VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MATEMATIKY
INSTITUTE OF MATHEMATICS

METODY FFD
FFD METHODS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. Jiří Novák

VEDOUCÍ PRÁCE
SUPERVISOR

Mgr. Jana Procházková, Ph.D.

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav matematiky
Student:	**Bc. Jiří Novák**
Studijní program:	Aplikované vědy v inženýrství
Studijní obor:	Matematické inženýrství
Vedoucí práce:	**Mgr. Jana Procházková, Ph.D.**
Akademický rok:	2016/17

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Metody FFD

Stručná charakteristika problematiky úkolu:

Metody FFD (Free Form Deformation) se používají pro editaci sítí řídících bodů a úpravy objektů. Existuje několik algoritmů založených na Bernsteinových polynomech, B–spline funkcích nebo NURBS. Úkolem práce bude zpracovat tyto algoritmy, srovnat statisticky jejich výsledky, navrhnout implementační zlepšení.

Cíle diplomové práce:

– Zpracování tématu v obecné rovině.
– Algoritmizace jednotlivých přístupů a jejich srovnání.

Seznam doporučené literatury:

SEDERBERG, Thomas W. and PARRY, Scott R. Free-form deformation of solid geometric models. SIGGRAPH Computer Graphics (ACM) 20 (4): 151–160. 1986.

PIEGL, Tiller. NURBS Book. 2nd ed. 2003. ISBN 3-540-61545-8.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2016/17

V Brně, dne

L. S.

prof. RNDr. Josef Šlapal, CSc. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

Abstrakt
Diplomová práce se věnuje tématu free-form deformací. Hlavními cíli této práce bylo zpracování teoretických poznatků o této problematice a naprogramování vybraných metod free-form deformací. V první části je popsána potřebná teorie splajnů, maticového počtu a free-form deformací. Ve výsledné verzi se nachází trojice programů. První program porovnává vybrané metody free-form deformací na příkladu mřížky 4×4 řídicích bodů. Druhý program slouží k zobecnění pro obecný případ mřížky řídicích bodů. Poslední program slouží pro zadávání plochy nikoliv pomocí řídicích bodů, nýbrž pomocí zadání konkrétního bodu, kterým program plochu proloží, tak aby výsledná plocha splňovala definované požadavky.

Summary
The diploma thesis deals with the topic of free-form deformations. The main goal of this work were elaboration of theoretical knowledge about this issue and the programming of selected methods od free-form deformations. The first part describes the required spline theory, matrix calculus and free-form deformations. The resulting version shows three programs. The first program compares the selected free-form deformation methods to the example of the 4×4 control point grid. The second program serves as a generalization for the general case of grid of control points. The last program is based on direct manipulation of arbitrary surface point and following recomputation of the control points to obtain demanded shape.

Klíčová slova
FFD, volnotvará deformace, B-splajn, NURBS, inverzní matice, pseudoinverzní matice, Moore-Penroseova inverze, M-P inverze, Object Pascal, Borland Delphi, Gnuplot

Keywords
FFD, free-form deformation, B-spline, NURBS, inverse matrix, pseudoinverse matrix, Moore-Penrose inverse, M-P inverse, Object Pascal, Borland Delphi, Gnuplot

NOVÁK, J.*Metody FFD*. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2017. 65 s. Vedoucí Mgr. Jana Procházková, Ph.D.

Prohlašuji, že jsem diplomovou práci vypracoval samostatně pod vedením vedoucí práce paní Mgr. Jany Procházkové, Ph.D. a uvedl jsem všechny literární prameny, ze kterých jsem při psaní této práce čerpal.

Jiří Novák

Chtěl bych poděkovat především paní Mgr. Janě Procházkové, Ph.D. za nápad na téma diplomové práce, dále cenné rady, připomínky k práci, pomoc a veškerou trpělivost při vedení diplomové práce.
Dále bych chtěl poděkovat své rodině za pečlivé pročtení a korekturu práce a v neposlední řadě i za podporu, jíž se mi během studia dostalo.

Jiří Novák

Obsah

1 Úvod

Hlavním tématem diplomové práce jsou Free-Form Deformataions (FFD), které se v praxi používají jako editační nástroj v různých CAD systémech. FFD v překladu znamená volnotvará deformace. Jedná se o skupinu metod, které slouží k navrhování objektů, jejichž tvar neodpovídá klasickým analyticko-geometrickým formám. Použití je velmi široké, od filmového průmyslu (postavy, přírodní objekty, atp.), design nových výrobků až po architekturu s organickými tvary budov. Výhodou metod FFD je snadná manipulace s modelovanými objekty a poměrně jednoduché vytvoření požadovaného tvaru. Jejich další velkou výhodou je, že máme k dispozici jednoznačný matematický popis výsledného objektu.

Potřeba FFD vyvstala díky rozvoji leteckého a automobilového průmyslu v období po druhé světové válce. Především konstruování letadel si se základními geometrickými tvary nevystačí – důvodem je požadavek na aerodynamičnost tvarů. V automobilovém průmyslu se spíše jedná o design, který produkt prodává.

Největší význam má FFD v oblasti průmyslového designu. Pokud vynecháme již zmíněné automobilové a letecké inženýrství, nesmíme zapomenout ani na navrhování zcela běžných a každodenních potřeb pro spotřebitelský průmysl (žehličky, ruční mixéry, nářadí, vrtačky, brusky, ...), kde je důležitým aspektem ergonomie a z toho vyplývající praktičnost.

Ať už se jedná o návrhy karoserií automobilů, trupy letadel, nebo návrh ergonomie domácího spotřebiče, průmyslový design se v současné době neobejde bez programů pro 3D modelování (především různé programy z kategorií CAD/CAM), které právě využívají možnosti, které metody FFD přinášejí. FFD je tak neoddělitelnou částí grafického počítačového modelování a tím i všech odvětví, které grafické modelování v počítači využívají.

V posledních dekádách se objevují snahy o využití netradičních tvarů namísto klasických rovinných a ortogonálních i v architektuře. Mezi nejznámější příklady z poslední doby můžeme zmínit zamýšlenou, ale neuskutečněnou, stavbu Národní knihovny na Letné v Praze (přezdívaná „Chobotnice“) navrženou architektem českého původu Janem Kaplickým. K realizovaným projektům lze uvést například tzv. „Tančící dům“ v Praze českého architekta chorvatského původu Vlada Miluniće nebo obchodní centrum Letmo v Brně podle projektu architektů Tomáše Dvořáka, Martina Klimeckého a Davida Fišera.

Poměrně netradiční využití se metodám FFD naskýtá v lékařství. Jako příklad zde zmiňme metodu „registrace obrazu“, která slouží k hledání transformací mezi snímky z moderních diagnostických zobrazovacích metod (např. počítačové tomografie, nebo magnetická rezonance). Registrace obrazu se zde používá k zachycení růstu nádorů v těle.

Mým úkolem bylo vyhledat a nastudovat odbornou literaturu, která se této oblasti věnuje a popsat konkrétní metody používané pro Free-form modelování. Dalším krokem bylo popsat jednotlivé typy aproximací používané v počítačovém modelování, včetně jejich matematického pozadí, jejich výhod a nevýhod.

V rámci práce jsem vytvořil několik programových knihoven, které provádí samotné výpočty podle daných metod. Další knihovna potom slouží k ovládání výstupů do datových souborů a na obrazovku počítače. Tato knihovna umožňuje vykreslit zadaná data pomocí softwaru Gnuplot na obrazovku nebo naformátovat je pro výstup LaTeXovského kódu. Zmíněné knihovny jsem využil pro programování trojice programů (popsané v části

3.3). Tyto programy slouží k ověření, že popsané knihovny jsou navrženy korektně a umožňují testování navržených metod.

První program FFD1 (viz část 3.3.1) slouží k porovnání používaných základních metod na mřížce 4×4 řídicích bodů. Uživatel nastaví souřadnice řídicích bodů, zvolí transformaci a nechá příslušnou plochu programem vykreslit. Druhý program FFD2 (popsaný v části 3.3.2) slouží k demonstraci funkčnosti těchto metod pro libovolně velkou mřížku. Poslední program FFD3 je založen na tzv. zpětné metodě. Uživatel pouze zadá, kam se daný bod má přemístit a program automaticky spočítá nové polohy všech řídících bodů tak, aby výsledná plocha daným bodem procházela. Celá práce je programována pouze pro 2D – tedy pro rovinný případ. Důvodem je vyšší názornost vůči obecnému trojrozměrnému případu. Rozšíření pro 3D případ lze však snadno provést přidáním další souřadnice, formální stránka problematiky je shodná s rovinným případem.

Během práce jsem využil poznatky z různých odvětví matematiky. Základem pro metody užívané k FFD je oblast počítačové grafiky (kubické interpolační křivky, splajny), využity jsou však závěry i z lineární algebry (maticový počet a řešení nedourčených soustav rovnic, podrobně je rozepsán výpočet pseudoinverzní matice) společně s poznatky matematické analýzy (hledání extrémů).

2 Teorie

V následující části popíšeme teoretické základy pro splajny a teorii týkající se inverzní a pseudoinverzní matice, kterou budeme používat v další práci. Předpokládáme základní znalosti lineární algebry.

2.1 Splajny

Kořeny matematického odvětví zabývajícího se kubickými polynomiálními křivkami a plochami spadají na přelom padesátých a šedesátých let minulého století. V té době si technická praxe, převážně v automobilovém a leteckém odvětví, začala žádat prostředky pro matematický popis tvarů různých součástek, jako například dílů karoserií, nebo částí trupů letadel či lodí. Do té doby návrhy těchto a dalších součástí prováděli projektanti s pomocí důmyslného náčiní sestávajícího se z dlouhých a tenkých (tedy dobře ohebných) proužků kovu, který byl tvarován pomocí závaží zhotovených z pochopitelných důvodů z olova. Tento přístup měl velkou výhodu v názornosti, rychlosti (pokud uvážíme tehdejší výpočetní kapacity) a relativní nenáročnosti na technické vybavení projekční kanceláře.

Největší slabinou těchto způsobů navrhování tvarů je fakt, že tyto modelovací metody nám nedávají matematický popis výsledných ploch, což bylo z hlediska následujícího vývoje rozhodující. Právě přesný matematický popis rychle se rozvíjející inženýrské obory, jakými automobilní a letecký průmysl bezpochyby byly a jsou, totiž vyžadovaly z důvodu rozšiřující se automatizace ve výrobě. Klasické objekty známé z analytické geometrie (které umíme snadno matematicky vyjádřit) však např. pro popis karosářských dílů značně nedostačují.

2.1.1 Předchůdci splajnů

Konstrukční kanceláře různých průmyslových společností proto hledaly vlastní cesty. Přes fakt, že se jednalo o inženýrská řešení, která byla vyvíjena různými firmami (Citroën, Renault, Boeing), mají všechny způsoby, které se prakticky prosadily a které jsou uvedeny níže, mnoho společného. V prvé řadě je společným znakem to, že tyto nově popsané křivky jsou ve tvaru kubických polynomů. Tato vlastnost je společná z prostého důvodu: základem úvah totiž byla už popsaná práce s plechovými proužky, jejichž chování lze právě nejlépe popsat kubickým polynomem. Jednotlivé popisy (Fergusonův, Bézierův, Coonsův) křivek se vyznačují pevně danými řídicími polynomiálními funkcemi (známými pod pojmem bázové polynomy). Dalším společným znakem je, že samotný tvar konkrétní křivky je popsán pomocí tzv. řídicího polygonu, skládajícího se z řídicích bodů. Dále se budeme věnovat jednotlivým typům křivek a jejich rozšířením na plochy.

Fergusonovy křivky a plochy

Fergusonova křivka je pojmenována po J. C. Fergusonovi, který ji, společně se svým kolegou ze společnosti Boeing D. MacLarenem, roku 1964 popsal.

Základní oblouk Fergusonovy křivky je popsán pomocí čtyř bodů P_0, P_1, P_2, P_3, které mají následující význam: body P_0 a P_2 jsou krajní body křivky a dvojice bodů P_0 a P_1, resp. P_2 a P_3 tvoří vektor $\overrightarrow{P_0P_1}$, resp. $\overrightarrow{P_2P_3}$. Tyto vektory určují tečný směr v krajních

bodech základního oblouku křivky a jejich velikost určuje druhou derivaci, tedy čím větší je velikost zmíněných vektorů, tím rychleji se křivka k danému tečnému směru přimyká.

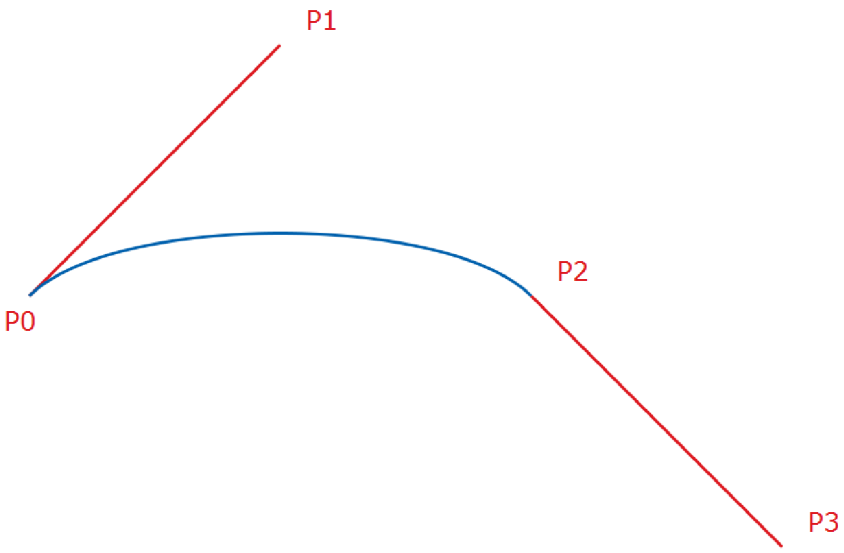
Hladké napojení dalšího oblouku křivky lze provést použitím bodů P_2 a P_3 předchozího oblouku jako bodů P'_0 a P'_1 následujícího oblouku.

Bázové polynomy jsou definovány jako:

$$\begin{aligned} F_0(t) &= t^3 - t^2 - t + 1 \\ F_1(t) &= t^3 - 2t^2 + t \\ F_2(t) &= -3t^3 + 4t^2 \\ F_3(t) &= t^3 - t^2 \end{aligned} \tag{2.1}$$

Parametrická rovnice Fergusonovy křivky je dána následovně:

$$Q(t) = \sum_{i=0}^{3} P_i F_i(t), t \in \langle 0, 1 \rangle \tag{2.2}$$



Obrázek 2.1: Fregusonova křivka a grafické znázornění významu řícicích bodů

Pokud Fergusonovu křivku dvourozměrně zobecníme, dostáváme Fergusonovu plochu. Ta je zadaná sítí řídicích bodů P_{ij} a parametrickou rovnicí:

$$Q(r, s) = \sum_{i=0}^{3} \sum_{j=0}^{3} P_{ij} F_i(r) F_j(s), r, s \in \langle 0, 1 \rangle \tag{2.3}$$

Bézierovy křivky a plochy

Tento způsob popisu křivek popsali nezávisle na sobě Francouzi Paul de Casteljau a Pierre Étienn Bézier. Pojmenována je po druhém jmenovaném, který ji roku 1962 patentoval. Ovšem již roku 1959 popsal stejný princip P. de Casteljau, který si však svůj nápad patentovat nenechal, proto se používá název Béziérova křivka.

Základní oblouk Bézierovy křivky je, podobně jako u Fergusonovy křivky, zadán čtyřmi body P_0, P_1, P_2, P_3, zde však s odlišným významem. Krajními body tu jsou body P_0 a P_3. Dvojice bodů P_0 a P_1, resp. P_3 a P_2 tvoří vektory $\overrightarrow{P_0P_1}$, resp. $\overrightarrow{P_3P_2}$, které určují směry tečen v krajních bodech. Dále pro tyto vektory platí, že jejich velikost je rovna trojnásobku směrnice.

Hladké napojení dalšího oblouku křivky zajistíme tím, že jako bod P_0' následujícího oblouku použijeme bod P_3 předchozího oblouku a bod P_1' nového oblouku vytvoříme obrazem bodu P_2 středovou souměrností bodu se středem P_3.

Bázové polynomy, nazývané v tomto případě Bernsteinovy polynomy, jsou definovány jako binomický rozvoj:

$$\begin{array}{rcl} B_0(t) & = & (1-t)^3 = 1 - 3t + 3t^2 - t^3 \\ B_1(t) & = & 3t(1-t)^2 = 3t - 6t^2 + 3t^3 \\ B_2(t) & = & 3t^2(1-t) = 3t^2 - 3t^3 \\ B_3(t) & = & t^3 \end{array} \tag{2.4}$$

Parametrická rovnice Bézierovy křivky je dána jako:

$$Q(t) = \sum_{i=0}^{3} P_i B_i(t), t \in \langle 0, 1 \rangle \tag{2.5}$$

Vzhledem k tomu, že Bernsteinovy polynomy tvoří binomický rozvoj, lze snadno Bézierovu křivku zobecnit pro případ $n + 1$ řídicích bodů. Pak její bázové polynomy jsou dány:

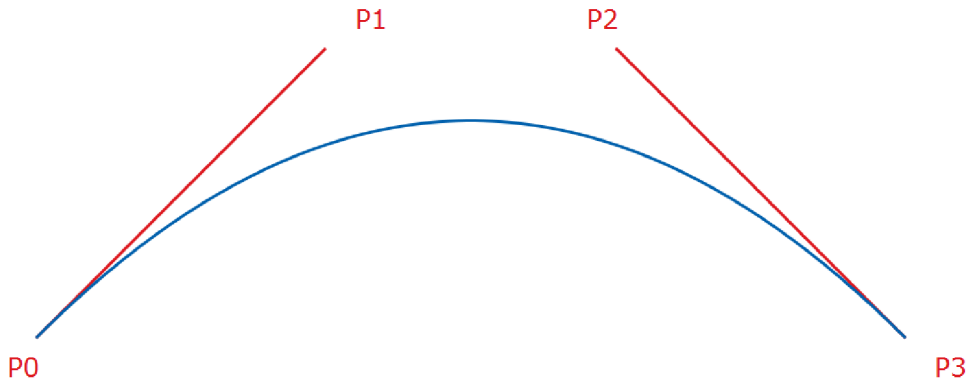
$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i} \text{ pro } i = 0, 1, \ldots, n. \tag{2.6}$$

Ovšem parametrická rovnice takto zobecněné Bézierovy křivky se téměř nezmění:

$$Q(t) = \sum_{i=0}^{n} P_i B_i(t), t \in \langle 0, 1 \rangle \tag{2.7}$$

Dvourozměrným zobecněním Bézierovy křivky získáme Bézierovu plochu. Zadaná je sítí řídicích bodů P_{ij} a parametrickou rovnicí:

$$Q(r, s) = \sum_{i=0}^{3} \sum_{j=0}^{3} P_{ij} B_i(r) B_j(s), r, s \in \langle 0, 1 \rangle \tag{2.8}$$



Obrázek 2.2: Bezierova křivka a grafické znázornění významu řídicích bodů

Coonsovy křivky a plochy

Tento způsob popisu křivek navrhl Steven Anson Coons (1912-1979), profesor na Massachusetts Institute of Technology a vizionář interaktivní počítačové grafiky jako nástroje pro inženýry.

Čtyřbodový řídicí polygon P_0, P_1, P_2, P_3 v tomto případě uvažujeme jako dvojici trojúhelníků P_0, P_1, P_2 a P_1, P_2, P_3. V obou těchto trojúhelnících lze nalézt tři tzv. antitěžiště. Mezi nimi je pro popis této Coonsovy křivky důležité to, které přísluší prostřednímu z vrcholů daného trojúhelníka (tzn. přísluší cí bodu P_1 v trojúhelníku P_0, P_1, P_2 a přísluší cí bodu P_2 v trojúhelníku P_1, P_2, P_3). Toto antitěžiště je krajním bodem oblouku Coonsovy křivky.

Hladké napojení dvou oblouků Coonsovy křivky zajistíme tím, že použijeme trojúhelník P_1, P_2, P_3 předchozího oblouku jako trojúhelník P'_0, P'_1, P'_2 následujícího oblouku. Bod P'_3 následujícího oblouku zvolíme libovolně.

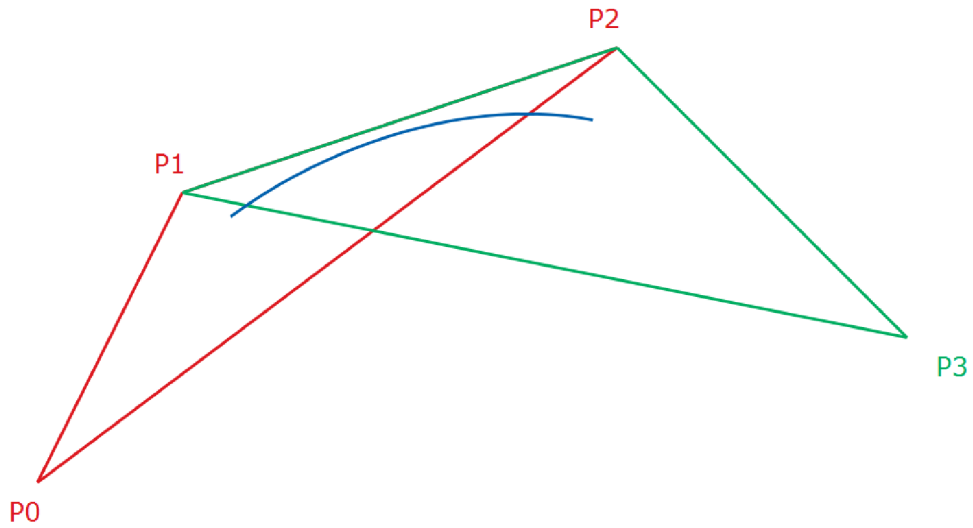
Pokud $P_0 = P_1$, resp. $P_2 = P_3$, pak první, resp. druhý, trojúhelník degeneruje v úsečku a počáteční, resp. koncový, bod se nachází v její první, resp. předposlední, šestině. Pokud $P_0 = P_1 = P_2$, resp. $P_1 = P_2 = P_3$, pak první, resp. druhý, trojúhelník degeneruje v jediný bod a počáteční, resp. koncový, bod je právě v tomto bodě.

