ZÁPADOČESKÁ UNIVERZITA V PLZNI

FAKULTA APLIKOVANÝCH VĚD

KATEDRA MATEMATIKY

DIPLOMOVÁ PRÁCE

Květen 2002

Pavel Čermák

ZÁPADOČESKÁ UNIVERZITA V PLZNI

FAKULTA APLIKOVANÝCH VĚD

KATEDRA MATEMATIKY

VÝPOČET VRSTEVNIC NA TROJÚHELNÍKOVÉ SÍTI

(DIPLOMOVÁ PRÁCE)

21.5.2002 Pavel Čermák

Předkládám tímto k posouzení a obhajobě diplomovou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem předloženou diplomovou práci vypracoval samostatně a s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

Na tomto místě bych také rád poděkoval vedoucí diplomové práce Doc. Dr. Ing. Ivaně Kolingerové za poskytnuté materiály, rady a připomínky.

V Plzni 21.5.2002 _____

Obsah

1. Úvod

Nepochybně důležitou vědní disciplínou je kartografie, která se zabývá zobrazením zemského povrchu. Jedním z produktů kartografie je mapa. Zde se však setkáváme s problémem, jak v mapě zobrazit a znázornit výškovou složku zobrazovaného povrchu. Existuje více metod, jednou z nich jsou vrstevnice. Vrstevnice mají skvělou vypovídací hodnotu a dávají dobrou prostorovou představu o daném terénu, proto jsou široce používány jak v klasické, tak i v digitální kartografii.

Tato diplomová práce se zabývá výpočtem a zobrazením vrstevnic na trojúhelníkové síti z hlediska digitální kartografie a zkoumá a porovnává různé metody a přístupy k dané problematice. Na základě známých teoretických poznatků z oblasti kartografie, matematiky, geografických informačních systémů a počítačové grafiky jsme se věnovali jak teoretickému odvození možných metod výpočtu, tak i jejich praktické realizaci. Bylo vytvořeno několik programových modulů pro systém MVE. Tyto moduly tvoří podstatnou část diplomové práce a slouží k výpočtu vrstevnic a jejich vizualizaci.

Práce si kladla za cíl vytvořit programový nástroj, kterým by bylo možno prohlížet trojrozměrná geografická data zobrazená pomocí vrstevnic, samozřejmě při dodržení kartografických a geografických zásad.

Práce je rozdělena do několika částí. V úvodních kapitolách objasňujeme některé základní pojmy a vysvětlujeme význam vrstevnic. V dalších kapitolách uvádíme možné algoritmy, které se používají pro řešení daného problému. Dále jsou vysvětleny principy námi užitých algoritmů a jejich otestování. V přílohách přikládáme uživatelskou příručku, ukázky výměnných formátů a použitých datových struktur. Na závěr jsou přiloženy příklady grafických výstupů programů. Součástí práce je CD se zdrojovými kódy programů a ukázkovými daty.

Za pomoc při realizaci programové části děkuji vedoucí diplomové práci Doc. Dr. Ing. Ivaně Kolingerové, která kromě teoretických rad poskytla většinu implementovaných algoritmů. Dále děkuji vývojářskému týmu grafiků na katedře informatiky za jejich pomoc při práci se systémem MVE a s grafickou knihovnou OpenGL.

2. Reliéf a povrch

Kartografické znázornění reliéfu, tj. průběhu zemského povrchu, je složitá záležitost. Zemský reliéf je výsledkem dlouhodobého působení přírodních sil i zásahů člověka. Ve své podstatě představuje plochu nezměřitelné složitosti (viz písečné přesypy v poušti, zoraná pole). Při topografickém mapování je reálný zemský povrch nahrazován **topografickou plochou**. Ta je výsledkem účelové schematizace zemského reliéfu ovlivněné měřítkem mapy a typizací reliéfních tvarů.

Na reliéfu se rozlišují tvary vyvýšené (vyvýšeniny, např. horstva) a tvary snížené neboli vhloubené (např. údolí). Vyvýšené tvary mají část vrcholovou (vrchol), boční (úbočí) a úpatní (úpatí). Snížené tvary mají úbočí části a dno. Ve vyvýšeninách se rozeznávají pahorky, kopce (vrchy), hory. Hory jsou jednoduché nebo složené, s vrcholovou částí plochou, kupovitou nebo ostrou (špičky, věže). Uvedené tvary pokrývají v různém uspořádání celé území.

