 9{,}1) - 0{,}2(h - 9{,}1)^2 + 0{,}03(h - 9{,}1)^3, h \in \langle 9{,}1; 13{,}7\rangle \tag{71}$$

Dvakrát derivujeme:

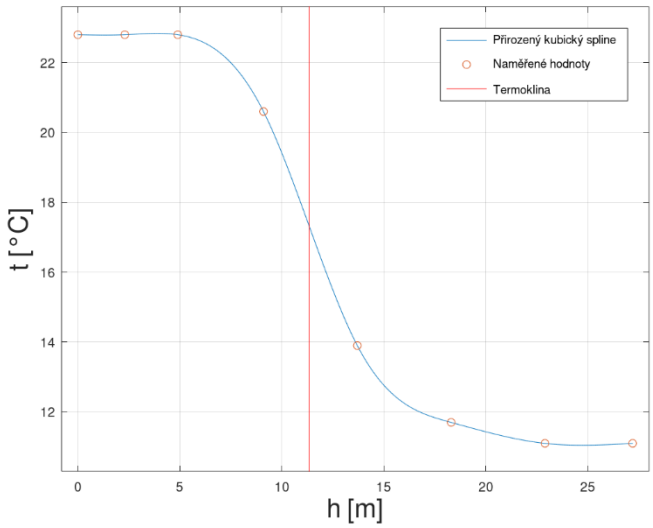
$$t^{''}(h) = -0{,}4 + 0{,}18(h - 9{,}1)\,, h \in \langle 9{,}1; 13{,}7\rangle \tag{72}$$

Rovnice výše se rovná 0, když $h = 11{,}35$ m. Tento bod však nemusí být nutně inflexní, proto musíme prozkoumat jeho pravé a levé okolí.

h	$(9{,}1;\ 11{,}35)$	$(11{,}35\ ;\ 13{,}7)$
$sign\ t''(h)$	$-$	$+$

Tabulka 4: Konvexnost, konkávnost na intervalu (vlastní zpracování)

Z tabulky 4 jde vidět, že funkce mění svou vypuklost v tomto bodě, to znamená, že se jedná o inflexní bod. Teplotu vypočítáme dosazením do (71) - $t(11{,}35) = 17{,}3$ °C. Grafické řešení je uvedeno na obrázku 16.



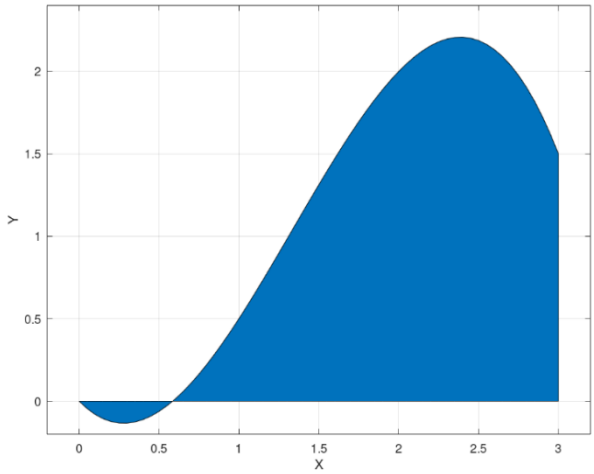
Obrázek 16: Aplikace – teplotní stratifikace (vlastní zpracování)

Úloha č.2 – Integrace interpolované funkce a její využití

Podle Kočandrlové a Černého v [3] lze vypočítat obsah plochy P vymezené osou x a funkcí $f(x)$ pomocí určitého integrálu. Grafické znázornění je na obrázku 17. Protože *interpolační spline* se skládá z n různých polynomů, je nutné vypočítat určitý integrál pro každý dílčí mnohočlen:

$$P = \sum_1^n \int_{x_k}^{x_{k+1}} S_k(x)dx\ ,k \in \{0{,}1{,}2{,}\dots{,}n\}. \qquad (73)$$

Využití výpočtu určitého integrálu je uvedeno na příkladu z oblasti ekonomie.