Bázové polynomy Coonsovy křivky jsou zadány následovně:

$$\begin{aligned} C_0(t) &= (1-t)^3 = 1 - 3t + 3t^2 - t^3 \\ C_1(t) &= 4 - 6t^2 + 3t^3 \\ C_2(t) &= 1 + 3t + 3t^2 - 3t^3 \\ C_3(t) &= t^3 \end{aligned} \tag{2.9}$$

Parametrické vyjádření Coonsovy křivky je:

$$Q(t) = \frac{1}{6}\sum_{i=0}^{3} P_i C_i(t), t \in \langle 0, 1 \rangle \tag{2.10}$$



Obrázek 2.3: Coonsova křivka a grafické znázornění významu řícicích bodů

Dvourozměrným zobecněním Coonsovy křivky dostáváme Coonsovu plochu. Zadaná je sítí řídicích bodů P_{ij} a parametrickou rovnicí:

$$Q(r, s) = \sum_{i=0}^{3}\sum_{j=0}^{3} P_{ij} C_i(r) C_j(s), r, s \in \langle 0, 1 \rangle \tag{2.11}$$

2.1.2 B-splajny

Hladké napojování oblouků Coonsových křivek umožňuje spojení libovolného počtu $n \geq 4$ řídicích bodů pomocí $n-3$ oblouků křivky. Jejich spojením vznikne po částech polynomiální křivka nazývaná Coonsův kubický B-splajn. Krajní body jednotlivých oblouků ve splajnu se nazývají uzly a značí se t. Parametry t lze uspořádat v konečnou posloupnost nazývanou uzlový vektor. Sousední uzly se od sebe liší o konstantní krok a uzlový vektor je tedy ekvidistantní. Coonsovu kubickou B-splajn křivku, resp. plochu, lze zobecnit pro libovolný stupeň polynomu a tím získáme B-splajn křivku, resp. plochu. Volbou neekvidistantního rozložení uzlů v uzlovém vektoru dostaneme další zobecnění, tzv. neuniformní splajn. Pokud přidáme kromě uzlového vektoru ještě váhový vektor, který ke každému bodu přiřadí váhu, získáme neuniformní racionální B-splajn, známější pod zkratkou NURBS.

B-splajn funkce

B-splajn funkce se využívají jako bázové funkce pro B-splajn (resp. od nich odvozené NURBS) křivky a plochy.

Definice 1. Mějme uzlový vektor $\mathbf{t} = (t_0, t_1, \ldots, t_n)$. B-splajn funkci definujeme rekurentně jako:

$$N_i^0(t) = \begin{cases} 1 & \text{pro} \quad t \in \langle t_i, t_{i+1}) \\ 0 & \text{jinak} \end{cases}$$

$$N_i^k(t) = \frac{t - t_i}{t_{i+k} - t_i} N_i^{k-1}(t) + \frac{t_{i+k+1} - t}{t_{i+k+1} - t_{i+1}} N_{i+1}^{k-1}(t) \tag{2.12}$$

kde $0 \leq i \leq n-k-1, 1 \leq k \leq n-1$, přičemž definujeme $\frac{0}{0} := 0$.

Uzlový vektor je tvořen jako neklesající posloupnost nezáporných reálných čísel, obvykle tvoří interval $\langle 0, 1)$, ale obecně lze volit interval libovolně.

Definice 2. Uzlový vektor $\mathbf{t} = (t_0, t_1, \ldots, t_n)$ nazýváme **ekvidistantní** (nebo též **uniformní**), splňuje-li následující rovnost:

$$t_i - t_{i-1} = t_{i-1} - t_{i-2} \text{ pro } i = 2, 3, \ldots, n. \tag{2.13}$$

Nesplňuje-li tuto rovnost, pak se takový uzlový vektor nazývá **neekvidistantní** (**neuniformní**).

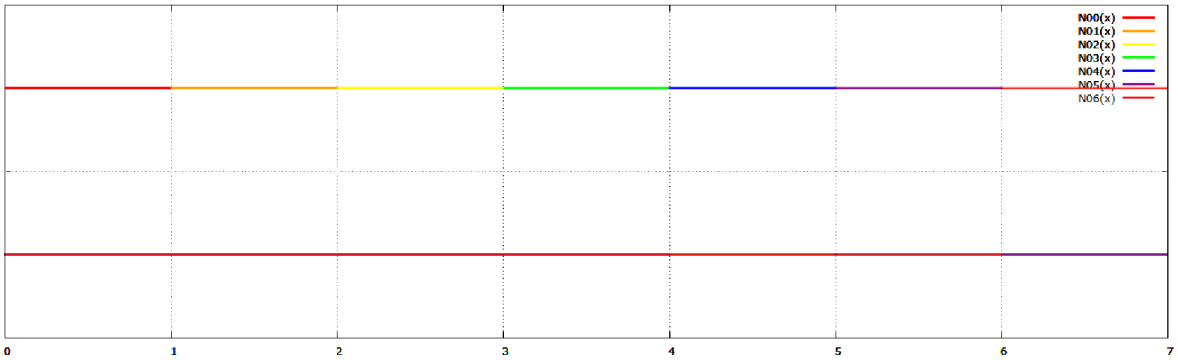
Definice 3. Nechť $\mathbf{t} = (t_0, t_1, \ldots, t_n)$ je uzlový vektor. Délkou uzlového vektoru se rozumí počet jeho prvků $n+1$.

Příklady B-splajn funkcí vybraných stupňů

Uvažujme uzlový vektor $\mathbf{t} = (t_0, t_1, \ldots, t_n) = (0, 1, \ldots, 5)$. V této části spočítáme a vykreslíme B-splajn funkce stupňů 0,1,2 a 3.

B-splajn funkce nultého stupně

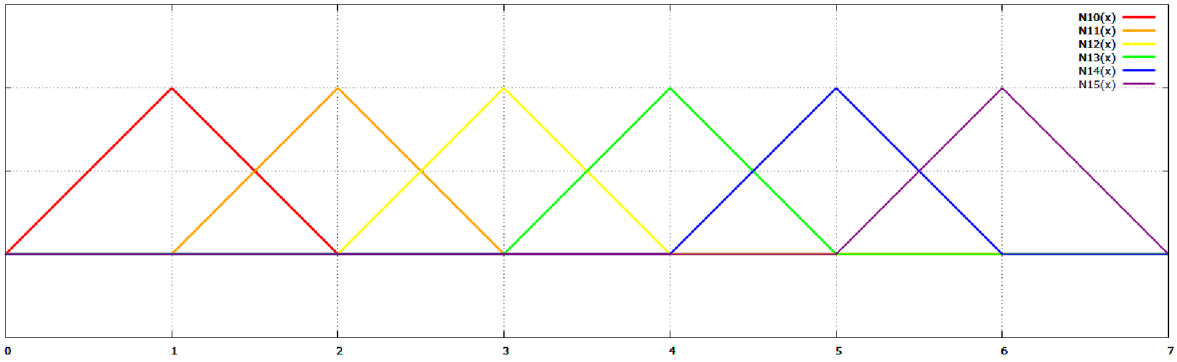
$$N_i^0(t) = \begin{cases} 1 & \text{pro} \quad t \in \langle t_i, t_{i+1}) \\ 0 & \text{jinak} \end{cases} \qquad (2.14)$$
$$\text{pro } \forall i = 0 \ldots n-1.$$



Obrázek 2.4: B-splajn funkce nultého stupně

B-splajn funkce prvního stupně

$$
\begin{aligned}
N_0^1(t) &= \begin{cases} t & \text{pro} \quad t \in \langle 0,1) \\ 2-t & \text{pro} \quad t \in \langle 1,2) \\ 0 & \text{jinak} \end{cases} \\
N_1^1(t) &= \begin{cases} t-1 & \text{pro} \quad t \in \langle 1,2) \\ 3-t & \text{pro} \quad t \in \langle 2,3) \\ 0 & \text{jinak} \end{cases} \\
N_2^1(t) &= \begin{cases} t-2 & \text{pro} \quad t \in \langle 2,3) \\ 4-t & \text{pro} \quad t \in \langle 3,4) \\ 0 & \text{jinak} \end{cases} \\
N_3^1(t) &= \begin{cases} t-3 & \text{pro} \quad t \in \langle 3,4) \\ 5-t & \text{pro} \quad t \in \langle 4,5) \\ 0 & \text{jinak} \end{cases} \\
N_4^1(t) &= \begin{cases} t-4 & \text{pro} \quad t \in \langle 4,5) \\ 6-t & \text{pro} \quad t \in \langle 5,6) \\ 0 & \text{jinak} \end{cases} \\
N_5^1(t) &= \begin{cases} t-5 & \text{pro} \quad t \in \langle 5,6) \\ 7-t & \text{pro} \quad t \in \langle 6,7) \\ 0 & \text{jinak} \end{cases}
\end{aligned}
\tag{2.15}
$$



Obrázek 2.5: B-splajn funkce prvního stupně

B-splajn funkce druhého stupně

$$
\begin{aligned}
N_0^2(t) &= \begin{cases} \frac{1}{2}t^2 & \text{pro} \quad t \in \langle 0,1) \\ \frac{1}{2}t(2-t) + \frac{1}{2}(t-1)(3-t) & \text{pro} \quad t \in \langle 1,2) \\ \frac{1}{2}(3-t)^2 & \text{pro} \quad t \in \langle 2,3) \\ 0 & \text{jinak} \end{cases} \\
N_1^2(t) &= \begin{cases} \frac{1}{2}(t-1)^2 & \text{pro} \quad t \in \langle 1,2) \\ \frac{1}{2}(t-1)(3-t) + \frac{1}{2}(t-2)(4-t) & \text{pro} \quad t \in \langle 2,3) \\ \frac{1}{2}(4-t)^2 & \text{pro} \quad t \in \langle 3,4) \\ 0 & \text{jinak} \end{cases} \\
N_2^2(t) &= \begin{cases} \frac{1}{2}(t-2)^2 & \text{pro} \quad t \in \langle 2,3) \\ \frac{1}{2}(t-2)(4-t) + \frac{1}{2}(t-3)(5-t) & \text{pro} \quad t \in \langle 3,4) \\ \frac{1}{2}(5-t)^2 & \text{pro} \quad t \in \langle 4,5) \\ 0 & \text{jinak} \end{cases} \\
N_3^2(t) &= \begin{cases} \frac{1}{2}(t-3)^2 & \text{pro} \quad t \in \langle 3,4) \\ \frac{1}{2}(t-3)(5-t) + \frac{1}{2}(t-4)(6-t) & \text{pro} \quad t \in \langle 4,5) \\ \frac{1}{2}(6-t)^2 & \text{pro} \quad t \in \langle 5,6) \\ 0 & \text{jinak} \end{cases} \\
N_4^2(t) &= \begin{cases} \frac{1}{2}(t-4)^2 & \text{pro} \quad t \in \langle 4,5) \\ \frac{1}{2}(t-4)(6-t) + \frac{1}{2}(t-5)(7-t) & \text{pro} \quad t \in \langle 5,6) \\ \frac{1}{2}(7-t)^2 & \text{pro} \quad t \in \langle 6,7) \\ 0 & \text{jinak} \end{cases}
\end{aligned}
\tag{2.16}
$$

Obrázek 2.6: B-splajn funkce druhého stupně

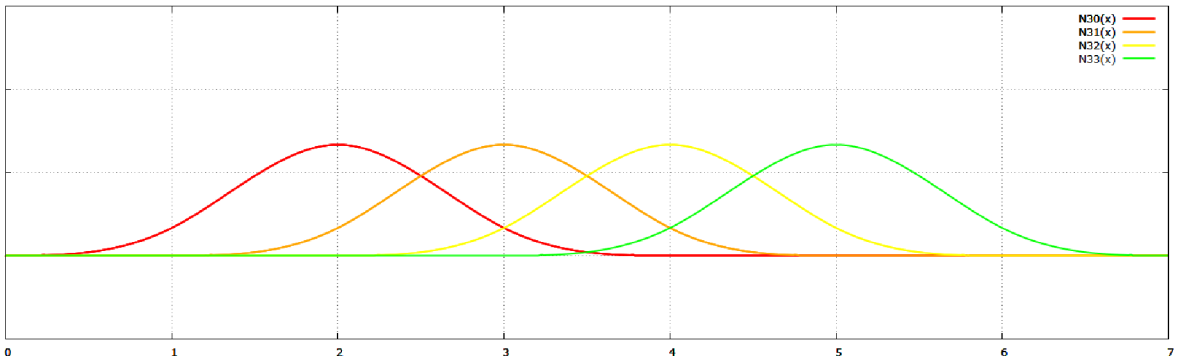
B-splajn funkce třetího stupně

$$
N_0^3(t) = \begin{cases} \frac{1}{6}t^3 & \text{pro} \quad t \in \langle 0,1) \\ \frac{1}{6}[t^2(2-t) + t(3-t)(t-1) + (4-t)(t-1)^2] & \text{pro} \quad t \in \langle 1,2) \\ \frac{1}{6}[t(3-t)^2 + (4-t)(t-1)(3-t) + (4-t)^2(t-2)] & \text{pro} \quad t \in \langle 2,3) \\ \frac{1}{6}(4-t)^3 & \text{pro} \quad t \in \langle 3,4) \\ 0 & \text{jinak} \end{cases}
$$

$$
N_1^3(t) = \begin{cases} \frac{1}{6}(t-1)^3 & \text{pro} \quad t \in \langle 1,2) \\ \frac{1}{6}[(t-1)^2(3-t) + (t-1)(4-t)(t-2) + (5-t)(t-2)^2] & \text{pro} \quad t \in \langle 2,3) \\ \frac{1}{6}[(t-1)(4-t)^2 + (5-t)(t-2)(4-t) + (5-t)^2(t-3)] & \text{pro} \quad t \in \langle 3,4) \\ \frac{1}{6}(5-t)^3 & \text{pro} \quad t \in \langle 4,5) \\ 0 & \text{jinak} \end{cases}
$$

$$
N_2^3(t) = \begin{cases} \frac{1}{6}(t-2)^3 & \text{pro} \quad t \in \langle 2,3) \\ \frac{1}{6}[(t-2)^2(4-t) + (t-2)(5-t)(t-3) + (6-t)(t-3)^2] & \text{pro} \quad t \in \langle 3,4) \\ \frac{1}{6}[(t-2)(5-t)^2 + (6-t)(t-3)(5-t) + (6-t)^2(t-4)] & \text{pro} \quad t \in \langle 4,5) \\ \frac{1}{6}(6-t)^3 & \text{pro} \quad t \in \langle 5,6) \\ 0 & \text{jinak} \end{cases}
$$

$$
N_3^3(t) = \begin{cases} \frac{1}{6}(t-3)^3 & \text{pro} \quad t \in \langle 3,4) \\ \frac{1}{6}[(t-3)^2(5-t) + (t-3)(6-t)(t-4) + (7-t)(t-4)^2] & \text{pro} \quad t \in \langle 4,5) \\ \frac{1}{6}[(t-3)(6-t)^2 + (7-t)(t-4)(6-t) + (7-t)^2(t-5)] & \text{pro} \quad t \in \langle 5,6) \\ \frac{1}{6}(7-t)^3 & \text{pro} \quad t \in \langle 6,7) \\ 0 & \text{jinak} \end{cases}
\tag{2.17}
$$



Obrázek 2.7: B-splajn funkce třetího stupně

Vliv uzlového vektoru

Uzly v uzlovém vektoru z předchozí části byly vytvořeny ekvidistantně, proto jsou B-splajn funkce příslušného stupně tvarově totožné a liší se pouze posunutím nad příslušný interval v uzlovém vektoru. V případě neekvidistantního uzlového vektoru dochází ke změně tvaru křivky. Výčet jednotlivých způsobů ovlivnění křivky formou uzlového vektoru byl mnohokrát publikován, např. v [15].

Vlastnosti B-splajn funkcí

1. **Nezápornost** $N_i^k(t) \geq 0$ pro libovolná přípustná i, k, t.
2. **Lokálnost nosičů** $N_i^k(t) \neq 0$ pro $t \in \langle t_i, t_{i+k+1})$.
3. **Vlastnost intervalu** Na libovolném intervalu $\langle t_i, t_{i+k+1})$ existuje nejvýše $k+1$ nenulových bázových funkcí stupně k a to následující:

$$\{N_{i-j}^k\}_{j=0}^k \tag{2.18}$$

   tedy $N_{i-k}^k, N_{i-k+1}^k, \ldots, N_i^k$.

4. **Rozklad jednotky** Součet B-splajn funkcí stupně k na intervalu $\langle t_i, t_{i+1})$ je roven jedné

$$\sum_{j=0}^{k} N_{i-j}^k = 1. \tag{2.19}$$

5. **Vlastnost uniformního uzlového vektoru** Funkce N_i^k jsou až na posun podél osy parametrů identické

$$N_i^k(t) = N_{i+1}^k(t + t_1). \tag{2.20}$$

6. **Spojitost** V s-násobném uzlu má bázová funkce N_i^k třídu spojitosti C^{k-s}.

Algoritmus výpočtu B-splajn funkcí

Na začátku této části je uvedena def. 1 B-splajn funkce, která je dána rekurentně (viz rovnice (2.12)). To způsobuje její omezení pro praktické výpočty, neboť rekurzivní výpočet je časově náročný. Proto je výhodné nalézt jiný algoritmus, který by výpočet urychlil. Budeme vycházet z vlastnosti intervalu, která je uvedena v předchozím odstavci.

Algoritmus můžeme popsat následujícím způsobem:

Vstup: stupeň n, index i, parametr t, uzlový vektor $\mathbf{t} = (t_0, \ldots, t_m)$ (označen jako pole `T[0],T[1],...,T[m]`)

Výstup: koeficienty $N_0^n(t), N_1^n(t), \ldots, N_i^n(t)$ (jako pole `N[0],N[1],...,N[i]`)

```
N[0]:=1.0;
for j:=1 to n do
  {
    left[j]:=t-T[i+1-j];
    right[j]:=T[i+j]-t;
    saved:=0.0;
    for r:=0 to j-1
      {
        temp:=N[r]/(right[r+1]+left[j-r]);
        N[r]:=saved+right[r+1]*temp;
        saved:=left[j-r]*temp;
      }
    N[j]:=saved;
  }
```

B-splajn křivky

Kombinací B-splajn funkcí s řídicími body lze zadefinovat B-splajn křivky.

Definice 4. B-splajn křivkou stupně n pro řídicí body $P_i \in \mathbb{R}^d, i = 0, 1, \ldots, m$ a uzlový vektor $\mathbf{t} = (t_0, t_1, \ldots, t_{m+n+1})$ rozumíme:

$$Q(t) = \sum_{i=0}^{m} P_i N_i^n(t) \tag{2.21}$$

kde N_i^n jsou bázové B-splajn funkce popsané vztahem (2.12) v kapitole 2.1.2.

Definice 5. B-splajn křivku stupně n nazveme **uzavřenou**, je-li n-tice počátečních bodů shodná s n-ticí koncových bodů. Není-li B-splajn křivka uzavřenou, pak ji nazveme **otevřenou**.

Definice 6. Řídicí body $P_0, P_1, \ldots, P_m$ nazýváme **deBoorovými body** a jako celek **deBoorovým polygonálním tahem**

Vlastnosti B-splajn křivek

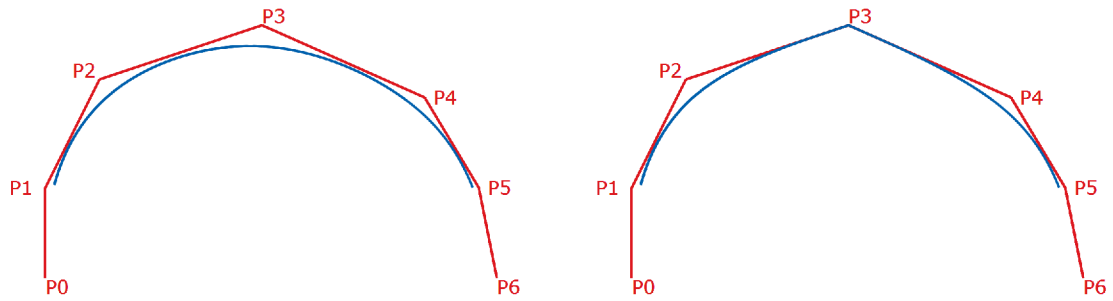
1. **Lokálnost vlivu změny řídicího bodu** Změna řídicího bodu P_i ovlivní tvar křivky n-tého stupně pouze na obloucích odpovídajících intervalu $\langle t_i, t_{i+n+1}\rangle$.

2. **Délka uzlového vektoru** Počet uzlů v uzlovém vektoru musí být roven součtu počtu řídicích bodů a stupně křivky plus jedna. Vnitřní uzly v uzlovém vektoru se nesmí opakovat vícekrát než je stupeň křivky.

3. **Symetrie**
$$\sum_{i=0}^{m} P_i N_i^n(t) = \sum_{i=0}^{m} P_{m-i} \overline{N}_{m-i}^n(1-t) \tag{2.22}$$
kde $N_i^n(t)$ je bázová funkce pro uzlový vektor $\mathbf{t} = (t_0, t_1, \ldots, t_{m+n+1}), t_j \in \langle 0; 1\rangle, \forall j \in \langle 0; m+n+1\rangle$. a $\overline{N}_{m-i}^n(1-t)$ je bázová funkce pro uzlový vektor $\mathbf{1} - \mathbf{t} = (1 - t_{m+n+1}, \ldots, 1 - t_1, 1 - t_0)$

   Jedná se o vlastnost všech bázových funkcí (např. Bernsteinových polynomů).

4. **Spojitost** V bodě odpovídajícím s-násobném uzlu má B-splajn křivka stupně n třídu spojitosti C^{n-s}. Na obrázku 2.8 je názorně vidět změna spojitosti v bodě při opakování vnitřního uzlu. Pro stupeň třetí při trojnásobném opakování uzlu dostaneme v příslušném bodě křivky spojitost C^0.



Obrázek 2.8: Znázornění spojitosti B-splajn křivky pro stupeň křivky 3: vlevo mají všechny uzly násobnost 1, vpravo má bod P_3 násobnost 3

B-splajn plochy

Definice 7. Nechť máme síť $(r+1) \times (s+1)$ řídicích bodů $P_{ij} \in \mathbb{R}^d, i = 0, 1, \ldots, r, j = 0, 1, \ldots, s$, stupeň pro řádky m, řádkový uzlový vektor $\mathbf{t} = (t_0, t_1, \ldots, t_{m+s+1})$, stupeň pro sloupce n, sloupcový uzlový vektor $\mathbf{u} = (u_0, u_1, \ldots, u_{n+r+1})$

B-splajn plochou pro síť $(r+1) \times (s+1)$ řídicích bodů $P_{ij} \in \mathbb{R}^d, i = 0, 1, \ldots, r, j = 0, 1, \ldots, s$ rozumíme:

$$Q(t) = \sum_{i=0}^{r} \sum_{j=0}^{s} P_{ij} N_i^n(t) N_j^m(u) \tag{2.23}$$

kde N_i^n jsou bázové B-splajn funkce popsané vztahem (2.12) v kapitole 2.1.2.

Vlastnosti B-splajn ploch

1. **Rozklad jednotky** Součet B-splajn funkcí $N_i^m(t)N_j^n(u)$ je roven jedné $\forall u, v \in \langle 0; 1 \rangle$

$$\sum_{i=0}^{r} \sum_{j=0}^{s} N_i^m N_j^n = 1 \tag{2.24}$$

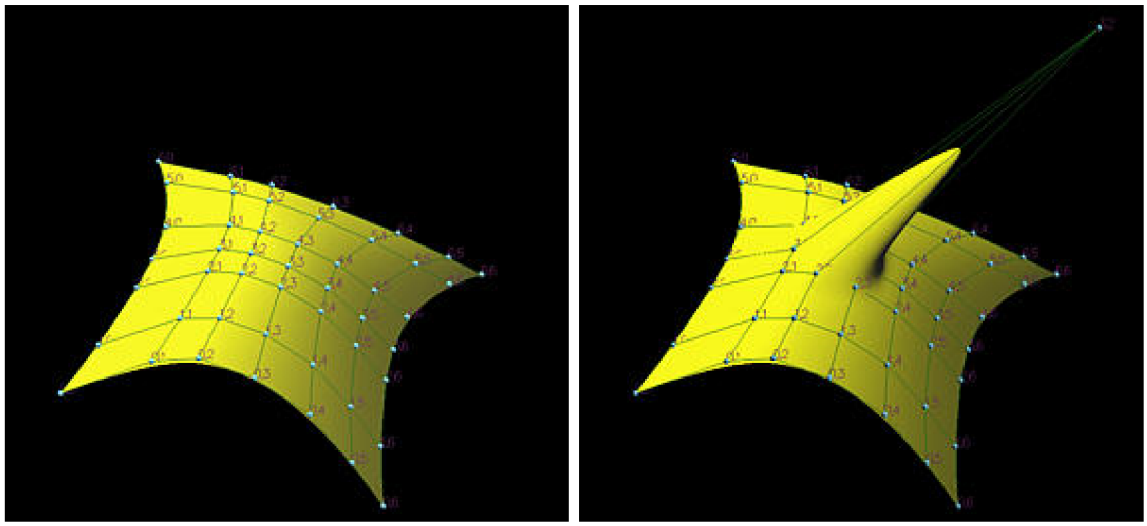
2. **Nezápornost** $N_i^m(t)N_j^n(u)$ jsou nezáporná $\forall i, j, m, n$.