2.1 Metody zobrazování reliéfů

Přístupy k vyjadřování členitosti zemského povrchu prodělaly dlouhý vývoj a tvoří specifickou skupinu metod jazyka mapy. Zobrazování zemského reliéfu, často nazývaného třetí rozměr mapy, se většinou vzájemně kombinují. V mapě nelze zobrazit reliéf se všemi jeho podrobnostmi. Jeho modelem je dříve zmíněná topografická plocha, která je obecnou plochou s nepravidelným průběhem ve vodorovném i svislém směru. Uvedeme přehled základních možností zachycení této plochy v mapovém obrazu.

Výškové kótování

Kóty jsou číselně vyjádřené výšky nebo hloubky bodů, vrstevnic nebo vodních ploch vůči zvolené hladinové (srovnávací) ploše.

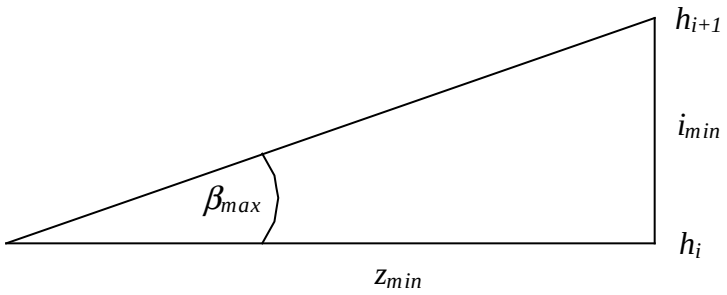
Vrstevnice

Jsou uzavřené linie spojující na topografické ploše body o stejné, vhodně zaokrouhlené výšce. Vrstevnice jsou půdorysné obrazy průniků hladinových ploch (zjednodušeně vodorovných rovin) vedených v určitém výškovém intervalu.
Vrstevnice se dělí na **základní** (výška je dělitelná zvoleným výškovým intervalem), **hlavní** (výška je násobek výškového intervalu, kreslí se zesílenou čarou a kótují se), **doplňkové**

(pomocné vrstevnice s intervalem rovným zlomku výškového intervalu, užívají se v plochých terénech, kde by použití základních vrstevnic nedokázalo vyjádřit reliéf). Mezi vrstevnice patří i **horizontály**, mající obecnou výšku, omezují např. stojaté vodní plochy.

Spolu s kótami dávají vrstevnice geometricky nejpřesnější obraz reliéfu, používají se při projekčních pracích (profily, čáry určitého sklonu, řezy, kubatury, řešení viditelnosti aj.). Kartografům slouží jako podklad pro ostatní metody zobrazování reliéfu (barevná hypsometrie, stínování, blokdiagramy).

Interval vrstevnic (rozestup sousedních hladinových ploch) se volí v topografických mapách konstantní, na zeměpisných mapách se jedná o rozhraní typických výškových stupňů. Při jeho volbě je nutné brát v úvahu sklonové poměry reliéfu. Z maximální hodnoty sklonu β_{max} a minimální zobrazitelné vzdálenosti mezi dvěma vrstevnicemi z_{min} = 0.3 mm se určí minimální hodnota intervalu i_{min} (viz obr.2.1).



Obr. 2.1: Sklonový trojúhelník

V rámci topografického mapového díla bývá vrstevnicový interval konstantní. V našich výškových podmínkách se běžně volí v závislosti na měřítku mapy hodnota

$$i = M/5000 \qquad (2.1)$$

tj. např. pro mapu 1 : 25 000 je i = 5 m.

Barevná hypsometrie

V zeměpisných mapách středních a malých měřítek nelze volit jednotný interval vrstevnic, v horských partiích by docházelo k jejich neúměrnému přehuštění , zatímco rovinné oblasti by byly vyjádřeny nevýrazně. Barevná hypsometrie vychází z metody vrstevnic. Její princip spočívá ve vykreslení vrstevnic ohraničujících typické výškové intervaly získané z průběhu hypsografické křivky reliéfu. Plochy mezi sousedními

vrstevnicemi se vykrývají barevně. Počet barevných vrstev a barva vykrytí se řídí účelem mapy a výškovou členitostí území.

Stínování

Obraz reliéfu se často doplňuje stínováním založeným na konvenčním osvitu terénu, pro lepší prostorový vjem. Při volbě úhlu dopadu paprsků pod úhlem 45° dostane mapa podobu leteckého snímku, kdy strany přivrácené ke světlu jsou světlé a odvrácené ztemněné.