Obrázek 17: Plošný obsah (vlastní zpracování)

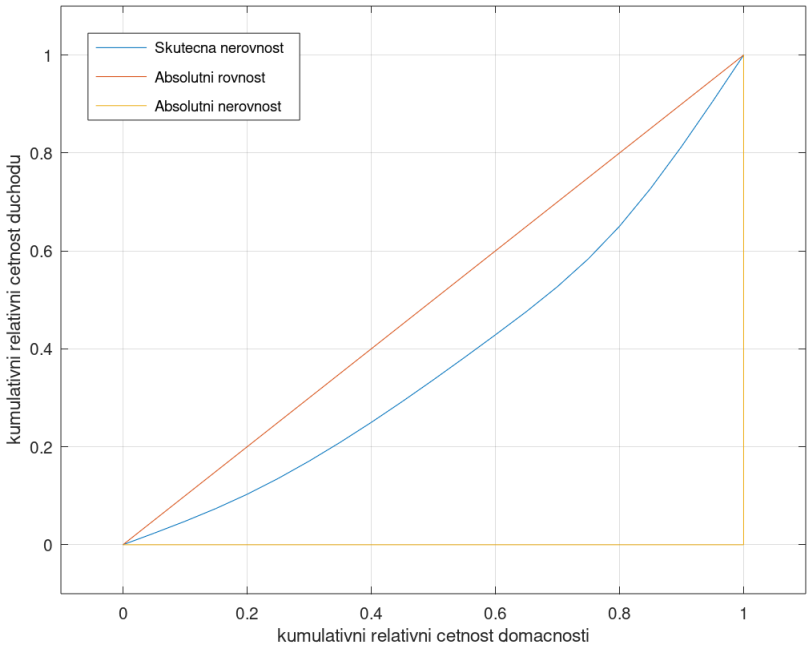
Úvod do problematiky:

Podle Hladkého a Faltové [10] slouží k zobrazení nerovnosti v důchodech domácností *Lorenzova křivka* (LC), pojmenovaná po americkém ekonomovi Max Otto Lorenzovi. Její definice z [11]: „*Lorenzova křivka* vyjadřuje vztah mezi procentním vyjádřením domácností a procentním vyjádřením důchodu. Na vodorovné ose jsou domácnosti jako příjemci důchodů, na svislé ose jsou přijaté důchody. Domácnosti jsou seřazeny od nejnižších důchodových příjmů k nejvyšším. Každý bod křivky udává, jak se příslušné procento domácností podílí na celkovém důchodu".

Podle toho, jaký má *Lorenzova křivka* tvar rozlišujeme 3 typy rozdělení (viz obrázek 18):

1. Absolutní rovnost – každá domácnost má stejný důchod
2. Absolutní nerovnost – veškerý důchod dostává jediná domácnost
3. Skutečná nerovnost

Poznámka: Při výpočtech se používají místo procent kumulativní relativní četnosti.



Obrázek 18: Lorenzova křivka – rovnosti (vlastní zpracování)

Jakmile známe funkční předpis Lorenzovy křivky, pak můžeme vypočítat *Giniho koeficient*. Ten oproti Lorenzově křivce nepopisuje nerovnost graficky, ale číselně [11]. Označíme-li plochu pod *Ideální Lorenzovou křivkou* (absolutní rovnost) jako A. Dále plochu pod skutečnou nerovností jako B, pak můžeme vypočítat *Giniho koeficient* (GC) podle vztahu:

$$GC = \frac{A-B}{A} \tag{74}$$

Hodnota GC se nachází v intervalu $\langle 0,1 \rangle$. Je-li roven nule, pak se jedná o absolutní rovnost rozdělení důchodu. Absolutní nerovnost nastane, když GC = 1.

Zadání:

Máme aproximovat *Lorenzovu křivku* a vypočítat *Giniho koeficient* pro důchody domácností České republiky v roce 2017 z dat v tabulce 5.

Kumulativní relativní četnost domácností – x	0,2	0,4	0,6	0,8	1
Kumulativní relativní četnost důchodu – $LC(x)$	0,1030	0,2498	0,4286	0,6503	1

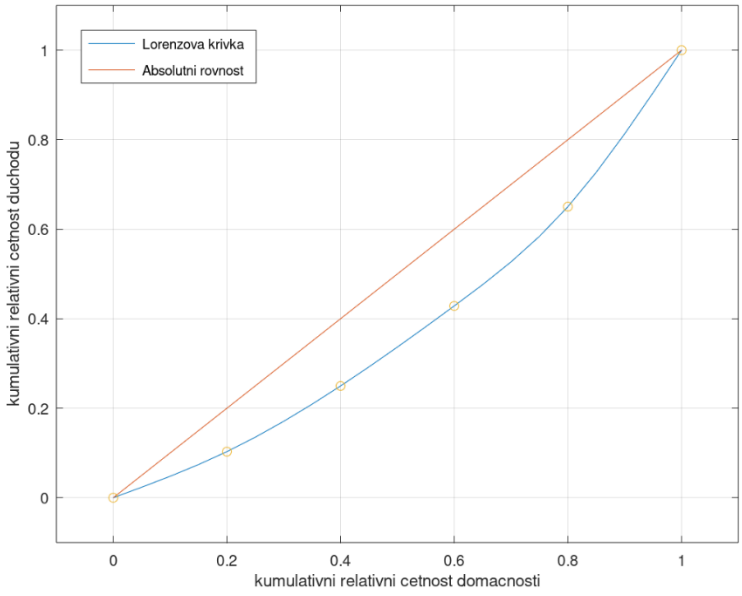
Tabulka 5: Rozdělení důchodu domácností České republiky, zdroj [12]