3. **Vlastnost konvexního obalu** Pokud $(u, v) \in \langle u_i; u_{i+1}) \times \langle v_j; v_{j+1})$ potom bod $Q(u, v)$ leží v konvexním obalu definovaném řídicími body $P_{h,k}$ kde $i - m \leq h \leq i$ a $j - n \leq k \leq j$.

4. **Lokální kontrolovatelnost** Pokud $(u, v) \notin \langle u_i; u_{i+m+1}) \times \langle v_j; v_{j+n+1})$ potom $N_i^m(t)N_j^n(u) = 0$.

5. **Afinní invariantnost** Při libovolné afinní transformaci B-splajn plochy můžeme pracovat pouze s řídicími body této plochy.

6. Pokud zvolíme tvar uzlových vektorů $\mathbf{u} = (0, 0, \ldots, 0, 1, 1, \ldots, 1)$ a $\mathbf{v} = (0, 0, \ldots, 0, 1, 1, \ldots, 1)$, pak se B-splajn plocha stává Béziérovou plochou a platí $m = r$ a $n = s$



Obrázek 2.9: Znázornění významu vlastnosti popsané jako lokální kontrolovatelnost. Zdroj: [22]

Algoritmus de Boor

De Boorův algoritmus slouží k rychlému a numericky stabilnímu výpočtu tvaru B-splajn křivek. Tato část čerpá z [15], [21].

B-splajn křivky jsou definovány řídicími body a B-splajn bázovými funkcemi. Tyto B-splajn funkce jsou definovány rekurzivním způsobem (viz část 2.1.2), což způsobuje výpočetní náročnost. V této kapitole je popsán také algoritmus (viz strana 14), který výpočet B-splajn funkcí částečně urychluje. Pro přímý výpočet bodu na B-splajn křivce slouží dále popsaný de Boorův algoritmus. Tento algoritmus je založen na vhodném způsobu vkládání uzlů do uzlového vektoru.

Vložením uzlu do uzlového vektoru naroste počet složek vektoru. Aby byla zachována rovnost popsaná mezi vlastnostmi B-splajn křivek (viz strana 16), pak s každým přidaným uzlem (tedy složkou uzlového vektoru) musí narůst stupeň křivky, nebo počet řídicích bodů.

Zvýší-li se stupeň křivky, pak se tvar křivky změní, proto je vhodnější volbou přidání řídicího bodu. Uzly do uzlového vektoru přidáváme tak dlouho, dokud násobnost uzlu odpovídajícího vloženému řídicímu bodu není rovna stupni křivky. Pak lze použít následující větu:

Věta 1. *Má-li uzel t uzlového vektoru* $\mathbf{t}$ *násobnost rovnu stupni křivky, pak platí, že jemu odpovídající bod $C(t)$ je bod, který na křivce leží.*

Způsob vkládání si můžeme představit na jednoduchém vložení uzlů:

Jednoduché vložení uzlu Pro vložení uzlu t do uzlového vektoru $\mathbf{t} = (t_0, t_1, \ldots, t_n)$ postupujeme v následujících krocích:

1. Nalezneme interval $\langle t_k, t_{k+1})$ tak, aby $t \in \langle t_k, t_{k+1})$.

2. Vytvoříme posloupnost bodů $Q_i, i = k, k-1, \ldots, k-n+1$ jako lineární kombinaci sousedních řídicích bodů:

$$Q_i = (1 - \alpha_i)P_{i-1} + \alpha_i P_i \tag{2.25}$$

kde

$$\alpha_i = \frac{t - t_i}{t_{i+n} - t_i} \tag{2.26}$$

3. B-splajn křivka je určena posloupností $m + 2$ řídicích bodů:

$$P_0, P_1, \ldots, P_{k-n}, Q_{k-n+1}, Q_{k-n+2}, \ldots, Q_{k-1}, Q_k, P_k, P_{k+1}, \ldots, P_m \tag{2.27}$$

Násobné vkládání uzlů Násobné vkládání je zcela analogické. Liší se pouze vzorci užitými v 2. kroku postupu.

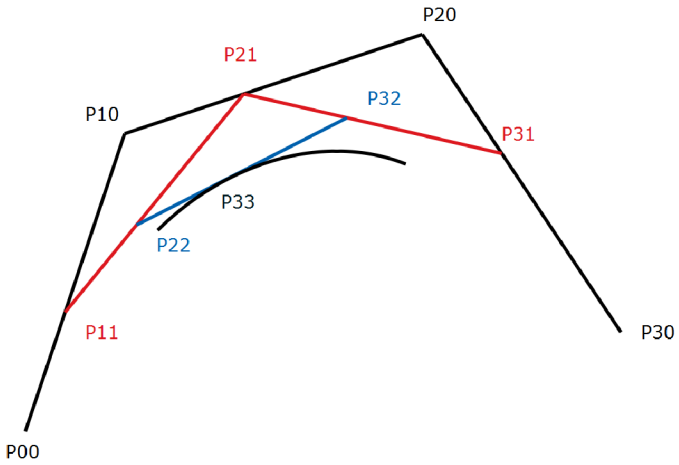
Chceme-li h-krát vložit uzel t do uzlového vektoru $\mathbf{t} = (t_0, t_1, \ldots, t_n)$, pak předpisy pro výpočet vypadají následovně:

$$Q_{i,h} = (1 - \alpha_{i,h})P_{i-1,h-1} + \alpha_{i,h}P_{i,h-1}, \tag{2.28}$$

kde

$$\alpha_{i,h} = \frac{t - t_i}{t_{i+n-(h-1)} - t_i} \tag{2.29}$$

Popis algoritmu pro výpočet bodu na křivce Myšlenka de Boorova algoritmu je založena na násobném vložení uzlu popsaném v předchozí části. Pro zvolený uzel provedeme jeho vložení tolikrát, aby jeho násobnost byla rovna stupni křivky. Nejprve tedy musíme zjistit, zda již není mezi složkami uzlového vektoru a to včetně jeho násobnosti. Pokud je násobnost uzlu nenulová, sníží se o stejnou hodnotu počet vložení uzlu.



Obrázek 2.10: Grafické vysvětlení vkládání uzlů

Algoritmus (de Boor)
Vstup: vkládaný uzel t, uzlový vektor $\mathbf{t}$, posloupnost řídicích bodů a stupeň křivky n
Výstup: bod křivky $C(t)$

1. Ověříme, že zadané vstupní hodnoty jsou řešitelné.
2. Nalezneme interval $\langle t_k, t_{k+1})$ tak, aby $t \in \langle t_k, t_{k+1})$.
3. Pokud $t = t_k$, pak musíme určit jeho násobnost s.
4. Provedeme $(n - s)$-násobné vložení uzlu:

```
for r := 1 to h do
  for i := k-n+r to k-s do
    begin
      a[i,r] = (u - u[i]) / (u[i+n-r+1] - u[i])
      p[i,r] = (1 - a[i,r])*p[i-1,r-1] + a[i,r]*p[i,r-1]
    end
 p(u) = p[k-s,n-s].
```

Popis algoritmu pro výpočet bodu na ploše Algoritmus probíhá ve dvou krocích

1. K mřížce řídicích bodů přistupujeme po řádcích (lze však ekvivalentně postupovat i po sloupcích). Pro každý řádek řídicích bodů provedeme de Boorův algoritmus pro výpočet bodu na křivce. Výsledkem je sloupec bodů na příslušných B-splajn křivkách.
2. Takto získané body použijeme jako řídicí body a aplikujeme na ně De Boorův algoritmus pro výpočet bodu na křivce. Výsledný bod je potom hledaných bodem na ploše.

2.2 Inverzní a pseudoinverzní matice

Část této práce se zabývá úkoly, pro které je nutná znalost maticového počtu. Použití maticového zápisu je výhodné pro jeho názornost a srozumitelnost v daných problémech. V následující části jsou uvedeny základní definice a věty maticového počtu, které budeme v práci využívat. Tato kapitola čerpá z [23] a [7].

2.2.1 Základní pojmy maticového počtu

V první části uvedeme základní pojmy z maticového počtu a lineární algebry, na které budeme navazovat kapitolami 2.2.2, 2.2.3 a 2.2.4 o inverzi a pseudoinverzi matice. Na úvod připomeneme standardní značení obvykle užívaných druhů matic a standardní maticové operace.

Označení 8. Nechť A a B jsou matice shodných typů. Symbolem $A + B$ budeme značit sčítání matic.

Označení 9. Nechť A a B jsou matice vhodných typů. Symbolem $A \cdot B$ budeme značit násobení matic.

Označení 10. Symbolem O_{mn} budeme označovat nulovou matici typu $m \times n$:

$$O_{mn} = \begin{pmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{pmatrix} \tag{2.30}$$

Označení 11. Symbolem I_n budeme v označovat jednotkovou matici řádu n:

$$I_n = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{pmatrix} \tag{2.31}$$

V další části této kapitoly jsou uvedeny základní pojmy maticového počtu a definice Hermitovsky sdružené matice, která je nezbytná pro definici pseudoinverze matice (viz 2.2.4).

Definice 12. Nechť $A = \{a_{ij}\}$ je matice typu $m \times n$. Pro $i, j \in \mathbb{N}$ $(1 \leq i \leq m, 1 \leq j \leq n)$ položme $b_{ji} = a_{ij}$. Pak matici $B = \{b_{ji}\}$ typu $n \times m$ nazveme maticí **transponovanou** k matici A a značíme A^T.

Věta 2. *Pro matice A, B vhodných typů platí:* $(A \cdot B)^T = B^T \cdot A^T$

Důkaz. Důkaz je uveden v [7].

Definice 13. Pokud pro prvky matice A platí $a_{ij} = a_{ji}$ (tedy $A^T = A$), pak matici A nazveme maticí **symetrickou**.

Definice 14. Pokud pro prvky matice A platí $a_{ij} = -a_{ji}$ (tedy $A^T = -A$), pak matici A nazveme maticí **antisymetrickou**.

Definice 15. Nechť $A = \{a_{ij}\}$ je matice typu $m \times n$. Pro $i, j \in \mathbb{N}$ $(1 \leq i \leq m, 1 \leq j \leq n)$ položme $b_{ij} = \overline{a_{ij}}$. Pak matici $B = \{b_{ij}\}$ typu $m \times n$ nazveme maticí **komplexně sdruženou** s maticí A a značíme $\bar{A}$.

Definice 16. Pokud pro prvky matice A platí $a_{ij} = \overline{a_{ji}}$ (tedy $\bar{A} = A$), pak matici A nazveme maticí **reálnou**.

Definice 17. Matici $\bar{A}^T$, kterou vytvoříme z matice A transpozicí a komplexním sdružením nazveme maticí **Hermitovsky sdruženou** s maticí A a značíme A^*.

Věta 3. *Pro matice A, B vhodných typů platí:* $(A \cdot B)^* = B^* \cdot A^*$

Důkaz. Přímo plyne z věty 2 a vlastností operace násobení komplexních čísel.

V následující části budou zavedeny pojmy hodnosti matice a úplné řádkové, resp. sloupcové hodnosti matice. Hodnost matice je v literatuře převážně definována jako nejvyšší řád nenulového minoru matice. Pro naše účely postačí ekvivalentní definice pomocí pomocí maximálního počtu lineárně nezávislých řádků, resp. sloupců.

Definice 18. Nechť A je matice typu $m \times n$. Hodností matice se rozumí maximální počet lineárně nezávislých řádků (což je zároveň rovno maximálnímu počtu lineárně nezávislých sloupců).

Definice 19. Nechť A je matice typu $m \times n$ a má hodnost $r(A)$.

1. Řekneme, že matice A má úplnou řádkovou hodnost, jestliže počet řádků je roven hodnosti matice, tedy $m = r(A)$
2. Řekneme, že matice A má úplnou sloupcovou hodnost, jestliže počet sloupců je roven hodnosti matice, tedy $n = r(A)$

Věta 4. *Nechť A, B jsou matice vhodných typů. Pak pro hodnost jejich součinu platí:*

$$r(A \cdot B) \leq \min(r(A), r(B)). \tag{2.32}$$

Důkaz. Důkaz je uveden v [23].

2.2.2 Inverzní matice

Častým úkolem v různých matematických odvětvích je tzv. inverzní úloha. Spočívá v tom, že již máme sestaven matematický model dané problematiky popisující vztah, kterým ze zadaných vstupních parametrů získáme výsledný stav. V praxi se však mnohdy pokládá otázka opačného charakteru, tedy že známe z pozorování či měření výsledný stav a chceme zjistit vstupní parametry. Takovým problémům se říká inverzní problémy. Jsou-li matematické modely popisující problém vyjádřitelné maticově, pak lze pro řešení inverzního problému s výhodou využít pojem inverzní matice.

Definice 20. Nechť A a X jsou čtvercové matice řádu n. Matici X splňující vlastnost $X \cdot A = I_n = A \cdot X$ nazveme **inverzní maticí** matice A. Má-li matice A inverzní matici, pak matici A nazveme maticí **invertibilní**. Inverzní matice k matici A je určena jednoznačně a značí se A^{-1}.

Věta 5 (o existenci)**.** *Čtvercová matice A má inverzi právě tehdy, je-li regulární (čili platí-li $\det(A) \neq 0$).*

Důkaz. 1. Dokažme, že z regularity plyne invertibilita. Předpokládejme, že matice A je regulární, tedy $\det A \neq 0$. Protože pro čtvercové matici C platí $C \cdot (\operatorname{adj} C) = (\operatorname{adj} C) \cdot C = (\det C) \cdot E$ (důkaz viz [7]), tak lze vyvodit, že:

$$A \cdot (\operatorname{adj} A) = (\operatorname{adj} A) \cdot A = (\det A) \cdot E$$

protože $\det A \neq 0$, tak lze předchozí rovnost podělit a získáme:

$$\frac{1}{\det A} A \cdot (\operatorname{adj} A) = \frac{1}{\det A} (\operatorname{adj} A) \cdot A = E$$

Označme $B = \frac{1}{\det A} \operatorname{adj} A$ a získáváme:

$$A \cdot B = B \cdot A = E$$

z čehož plyne, že matice $B = \frac{1}{\det A} \operatorname{adj} A$ je inverzí matice A.

2. Dokažme, že z invertibility plyne regularita. Předpokládejme, že matice A je invertibilní a B je její inverzní matice. Potom platí $A \cdot B = E$. Protože pro čtvercové matice C, D vhodné k násobení (tj. shodného řádu) platí $\det(C \cdot D) = \det(C) \cdot \det(D)$ (důkaz viz [7]), tak lze vyvodit, že $\det A \cdot \det B = \det E = 1$, z čehož plyne, že $\det A \neq 0$.

Poznámka 1. V důkazu předchozí věty je mimo jiné uveden jeden ze způsobů výpočtu inverzní matice (konkrétně využívající matici adjungovanou):

$$A^{-1} = \frac{1}{\det A} \operatorname{adj} A. \tag{2.33}$$

V dalším uvedeme větu popisující záměnu pořadí inverzního a Hermitovského operátoru.

Věta 6. *Pro regulární matici A platí*

$$(A^*)^{-1} = (A^{-1})^* \tag{2.34}$$

Důkaz. Důkaz plyne z definice Hermitovkého operátoru (viz definice 17) a z rovnosti (2.33) s využitím vlastností komplexního sdružení vzhledem k operacím sčítání a násobení komplexních čísel.

Myšlenka důkazu. Pro levou stranu platí:

$$(A^*)^{-1} = \frac{1}{\det(A^*)} \operatorname{adj}(A^*) = \frac{1}{\det\left(\overline{A}^T\right)} \operatorname{adj}(\overline{A}^T) = \frac{1}{\det(\overline{A})} \operatorname{adj}\left(\overline{A}^T\right)$$

Pro pravou stranu platí:

$$(A^{-1})^* = \left(\frac{1}{\det A} \operatorname{adj} A\right)^* = \overline{\left(\frac{1}{\det A} \operatorname{adj} A\right)}^T = \overline{\frac{1}{\det A} (\operatorname{adj} A)^T} = \frac{1}{\overline{\det A}} \left(\overline{\operatorname{adj} A}\right)^T$$

Tedy potřebujeme dokázat, že:

$$\frac{1}{\det(\overline{A})} \operatorname{adj}\left(\overline{A}^T\right) = \frac{1}{\overline{\det A}} \left(\overline{\operatorname{adj} A}\right)^T$$

Platnost rovnosti

$$\frac{1}{\det(\overline{A})} = \frac{1}{\overline{\det A}}$$

plyne z vlastností operací + a $\cdot$ nad oborem komplexních čísel.

Zbývá tedy dokázat, že:

$$\operatorname{adj}\left(\overline{A}^T\right) = \left(\overline{\operatorname{adj} A}\right)^T$$

což lze dokázat rozepsáním obou matic.

2.2.3 Inverzní matice zleva a zprava

Požadavky na invertibilitu matice (regularita, resp. i čtvercovost) jsou poměrně silné a tudíž značně omezující. Proto má smysl hledat mírnější požadavky v definici, tak aby zůstaly důležité vlastnosti zachovány, ale rozšířili jsme okruh matic splňujících definici. Jednoduchým zobecněním je omezení se na inverzi zprava, resp. zleva.

Definice 21. Nechť A je matice typu $m \times n$ a X je matice typu $n \times m$. Nechť dále platí:

1. $A \cdot X = I_m$. Pak matici X nazveme **inverzní maticí zprava** matice A.
2. $X \cdot A = I_n$. Pak matici X nazveme **inverzní maticí zleva** matice A.

Věta 7 (o existenci inverze zprava)**.** *Nechť A je matice, pak následující tvrzení jsou navzájem ekvivalentní:*

1. *matice $A \cdot A^*$ je regulární*
2. *matice A má inverzi zprava*
3. *matice A je úplné řádkové hodnosti*

Věta 8 (o existenci inverze zleva)**.** *Nechť A je matice, pak následující tvrzení jsou navzájem ekvivalentní:*

1. *matice $A^* \cdot A$ je regulární*
2. *matice A má inverzi zleva*
3. *matice A je úplné sloupcové hodnosti*

Důkaz. Důkazy těchto vět jsou technického charakteru, proto je zde nebudeme vypisovat a jsou k dispozici např. v [23].

Pro inverzní matici platí jednoduché pravidlo existence a jednoznačnosti. Zobecněním na inverzi zprava, resp. zleva se toto pravidlo rozšíří.

U čtvercových matic zůstává existence a jednoznačnost v souladu s poznatky z kapitoly o inverzi matice. To znamená, že je-li matice regulární, pak má právě jednu inverzní matici zleva a má právě jednu inverzní matici zprava. Navíc platí, že tyto matice jsou si rovny a jsou tedy přímo inverzí matice. Je-li čtvercová matice singulární, pak nemá žádnou inverzní matici zleva ani zprava (singulární matice totiž není úplné sloupcové ani řádkové hodnosti).

U obdélníkových matic je situace složitější. Existenci popisují dvě předchozí věty 7 a 8. Jednoznačnost však zaručena není a může existovat neprázdná a dokonce i více než jednoprvková množina inverzních matic zprava, resp. zleva. Množiny všech inverzních matic zprava, resp. zleva lze popsat pomocí permutačních matic, jak je uvedeno např. v [23].

2.2.4 Pseudoinverzní matice

V předcházející části jsme se zabývali pojmem inverze matice a inverze matice zleva, resp. zprava. Množina invertibilních matic je relativně malá, určení inverze zleva resp. zprava není často jednoznačné. Proto se zavádí pojem pseudoinverze, která je jednoznačně určena a lze ji určit pro obecnou matici.

Definice 22. Nechť A je matice typu $m \times n$ a X je matice typu $n \times m$, splňující následující, tzv. **Penroseovy podmínky**:

1. $A \cdot X \cdot A = A$
2. $X \cdot A \cdot X = X$
3. $(A \cdot X)^* = A \cdot X$
4. $(X \cdot A)^* = X \cdot A$

Pak matici X nazveme **Moore-Penroseovou inverzí** (zkráceně **M-P inverzí**) matice A, nebo též pseudoinverzní maticí matice A.

Věta 9 (o jednoznačnosti pseudoinverze)**.** *Má-li matice pseudoinverzi, pak je tato pseudoinverze určena jednoznačně.*

Důkaz. Předpokládejme, že matice A má dvě pseudoinverze, označené B a C. Vhodně užijeme Penroseových podmínek:

$$B \overset{(2)}{=} BAB = (BA)B \overset{(4)}{=} (A^*B^*)B = (A^*)B^*B \overset{(1)}{=} (A^*C^*A^*)B^*B \tag{2.35}$$

$$= (A^*C^*)(A^*B^*)B \overset{(4)}{=} (CA)(BA)B = (CA)(BAB) \overset{(2)}{=} CAB. \tag{2.36}$$

A tedy $B = CAB$. Obdobně ukážeme, že $C = CAB$:

$$C \overset{(2)}{=} CAC = C(AC) \overset{(3)}{=} C(C^*A^*) = CC^*(A^*) \overset{(1)}{=} CC^*(A^*B^*A^*) \tag{2.37}$$

$$= C(C^*A^*)(B^*A^*) \overset{(3)}{=} C(AC)(AB) = (CAC)(AB) \overset{(2)}{=} CAB. \tag{2.38}$$

Protože $B = CAB$ a $C = CAB$, tak $B = C$, tudíž je jednoznačnost pseudoinverze dokázána.