Šrafy

Jsou krátké spádnice kreslené jako úsečky nebo malé geometrické obrazce (např. trojúhelníčky), kreslené hustě vedle sebe. Jejich vypovídací hodnota je spíše umělecká. Graficky značně zatěžují mapu a dnes se jich používá ke znázornění drobných terénních tvarů, které nelze vystihnout vrstevnicemi.
[Veverka B., 1997]

2.2 Reliéf z hlediska digitální kartografie a digitální modely terénu (povrchy)

Veličiny, které se v prostoru souvisle mění (např. výška bodů na terénní ploše, tlak, teplota atd.) mohou být zpracovány jako souvislé „povrchy“.

Povrchy popisují souvisle se měnící hodnotu atributu v prostoru, bez náhlých změn (nespojitosti). Atribut se ukládá jako jednoduchá proměnná, jejíž hodnota je definována v poloze X,Y. Vzhledem k takovému charakteru existují pokusy popsat atributy matematickými funkcemi. Hovoří se proto i o funkcionálních površích. Rozumí se tím kromě jiného skutečnost, že daným souřadnicím X,Y odpovídá jen jedna hodnota – funkce proměnné Z. V obecných případech je však průběh ploch povrchů v prostoru vcelku příliš složitý, prakticky analyticky nepopsatelný. Proto se používá rozdělení ploch na menší části. Někdy se předpokládá, že tyto dílčí plošky jsou rovinné. Popis průběhu je pak jednodušší, samozřejmě na úkor přesnosti popisu povrchu jako celku. Důležitou úlohu přitom sehrávají dělící čáry mezi dílčími ploškami. Tyto jsou vedeny tam, kde se výrazně mění průběh povrchu jako celku. Hodnota funkce proměnné Z na dělicích liniích může být stejná (např. modelování pobřežní čáry) nebo častěji se mění (modelování údolnice,

hřebenu). Dělicí linie rozeznáváme tvrdé – jde skutečně o hrany, zlomy a měkké – přechod je plynulý.

Všeobecně můžeme říci, že pro jejich modelování jsou nejvhodnější mozaiky – pravidelné **rastrové reprezentace**, ale i nepravidelné – **nepravidelné trojúhelníkové sítě**. Z praktických aplikací má největší význam modelování povrchu – reliéfu terénu ve formě digitálních modelů terénu nebo digitálních výškových modelů.
[Tuček J., 1998]

2.2.1 Definice a systemizace digitálních modelů terénu

Podle slovníku geodetického a kartografického názvosloví **digitální model reliéfu terénu** „je soubor číselných informací o něm doplněný pravidly na jejich používání."

Podle Krchy, (1979) jde o „reprezentativní soubor bodů reliéfu terénu vybraných podle určitých pravidel, polohově lokalizovaných s přiřazeným vektorem (sloupcem hodnot) parametrů reliéfu terénu. Jde tedy o body, informace o nich a pravidla používání těchto informací."

Podle Burrougha, (1986), „každá číselná reprezentace souvislých změn reliéfu terénu v prostoru je jeho digitálním modelem."

Urban (1991) klasifikuje modely terénů takto:

- Terénní plocha je velmi nepravidelná – jsou místa, kde je průběh plynulý, jinde jsou změny prudší
- Lomové, dělící čáry oddělující plochy s víceméně plynulým průběhem se nazývají singularity
- Popsat plochu jako celek je problém, proto se popisují jednotlivé dílčí plošky, hranice dělení se vedou podle možnosti po singularitách
- Podle způsobu dělení jde o pravidelné nebo nepravidelné plošky.

Vzhledem k uvedenému pak rozeznáváme:

- **Rastrové modely** – využívají dělení plochy na pravidelné, stejně veliké plošky.

- **Polyedrické modely** – využívají dělení plochy na nepravidelné, různě veliké plošky, obvykle trojúhelníkového tvaru, na dělících ploškách se používají lineární interpolace.
- **Plátové modely** – využívají dělení plochy na nepravidelné, různě veliké plošky, obvykle trojúhelníkového tvaru. Na dílčích plochách se používá nelineární interpolace, zohledňuje se průběh plochy na sousedních plátech.