Řešení

Aproximovanou *Lorenzovu křivku* získáme proložením bodů z tabulky 5 přirozeným kubickým splinem:

$$LC(x) = \begin{cases} 0{,}467x + 1{,}21x^3 & , x \in \langle 0;\ 0{,}2\rangle \\ 0{,}1030 + 0{,}612(x-0{,}2) + 0{,}724(x-0{,}2)^2 - 0{,}555(x-0{,}2)^3 & , x \in \langle 0{,}2;\ 0{,}4\rangle \\ 0{,}2498 + 0{,}834(x-0{,}4) + 0{,}391(x-0{,}4)^2 - 0{,}461(x-0{,}4)^3 & , x \in \langle 0{,}4;\ 0{,}6\rangle \\ 0{,}4286 + 0{,}935(x-0{,}6) + 0{,}114(x-0{,}6)^2 + 3{,}763(x-0{,}6)^3 & , x \in \langle 0{,}6;\ 0{,}8\rangle \\ 0{,}6503 + 1{,}432(x-0{,}8) + 2{,}372(x-0{,}8)^2 - 3{,}953(x-0{,}8)^3 & , x \in \langle 0{,}8;\ 1\rangle \end{cases} \quad (75)$$

Graf funkce LC je uveden na obrázku 19.



Obrázek 19: Aplikace – aproximace Lorenzovy křivky (vlastní zpracování)

Prvním krokem pro získání *Giniho koeficientu* je výpočet určitých integrálů dílčích polynomů:

$$\begin{aligned} \int_0^{0,2} LC_0(x)dx &= 0{,}01j^2 \\ \int_{0,2}^{0,4} LC_1(x)dx &= 0{,}03j^2 \\ \int_{0,4}^{0,6} LC_2(x)dx &= 0{,}07j^2 \\ \int_{0,6}^{0,8} LC_3(x)dx &= 0{,}11j^2 \\ \int_{0,8}^{1} LC_4(x)dx &= 0{,}16j^2 \end{aligned} \quad (76)$$

Jejich suma je rovna obsahu plochy B:

$$B = 0{,}38j^2 \quad (77)$$

Plocha A má obsah $\frac{1}{2}$, protože rozděluje čtverec s délkou strany 1 na dvě poloviny. *Giniho koeficient* spočítáme dosazením do (74):

$$GC = \frac{0,5-0,38}{0,5} = 0,24 \tag{78}$$

Což je to hodnota pod průměrem Evropské unie a také je nižší než u všech sousedních států kromě Slovenska – viz tabulka 6. Nejvyšší *Giniho koeficient* má v EU Bulharsko.

Země	**GC**
Polsko	0,29
Slovensko	0,23
Rakousko	0,27
Německo	0,29
Bulharsko	0,40
EU	0,30

Tabulka 6: Giniho koeficient vybraných států, zdroj [12]

6 Závěr

Cílem této bakalářské práce bylo popsat a charakterizovat spline funkce, v Matlabu naprogramovat jejich výpočet a uvést aplikaci této interpolační metody.

U každého typu splinu se v teoretické části vycházelo z jeho matematické definice. Z ní byl odvozen výpočet koeficientů dílčích mnohočlenů. V této části závěrečné práce byly také popsány rozdíly a výhody oproti interpolaci polynomy vyšších řádů.

Pro snazší pochopení napsaných skriptů byly před samotnou implementací popsány použité funkce v jazyce Matlab. Následně byly popsány algoritmy výpočtů jednotlivých spline funkcí. Praktická část neobsahuje pouze tyto skripty, ale i pomocné. Ty byly napsány z důvodu odstranění částí kódu, které se opakují, a pro validaci vstupních dat. Výstupem výpočetních skriptů jsou, kromě koeficientů dílčích polynomů, také grafy s popisy, vysvětlivkami a se zvýrazněnými uzlovými body.

V poslední kapitole je uvedeno využití spline funkcí. Ukázalo se, že pro praktické účely první dva typy, lineární a kvadratický, nejsou příliš vhodné.