Poznámka 2. Důsledkem této věty je, že pro pseudoinverzi matice lze zavést označení. Obvykle se psedudoinverzní matice k matici A značí A^+

Vlastnosti pseudoinverzní matice

Věta 10 (Vztah pseudoinverze a inverze)**.** *Nechť A je regulární matice a A^{-1} je matice k ní inverzní. Pak tato inverzní matice splňuje Penroseovy podmínky a tudíž platí: $A^+ = A^{-1}$.*

Důkaz. Je zřejmý. Probíhá ověřením Penroseových podmínek pro $X = A^{-1}$:

1. $A \cdot X \cdot A = A \cdot A^{-1} \cdot A = A \cdot I = A$
2. $X \cdot A \cdot X = A^{-1} \cdot A \cdot A^{-1} = A^{-1} \cdot I = A^{-1} = X$
3. $(A \cdot X)^* = (A \cdot A^{-1})^* = I^* = I = A \cdot A^{-1} = A \cdot X$
4. $(X \cdot A)^* = (A^{-1} \cdot A)^* = I^* = I = A^{-1} \cdot A = X \cdot A$

Věta 11. *Jestliže matice A má pseudoinverzi, pak tato její pseudoinverze A^+ má pseudoinverzi a platí $(A^+)^+ = A$*

Důkaz. Probíhá ověřením Penroseových podmínek:

1. $A^+ \cdot A^{++} \cdot A^+ = A^+ \cdot A \cdot A^+ = A^+$ (podle Penroseovy podmínky (2))
2. $A^{++} \cdot A^+ \cdot A^{++} = A \cdot A^+ \cdot A = A^{++} = A$ (podle Penroseovy podmínky (1))
3. $(A^+ \cdot A^{++})^* = (A^+ \cdot A)^* = A^+ \cdot A = A^+ \cdot A^{++}$ (podle Penroseovy podmínky (4))
4. $(A^{++} \cdot A^+)^* = (A \cdot A^+)^* = A \cdot A^+ = A^{++} \cdot A^+$ (podle Penroseovy podmínky (3))

Věta 12. *Nechť $A = O_{m,n}$ je nulová matice typu $m \times n$. Pro $X = O_{n,m}$ lze ověřit Penroseovy podmínky a tudíž platí $O^+_{m,n} = O_{n,m}$*

Důkaz. Probíhá ověřením Penroseových podmínek:

1. $O_{m,n} \cdot O_{n,m} \cdot O_{m,n} = O_{m,n}$
2. $O_{n,m} \cdot O_{m,n} \cdot O_{n,m} = O_{n,m}$
3. $(O_{m,n} \cdot O_{n,m})^* = O_{m,n} \cdot O_{n,m}$
4. $(O_{n,m} \cdot O_{m,n})^* = O_{n,m} \cdot O_{m,n}$

Věta 13. *Nechť $A = D_n$ je diagonální matice řádu n vypadající následovně:*

$$D = \begin{pmatrix} d_1 & 0 & \cdots & 0 \\ 0 & d_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & d_n \end{pmatrix} \tag{2.39}$$

Pak Moore-Penroseova inverze D^+ matice D se vytvoří podle následujícího schématu:

$$D^+ = \begin{pmatrix} d_1^+ & 0 & \cdots & 0 \\ 0 & d_2^+ & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & d_n^+ \end{pmatrix}, d_i^+ = \begin{cases} d_i^{-1} & \text{pro} \quad d_i \in \mathbb{C} \setminus \{0\} \\ 0 & \text{pro} \quad d_i = 0 \end{cases} \tag{2.40}$$

Důkaz. Pokud jsou všechny prvky diagonály matice nulové, pak se jedná o nulovou matici řádu n (tedy typu $n \times n$) a dle předchozí věty 12 je sama sobě pseudoinverzí. Uvažujme, že je alespoň jeden z prvků diagonály nenulový.

Platí, že součinem dvou diagonálních matic shodných řádů, je opět diagonální matice totožného řádu a daný prvek diagonály výsledné matice je součinem odpovídajících prvků diagonál obou matic. Provedeme-li násobení matice D a D^+ (platí i pro opačné pořadí) máme konkrétně:

$$d_i \cdot d_i^+ = \begin{cases} d_i \cdot d_i^{-1} = 1 & \text{pro} \quad d_i \in \mathbb{C} \setminus \{0\} \\ 0 & \text{pro} \quad d_i = 0 \end{cases} \tag{2.41}$$

Výsledkem je tedy diagonální matice na jejíž diagonále jsou pouze nuly a jedničky.

Nuly i jedničky jsou reálná čísla, tedy jsou sama sobě komplexně sdružená, matice je tedy reálná a tím pádem je sama sobě maticí komplexně sdruženou. Diagonální matice je matice symetrická, tudíž transpozice ji nezmění. Zřejmě jsou tedy splněny Penroseovy podmínky (3) a (4).

Zbývá ověřit podmínky (1) a (2). Již víme, že vynásobením matic D a D^+ získáme diagonální matici s pouze jedničkami a nulami na diagonále. Navíc nuly se vyskytují jen na místech, kde byly v matici D (tudíž i v matici D^+). Tam, kde byly v matici D (tudíž i v matici D^+) nenulová čísla, je v jejich násobku jednička. Proto platí:

$$d_i \cdot d_i^+ \cdot d_i = \begin{cases} 1 \cdot d_i = d_i & \text{pro} \quad d_i \in \mathbb{C} \setminus \{0\} \\ 0 & \text{pro} \quad d_i = 0, \end{cases} \tag{2.42}$$

čímž dokážeme podmínku (1) a podmínka (2) se dokáže obdobně.

Věta 14. *1. Nechť A je matice typu $m \times n$ $(m \leq n)$ s úplnou řádkovou hodností. Pak je matice $A \cdot A^*$ regulární, tudíž i invertibilní a pro pseudoinverzi matice A platí:*

$$A^+ = A^* \cdot (A \cdot A^*)^{-1} \text{ a } A \cdot A^+ = I_m \tag{2.43}$$

2. Nechť A je matice typu $m \times n$ $(m \geq n)$ s úplnou sloupcovou hodností. Pak je matice $A^* \cdot A$ regulární, tudíž i invertibilní a pro pseudoinverzi matice A platí:

$$A^+ = (A^* \cdot A)^{-1} \cdot A^* \text{ a } A^+ \cdot A = I_n \tag{2.44}$$

Důkaz. 1. Uvažujme, že matice A splňuje předpoklady prvního tvrzení této věty. Pro libovolnou matici M platí, že matice M, M^*, $M \cdot M^*$ a $M^* \cdot M$ mají stejnou hodnost.V našem případě tedy máme $m = r(A) = r(A \cdot A^*)$, z čehož vyplývá regularita matice $A \cdot A^*$. Položme tedy

$$X = A^* \cdot (A \cdot A^*)^{-1} \tag{2.45}$$

a ověřme Penroseovy podmínky:

(a) $A \cdot X \cdot A = A \cdot A^* \cdot (A \cdot A^*)^{-1} \cdot A = I_m \cdot A = A$

(b) $X \cdot A \cdot X = X \cdot A \cdot A^* \cdot (A \cdot A^*)^{-1} = X \cdot I_m = X$

(c) $(A \cdot X)^* = (A \cdot A^* \cdot (A \cdot A^*)^{-1})^* = I_m^* = I_m = A \cdot A^* \cdot (A \cdot A^*)^{-1} = A \cdot X$

(d) $(X \cdot A)^* = A^* \cdot X^* = A^* \cdot (A^* \cdot (A \cdot A^*)^{-1})^* = A^* \cdot ((A \cdot A^*)^{-1})^* \cdot A^{**} = A^* \cdot ((A \cdot A^*)^*)^{-1} \cdot A \overset{=}{} A^* \cdot (A^{**} \cdot A^*)^{-1} \cdot A \overset{=}{} A^* \cdot (A \cdot A^*)^{-1} \cdot A \overset{=}{} X \cdot A$

2. Důkaz probíhá zcela analogicky

Pseudoinverze a skeletní rozklad

Výpočet pseudoinverzní matice lze provést několika způsoby. Pro ilustraci zde představíme výpočet pseudoinverze s využitím skeletního rozkladu. Mezi dalšími možnostmi je možno zmínit využití singulárního rozkladu nebo Grevilleův algoritmus (tyto i další způsoby výpočtu lze nalézt např. v [9]). Tato část čerpá z [23].

Věta 15 (O skeletním rozkladu)**.** *Nechť A je nenulová matice typu $m \times n$ s hodností $r \geq 1$. Pak $\exists$ matice B, C typu $m \times r, r \times n$ splňující:*

$$A = B \cdot C,\ r(B) = r(C) = r. \tag{2.46}$$

Rozkladu $A = B \cdot C$ se říká ***skeletní rozklad matice A****.*

Důkaz. Sloupce matice A zapišme bloky blokové matice: $A = (s_1 \dots s_n)$. Ze sloupců matice A vyberme r lineárně nezávislých sloupců $\sigma_1, \dots, \sigma_r$ a vytvoříme z nich matici $B = (\sigma_1 \dots \sigma_r)$. Tato matice B je pak zjevně matice typu $m \times r$ a je plné hodnosti, tedy $r(B) = r$.

Matici C vytvoříme následovně: uvažujme ji v požadovaném tvaru – tedy jako blokovou matici se sloupcovými vektory $\xi_1, \dots, \xi_n$ dimenze r, tedy

$$C = (\xi_1 \cdots \xi_n) \tag{2.47}$$

tak aby splňovala rovnici $A = B \cdot C$, tedy

$$A = (B\xi_1 \cdots B\xi_n), \tag{2.48}$$

přičemž $B\xi_j = s_j$ $(1 \leq j \leq n)$. Tím dostáváme n soustav lineárních rovnic o r neznámých.

Pro matici B těchto soustav platí $r(B) = r$. Rozšířená matice j-té $(1 \leq j \leq n)$ soustavy je v blokovém zápisu tvaru (Bs_j) a je také hodnosti r (protože sloupcový vektor, kterým je matice B rozšířena, je lineární kombinací sloupců matice B).

Protože se hodnosti matic soustav shodují s hodnostmi příslušných rozšířených matic soustav, tak jsou dané soustavy řešitelné, pak ξ_j označme jejich řešení j-té soustavy. Pro tato řešení platí:

$$A = B \cdot C = B \cdot (\xi_1 \cdots \xi_n) \tag{2.49}$$

Na závěr je ještě třeba ověřit platnost rovnosti $r(C) = r$. Podle věty 4 platí: $r = r(A) = r(B \cdot C) \leq r(C) \leq r(C) \leq r$, z čehož přímo plyne požadovaný vztah.

Příklad 1. Nalezněte skeletní rozklad matice A zadané následovně:

$$A = \begin{pmatrix} 1 & 4 & 3 \\ -1 & 1 & 2 \\ -2 & -2 & 0 \end{pmatrix} \tag{2.50}$$

Řešení. Libovolným způsobem ověříme, že hodnost matice je 2. Vybereme-li tedy libovolnou dvojici sloupců, pak budou lineárně nezávislé. Označíme-li sloupce matice A jako s_1, s_2, s_3, pak je zřejmé, že platí např. $s_2 = s_1 + s_3$. Vybereme tedy s_1 a s_3 jako slupce matice B:

$$B = (s_1 s_3) = \begin{pmatrix} 1 & 3 \\ -1 & 2 \\ -2 & 0 \end{pmatrix}, \xi_1 = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}, \xi_2 = \begin{pmatrix} x_2 \\ y_2 \end{pmatrix}, \xi_3 = \begin{pmatrix} x_3 \\ y_3 \end{pmatrix}. \tag{2.51}$$

Nyní hledáme $x_i, y_i (i = 1, 2, 3)$ tak, aby $A = B \cdot C$, přičemž $C = (\xi_1 \xi_2 \xi_3)$ Dostáváme tak trojici soustav lineárních rovnic:

$$B \cdot \xi_1 = s_1 \quad B \cdot \xi_2 = s_2 \quad B \cdot \xi_3 = s_3 \tag{2.52}$$

tedy:

$$\begin{array}{rlrlrl} x_1 + 3y_1 & = 1 & x_2 + 3y_2 & = 4 & x_3 + 3y_3 & = 3 \\ -x_1 + 2y_1 & = -1 & -x_2 + 2y_2 & = 1 & -x_3 + 2y_3 & = 2 \\ -2x_1 + 0y_1 & = -2 & -2x_2 + 0y_2 & = -2 & -2x_3 + 0y_3 & = 0 \end{array} \tag{2.53}$$

tyto soustavy vyřešíme a získáváme: $x_1 = 1, y_1 = 0; x_2 = 1, y_2 = 1; x_3 = 0, y_3 = 1$. Matice C tedy vypadá následovně:

$$C = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \tag{2.54}$$

a $A = B \cdot C$ je skeletní rozklad matice A.

Věta 16 (O existenci pseudoinverzní matice)**.** *Nechť A je nenulová matice se skeletním rozkladem $A = B \cdot C$. Pak matice A má pseudoinverzi zadanou následujícím vztahem:*

$$A^+ = C^+ \cdot B^+ \tag{2.55}$$

Důkaz. Matice B, C jsou plné (sloupcové, řádkové) hodnosti. Podle tvrzení věty 14 platí: $B^+ \cdot B = C \cdot C^+ = I_r$. Položme tedy $X = C^+ \cdot B^+$ a ověřme, že X je pseoudoinverzní maticí k matici A ověřením Penroseových podmínek:

1. $A \cdot X \cdot A = (B \cdot C) \cdot (C^+ \cdot B^+) \cdot (B \cdot C) = B \cdot (C \cdot C^+) \cdot (B^+ \cdot B) \cdot C = B \cdot I_r \cdot I_r \cdot C = B \cdot C = A$
2. $X \cdot A \cdot X = (C^+ \cdot B^+) \cdot (B \cdot C) \cdot (C^+ \cdot B^+) = C^+ (\cdot B^+ \cdot B)(\cdot C \cdot C^+) \cdot B^+ = C^+ I_r \cdot I_r \cdot B^+ = C^+ \cdot B^+ = X$
3. $(A \cdot X) = (B \cdot C) \cdot (C^+ \cdot B^+) = B \cdot (C \cdot C^+) \cdot B^+ = B \cdot I_r \cdot B^+ = B \cdot B^+$
   tedy $(A \cdot X)^* = (B \cdot B^+)^* = B \cdot B^+ = A \cdot X$
4. $X \cdot A = (C^+ \cdot B^+) \cdot (B \cdot C) = C^+ \cdot (B^+ \cdot B) \cdot C = C^+ \cdot I_r \cdot C = C^+ \cdot C$
   tedy $(X \cdot A)^* = (C^+ \cdot C)^* = C^+ \cdot C = X \cdot A$

Příklad 2. Vypočtěte pseudoinverzní matici k matici A zadané následovně:

$$A = \begin{pmatrix} 1 & 4 & 3 \\ -1 & 1 & 2 \\ -2 & -2 & 0 \end{pmatrix} \tag{2.56}$$

Řešení. V předchozím příkladu 1 jsme spočítali skeletní rozklad matice A:

$$A = B \cdot C, B = \begin{pmatrix} 1 & 3 \\ -1 & 2 \\ -2 & 0 \end{pmatrix}, C = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \tag{2.57}$$

Pseudoinverze matic B, C určíme pomocí tvrzení věty 14:

$$\begin{aligned} B^+ = (B^* \cdot B)^{-1} \cdot B^* &= \left(\begin{pmatrix} 1 & -1 & -2 \\ 3 & 2 & 0 \end{pmatrix} \begin{pmatrix} 1 & 3 \\ -1 & 2 \\ -2 & 0 \end{pmatrix} \right)^{-1} \begin{pmatrix} 1 & -1 & -2 \\ 3 & 2 & 0 \end{pmatrix} \\ &= \begin{pmatrix} 6 & 1 \\ 1 & 13 \end{pmatrix}^{-1} \begin{pmatrix} 1 & -1 & -2 \\ 3 & 2 & 0 \end{pmatrix} = \frac{1}{77} \begin{pmatrix} 13 & -1 \\ -1 & 6 \end{pmatrix} \begin{pmatrix} 1 & -1 & -2 \\ 3 & 2 & 0 \end{pmatrix} \\ &= \frac{1}{77} \begin{pmatrix} 10 & -15 & -26 \\ 17 & 13 & 2 \end{pmatrix} \end{aligned} \tag{2.58}$$

$$\begin{aligned} C^+ = C^* \cdot (C \cdot C^*)^{-1} &= \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix} \left(\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix} \right)^{-1} \\ &= \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}^{-1} = \frac{1}{3} \begin{pmatrix} 1 & 0 \\ 0 & \frac{4}{3} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} \\ &= \frac{1}{3} \begin{pmatrix} 2 & -1 \\ 1 & 1 \\ -1 & 2 \end{pmatrix} \end{aligned} \tag{2.59}$$

Pro pseudoinverzní matici platí:

$$A^+ = C^+ \cdot B^+ = \frac{1}{3} \begin{pmatrix} 2 & -1 \\ 1 & 1 \\ -1 & 2 \end{pmatrix} \frac{1}{77} \begin{pmatrix} 10 & -15 & -26 \\ 17 & 13 & 2 \end{pmatrix} = \frac{1}{231} \begin{pmatrix} 3 & -43 & -54 \\ 27 & -2 & -24 \\ 24 & 41 & 30 \end{pmatrix} \tag{2.60}$$

2.2.5 Metoda nejmenších čtverců minimální v normě

Pro výpočet zpětné FFD transformace v části 3.1.1 budeme potřebovat řešení poddeterminovaného (nedourčeného) systému lineárních rovnic. To znamená, že matice soustavy nemá úplnou sloupcovou hodnost, ale má úplnou řádkovou hodnost. Máme tedy více neznámých než rovnic. Z toho vyplývá, že existuje více než jedno řešení. Našim úkolem je nalézt řešení, které je nejlepší, tedy které má nejmenší normu.

Hledáme tedy vektor, který splňuje:

$$A\mathbf{x} = \mathbf{b}, \text{ kde } \mathbf{x} = \min(\mathbf{x}^T\mathbf{x}) \tag{2.61}$$

Tuto úlohu lze chápat jako optimalizační problém pro Lagrangeovu podmínku (viz např. [8]):

$$\mathcal{F} = \mathbf{x}^T\mathbf{x} + \lambda^T(A\mathbf{x} - \mathbf{b}) \tag{2.62}$$

kde λ ve vektor tzv. **Lagrangeových multiplikátorů** (Lagrangeových násobků).

Jedná se o optimalizační úlohu. Optimum je extrém funkce $\mathcal{F}$. Funkce $\mathcal{F}$ nabývá svého extrému, jsou-li její parciální derivace rovny nule:

$$\begin{aligned} \frac{\partial \mathcal{F}}{\partial \mathbf{x}} &= \mathbf{0} \\ 2\mathbf{x}^T + \lambda^T A &= \mathbf{0} \end{aligned} \tag{2.63}$$

z čehož plyne:

$$\mathbf{x} = \frac{1}{2}A^T\lambda \tag{2.64}$$

Dosadíme-li nyní (2.64) do (2.61), dostaneme:

$$\begin{aligned} A\left(\tfrac{1}{2}A^T\lambda\right) &= \mathbf{b} \\ \tfrac{1}{2}AA^T\lambda &= \mathbf{b} \end{aligned} \tag{2.65}$$

Předpokládejme, že matice A je úplné řádkové hodnosti. Potom je matice AA^T podle věty 7 maticí regulární a tudíž invertibilní. Proto lze z rovnice (2.65) osamostatnit člen λ:

$$\begin{aligned} \tfrac{1}{2}AA^T\lambda &= \mathbf{b} \\ \underbrace{\underbrace{(AA^T)^{-1}}_{M^{-1}}\underbrace{AA^T}_{M}}_{I}\lambda &= 2(AA^T)^{-1}\mathbf{b} \\ \lambda &= 2(AA^T)^{-1}\mathbf{b} \end{aligned} \tag{2.66}$$

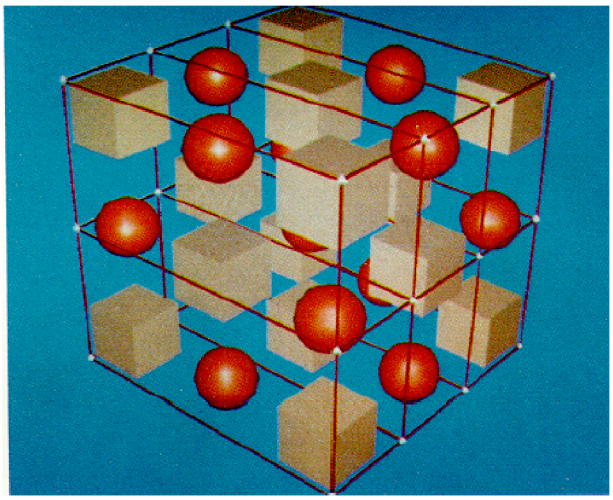
což dosadíme do rovnice (2.64) a s využitím věty 14 dostaneme:

$$\begin{aligned} \mathbf{x} &= \tfrac{1}{2}2\underbrace{A^T(AA^T)^{-1}}_{A^+}\mathbf{b} \\ \mathbf{x} &= A^+\mathbf{b} \end{aligned} \tag{2.67}$$

2.3 Metody FFD

Free-form deformation, zkráceně FFD, je technika sloužící k úpravě tvaru křivek a ploch při jejich počítačovém modelování. Nejvýznamnější použití má v oblasti počítačové grafiky v oblasti tzv. free-form modelování, kde různými nástroji můžeme volně upravovat tvar [1], [12]. Návrhy, které vznikají, mohou mít další omezující podmínky, například zachovávající objem daného tělesa [4], [16]. FFD je také možné použít pro vytváření aerodynamických tvarů v leteckém průmyslu [18]. FFD se využívá také v lékařství pro registraci obrazů vzniklých při magnetické rezonanci [17] nebo CT skenech [20] pro výzkum v oblasti nádorových onemocnění. Tato část čerpá z [19], [13] a [5].

Objekt, se kterým chceme manipulovat, umístíme do rovnoběžnostěnu. Po přemístění řídicího bodu je nová pozice bodů objektu určena váhovým součtem řídicích bodů. Váhy jsou funkce lokálních souřadnic přiřazené ke každému z řídicích bodů. Změna pozice řídicího bodu provede zároveň změnu tvaru tělesa, tedy změnu pozic bodů uvnitř tělesa.

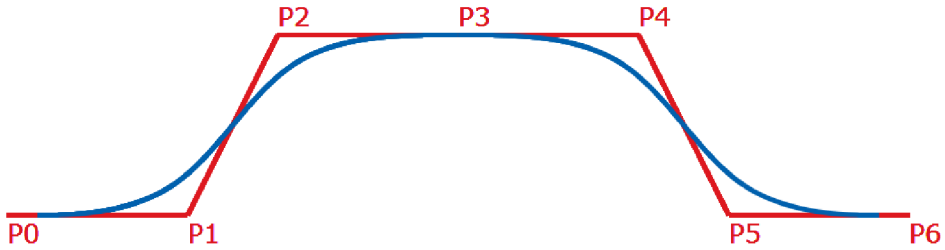


Obrázek 2.11: Ukázka objektu vsazeného v rovnoběžnostěnu. Zdroj [19]

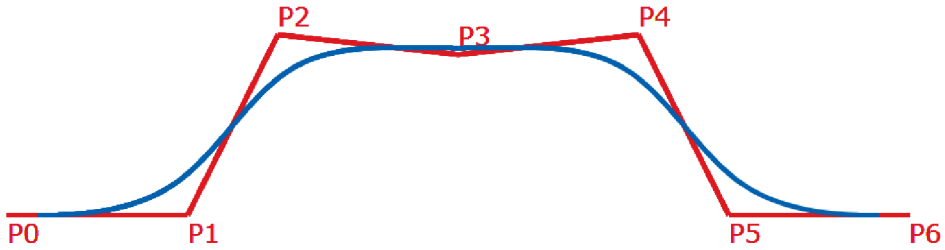
Přímá metoda manipulace s plochou je značně obtížná. Metoda totiž spočívá v pohybování bodů přímo na ploše. Velkým problémem je také například vytvoření úsečky z části splajn křivky, resp. roviny z části splajn plochy: řídicí body takového útvaru jsou uspořádány výrazně neintuitivně – viz obr. 2.12 a 2.13

Při práci s řídícími body obecných ploch se setkáváme s těmito problémy:

1. problém dosažení požadovaného tvaru křivky, resp. plochy,
2. problém dosažení přesného umístění daného bodu,
3. nesnadnost práce s matematickým modelem popisujícím deformaci,
4. nemožnost pracovat i s komplexními modely.