Podle Burrougha, (1986) je možné povrchy popsat jako:

1. Matematicky definované plochy v prostoru. Jako celek to není možné, proto je dělíme na menší, jednodušší části. Hodnoty funkce Z získáme různými typy interpolace na dílčích plochách.
2. Obrazy:

- **bodové** - s pravidelnou strukturou **– rastry nebo lattices**,
  - s nepravidelnou strukturou – **nepravidelné trojúhelníkové sítě** (např. systém TIN – trianguled irregular network)
- **liniové** - vrstevnicové
  - profilové
  - kritických čar (údolnice, hřebeny).

Podle materiálů k systému ARC/INFO existují modely:

1. S pravidelně rozmístěným bodovým polem (rastry, lattices).
2. S nepravidelně rozmístěným bodovým polem (trojúhelníkové sítě, např. TIN).

[Tuček J., 1998]

2.2.2 Rastrové modely terénu

Svojí podstatou vycházejí z běžných rastrových datových struktur. Zkoumaná oblast je rozdělena na pravidelný rastr, jehož buňky jsou prostorově lokalizovány. V každé buňce je uložena hodnota výšky terénu. Protože hodnota výšky se v prostoru plynule mění, není možné použít vylepšené způsoby uložení dat. Nejčastěji se jedná o matici, tedy přímé datování informační vrstvy. Prostorové údaje – lokalizace buněk rastru je obvykle uložená ve specifické části souboru (header) nebo ve zvláštním dokumentačním souboru. Mají všechny výhody rastrových datových reprezentací, tedy zvláště jednoduchost a lehkou pochopitelnost struktury. Stejně tak jednoduchost postupů při následném zpracování pro

výpočty odvozených parametrů a pro potřeby zobrazení. Stejně tak je to s nevýhodami. Především s potřebou uložit obrovská kvanta dat pro větší území, dále s nepřesností při hrubém rastru, nadbytečností údajů na plochách, kde je průběh terénu pravidelný a nevhodností, resp. nepřesností při výpočtech pro reprezentované liniové a plošné objekty (nesledují strukturu rastru).

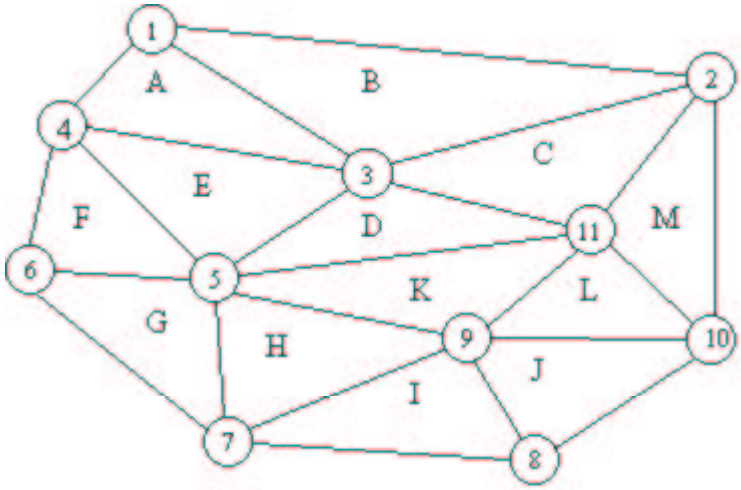
Ve většině případů se předpokládá, že výška uložená v každé z buněk rastru platí pro celou buňku. V jiných případech uložená výška platí pro střed buňky a v modelu je možné dále interpolovat. Potom hovoříme o tzv. lattices. Lattices můžeme považovat za (vektorový) bodový model terénu s pravidelným rozmístěním bodů.
[Tuček J. 1998]

2.2.3 Nepravidelné trojúhelníkové sítě

Využívají rozdělení terénní plochy na dílčí plochy, nejčastěji trojúhelníkového tvaru. Hranice dělení jsou vedeny po singularitách, liniích, na kterých dochází k výrazným změnám v průběhu terénní plochy jako celku. Prvotním zdrojem informací jsou údaje o prostorových vrcholech těchto trojúhelníků – jejich poloha v souřadnicovém systému a hodnota jejich výšky. Spojnice vrcholů trojúhelníků by měly co nejvýstižněji sledovat linie, na kterých dochází k výrazným změnám v průběhu terénní plochy jako celku. Ve vnitřku trojúhelníků předpokládáme pravidelný rovnoměrný průběh změn výšek. Pro tyto entity vybudujeme soustavu definovaných topologických vztahů tak, aby povrchy trojúhelníků bylo možné popsat a přitom nebylo potřebné údaje o vrcholech, jejich spojnicích a plochách ukládat vícenásobně.