Cíle byly tedy splněny. Na tuto bakalářskou práci by mohlo být navázáno tématem zabývající se B-spline funkcemi, které mají větší využití zejména v oblasti počítačové grafiky.

7 Seznam použité literatury

[1] PRAŽÁK, Pavel. *Matematika 1*. Hradec Králové: Gaudeamus, 2012. ISBN 978-80-7435-227-0.

[2] MATHEWS, John H. a Kurtis D. FINK. *Numerical methods using MATLAB*. 4th ed. Upper Saddle River: Pearson Prentice Hall, c2004. ISBN 9780131911789.

[3] KOČANDRLOVÁ, Milada a Jaroslav ČERNÝ. *Geo-matematika I*. Praha: Česká technika - nakladatelství ČVUT, 2008. ISBN 978-80-01-03936-6.

[4] ČERMÁK, Libor a Rudolf HLAVIČKA. *Numerické metody*. Brno: Akademické nakladatelství CERM, 2005. ISBN 80-214-3071-0.

[5] CHAPRA, Steven C. *Applied numerical methods with MATLAB for engineers and scientists*. 3rd ed. New York: McGraw-Hill, c2012. ISBN 978-0-07-340110-2.

[6] *MATLAB Documentation* [online]. The MathWorks, 2020 [cit. 2020-05-25]. Dostupné z: https://www.mathworks.com/help/matlab/

[7] JANG, Wŏn-jŏng. *Applied numerical methods using MATLAB*. Hoboken: Wiley-Interscience, c2005. ISBN 0-471-69833-4.

[8] ZAPLATÍLEK, Karel a Bohuslav DOŇAR. *MATLAB: tvorba uživatelských aplikací*. Praha: BEN - technická literatura, 2004. ISBN 80-7300-133-0.

[9] DODDS, Walter K. a Matt R. WHILES. *Freshwater ecology: concepts and environmental applications of limnology*. 2nd ed, Academic Press, 2010. ISBN 978-0-12-374724-2.

[10] HLADKÝ, Jan a Ivana FALTOVÁ LEITMANOVÁ. *Mikroekonomie I*. České Budějovice: Jihočeská univerzita, 1997. ISBN 80-7040-201-6.

[11] FUCHS, Kamil. *Makroekonomie I: distanční studijní opora*. Brno: Masarykova univerzita, Ekonomicko-správní fakulta, 2006. ISBN 80–210–3959–0.

[12] Income and living conditions. *Eurostat* [online]. [cit. 2020-07-30]. Dostupné z: https://ec.europa.eu/eurostat/web/income-and-living-conditions/data/database

8 Přílohy

1) Soubor zip se skripty:
    - koeficienty.m
    - kubicky_extrapolovany.m
    - kubicky_konstantni_derivace.m
    - kubicky_prirozeny_spline.m
    - kubicky_urcene_druhe.m
    - kubicky_urcene_prvni.m
    - kubicky_zaklad.m
    - kvadraticky_spline.m
    - linearni_spline.m
    - test_derivace.m
    - test_uzlovych_bodu.m
    - vykresleni_grafu.m

Univerzita Hradec Králové
Fakulta informatiky a managementu

Zadání bakalářské práce

Autor: **David Pikal**

Studium: I1700127

Studijní program: B1802 Aplikovaná informatika

Studijní obor: Aplikovaná informatika

Název bakalářské práce: **Spline funkce a jejich aplikace**

Název bakalářské práce AJ: Spline functions and their applications

Cíl, metody, literatura, předpoklady:

Cílem práce je popsat a charakterizovat spline funkce, implementovat algoritmy pro jejich výpočet a popsat použití spline funkcí

Osnova: 1. Matematická teorie spline funkcí 2. Matlab - důležité příkazy 3. Implementace algoritmů v Matlabu 4. Vybrané aplikace spline funkcí

Mathews J.H., Fink, K.: Numerical Methods Using MATLAB, Pearson Prentice Hall, New Jersey, 2004

Čermák, L., Hlavička, R.: Numerické metody, Akademicke nakladatelsví CERM, Brno, 2005

BUCHANAN, James L. a P. R. TURNER. Numerical methods and analysis. New York: McGraw-Hill, 1992

CHAPRA, Steven C. a Raymond P. CANALE. Numerical methods for engineers. Seventh edition. New York, NY: McGraw-Hill Education, 2015

Garantující pracoviště: Katedra informatiky a kvantitativních metod, Fakulta informatiky a managementu

Vedoucí práce: doc. RNDr. Pavel Pražák, Ph.D.

Oponent: Ing. Petr Bauer, Ph.D.

Datum zadání závěrečné práce: 14.1.2018