Obrázek 2.12: Řídicí body P_2, P_3 a P_4 jsou uspořádány v přímce, B-splajn křivka netvoří nikde rovný úsek



Obrázek 2.13: Řídicí body P_2, P_3 a P_4 neleží na přímce, část B-splajn křivky však tvoří rovný úsek

2.3.1 Transformace pomocí Bernsteinových polynomů

Tato kapitola čerpá z původní práce profesora Sederberga [19], který poprvé uvedl termín Free-form deformation a z tohoto článku celý následující vývoj FFD vychází.

Nejdříve zavedeme lokální souřadnicovou soustavu. Libovolný bod X má polohový vektor $\mathbf{X}$, který lze zapsat jako:

$$\mathbf{X} = \mathbf{X}_0 + s\mathbf{s} + t\mathbf{t} + u\mathbf{u} \tag{2.68}$$

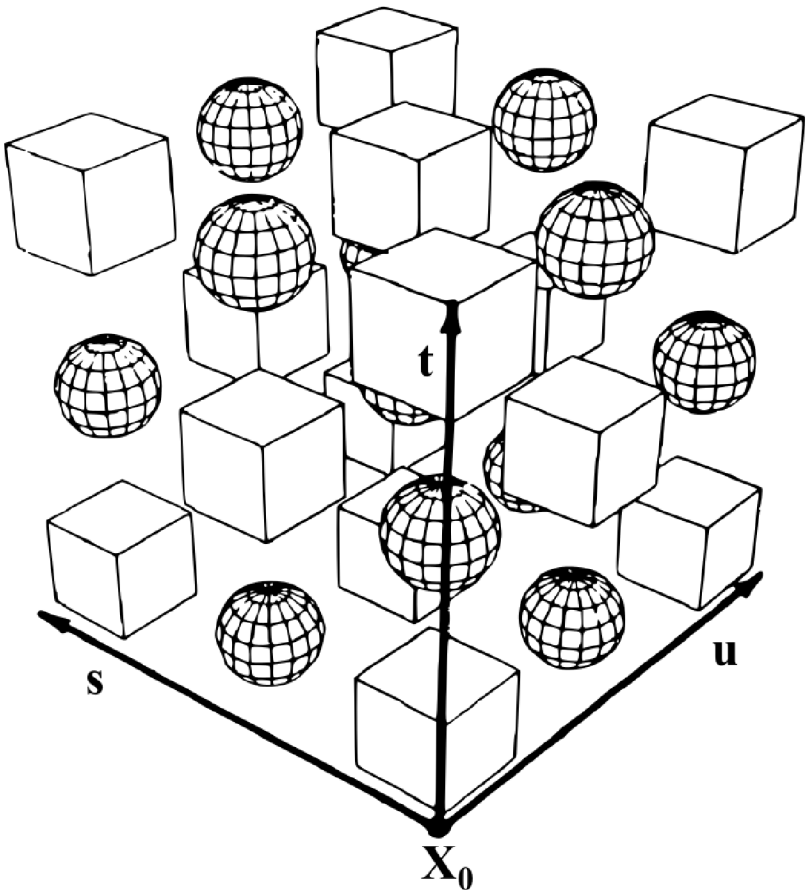
kde $\mathbf{X}_0$ je polohový vektor počátku lokální souřadnicové soustavy a $\mathbf{s}$, $\mathbf{t}$ a $\mathbf{u}$ jsou jednotkové vektory navzájem kolmých souřadných směrů. Pak s, t a u jsou souřadnice lokální souřadnicové soustavy. Pro vnitřní body rovnoběžnostěnu v této lokální souřadnicové soustavě platí: $0 < s < 1$, $0 < t < 1$ a $0 < u < 1$ (viz obr. 2.14).

Souřadnice lokální souřadnicové soustavy pro libovolný bod lze získat následovně:

$$s = \frac{\mathbf{t} \times \mathbf{u} \cdot (\mathbf{X} - \mathbf{X}_0)}{\mathbf{t} \times \mathbf{u} \cdot \mathbf{s}} \tag{2.69}$$

$$t = \frac{\mathbf{s} \times \mathbf{u} \cdot (\mathbf{X} - \mathbf{X}_0)}{\mathbf{s} \times \mathbf{u} \cdot \mathbf{t}} \tag{2.70}$$

$$u = \frac{\mathbf{s} \times \mathbf{t} \cdot (\mathbf{X} - \mathbf{X}_0)}{\mathbf{s} \times \mathbf{t} \cdot \mathbf{u}} \tag{2.71}$$



Obrázek 2.14: Zavedení lokální souřadnicové soustavy (podle [19])

V dalším kroku rovnoběžnostěnu připojíme mřížku řídicích bodů. Tato mřížka má $l+1$ bodů ve směru vektoru $\mathbf{s}$, $m+1$ bodů ve směru vektoru $\mathbf{t}$ a $n+1$ bodů ve směru vektoru $\mathbf{u}$. Řídicí bod P_{ijk} má potom souřadnice $\mathbf{P}_{ijk}$ určené podle (2.68):

$$\mathbf{P}_{ijk} = \mathbf{X}_0 + \frac{i}{l}\mathbf{s} + \frac{j}{m}\mathbf{t} + \frac{k}{n}\mathbf{u} \tag{2.72}$$

Deformace rovnoběžnostěnu je dána změnou polohy řídicích bodů P_{ijk} vůči jejich počáteční poloze v pravidelné mřížce. Pro libovolný bod X rovnoběžnostěnu lze jeho deformovanou polohu $\mathbf{X}_{pos}$ nalézt podle následujícího vztahu:

$$\mathbf{X}_{pos} = \sum_{i}^{l} \binom{l}{i} (1-s)^{l-i} s^i \left(\sum_{j}^{m} \binom{m}{j} (1-t)^{m-j} t^j \left(\sum_{k}^{n} \binom{n}{k} (1-u)^{n-k} u^k \mathbf{P}_{ijk} \right) \right) \tag{2.73}$$

kde $\mathbf{P}_{ijk}$ je polohový vektor i,j,k-tého řídicího bodu vyjádřený ve výše zavedené lokální kartézské souřadné soustavě. Tento vztah udává polohový vektor $\mathbf{X}_{pos}$ bodu X po deformaci.

2.3.2 Transformace pomocí B-splajn funkcí

Tato kapitola se zabývá transformací pomocí B-splajn funkce a vychází z článku [13]. V tomto článku se uvažuje dvourozměrná free-form deformace rovnoběžníku, zatímco v článku [19], který slouží jako základ k předchozí kapitole, je uvažována trojrozměrná free-form deformace rovnoběžnostěnu. Zobecnění případu dvojrozměrné FFD na trojrozměrnou lze snadno provést, stejně tak jako zúžení prostorového případu na rovinný.

Mějme rovnoběžník Ω v rovině xy. K tomuto rovnoběžníku vytvoříme lokální souřadnicovou soustavu u, v tak, že libovolný bod X rovnoběžníku bude mít v této lokální souřadné soustavě následující vyjádření:

$$\mathbf{X} = (u, v) \text{ kde } 1 \leq u \leq m \text{ a } 1 \leq v \leq n \tag{2.74}$$

Tento rovnoběžník propojíme s mřížkou P řídicích bodů P_{ij} následujícím způsobem. Mřížka řídicích bodů nechť je tvořena $(m+2) \times (n+2)$ řídicími body P_{ij} tak, že i, j-tý řídicí bod leží v počáteční konfiguraci v $\mathbf{P}^0_{ij} = (i, j)$.

Mějme tedy mřížku P řídicích bodů P_{ij} umístěnou na rovnoběžníku, který chceme deformovat. Body ležící na B-splajn ploše určené těmito řídicími body lze spočítat podle následující formule:

$$\mathbf{w}(u, v) = \sum_{k=0}^{3} \sum_{l=0}^{3} N^3_k(s) N^3_l(t) \mathbf{P}_{(i+k)(j+l)} \tag{2.75}$$

kde $i = \lfloor u \rfloor - 1$, $j = \lfloor v \rfloor - 1$, $s = u - \lfloor u \rfloor$ a $t = v - \lfloor v \rfloor$. V předchozím vztahu se uvažují $N^3_k(s)$ a $N^3_l(t)$ jako uniformní B-splajn bázové funkce (používají ekvidistantní uzlový vektor) vyčíslené pro s a t. Definici 1 B-splajn bázových funkcí lze nalézt v kapitole 2.1.2.

V celé práci používáme bázové funkce třetího stupně, tedy kubické B-splajn bázové funkce, proto pracujeme pouze s mřížkou 4×4. Podle vlastnosti B-splajn křivek je nutné mít pro výpočet bázové funkce stupně n právě $n+1$ řídicích bodů. Pro výpočet B-splajn křivek a ploch bývá využíván de Boorův algoritmus popsaný v kapitole 2.1.2, který přináší zrychlení výpočtu B-splajn funkcí oproti výpočtu pomocí jejich rekurentní definice. Zde však pracujeme se zcela specifickým typem B-splajn funkcí. Uniformní B-splajn funkce třetího stupně jsou totiž totožné s bázovými polynomiálními funkcemi Coonsovy křivky. Pak lze místo $N^3_l(t)$ psát $C_0(t)$. Bázové polynomy Coonsovy křivky jsou popsány již v kapitole 2.1.1 a jejich definice je:

$$C_0(t) = \frac{1}{6}(-t^3 + 3t^2 - 3t + 1) \tag{2.76}$$

$$C_1(t) = \frac{1}{6}(3t^3 + 6t^2 + 4) \tag{2.77}$$

$$C_2(t) = \frac{1}{6}(-3t^3 + 3t^2 + 3t + 1) \tag{2.78}$$

$$C_3(t) = \frac{1}{6}t^3 \tag{2.79}$$

kde $0 \leq t < 1$.

Z rovnice (2.75) vyplývá, že posunutá poloha $\mathbf{w}(\mathbf{X})$ bodu X rovnoběžníku Ω bude ovlivněna jen 16 okolními řídicími body.

Máme-li B-splajn křivku určenou řídicími body uspořádanými do přímky, pak je tato křivka úsečkou. Tím pádem při počátečním nastavení mřížky P dostaneme nedeformovaný tvar rovnoběžníku, tedy:

$$\mathbf{w}^0(u,v) = (u,v) = \sum_{k=0}^{3}\sum_{l=0}^{3} B_k(s)B_l(t)\phi^0_{(i+k)(j+l)}. \tag{2.80}$$

Vlastnost one-to-one

Goodman a Unsworth [2] představili postačující podmínku pro dvoudimenzionální Bézierovu plochu tak, aby měla vlastnost one-to-one, tedy aby existovalo injektivní zobrazení mezi původní a transformovanou plochou. Ve své práci uvádějí, že tuto podmínku lze použít i pro dvoudimenzionální B-splajn plochu. Pro mřížku $m \times n$ řídicích bodů uvažujeme v podmínce $2m(m+1)+2n(n+1)$ lineárních nerovnic. To znamená, že při nárůstu počtu řídicích bodů výrazně naroste čas potřebný k ověření těchto podmínek. Navíc pokud podmínka není splněna, neexistuje snadný způsob manipulace s řídicí mřížkou pro splnění podmínek.

Věta 17 (one-to-one podmínka)**.** *Označme* $\Delta\mathbf{P}_{ij} = \mathbf{P}_{ij} - \mathbf{P}^0_{ij}$ *posunutí* i,j*-tého řídicího bodu vůči počáteční poloze. Pak funkce* $\mathbf{w}$ *zavedená v* (2.75) *má vlastnost one-to-one tehdy, když* $|\Delta\phi_{ij}|_\infty \leq 0,48$.

Důkaz. Náznak důkazu. Nechť $\Delta\mathbf{P}_{ij} = \mathbf{P}_{ij} - \mathbf{P}^0_{ij}$ a $\mathbf{w} = (x,y)$. Předpokládejme dále, že $\frac{\partial x}{\partial u} > |\frac{\partial x}{\partial v}|$ a $\frac{\partial y}{\partial v} > |\frac{\partial y}{\partial u}|$ platí pro každý bod oblasti Ω, kde platí $(-0,48;-0,48) \leq \mathbf{P}_{ij} \leq (0,48;0,48)$.

Potom je Jakobián $J = \frac{\partial x}{\partial u}\frac{\partial y}{\partial v} - \frac{\partial x}{\partial v}\frac{\partial y}{\partial u} > 0$ pro všechny body oblasti Ω včetně hranice oblasti. Z toho podle [2] vyplývá, že funkce $\mathbf{w}$ splňuje vlastnost one-to-one.

Podrobný důkaz je uveden v [13].

Tato věta nám podává dostatečnou, ne však nutnou podmínku pro splnění vlastnosti one-to-one.

2.3.3 Zpětná metoda FFD

V této části budeme řešit složitější úlohu. Transformace nebude dána pouze pozměněnou polohou řídících bodů v mřížce, ale bude zadána vektorem posunutí libovolného bodu. Algoritmus zpětné FFD dokáže následně určit vektory posunutí všech řídících bodů v okolí, abychom docílili požadované transformace. Tento výpočet můžeme zapsat pomocí maticových operací s maticemi B-splajn funkcí (což vede na myšlenku pseudoinverze) nebo pomocí sumační symboliky B-splajn funkcí.

Předpokládejme, že máme dánu mřížku řídicích bodů P_{ij}. Jako deformační funkce použijeme B-splajn, což je výhodné především díky její vlastnosti lokální kontrolovatelnosti.

$$\mathbf{Q}_{i,j,k}(s,t,u) = \sum_{l,m,n=-3}^{0} \mathbf{P}_{i+l,j+m,k+n} B_l(s)B_m(t)B_n(u) \tag{2.81}$$

Určení lokálních souřadnic (s_0,t_0,u_0) bodů

$$\overline{\mathbf{Q}}_{i,j,k}(s_0,t_0,u_0) = \mathbf{w} \tag{2.82}$$

Pokud se však nejprve omezíme na dvoudimenzionální případ (což je v souladu se způsobem zpracování u předchozích metod FFD), tak se předchozí vztahy zjednoduší:

$$\mathbf{Q}_{i,j}(s,t) = \sum_{k,l=-3}^{0} \mathbf{P}_{i+k,j+l} B_k(s) B_l(t) \tag{2.83}$$

Což lze zapsat jako

$$\mathbf{Q} = B \cdot \mathbf{P} \tag{2.84}$$

kde B je řádkový vektor (matice typu 1×16) B-splajn funkcí vytvořený následovně:

$$\begin{aligned} B &= (B_i(u)B_j(v))_{i,j=0}^{3} \\ &= (B_0(u)B_0(v), B_0(u)B_1(v), \ldots, B_0(u)B_3(v), B_1(u)B_0(v), \ldots, B_3(u)B_3(v)) \end{aligned} \tag{2.85}$$

a P je sloupcový vektor, jehož prvky jsou řádkové vektory souřadnic řídicích bodů P_{ij}, tedy tvořící matici typu 16×2:

$$\mathbf{P} = \left(P_{ij}^x, P_{ij}^y\right)_{i,j=0}^{3} = \begin{pmatrix} P_{00}^x & P_{00}^y \\ P_{01}^x & P_{01}^y \\ \vdots & \vdots \\ P_{03}^x & P_{03}^y \\ P_{10}^x & P_{10}^y \\ \vdots & \vdots \\ P_{33}^x & P_{33}^y \end{pmatrix} \tag{2.86}$$

Výsledkem operace je tedy bod Q o souřadnicích $\mathbf{Q} = (Q^x, Q^y)$:

$$\mathbf{Q} = (Q^x, Q^y) = (B_0(u)B_0(v), \ldots, B_3(u)B_3(v)) \begin{pmatrix} P_{00}^x & P_{00}^y \\ \vdots & \vdots \\ P_{33}^x & P_{33}^y \end{pmatrix} \tag{2.87}$$

Pro náš případ bereme $\mathbf{Q}_{puv}$ jako původní souřadnice bodu Q a $\mathbf{Q}_{pos}$ jeho souřadnice po provedeném posunutí. Obdobně symbolem $\mathbf{P}_{puv}$ rozumíme matici (typu 16×2) původních souřadnic řídicích bodů P_{ij} a symbolem $\mathbf{P}_{pos}$ matici (typu 16×2) jejich posunutých souřadnic. Pak z rovnice (2.84) můžeme vyjádřit souřadnice bodu Q_{puv} jako:

$$\mathbf{Q}_{puv} = B \cdot \mathbf{P}_{puv} \tag{2.88}$$

a pro souřadnice posunutých bodů musí platit:

$$\mathbf{Q}_{pos} = B \cdot \mathbf{P}_{pos} \tag{2.89}$$

Výsledné vektory posunutí ΔP_{ij} pro všechny body P_{ij} dostáváme jako:

$$\Delta\mathbf{P} = B^{+} \cdot \Delta\mathbf{Q}, \tag{2.90}$$

kde $\Delta\mathbf{Q} = \mathbf{Q}_{pos} - \mathbf{Q}_{puv}$.

Podrobné odvození včetně použití pseudoinverze je uvedeno v části 3.1.1.

3 Programové zpracování metod FFD

V rámci diplomové práce bylo mým hlavním úkolem nastudovat metody používané pro FFD, naprogramovat tyto metody a otestovat je v programech, které demonstrují možnosti FFD a porovnávají výhody a nevýhody používaných metod.

V kapitole 2 s popisem teoretických základů jsem popsal dvě možné metody výpočtu transformací mřížky:

- pomocí B-splajnů s využitím de Boorova algoritmu
- s využitím Bernsteinových polynomů

Tyto dvě metody jsou naprogramovány a podrobně popsány v kapitolách 3.3.1 a 3.3.1.

Ve třetí části 3.3.3 je popsáno využití pseudoinverze matic k výpočtu kontrolních bodů pro B-splajn plochu.

V celé práci pracujeme s polynomy třetího stupně z důvodu spojitosti druhého řádu a stále ještě jednoduché výpočetní náročnosti.

Programátorská práce na všech těchto metodách byla provedena v jazyku Object Pascal ve vývojovém prostředí Delphi 7. Vytvořené programy byly testovány na notebooku běžícím pod 64-bitovým operačním systémem Windows 7 s procesorem AMD Athlon II P360 Dual-Core 2,30GHz a 4GB RAM.

Výsledky navržených metod jsou prezentovány v následujících částech včetně jejich časové náročnosti.

3.1 Matematický základ naprogramovaných metod

3.1.1 Zpětná FFD – princip metody

V následující části bude podrobně popsána metoda zpětné FFD, kdy jako vstup máme zadán vektor posunutí libovolného bodu. Výstupem této metody je matice vektorů posunutí pro vstupní mřížku řídicích bodů.

Označme $\mathbf{Q}_{puv}$ původní souřadnice bodu Q, $\mathbf{Q}_{pos}$ jeho souřadnice po vychýlení a definujme jeho posunutí jako $\Delta\mathbf{Q} = \mathbf{Q}_{pos} - \mathbf{Q}_{puv}$. Dále označme $\mathbf{P}_{puv}$ matici (typu 16×2) původních souřadnic řídicích bodů P_{ij}, $\mathbf{P}_{pos}$ matici (typu 16×2) jejich posunutých souřadnic a $\Delta\mathbf{P} = \mathbf{P}_{pos} - \mathbf{P}_{puv}$ bude značit matici (typu 16×2) posunutí jednotlivých řídicích bodů.

Pro úplnost připomeňme, že maticí B rozumíme:

$$B = (b_k)_{k=0}^{16} \text{ kde } b_{4i+j+1} = B_i(u)B_j(v) \text{ pro } i, j \in \{0, 1, 2, 3\} \tag{3.1}$$

Poznamenejme, že na pořadí prvků $B_i(u)B_j(v)$ v matici B nezáleží – důležité je, aby měly odpovídající pořadí prvky $\mathbf{P}_{ij}$ v matici $\mathbf{P}$.

Pak z rovnice (2.84) můžeme vyjádřit původní souřadnice $\mathbf{Q}_{puv}$ bodu Q jako:

$$\mathbf{Q}_{puv} = B \cdot \mathbf{P}_{puv} \tag{3.2}$$

a pro souřadnice posunutého bodu musí platit:

$$\mathbf{Q}_{pos} = B \cdot \mathbf{P}_{pos} \tag{3.3}$$

Označili jsme $\Delta\mathbf{P} = \mathbf{P}_{pos} - \mathbf{P}_{puv}$ a předchozí rovnici můžeme přepsat jako:

$$\mathbf{Q}_{pos} = B \cdot (\mathbf{P}_{puv} + \Delta\mathbf{P}) = B \cdot \mathbf{P}_{puv} + B \cdot \Delta\mathbf{P} \tag{3.4}$$

a odečtením (3.2) od rovnice (3.4) získáme:

$$\begin{array}{rcl} \underbrace{\mathbf{Q}_{pos} - \mathbf{Q}_{puv}}_{\Delta\mathbf{Q}} & = & B \cdot \Delta\mathbf{P} \\ \Delta\mathbf{Q} & = & B \cdot \Delta\mathbf{P} \end{array} \tag{3.5}$$

Chceme-li zjistit posuny ΔP_{ij} řídicích bodů P_{ij}, tak musíme člen $\Delta\mathbf{P}$ osamostatnit. Matice B však je obdélníková, tudíž není regulární a tedy nemá inverzi. Pro účel osamostatnění matice $\Delta\mathbf{P}$ by postačovalo, kdybychom nalezli zleva inverzní matici k matici B, neboť bychom násobením zleva touto maticí rovnici rozšířili a na pravé straně bychom docílili osamostatnění matice $\Delta\mathbf{P}$.

Podle věty 8 má matice inverzi zleva právě tehdy, je-li úplné sloupcové hodnosti. Naše matice B je však řádkový vektor, tudíž není úplné sloupcové hodnosti (ale je naopak úplné řádkové hodnosti), a proto nemá inverzi zleva. Obvykle lze podobné problémy řešit následujícím schématem: soustavu zleva vynásobíme $(B^*B)^{-1}B^*$ a upravujeme, jak je naznačeno níže:

$$\begin{array}{rcl} \Delta\mathbf{Q} & = & B \cdot \Delta\mathbf{P} \\ \underbrace{(B^*B)^{-1}B^*}_{B^+}\Delta\mathbf{Q} & = & \underbrace{\underbrace{(B^*B)^{-1}}_{A^{-1}}\underbrace{B^*B}_{A}}_{I}\Delta\mathbf{P} \\ B^+\Delta\mathbf{Q} & = & \Delta\mathbf{P} \end{array} \tag{3.6}$$

Nutnou podmínkou, abychom mohli toto schéma použít, je existence příslušných matic, kterými zleva násobíme soustavu. Aby existovala matice $(B^*B)^{-1}$, musí být matice B^*B invertibilní. To je podle věty 5 právě tehdy, je-li matice B^*B regulární. Podle věty 8 je matice B^*B regulární tehdy a jen tehdy, pokud je matice B úplné sloupcové hodnosti. Jak již bylo uvedeno výše, tento požadavek splněn není. Proto musíme hledat jiný způsob.