K popisu topologických vztahů průsečíků, spojnic a ploch lze použít několik postupů. Jen informativně uvádíme, že je to např. princip okřídlené hrany, kde jsou definovány všechny vzájemné topologické vztahy všech uzlů, hran i ploch a který se používá při detailních výpočtech pro potřeby projektování, výpočtů objemů zemských prací, architektonické účely apod. Datová struktura je však náročná na množství uložených dat a postupy výpočtů nejsou zcela triviální. GIS využívají pro tento účel tzv. TIN struktury – nepravidelné trojúhelníkové sítě, ve kterých je topologie popsána speciálním uspořádaným způsobem.

Příklad souborů a způsob uložení údajů pro TIN model terénu uvádíme na obr. 2.2 a v tabulkách tab.2.1, tab.2.2 a tab.2.2.



Obr. 2.2: Ukázka TIN

Identifikátor trojúhelníka	souřadnice x	y	z
1	x_1	y_1	z_1
2	x_2	y_2	z_2
3	x_3	y_3	z_3
4	x_4	y_4	z_4
.	.	.	.
.	.	.	.
11	x_{11}	y_{11}	z_{11}

a) seznam vrcholů

Tab. 2.1

Identifikátor trojúhelníka	vrcholy		
A	1	3	4
B	1	2	3
C	2	3	11
D	5	3	11
E	3	4	5
F	4	5	6
G	5	6	7
H	5	7	9
I	7	8	9
J	8	9	10
K	5	9	11
L	9	10	11
M	2	10	11

b) seznam vrcholů

Tab. 2.2

Identifikátor trojúhelníka	vrcholy		
A	B	E	
B	A	C	
C	B	D	M
D	C	E	K
E	A	D	F
F	E	G	
G	F	H	
H	G	I	K
I	H	J	
J	I	L	
K	D	H	J
L	J	K	M
M	C	L	

c) seznam hran

Tab. 2.3

Základním požadavkem pro danou strukturu je přitom vypočítat – odvodit hodnotu funkce Z pro kterýkoli bod zkoumané plochy. Interpolace na dílčích plochách může být buď lineární nebo nelineární.

Výhodou uvedených modelů v porovnání s rastrovými modely je zmenšený objem uložených údajů a možnost odvodit, vypočítat hodnotu výšky terénu (hodnota funkce Z) pro libovolný bod zkoumané plochy. Jsou tedy přesnější a vhodnější pro práci s liniovými objekty. Nevýhodou je naopak složitost struktury a postupu jejího vzniku. Stejně tak odvození jakékoli informace je mimořádně výpočetně náročné.

V případě konstrukce tohoto typu modelu není nepravidelnost rozmístění vstupního bodového pole na závadu. Spíše je důležité, zda do výběru byly zahrnuty body ležící na všech důležitých singularitách. Z tohoto důvodu nejsou např. vhodným zdrojem pro tvorbu TIN zdigitalizované vrstevnice. Z nepravidelně rozmístěného bodového pole lze vytvořit mnoho možných trojúhelníkových sítí. Přičemž optimalizačním kritériem může být požadavek na co největší rovnostrannost trojúhelníků, jejich minimální obvod apod.
[Tuček J. 1998]

2.2.4 Analýza modelů terénu

Základním požadavkem na model terénu je možnost odvodit výšku terénu pro definované místo, objekty apod. Způsob řešení tohoto požadavku výrazně souvisí s typem modelu terénu a použité datové struktury, které jsme popsali výše.

Pokud je rastrový model ukládán a zpracováván jako lattices, používá se k odvození výšek bodů mezi středy buněk, zvláště tzv. bilineární (dvojnásobná lineární) interpolace. Podrobněji tuto problematiku rozebírá Urban, (1988), který uvádí detailní matematický aparát pro interpolaci na troj nebo čtyřúhelníkové rovinné plošce.