Systém rovnic, který řešíme, má totiž dvě rovnice o 32 neznámých, jedná se tedy o poddeterminovaný systém. Z toho vyplývá, že existuje více než jedno řešení. Našim úkolem je nalézt nejlepší řešení ve smyslu nejmenších čtverců minimální v normě. Způsob nalezení řešení je popsán v kapitole 2.2.5. Využitím poznatků popsaných ve zmíněné kapitole dostáváme následující řešení:

$$\Delta\mathbf{P} = B^+\Delta\mathbf{Q} \tag{3.7}$$

Ověříme, že tyto matice mají vhodné rozměry pro násobení: $(16\times1)\cdot(1\times2) = (16\times2)$.

Zbývá tedy spočítat pseudoinverzní matici matice B. V kapitole 2.2.4 jsou uvedeny některé způsoby výpočtu pseudoinverzní matice: pomocí skeletního rozkladu (ten je ve zmíněné kapitole popsán podrobněji), Grevilleovým algoritmem (založen na rekurzi, podrobněji popsán např. v [23]), nebo pomocí singulárního rozkladu (např. v [3]). Různé způsoby výpočtu pseudoinverzní matice lze najít přehledně např. v [9].

V našem případě však počítáme pseudoinverzi velice specifických matic. Proto lze s výhodou využít tvrzení věty 14 (poznamenejme, že tato věta základem rekurentního Grevilleova algoritmu). Matice B je vždy typu 1×16, jedná se tedy o řádkový vektor. Ten má vždy hodnost 1, stejně jako řádkovou hodnost a je tedy úplné řádkové hodnosti. Podle věty 14 pak lze pseudoinverzi matice B spočítat následovně:

$$B^+ = B^* \cdot (B \cdot B^*)^{-1} \tag{3.8}$$

Existence příslušných matic je na rozdíl od schématu (3.6) zaručena (přímo větou 14, nebo také větou 7). Rozeberme si podrobněji člen $B \cdot B^*$. Matice je B je typu 1×16, matice je B^* je typu 16×1, výsledná matice je tedy typu 1×1. Matice B je reálná, lze tedy Hermitovský operátor zjednodušit na transpozici.

Maticové násobení $B \cdot B^T$ je ekvivalentní se skalárním součinem (řádkového) vektoru B sama se sebou. Proto platí:

$$B \cdot B^T = \sum_{k=1}^{16} b_k^2 \tag{3.9}$$

Matice $B \cdot B^T$ je tedy číslo spočtené jako suma kvadrátů členů matice B. Snadno lze pak vidět, že inverzní matice $(B \cdot B^T)^{-1}$ je převrácenou hodnotou této sumy, tedy:

$$(B \cdot B^T)^{-1} = \frac{1}{\sum_{k=1}^{16} b_k^2} \tag{3.10}$$

Pseudoinverzní matici B^+ lze díky tomu, že jsme z násobení matic dostali násobení matice skalárem, spočítat následovně:

$$B^+ = \frac{1}{\sum_{k=1}^{16} b_k^2} B^T \tag{3.11}$$

a hledané řešení je tedy:

$$\Delta\mathbf{P} = \frac{1}{\sum_{k=1}^{16} b_k^2} B^T \Delta\mathbf{Q} \tag{3.12}$$

což lze psát následovně:

$$\Delta\mathbf{P} = \left(\frac{b_i \Delta\mathbf{Q}}{\sum_{k=1}^{16} b_k^2}, \right)_{i=1}^{16} = \left(\frac{b_i \Delta Q^x}{\sum_{k=1}^{16} b_k^2}, \frac{b_i \Delta Q^y}{\sum_{k=1}^{16} b_k^2} \right)_{i=1}^{16} \tag{3.13}$$

Pro naše potřeby je však vhodnější posunutí jednotlivých řídicích bodů zapisovat v původním značení:

$$\Delta\mathbf{P}_{ij} = \left(\frac{B_i(u) B_j(v) \Delta Q^x}{\sum_{k=0}^{3} \sum_{l=0}^{3} (B_k(u) B_l(v))^2}, \frac{B_i(u) B_j(v) \Delta Q^y}{\sum_{k=0}^{3} \sum_{l=0}^{3} (B_k(u) B_l(v))^2} \right) \tag{3.14}$$

pro $i, j = 0, 1, 2, 3$.

3.2 Popis použitého softwaru

V této části popíši v práci používané softwarové vybavení.

3.2.1 Delphi

Praktická část této práce spočívá v naprogramování metod FFD a vytvoření programů sloužících k jejich testování. K tomuto účelu byl použit programovací jazyk Object Pascal. Jako vývojové prostředí bylo zvoleno Borland Delphi 7.

Rozebírat možnosti programování v jazyce Object Pascal není cílem této práce. Základy programování v jazyce Object Pascal lze najít např. v [10].

3.2.2 Gnuplot

Velký význam má ve výsledných programech zapojení programu Gnuplot. Jedná se o program, který zprostředkovává grafický výstup na obrazovku.

Gnuplot je programem pro tvorbu plošných i prostorových grafů zadaných dat. Jedná se o volně dostupný software.

Pro základy práce s Gnuplotem lze v češtině najít např. [11], nebo na stránkách [25], podrobnější popis práce se softwarem nalezneme v [24].

Formát dat

Data, která chceme pomocí Gnuplotu zpracovat je nutno naformátovat do přesně daného tvaru. Sloupce se oddělují tabulátorem. Datový soubor lze pak členit na části vynecháním prázdného řádku. Jako desetinný oddělovač je nutno použít tečku.

Popis použitých příkazů

Nejprve je vždy potřeba uvést Gnuplot do počátečního stavu. To provedeme pomocí příkazu `reset`. Poté musíme nastavit požadovaný výstupní formát. V našem případě chceme vykreslovat graf do okna na obrazovce. To provedeme pomocí příkazu `set terminal wxt size 600,400 persist`, čímž nastavíme vykreslovacímu oknu rozměry 600×400 pixelů. Jako další možnosti lze zmínit výstup do bitmapového formátu PNG nebo vektorového SVG.

Dále provedeme nastavení rámečku kolem vykresleného grafu. To provedeme pomocí příkazu `set nmargin x` kde místo znaku `n` píšeme některý ze znaků množiny $\mathtt{l}, \mathtt{r}, \mathtt{t}, \mathtt{b}$ s následujícím významem:

- `lmargin` nastaví levý okraj
- `rmargin` nastaví pravý okraj
- `tmargin` nastaví horní okraj
- `bmargin` nastaví dolní okraj

Místo znaku `x` se potom píše číslo, udávající požadovanou tloušťku čáry. Tloušťka je udána v jednotkách velikosti znaku použitého fontu terminálu.

Dále lze nastavit velikost grafu a měřítko na osách. To se definuje pomocí příkazu `set size a,b ratio c`, kde `a`, `b` jsou poměry velikosti os grafu vůči velikosti okna (obecně vůči velikosti terminálu). Parametr `c` má význam poměru šířky a výšky rámečku, resp. je-li záporný, pak má význam poměru měřítek os. Jeli nastaven na 1, pak jsme nastavili čtverec, pokud nastavíme -1, pak jsme nastavili shodné měřítko na obou osách.

Posledním použitým formátovacím příkazem je příkaz pro zobrazení mřížky `set grid` pro snadnější odečítání z grafu.

Na závěr provedeme vykreslení našeho datového souboru pomocí příkazu `plot`. Tento příkaz se používá následovně:

```
plot "nazev" with points linetype rgb "red"
```

3.2.3 Matlab

Jeden z možných naprogramovaných výstupů je do formátu pro zpracování Matlabem. Matlab je interaktivní programové prostředí se stejnojmenným skriptovacím jazykem. Vyvíjen je společností MathWorks.

Formát dat

Syntaxe pro jazyk Matlabu je jednoduchá. Matici nazveme jménem a pomocí přiřazovacího příkazu `=` jí přiřadíme matici zapsanou v hranatých závorkách (`[]`), kde řádky jsou odděleny entrem a buňky jsou odděleny prázdným znakem (např. mezerou). Jako desetinný oddělovač je nutno použít tečku.

3.2.4 LaTeX

Jako jeden z možných výstupů vytvořených programů je vytvoření matice (tedy dvourozměrné tabulky) bodů, resp. posunutím, ve formátu vyhovujícímu jazyku LaTeX. TeX je program pro počítačovou sazbu vytvořený profesorem Donaldem Ervinem Knuthem. LaTeX slouží k usnadnění sazby oproti původnímu TeXu. LaTeX je pojmenován po svém tvůrci Lesliem Lamportovi. Soubory obsahující kód v TeXu mají příponu `.tex`.

Popis použitých příkazů

Samotná tabulka je tvořena řádky, jenž jsou od sebe odděleny znakem `\\`. V řádku jsou jednotlivé buňky odděleny znakem `&`.

Takto vytvořenou tabulku je potřeba uvodit pomocí příkazu `\begin{array}{...}` a uzavřít pomocí `\end{array}`. Místo ... je potřeba napsat n-tici znaků z množiny $\{\texttt{l}, \texttt{c}, \texttt{r}\}$, kde n je počet sloupců tabulky. Jednotlivé symboly z této množiny definují různé zarovnání daného sloupce:

- `l` – zarovná sloupec do leva
- `c` – zarovná sloupec na prostředek
- `r` – zarovná sloupec do prava

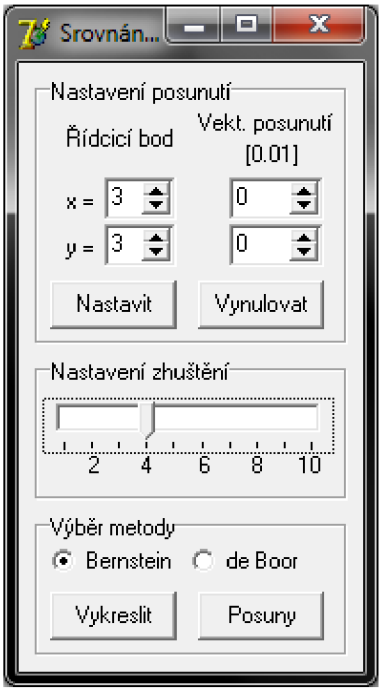
Celý tento kód lze použít pouze v matematickém módu. Proto musí být v prostředí `equation` (uzavřen mezi příkazy `\begin{equation*}` a `\end{equation*}`).

3.3 Popis jednotlivých programů

Zde popíši ovládání jednotlivých programů. Připomeňme, že pro jejich správnou funkci je nutné mít nainstalovaný Gnuplot.

3.3.1 Program FFD1 – porovnání transformačních mřížek

V rámci diplomové práce jsem vytvořil program, který porovnává výpočet FFD pomocí de Boorova algoritmu a pomocí Bernsteinových polynomiálních bázových funkcí. Navržený algoritmus vychází z původní práce [19] profesora Sederberga, který danou metodu navrhl a poprvé v roce 1986 publikoval. Vstupem je mřížka 4×4 řídicích bodů. Velikost této mřížky byla zvolena kvůli Bernsteinovým polynomům, které v případě volby třetího stupně pracují právě na mřížce velikosti 4×4 řídicích bodů (dle vlastnosti v části 2.1.1).



Obrázek 3.1: Ukázka programu FFD1

Toto porovnání provedeme pomocí programu FFD1. Do programu je nutno nastavit souřadnice řídicích bodů. Pomocí příslušné dvojice prvků SpinEdit se zvolí řídicí bod, který chceme posouvat a v další dvojici prvků SpinEdit se nastaví požadovaný vektor posunutí řídicího bodu. Následně je nutno takto nastavené hodnoty zapsat do programu stiskem tlačítka „Nastavit“. Tento postup můžeme opakovat tak, abychom postupně nastavili všechny řídicí body na námi požadované pozice.

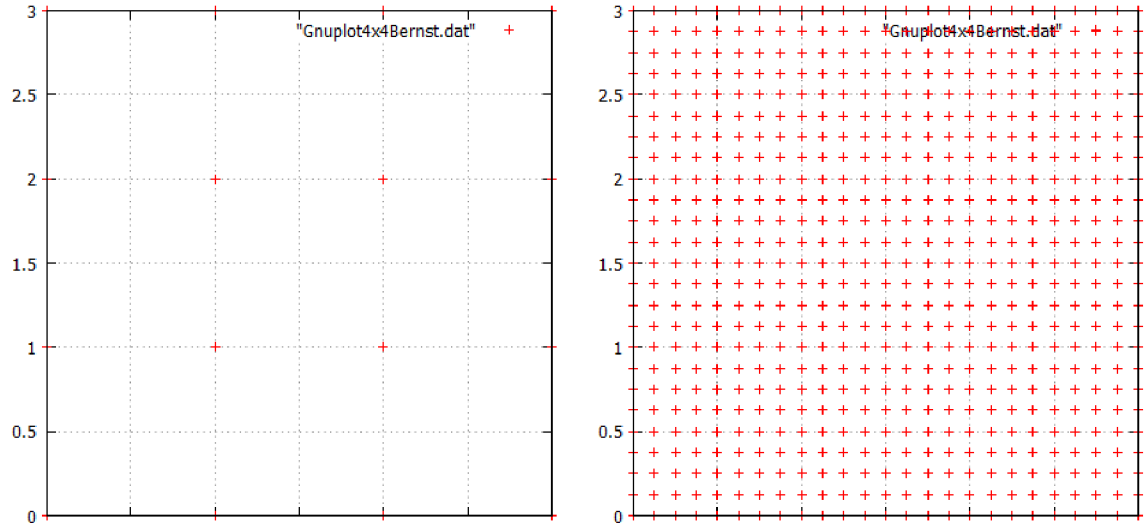
Složky vektoru posunutí jsou reálná čísla. Pro zpracování na počítači lze použít libovolného počtu desetinných míst v závislosti na typu proměnné. Vykreslení však probíhá na obrazovku, hodnoty se tedy zaokrouhlí na pixely. Proto je postačující nastavovat složky vektoru posunutí pouze s omezeným množstvím desetiných míst. Pro optimální zobrazení bylo použito nastavení vektoru posunutí s přesností na setiny.

Dále je možné zvolit nastavení hustoty vykreslení kontrolní mřížky, a to v celočíselném intervalu od 1 do 10. Hustotou je zde míněno zhuštění sítě vykreslovaných bodů vůči

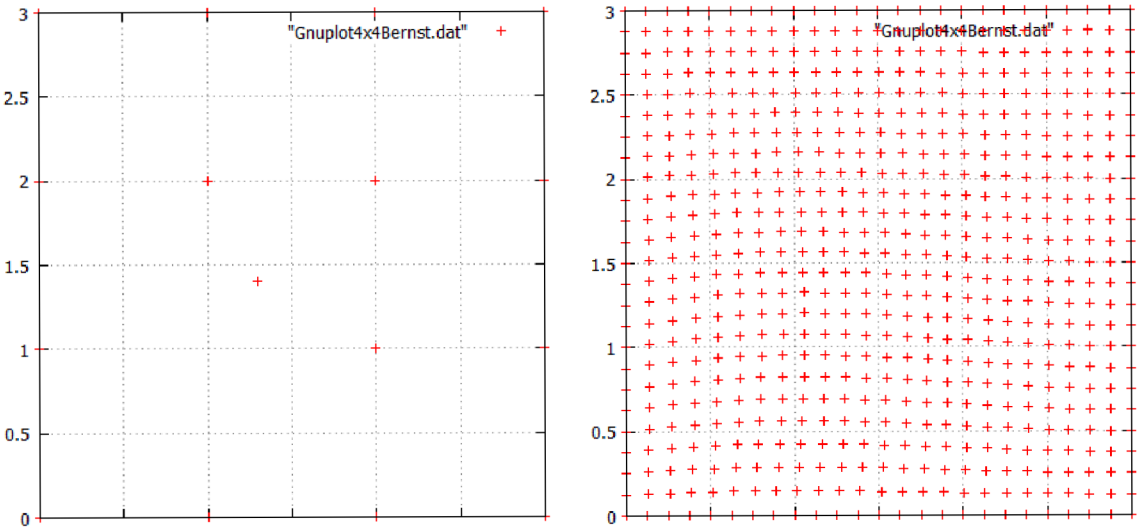
mřížce řídicích bodů. Například, máme-li zadáno zhustění 1, pak je síť vykreslovaných bodů stejně hustá jako mřížka řídicích bodů, tedy vykreslí se síť 4×4 bodů plochy. Máme-li zadáno zhuštění 5, pak to znamená, že díl mezi dvojicí řídicích bodů bude rozdělen na 5 dílků ohraničených vykreslovanými body křivky. Rozestup mezi vykreslovanými body bude $\frac{1}{5} = 0,2$. Výsledkem je pak síť 16×16 bodů plochy. Obecně, pokud máme mřížku 4×4 (ta je v tomto programu pevně dána) a zvolíme zhuštění h, pak dostaneme síť $n \times n$ bodů plochy, kde $n = 3 \cdot h + 1$.

Následně pomocí prvku RadioButton můžeme vybrat metodu zpracování transformace, a to de Boorovým algoritmem nebo Bernsteinovými polynomy. Volbou „de Boor“ nastavíme, že výpočet transformované plochy, resp. výpočet posunutí jednotlivých vykreslovaných bodů, proběhne pomocí výpočtu B-splajn ploch algoritmem de Boor. Volbou „Bernstein“ nastavíme, že výpočet proběhne pomocí předpočítaných bázových polynomiálních funkcí.

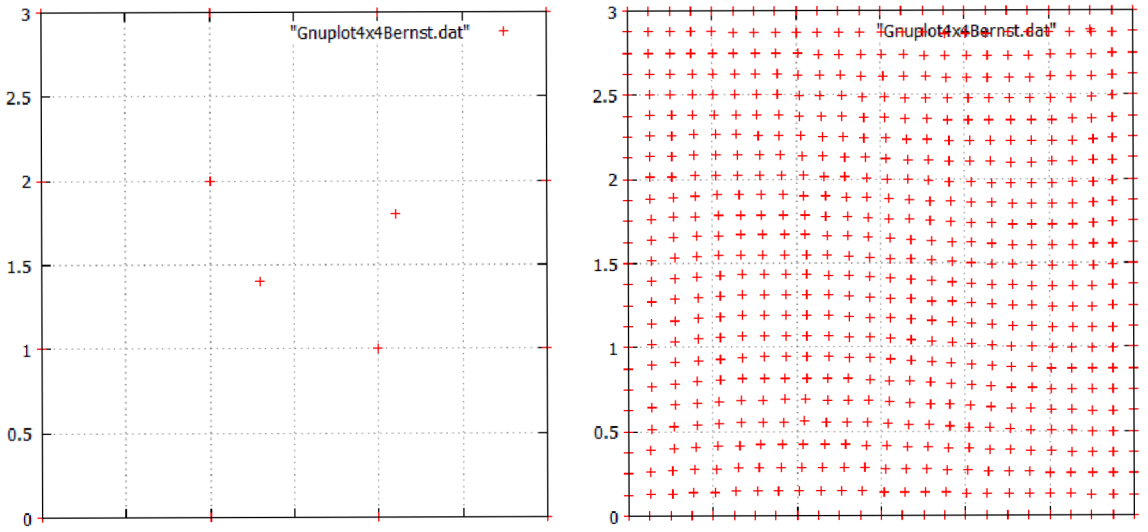
Tlačítko (prvek Button) „Vykreslit“ zobrazí výslednou transformovanou mřížku. Na obr. 3.2 vlevo je původní mřížka řídicích bodů velikosti 4×4. Tato mřížka definuje původní netransformovanou plochu, jak je možno vidět na stejném obrázku vpravo. Poté byly nastaveny následující transformace řídicích bodů: bod $[1; 1]$ posuneme vektorem posunutí $[0,3; 0,4]$ do pozice $[1,3; 1,4]$ (viz obrázek 3.3 vlevo) a poté přidáme další transformaci. Bod $[2; 2]$ pomocí vektoru posunutí $[0,1; -0,2]$ posuneme na pozici $[2,1; 1,8]$ (viz obrázek 3.4 vlevo). Tyto zmíněné transformace po řadě vytváří transformované plochy, které lze vidět na obrázcích 3.3 a 3.4 vpravo.



Obrázek 3.2: Vlevo: základní mřížka 4×4 řídicích bodů. Vpravo: plocha odpovídající základní mřížce



Obrázek 3.3: Vlevo: Mřížka řídicích bodů transformovaná vektorem posunutí $[0,3;0,4]$ v bodě $[1;1]$. Vpravo: Plocha odpovídající mřížce řídicích bodů vlevo



Obrázek 3.4: Vlevo: Mřížka řídicích bodů transformovaná vektorem posunutí $[0,1;-0,2]$ v bodě $[2;2]$ a vektorem $[0,3;0,4]$ v bodě $[1;1]$. Vpravo: Plocha odpovídající mřížce řídicích bodů vlevo

Button „Posuny“ vytvoří kód použitelný v LaTeXu, který popisuje posunutí jednotlivých bodů. Ukázkové posunutí je v tabulkách v kapitole 3.5.

3.3.2 Program FFD2 – de Boorův algoritmus na obecné velikosti mřížky

Jak už bylo uvedeno v předchozí kapitole, metoda FFD pomocí de Boorova algoritmu není na rozdíl od metody využívající Bernsteinovy polynomy omezena velikostí mřížky 4×4 řídicích bodů, ale lze obecně použít pro libovolnou velikost mřížky. V následující části popíšeme program, který je schopen vypočítat FFD transformaci pomocí de Boorova algoritmu pro libovolnou velikost mřížky. Ukázka okna programu je na obrázku 3.5.

Obrázek 3.5: Ukázka programu FFD2

Na rozdíl od předchozího programu, kde byl počet řídicích bodů v mřížce pevně dán, zde je v prvním kroku nutno zadat požadovanou velikost mřížky řídicích bodů, tedy počet řádků a sloupců mřížky. Velikost mřížky je vzhledem k paměťové náročnosti omezena na maximálně 25×25 řídicích bodů. Minimální velikost je nastavena na 4×4 řídicí body. Důvodem je, že z již uvedených důvodů používáme B-splajn plochy stupně 3, které pro svoji definici vyžadují právě 4×4 řídicí body. Nastavené hodnoty je nutno zapsat do programu stisknutím tlačítka „Nová mřížka“, čímž se vytvoří základní, nedeformovaná, mřížka řídicích bodů.

V dalším kroku se v souladu s předchozím programem nastavuje posunutí jednotlivých řídicích bodů v mřížce. Každé provedené posunutí řídicího bodu je nutno do programu zapsat stiskem tlačítka „Posunout“. Rozdíl oproti předchozímu programu je v tom, že

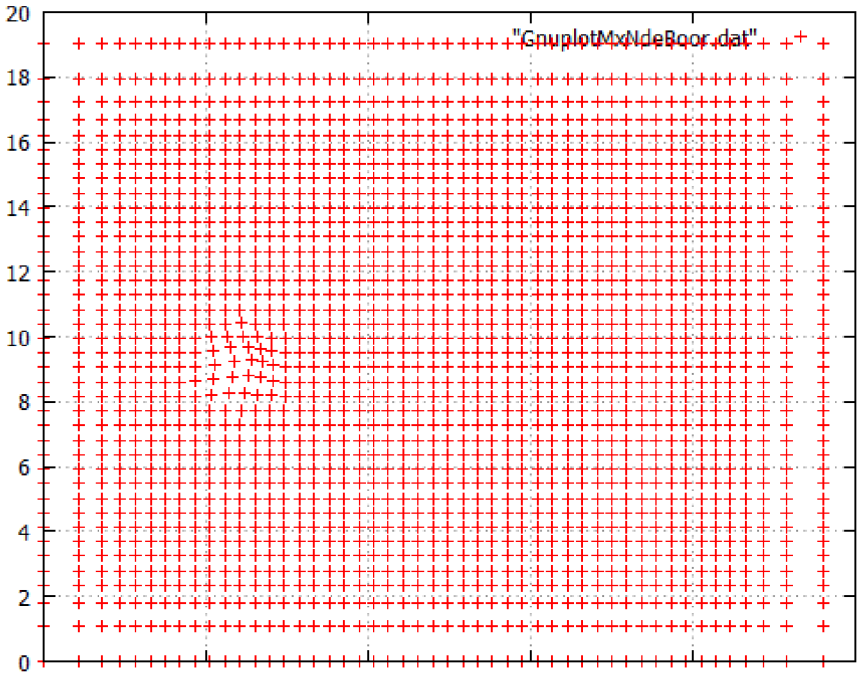
nyní nejsme omezeni čtvercovou mřížku 4×4 řídicích bodů, ale můžeme nastavit pozice všech bodů, které jsou v mřížce obsaženy. V případě, že chceme zadat zcela novou mřížku řídicích bodů, lze to provést stiskem již popsaného tlačítka „Nová mřížka“.

Posledním prvkem, který lze nastavit, je posuvník, kterým se určuje zhuštění výsledné sítě vykreslených bodů. Tento posuvník lze nastavit na celočíselné hodnoty mezi 1 a 4. Důvodem takto omezeného rozsahu hodnot je možnost nastavení až 25×25 řídicích bodů, což pro čtyřnásobné zhuštění znamená 97×97 vykreslených bodů křivky. Tím se dostáváme na hranici toho, co lze do deklarované proměnné typu `Pole2DBodu2D` uložit. Obecně pro zvolenou mřížku $p_1 \times p_2$ řídicích bodů a pro zvolené zhustění vykreslovaných bodů h dostáváme síť $n_1 \times n_2$ bodů plochy, kde $n_i = (p_i - 1) \cdot h + 1$ pro $i = 1, 2$.

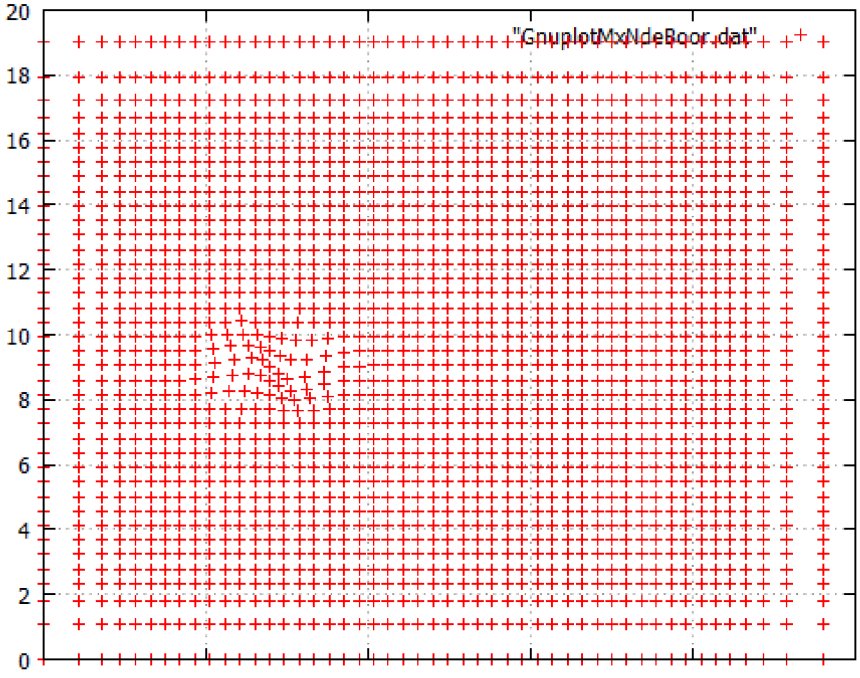
Obrázek 3.6: Plocha odpovídající nedeformované mřížce

Tlačítko „Vykreslit“ vypočítá deformace B-splajn plochy pomocí algoritmu de Boor a následně danou plochu vykreslí.

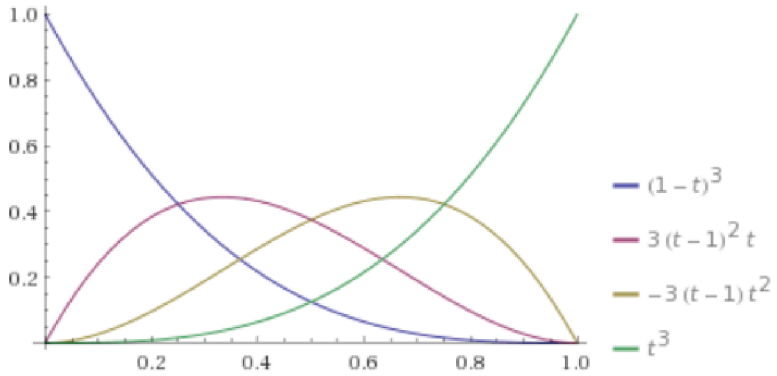
Na tomto místě je vhodné poznamenat, že mřížka vykreslovaných bodů plochy nemá konstatní hustotu na celé vykreslované oblasti (viz obrázky 3.6, 3.7, 3.8). Tento jev je způsoben nerovnoměrným přírůstkem plochy podgrafu bázových funkcí - viz obrázek 3.9.



Obrázek 3.7: Plocha odpovídající mřížce transformované vektorem $[0,8;0,6]$ v bodě $[6;8]$



Obrázek 3.8: Plocha odpovídající mřížce transformované vektorem $[-0,8;-0,9]$ v bodě $[8;9]$ a vektorem $[0,8;0,6]$ v bodě $[6;8]$



Obrázek 3.9: Vysvětlení nerovnoměrnosti vykreslovaných bodů na okraji mřížky (vytvořeno pomocí programu Wolfram Alpha)

3.3.3 Program FFD3 – metoda přímého ovládání FFD

V kapitole popisující teoretické základy free-form deformace byly popsány obtíže, které s sebou přináší snaha nastavit řídicí body mřížky tak, aby výsledek splňoval vytyčené požadavky. Tuto problematiku řeší metody nazývané jako metody ovládání/manipulace FFD. Součástí této práce je i navržení programu, který tento problém řeší. Ukázka výsledného programu je na obrázku 3.10, popis jeho funkce je v následující části.

Obrázek 3.10: Ukázka programu FFD3

Program se skládá ze dvou do jisté míry samostatných částí: první má za cíl navrhnout vhodné rozvržení řídicích bodů tak, aby požadovaný bod ležel na ploše, druhá část programu slouží pouze k vizualizaci vypočítané plochy. Tento program pracuje s mřížkou 4×4 řídicích bodů. Důvody jsou shodné jako u programu FFD1 popsaném v kapitole 3.3.1, tedy že používané kubické splajny vyžadují ke své definici čtveřici řídicích bodů. Tímto způsobem se postupuje i ve veškeré dostupné literatuře věnující se právě metodám ovládání FFD (viz např. [13] nebo [5]).

V prvním kroku je nutno nastavit, který konkrétní bod plochy chceme posunout. To se provede nastavením souřadnic požadovaného bodu ve čtveřici políček `SpinEdit`. První pár slouží k nastavení souřadnice ve směru x: v prvním poli nastavíme celou část, v druhém pak desetinou (v našem případě setiny). Druhý pár slouží k obdobnému nastavení y-ové souřadnice požadovaného bodu.

Následně je potřeba nastavit vektor posunutí námi (v předchozím kroku) zvoleného bodu. Prvky vektoru posunutí se nastavují v setinách. Mohou nabývat hodnot v intervalu $\langle -0,48; 0,48 \rangle$. Důvodem je zachování vlastnosti one-to-one, popsané v kapitole 2.3.2 a podrobně vysvětlené v [13].

Posledním krokem v první části je stisk tlačítka „Zpětná FFD“. Tímto se nastavení zvolené pomocí polí `SpinEdit` zapíše do programu a provede se algoritmus ovládání FFD popsaný v kapitole 3.1.1. Výsledkem je, že program vypočítá podle principu nejmenších čtverců nejlépe pozice řídicích bodů tak, aby definovaná plocha procházela zvoleným bodem. Úspěšné ukončení je signalizováno tlačítkem „Vykreslit“. To má původně šedý popisek, tedy nepoužitelné. Po úspěšném provedení první části programu potom jeho popisek ztmavne, což značí, že jej již lze použít.

Druhá část tohoto programu je z pohledu algoritmů méně zajímavá. Je však nezbytná pro ukázku toho, že algoritmy použité v předchozí části programu jsou funkční a že podávají smysluplné výsledky.

Nejprve je třeba zvolit zhuštění sítě vykreslovaných bodů křivky vůči mřížce řídicích bodů. Tento krok je totožný s krokem v programu FFD1. Stiskem tlačítka „Vykreslit“ se provede vykreslení bodů plochy. Vykreslení probíhá pomocí metody s předpočítanými Bernsteinovými polynomy, neboť je časově méně náročná než pomocí de Boorova algoritmu.

3.4 Popis naprogramovaných knihoven

Během programování vlastních programů demonstrujících a porovnávajících možnosti zmíněných metod FFD byl naprogramovaný kód z důvodu přehlednosti uspořádán do několika knihoven. Tyto jednotky se nachází ve stejném adresáři jako vlastní programy, uloženy v zdrojových souborech s názvem shodným s názvem jednotky a příponou .pas.

3.4.1 Pomocné knihovny

Nejprve představíme knihovny, které tvoří základ pro další práci. Zpravidla obsahují jednoduché konstrukce, bez nichž by však pokročilejší metody nefungovaly.

Knihovna UnitTypeDef

Tato jednotka obsahuje deklarace typů proměnných, které jsou společné pro všechny další jednotky využívané ve vytvořených programech. Jedná se o typy pro popis bodu v jedno-dvou- a trojrozměrném prostoru a pro jedno- a dvojrozměrná pole těchto bodů. Dále je zde uvedena metoda pro vytvoření pravidelné mřížky bodů. Součástí této knihovny jsou též metody, které slouží pro převod matice řídicích bodů a matice bázových polynomů do formátu, který vyhovuje použití v metodě zpětné FFD, pokud bychom ji chtěli aplikovat pomocí výpočtu pseudoinverze spuštěného pomocí MatLabu.

Jako první je zde zadefinována konstanta `MaxDim`, následují definice vytvořených typů. Prvním typem je typ `vektor` sloužící především pro ukládání uzlového vektoru.

Bod je zde definován jako záznam (`record`) skládající se ze složek typu `double` (tedy reálných čísel) vyjadřujících souřadnice bodu. Počet složek odpovídá dimenzi prostoru.

Pole bodů je zadefinováno jako záznam skládající se z počtu bodů (typu `1..MaxDim` - tj. celé číslo z intervalu $\langle 0; \texttt{MaxDim} \rangle$, resp. u dvourozměrného pole se jedná o počet bodů v řádku a počet bodů ve sloupci) a z pole (`array`) (jedno nebo dvourozměrného) složeného z příslušného typu bodů.

Funkce `GenerPole` vyžaduje dva parametry. Těmi je počet řádků a počet sloupců požadovaného pole. Výsledkem je pravidelné pole (například řídicích) bodů.

Knihovna UnitUtility

Tato jednotka obsahuje metody počítající hodnotu Bernsteinova polynomu (resp. bázové funkce pro zadaný uzlový vektor) zadaného stupně a pro zadaný parametr t. Dále jsou zde definovány metody pro tvorbu pole Bernsteinových polynomů (resp. bázových funkcí) pro zadané rozměry pole a parametry u, v.

Funkce `BernPoly` vyžaduje zadání pořadí a stupeň Bernsteinova polynomu a parametr t, pro nějž má být hodnota polynomu vyčíslena. Výstupem je hodnota Bernsteinova polynomu pro zadané parametry.

Obdobný vstup je i pro funkci `BazFca`, kde však musíme mezi vstupní parametry připsat i uzlový vektor, pro nějž má být B-splajn funkce sestavena a následně vyčíslena.

Funkce `GenerPoleBernPoly` a `GenerPoleBazFci` mají shodné ovládání. Zadat musíme počet řádků a počet sloupců požadovaného pole a řádkový a sloupcový parametr. Výsledkem těchto funkcí je matice v jejíž prvky jsou pronásobené Bernsteinovy polynomy, resp. B-splajn funkce pro daný řádek a sloupec.

Knihovna UnitUtilityKombin

Tato jednotka obsahuje funkce z oboru kombinatoriky, které se užívají při výpočtech B-splajn funkcí. Svým zaměřením je tato knihovna použitelná i mimo problematiku FFD a počítačové grafiky.

Knihovna UnitUtilityUzlVekt

Tato jednotka řeší problematiku uzlového vektoru. V prvé řadě se jedná o funkci generující uzlový vektor pro potřeby dalšího zpracování v metodách počítajících B-splajny. Dále obsahuje několik procedur sloužících k výpisu uzlového vektoru na obrazovku, čehož lze využít především u ladění.

Funkce `uzlovyVek` vyžaduje zadání počtu řídicích bodů, pro které má vytvořit uzlový vektor a požadovaný stupeň křivky. Výsledkem je uzlový vektor, jehož hodnoty se nachází v intervalu $\langle 0; 1\rangle$ tak, že prvních $n + 1$ souřadnic (n je stupeň B-splajn křivky, pro niž uzlový vektor tvoříme) je 0, posledních $n + 1$ souřadnic je 1 a zbylé souřadnice jsou ekvidistantně rozděleny na zmíněném intervalu. Tento tvar uzlového vektoru zajistí, že odpovídající křivka bude začínat v prvním řídicím bodě a končit bude v posledním.

3.4.2 Knihovna UnitBernstein

Knihovna UnitBernstein.pas patří mezi stěžejní části kódu. Nachází se zde jediná metoda zajišťující výpočet bodu na Béziérově ploše stupně 3 pro zadanou mřížku řídicích bodů a parametry u, v.

3.4.3 Knihovna UnitDeBoor

Knihovna UnitDeBoor se řadí k dalším stěžejním částem naprogramovaného kódu. První funkce zajišťuje výpočet bodu na B-splajn křivce pomocí de Boorova algoritmu. Druhá pak opakovaně používá předchozí funkci k výpočtu bodu na B-splajn ploše. Obě tyto metody jsou popsány v kapitole 2.1.2.

3.4.4 Knihovna UnitFileProc

Jedná se o nejrozsáhlejší z představovaných knihoven. Obsahuje metody pro práci se soubory. Jako první je uvedena dvojice procedur sloužících k výpisu pole bodů do prostého textového souboru. Podobný význam mají i další procedury. Mezi nimi lze jmenovat procedury pro výstup do Matlabu, do LaTeXu a několik, které slouží k výstupu pro Gnuplot. Aby Gnuplot či Matlab data správně interpretoval je nutno použít určitý, daným programem vyžadovaný, styl formátování dat, se kterým nás seznámila kapitola 3.2. Na závěr nutno zmínit, že v této knihovně se nachází i procedury, které slouží k vypsání příkazů pro Gnuplot do dávkového souboru a pro spuštění tohoto dávkového souboru.

Princip všech procedur v této knihovně je obdobný. Vysvětlíme si ho na proceduře `VypisSouboru`. Nejprve je potřeba pomocí `AssignFile` propojit soubor na disku počítače se souborem jako typem proměnné v Object Pascalu. Prvním parametrem je proměnná typu textový soubor (`TextFile`), druhým potom název souboru na disku, do kterého chceme data ukládat. Následně zavoláme příkaz `Rewrite` s parametrem proměnné typu

`TextFile`. Tento příkaz zajistí, že můžeme do zmíněného souboru zapisovat s tím, že pokud již soubor existuje, pak jej přepíšeme. Samotný zápis se provádí pomocí příkazů `Write` a `Writeln`. Parametry mají oba příkazy totožné: prvním je soubor, do nějž chceme zapisovat a druhým je text (posloupnost znaků - `string`), který chceme zapsat. Použijeme-li příkaz `Write`, pak se pouze zapíše zmíněný text, použijeme-li `Writeln`, pak se zapíše text a kurzor nastaví na nový řádek. Pokud příkaz `Writeln` napíšeme bez druhého parametru, pak dojde pouze k napsání nového řádku. Na závěr práce je potřeba otevřený soubor zavřít příkazem `CloseFile` s parametrem názvu proměnné typu textového souboru.

```
procedure VypisSouboru(nazev : String; Pole : Pole1DBodu2D);
var i : integer;
    soubor : TextFile;
begin
  // otevření souboru
  AssignFile(soubor, nazev);
  Rewrite(soubor);
  // zápis do souboru
  for i:=1 to Pole.pocet do
    begin
      Write(soubor,FloatToStr(Pole.pole[i].x));
      Write(soubor,tab);
      Write(soubor,FloatToStr(Pole.pole[i].y));
      Writeln(soubor);
    end;
  // zavření souboru
  CloseFile(soubor);
end;
```

Ostatní procedury v této knihovně se odlišují pouze strukturou, kterou vyžadují programy, které s výslednými soubory dále pracují. Vysvětlení významu jednotlivých příkazů používaných v procedurách této knihovny je uvedeno v kapitole 3.2.

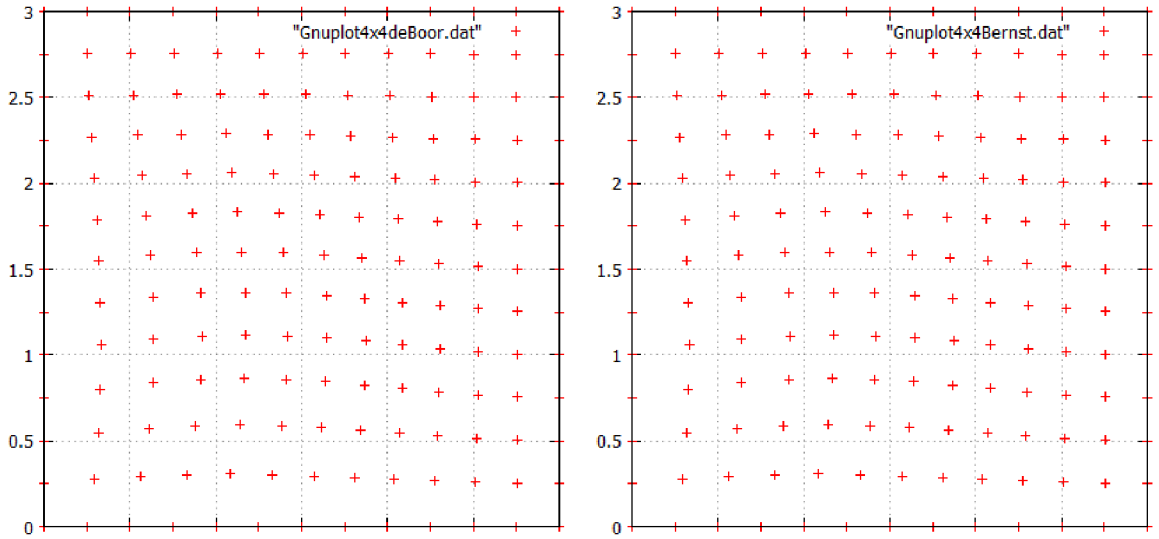
3.5 Výsledky

Jedním z cílů této diplomové práce bylo použité metody FFD porovnat z hlediska ekvivalence jejich výstupů a z hlediska časové náročnosti.

3.5.1 Grafické porovnání de Boorova algoritmu s metodou používající Bernsteinovy polynomy

Nejprve porovnáme, zda de Boorův algoritmus a metoda počítající pomocí Bernsteinových polynomů dávají pro shodná zadání i totožné výsledky. K tomuto účelu použijeme program FFD1.

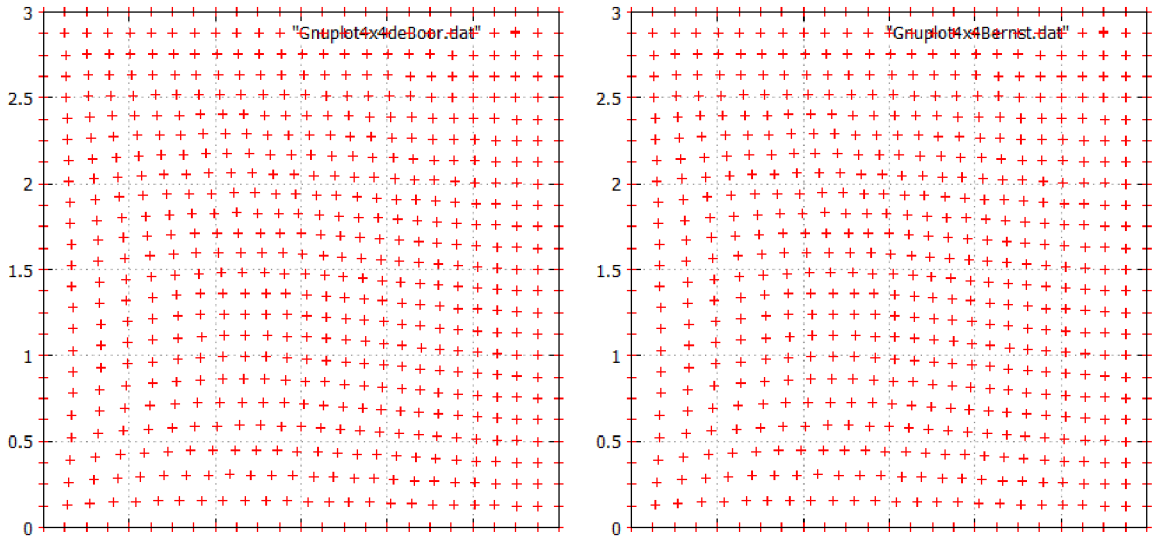
Nastavme například bod $[1; 1]$ do pozice $[1,9; 1,6]$, tedy vektor posunutí bude $(0,9; 0,6)$. Nyní pro takto deformovanou mřížku řídicích bodů vykresleme pomocí programu FFD1 body odpovídající plochy pro čtyřnásobné zhuštění. Plocha vykreslená de Boorovým algoritmem je na obr. 3.11 vlevo, plocha vykreslená pomocí Bernsteinových polynomů je na obr. 3.11 vpravo.



Obrázek 3.11: Srovnání ploch vykreslených pro mřížku řídicích bodů transformovanou posunem bodu $[1; 1]$ vektorem posunutí $(0,9; 0,6)$. Zhuštění vykreslených bodů 4-násobné. Vlevo: plocha vykreslená pomocí de Boorova algoritmu. Vpravo: plocha vykreslená metodou FFD používající Bernsteinovy polynomy

Totéž srovnání provedeme pro lepší průkaznost i s hustotou vykreslovaných bodů plochy navýšenou na osminásobek vůči mřížce řídicích bodů. Příslušné plochy lze nalézt na obr. 3.12.

Protože vizuální kontrola nemusí být průkazná, provedeme ještě kontrolu numerickou. Pomocí programu FFD1 si můžeme vygenerovat tabulky s napočítanými vektory posunutí pro obě porovnávané metody. Kvůli přehlednosti a rozsahu tabulky takovému, aby vešla na stránku bylo k tomuto účelu vybráno dvojnásobné zhuštění vykreslovaných bodů. Hodnoty vektorů posunutí v počítaných bodech jsou uvedeny v tabulce 3.1 (pro de Boorův algoritmus) a 3.2 (pro metodu s Bernsteinovými polynomy).



Obrázek 3.12: Srovnání ploch vykreslených pro mřížku řídicích bodů transformovanou posunem bodu $[1; 1]$ vektorem posunutí $(0,9; 0,6)$. Zhuštění vykreslených bodů 8-násobné. Vlevo: plocha vykreslená pomocí de Boorova algoritmu. Vpravo: plocha vykreslená metodou FFD používající Bernsteinovy polynomy

Tabulka 3.1: Vektory posunutí pro body plochy vykreslené pomocí de Boorova algoritmu. Mřížka řídicích bodů transformována posunem bodu $[1; 1]$ vektorem posunutí $(0,9; 0,6)$

0;0	0;0	0;0	0;0	0;0	0;0	0;0
0;0	0,11;0,07	0,14;0,09	0,12;0,08	0,07;0,05	0,02;0,01	0;0
0;0	0,14;0,09	0,18;0,12	0,15;0,1	0,09;0,06	0,03;0,02	0;0
0;0	0,12;0,08	0,15;0,1	0,13;0,08	0,07;0,05	0,02;0,02	0;0
0;0	0,07;0,05	0,09;0,06	0,07;0,05	0,04;0,03	0,01;0,01	0;0
0;0	0,02;0,01	0,03;0,02	0,02;0,02	0,01;0,01	0;0	0;0
0;0	0;0	0;0	0;0	0;0	0;0	0;0

Tabulka 3.2: Vektory posunutí pro body plochy vykreslené metodou Bernsteinových polynomů. Mřížka řídicích bodů transformována posunem bodu $[1; 1]$ vektorem posunutí $(0,9; 0,6)$

0;0	0;0	0;0	0;0	0;0	0;0	0;0
0;0	0,11;0,07	0,14;0,09	0,12;0,08	0,07;0,05	0,02;0,01	0;0
0;0	0,14;0,09	0,18;0,12	0,15;0,1	0,09;0,06	0,03;0,02	0;0
0;0	0,12;0,08	0,15;0,1	0,13;0,08	0,07;0,05	0,02;0,02	0;0
0;0	0,07;0,05	0,09;0,06	0,08;0,05	0,04;0,03	0,01;0,01	0;0
0;0	0,02;0,01	0,03;0,02	0,02;0,02	0,01;0,01	0;0	0;0
0;0	0;0	0;0	0;0	0;0	0;0	0;0

Z uvedených tabulek i grafů je zřejmé, že náš předpoklad je splněn. Tedy obě dvě porovnávané metody FFD pro shodný vstup podávají shodné výsledky.

3.5.2 Časové porovnání de Boorova algoritmu s metodou používající Bernsteinovy polynomy

Jedním z cílů bylo porovnat zvolené metody FFD z hlediska časové náročnosti jejich výpočtu. K porovnání byl zvolen případ mřížky 4×4 řídicích bodů. Důvody jsou popsány v kapitole 3.3.1. Jako základ pro toto porovnání byl použit program FFD1, který obsahuje měření pomocí funkce `GetTickCount`.

Nejprve byly řídicí body ponechány na původní pozici, tedy byla použita mřížka bez transformací. Tato mřížka byla použita pro výpočet splajn plochy zadané těmito řídicími body. Zhuštění vykreslovaných bodů vůči mřížce řídicích bodů bylo nastaveno na v programu přípustné celočíselné mocniny dvou, tedy postupně na hodnoty 1, 2, 4 a 8. Pro každou hodnotu zhuštění bylo provedeno 10 měření. Tímto způsobem byly naměřeny délky časových intervalů pro obě porovnávané metody: de Boorův algoritmus i metodu FFD užívající Bernsteinovy polynomy. Naměřené hodnoty jsou zapsány v tabulce 3.3.

Tabulka 3.3: Experimentálně naměřená délka průběhu FFD pomocí de Boorova algoritmu (nahoře) a pomocí metody FFD využívající Bernsteinovy polynomy (dole), pro základní mřížku [ms]

zhuštění	Algoritmus de Boor									
1	16	16	15	0	0	16	0	16	0	0
2	31	31	31	31	31	31	16	31	32	32
4	109	94	93	93	93	94	94	109	94	93
8	359	359	358	359	359	359	375	390	374	374

zhuštění	Metoda s Bernsteinovými polynomy									
1	0	0	15	0	0	0	0	0	0	0
2	0	16	0	0	0	0	0	0	0	0
4	16	16	15	16	31	16	15	16	16	16
8	63	63	63	78	63	78	63	63	62	47

V další části porovnání byla nastavena následující transformace mřížky řídicích bodů: řídicí bod $[1; 1]$ byl posunut vektorem posunutí $(0,9; 0,6)$. Pro takto transformovanou mřížku byla opět provedena měření výpočtu plochy se zhuštěním výsledné sítě vykreslovaných bodů po řadě 1, 2, 4 a 8. Pro každé zhuštění bylo provedeno 10 měření. Tímto způsobem byly naměřeny délky časových intervalů pro de Boorův algoritmus i metodu FFD užívající Bernsteinovy polynomy. Naměřené hodnoty jsou zapsány v tabulce 3.4.

Z tabulek 3.3 a 3.4 je zřejmé, že to, zda počítáme body plochy pro nedeformovanou mřížku řídicích bodů nebo pro mřížku transformovanou, nemá výrazný vliv na časovou náročnost výpočtu. Jako významná se naopak jeví zvolená metoda výpočtu, zvláště s narůstajícím zhuštěním počítaných bodů plochy. Pro jednonásobné zhuštění není rozdíl příliš patrný, pro zhuštění dvojnásobné je již zřejmé, že výpočet pomocí de Boorova algoritmu je přibližně dvakrát pomalejší oproti metodě používající předpočítané Bernsteinovy polynomy. Pro zhuštění nastavené na hodnoty 4 nebo 8 už lze pozorovat, že výpočet pomocí de Boorova algoritmu je přibližně šestkrát pomalejší oproti Bernsteinovým polynomům.

Hodnoty, které jsou zapsány v tabulkách 3.3 a 3.4, vykazují výrazné skokové změny – např.: v prvním datovém řádku tabulek je buď 0 nebo 15 (resp. 16), zcela však chybí

Tabulka 3.4: Experimentálně naměřená délka průběhu FFD pomocí de Boorova algoritmu (nahoře) a pomocí metody FFD využívající Bernsteinovy polynomy (dole), pro transformovanou mřížku [ms]

zhuštění	Algoritmus de Boor									
1	15	0	16	0	16	16	0	16	15	0
2	31	32	31	32	31	32	31	31	32	31
4	109	94	109	93	93	93	94	93	93	109
8	281	359	359	359	358	359	375	281	359	359

zhuštění	Metoda s Bernsteinovými polynomy									
1	0	0	0	16	0	0	0	16	0	0
2	0	0	16	0	0	15	16	15	0	15
4	16	16	16	15	16	15	15	15	15	16
8	63	47	63	63	63	62	47	47	62	46

jakékoliv hodnoty mezi nimi. Vysvětlením je k měření použitá funkce `GetTickCount`. Tato funkce vrací počet milisekund uplynulých od spuštění systému. Bohužel je však omezena vlastností systémového časovače. Přitom jedno „tiknutí“ proběhne přibližně jednou za 16 milisekund. Pokud tedy začátek i konec měření času leží v tomto intervalu, pak je jejich naměřená hodnota shodná, proto je jejich rozdíl – tedy naměřená délka průběhu algoritmu – nulový.

Proto byla provedena i další metoda měření času. Program FFD1 byl za tímto účelem modifikován tak, že umožní provést měření délky průběhu pro opakovaně spouštěné algoritmy. Počet opakování lze nastavit na mocniny čísla 10. Opět jako v předchozím měření lze zvolit požadované zhuštění počítaných bodů plochy a metodu, kterou má být výpočet proveden. Měření zde zajišťuje funkce `Time`, která jako výsledek vrací aktuální datum a čas. Nejmenším krokem je zde jedna sekunda. To uložíme do proměnné příslušného typu (datový typ `TDateTime`). Hodnoty uložené před započetím a po skončení zadaného počtu opakování odečteme. Na formát času je převedeme pomocí funkce `TimeToStr`. Toto měření bylo provedeno pouze pro nedeformovanou mřížku řídicích bodů, neboť – jak už bylo uvedeno – nemá tento fakt na délku průběhu dané metody výrazný vliv. Naměřené hodnoty jsou uvedeny v tabulce 3.5. Některá pole tabulky jsou ponechána prázdná. Důvodem je, že již předchozí měření poskytla dostatečně přesné výsledky a další měření by dávala shodné výsledky.

Z tabulky 3.5 vyplývají následující závěry. Jeden výpočet plochy (jedno opakování cyklu) pro jednonásobné zhustění trval v případě použití de Boorova algoritmu $\frac{564}{10^5}s = 5,64 \cdot 10^{-3}s = 5,64ms$, zatímco pomocí předpočítaných Bernsteinových polynomů pouze $\frac{56}{10^5}s = 5,6 \cdot 10^{-4}s = 0,56ms$, tedy 10-krát kratší čas.

Při dvojnásobném zhuštění dostáváme časy $\frac{147}{10^4}s = 1,47 \cdot 10^{-2}s = 14,7ms$ při použití de Boorova algoritmu, zatímco jen $\frac{354}{10^5}s = 3,54 \cdot 10^{-3}s = 3,54ms$ u Bernsteinových polynomů, které tedy v tomto případě přibližně 4-krát rychlejší. Podobný poměr časů dostáváme i pro zhuštění čtyřnásobné: $\frac{403}{10^4}s = 4,03 \cdot 10^{-2}s = 40,3ms$ u de Boorova algoritmu a $\frac{117}{10^4}s = 1,17 \cdot 10^{-2}s = 11,7ms$ u Bernsteinových polynomů.

A konečně při 8-násobném zhuštění trvá de Boorův algoritmus $\frac{275}{10^3}s = 2,75 \cdot 10^{-1}s = 275ms$, zatímco metoda s Bernsteinovými polynomy $\frac{540}{10^4}s = 5,4 \cdot 10^{-2}s = 54,ms$, je tedy přibližně pětkrát rychlejší.

Tabulka 3.5: Experimentálně naměřená délka průběhu FFD pomocí de Boorova algoritmu (nahoře) a pomocí metody FFD využívající Bernsteinovy polynomy (dole), [s]

algoritmus de Boor	počet opakování					
zhuštění	10^0	10^1	10^2	10^3	10^4	10^5
1	0	0	0	9	82	564
2	0	0	2	17	147	–
4	0	1	7	54	403	–
8	0	2	23	275	–	–

Bernstein. polynomy	počet opakování					
zhuštění	10^0	10^1	10^2	10^3	10^4	10^5
1	0	0	0	1	6	56
2	0	0	0	3	46	354
4	0	0	1	11	117	–
8	0	1	3	48	540	–

Zjištěné závěry jsou v souladu s předpokladem, že metoda používající předpočítané bázové (Bernsteinovy) polynomy bude rychlejší oproti značně univerzálnímu de Boorovu algoritmu.

Obě metody by však šlo do jisté míry optimalizovat, čímž by se jejich výpočetní časy zkrátily. Dalšího urychlení lze dosáhnout, pokud by tyto metody nebyly naprogramovány pro výpočet procesorem počítače, nýbrž pro výpočet pomocí grafické karty.

4 Závěr

V rámci práce jsem zpracoval problematiku FFD transformací. Práce je rozdělena na dvě části: teoretickou a praktickou. V první části jsem popsal teoretické základy metod Free-form deformace, především definice a vlastnosti kubických splajn křivek a ploch, dále je zde uvedena teorie z oblasti lineární algebry týkající se maticového počtu, pojmů inverzní a pseudoinverzní matice a řešení poddeterminovaných soustav lineárních rovnic. Na jejich řešení vede problém přímé manipulace plochou (zpětná FFD). Teoretická část je zakončena kapitolou popisující současné metody Free-form deformací.

Praktickou část této práce tvoří navržené knihovny, které obsahují metody sloužící k výpočtu splajn ploch pomocí postupů popsaných v teoretické části práce. Jedná se o knihovnu programující de Boorův algoritmus a knihovnu programující metodu s Bernsteinovými polynomy. Dále v průběhu práce vzniklo několik pomocných knihoven sloužících k výstupu dat do LaTeXu, Matlabu a Gnuplotu. Metody, které zprostředkovávají výstup do Gnuplotu, jsou velmi důležité, neboť slouží ke grafickému výstupu pro všechny vytvořené programy.

Všechny navržené programy FFD1, FFD2, FFD3 vytváří grafické uživatelské rozhraní pro výpočet a porovnání jednotlivých metod. Tyto programy využívají pravidelnou mřížku řídicích bodů, kterou si uživatel přizpůsobí svým požadavkům. Výsledky jsou uvedeny nejdříve graficky (ve formě ploch vykreslených oběma metodami pro shodně zadané transformace), potom ve formě tabulek porovnávajících samotné posunutí jednotlivých bodů výsledné plochy. Dále je v práci uvedeno porovnání metod z hlediska časové náročnosti výpočtu.

Metody, které jsou v práci zpracovány, jsou naprogramovány pro dvoudimenzionální případ. Ten byl zvolen pro svoji přehlednost a názornost. Snadno však lze použité metody zobecnit i pro obecné trojdimenzionální zadání.

Metody FFD slouží k matematickému a počítačovému modelování v mnoha oborech lidské činnosti: od návrhů automobilů a letadel, přes průmyslový design běžných předmětů, po modelování objektů při vývoji počítačových her. V nejnovějším výzkumu jsou také využívány v lékařství, kdy metody odvozené od FFD slouží k popisu změn nádorů při léčbě nádorových onemocnění.

Literatura

[1] COQUILLART, Sabine a Pierre JANCÉNE. Animated free-form deformation. In: *Proceedings of the 18th annual conference on Computer graphics and interactive techniques - SIGGRAPH '91* [online]. New York, New York, USA: ACM Press, 1991, s. 23-26 [cit. 2017-05-25]. DOI: 10.1145/122718.122720. ISBN 0897914368. Dostupné z: `http://portal.acm.org/citation.cfm?doid=122718.122720`

[2] GOODMAN, Tim N.T a Keith UNSWORTH. InjectiveBivariateMaps. Annals of Numerical Mathematics 3. Dundee, Scotland, UK GB, 1996, s. 91–104. Dostupné z: `https://www.researchgate.net/publication/235300492_Injective_bivariate_maps`

[3] GREGORČIČ, Gregor. *The Singular Value Decomposition and the Pseudoinverse* [online]. Corc, Ireland, 2001 [cit. 2017-05-25]. Dostupné z: `http://www.cs.bgu.ac.il/~na151/wiki.files/SVD_application_paper.pdf`

[4] HIROTA, G., R. MAHESHWARI a M.C. LIN. *Fast volume-preserving free-form deformation using multi-level optimization.* In: Computer-Aided Design [online]. 2000, 32(8-9), s. 499-512 [cit. 2017-05-25]. DOI: 10.1016/S0010-4485(00)00038-5. ISSN 00104485. Dostupné z: `http://linkinghub.elsevier.com/retrieve/pii/S0010448500000385`

[5] HSU, William M., John F. HUGHES a Henry KAUFMAN. Direct manipulation of free-form deformations. *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques.* New York, NY, USA, 1992, 177–184. DOI: http://doi.acm.org/10.1145/133994.134036. ISSN 0-89791-479-1. Dostupné z: `https://pdfs.semanticscholar.org/4b69/ac8f81ee87a7a72e17c4f02c7e2def499a28.pdf`

[6] JEŽEK, František. *Geometrické a počítačové modelování* [online]. Verze 8.1. Plzeň, 2006 [cit. 2017-05-25]. Dostupné z: `http://www.fd.cvut.cz/personal/voracsar/GM/PGR021/GM_Jezek.pdf`

[7] KARÁSEK, Jiří a Ladislav SKULA. *Lineární algebra: teoretická část.* Brno: Akademické nakladatelství CERM, 2005. ISBN 80-214-3100-8.

[8] KARÁSEK, Jiří. *Matematika II.* 2. vyd. Brno: PC-DIR, 1995. ISBN 80-214-0714-x.

[9] KOŠČÁKOVÁ, Petra. *Maticové numerické výpočty.* Brno, 2016. Dostupné z `https://is.muni.cz/th/394432/prif_m/DiplomovaPrace.pdf`

[10] KRČEK, Břetislav a Ivan KOLOMAZNÍK. *Algoritmy a datové struktury* [online]. Ostrava: Technická univerzita, 2008 [cit. 2017-05-25]. ISBN 978-80-248-1306-6. Dostupné z `http://homen.vsb.cz/~kol70/algoritmy/ads.pdf`

[11] LEDVINA, Lukáš a Pavel BROM. *Grafické zpracování dat a měření pomocí Gnuplotu* [online]. 2012 [cit. 2017-05-25]. Dostupné z: `http://fykos.cz/doc/gp_zaklady.pdf`

[12] LAMOUSIN, H.J. a N.N. WAGGENSPACK. NURBS-based free-form deformations. In: *IEEE Computer Graphics and Applications* [online]. 1994, 14(6), s.59-65 [cit. 2017-05-25]. DOI: 10.1109/38.329096. ISSN 0272-1716. Dostupné z: `http://ieeexplore.ieee.org/document/329096/`

[13] LEE, Seungyong, George WOLBERG, Kyung-Yong CHWA a Sung Yong SHIN. Image Metamorphosis with Scattered Feature Constraints. *IEEE Transactions on Visualization and Computer Graphics*. 1996, 2(No. 4), s. 337-354. Dostupné z: `http://www.academia.edu/14353190/Image_metamorphosis_with_scattered_feature_constraints`

[14] PIEGL, Les A. a Wayne TILLER. *The NURBS book*. Berlin: Springer-Verlag, 1997. Monographs in visual communication. ISBN 3-540-61545-8.

[15] PROCHÁZKOVÁ, Jana. *Modelování matematických ploch v CAD systémech* Brno: Vysoké učení technické v Brně, 2007. ISBN 978-80-214-3455-4.

[16] RAPPOPORT, Ari, Alla SHEFFER a Michel BERCOVIER. Volume-preserving free--form solid. In: *Proceedings of the third ACM symposium on Solid modeling and applications - SMA '95* [online]. New York, New York, USA: ACM Press, 1995, s. 361-372 [cit. 2017-05-25]. DOI: 10.1145/218013.218086. ISBN 0897916727. ISSN 0010-4485. Dostupné z: `http://portal.acm.org/citation.cfm?doid=218013.218086`

[17] RUECKERT, D., L.I. SONODA, C. HAYES, D.L.G. HILL, M.O. LEACH a D.J. HAWKES. Nonrigid registration using free-form deformations: application to breast MR images. In:*IEEE Transactions on Medical Imaging* [online]. 1999, 18(8), s. 712-721 [cit. 2017-05-25]. DOI: 10.1109/42.796284. ISSN 02780062. Dostupné z: `http://ieeexplore.ieee.org/document/796284/`

[18] SAMAREH, Jamshid. Aerodynamic Shape Optimization Based on Free-Form Deformation. In: *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference* [online]. Reston, Virigina: American Institute of Aeronautics and Astronautics, 2004 [cit. 2017-05-25]. DOI: 10.2514/6.2004-4630. ISBN 978-1-62410-019-2. Dostupné z: `http://arc.aiaa.org/doi/10.2514/6.2004-4630`

[19] SEDERBERG, Thomas W. a Scott R. PARRY. Free-Form Deformation of Solid Geometric Models. In: *Proceedings of the 20th annual conference on Computer graphics and interactive techniques - SIGGRAPH* '86. 1986, Vol. 20(No. 4), s. 151-160. Dostupné z: `http://pdfs.semanticscholar.org/aff0/330cb894430eb64a4e1eb2bb9dae7a73855f.pdf`

[20] SHEKHAR, Raj, Peng LEI, Carlos R. CASTRO-PAREJA, William L. PLISHKER a Warren D. D'SOUZA. Automatic segmentation of phase-correlated CT scans through nonrigid image registration using geometrically regularized free-form deformation. In: *Medical Physics* [online]. 2007, 34(7), s. 3054-3066 [cit. 2017-05-25]. DOI: 10.1118/1.2740467. ISSN 00942405. Dostupné z: `http://doi.wiley.com/10.1118/1.2740467`

[21] De Boor's Algorithm. *Computer Science* [online]. [cit. 2017-05-25]. Dostupné z: `http://www.cs.mtu.edu/~shene/COURSES/cs3621/NOTES/spline/de-Boor.html`

[22] B-spline Surfaces: Important Properties. *Computer Science* [online]. [cit. 2017-05-25]. Dostupné z: `http://www.cs.mtu.edu/~shene/COURSES/cs3621/NOTES/surface/bspline-properties.html`

[23] SKULA Ladislav *Pseudoinverzní matice, metoda nejmenších čtverců* [online]. Brno, 2015 [cit. 2017-05-25].

[24] WILLIAMS, Thomas a Colin KELLEY. *Gnuplot 4.6: An Interactive Plotting Program* [online]. Version 4.6.6. 2014 [cit. 2017-05-25]. Dostupné z: `http://www.gnuplot.info/docs/gnuplot.pdf`

[25] Gnuplot. *GeoWikiCZ* [online]. [cit. 2017-05-25]. Dostupné z: `http://gama.fsv.cvut.cz/gwiki/Gnuplot`

5 Seznam použitých zkratek a symbolů

FFD	Free-form deformation
obr., tab.	obrázek, tabulka
$\exists$,$\forall$	malý („existuje“) a velký („pro všechna“) kvantifikátor
$\leq$,$<$	je menší nebo rovno, je ostře menší
$\geq$,$>$	je větší nebo rovno, je ostře větší
$=$,$\neq$	je rovno, je různé od (není rovno)
$\in$,$\notin$	je prvkem, není prvkem
$\mathbb{N}, \mathbb{R}, \mathbb{C}$	množiny přirozených, reálných a komplexních čísel
$\overline{z}$	číslo komplexně sdružené s číslem z
$\langle a, b\rangle$	uzavřený (zleva i zprava) interval
(a, b)	otevřený (zleva i zprava) interval
$(a, b\rangle$	interval zleva otevřený zprava uzavřený
$\langle a, b)$	interval zleva uzavřený zprava otevřený
$\sum$	sumace
Δ	rozdíl
$\binom{n}{k}$	kombinační číslo „n nad k“
$\lfloor x \rfloor$	(dolní) celá část čísla x
$\min M$	minimum množiny M
$\overrightarrow{AB}$	vektor určený body A, B
$\mathbf{v}$	vektor
$\mathbf{0}$, $\mathbf{1}$	nulový vektor, jedničkový vektor
$\lvert\mathbf{v}\rvert_\infty$	∞-norma vektoru $\mathbf{v}$
M, $M_{m,n}$, M_n	matice, matice typu $m \times n$, čtvercová matice řádu n
O, $O_{m,n}$	nulová matice, nulová matice typu $m \times n$
I, I_n	jednotková matice, jednotková matice řádu n
$\det M$	determinant (čtvercové) matice M

$r(M)$	hodnost matice M
M^T	matice transponovaná k maticí M
$\overline{M}$	matice komplexně sdružená s maticí M
M^*	matice Hermitovsky sdružená s maticí M
$\operatorname{adj} M$	matice adjungovaná k matici M
M^{-1}	inverzní matice k matici M
M^+	pseudoinverzní matice k matici M