V TIN strukturách lze k interpolaci přistupovat dvěma způsoby. Buď se plocha trojúhelníku považuje za rovinnou nebo za obecně křivou. Předpokladem uskutečnění interpolace je identifikace trojúhelníka jako dílčí plošky, ve které se nachází bod, pro který se má interpolace uskutečnit. Z TIN struktury lze načíst všechny souřadnice polohy (X,Y,Z) vrcholů prostorového trojúhelníku. Pomocí těchto souřadnic lze vypočítat koeficienty obecné rovnice rovinné trojúhelníkové plochy v prostoru:

$$ax + by + cz + d = 0 \tag{2.2}$$

Kde *a*, *b*, *c*, *d* jsou koeficienty stanovené na základě souřadnic vrcholů trojúhelníku. Podle Urbana, (1988) pak:

$$\begin{vmatrix} x & y & z & 1 \\ x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \end{vmatrix} = 0 \tag{2.3}$$

odtud : $z = kx + ly + m$

$$k = \frac{\begin{vmatrix} y_1 & z_1 & 1 \\ y_2 & z_2 & 1 \\ y_3 & z_3 & 1 \end{vmatrix}}{\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}} \qquad l = \frac{\begin{vmatrix} x_1 & z_1 & 1 \\ x_2 & z_2 & 1 \\ x_3 & z_3 & 1 \end{vmatrix}}{\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}} \qquad m = \frac{\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}}{\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}} \tag{2.4}$$

V druhém případě se při vyjádření plochy analytického trojúhelníku berou v úvahu i sousední plochy trojúhelníku. K analytickému popisu obecně křivých plátů se využívají různé způsoby např. Bézierovy plochy, viz Urban (1991), viz Ježek (2000) .

Použití interpolační metody vychází z povahy zdrojových údajů pro tvorbu modelu a jeho určení. Lineární interpolace je výpočetně rychlejší, ale průběh povrchu je narušen na každé hraně mezi plochami trojúhelníků. Můžeme ji považovat za vhodnou pro interpolaci na površích, které byly vytvořeny z bodového pole získaného systematickým výběrem, přičemž výběrové bodové pole by mělo obsahovat co největší počet bodů, definujících lokální minima a maxima, inflexní body a zlomové čáry. Lineární interpolace je dostatečná pro většinu aplikací. Nelineární interpolace je výpočetně náročnější, ale dává realističtější výsledky pro rozptýlené vstupní (řídké) bodové pole.

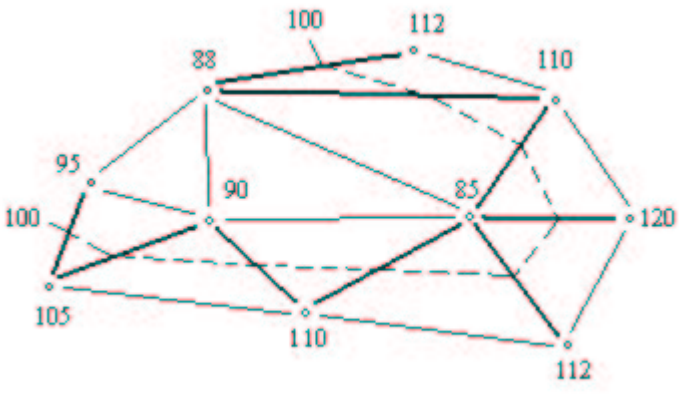
I když to tak na první pohled vypadá, interpolace, zjišťování výšky nemusí být triviální, jednorázová záležitost postupného dotazování pro jednotlivé polohy.
[Tuček J., 1998]

3. Používané algoritmy na výpočet vrstevnic

Digitální modely terénu se běžně využívají pro odvození vrstevnic – horizontálních řezů terénu pro zpracování vrstevnicových map, vertikálních řezů terénu – systematických pro generování pohledů na terén, případně pro zadanou profilovou čáru apod. Často na vygenerování řezové čáry navazuje algoritmus na její vyhlazení, nebo je vyhlazení přímo součástí výpočtu. Popis algoritmů pro generování vrstevnic z TIN uvádějí různí autoři. Jednoduché vysvětlení poskytují např. Cornelius a Heywood, (1994), kteří uvádějí, že TIN je „už svou podstatou vhodný pro exaktní interpolační postupy." Protože hodnota výšky je známá v každém vrcholu prostorového trojúhelníku a vzdálenost mezi nimi lze snadno dopočítat, je možné použít lineární interpolaci na základě vzdálenosti výpočtu výšky na spojnicích vrcholů trojúhelníků.

3.1 Algoritmy pro lineární interpolaci

Při generování izolinií z TIN modelu musíme nejdříve stanovit výšku izolinie, jejíž průběh se má identifikovat. Tu můžeme odvodit ze stanovené minimální výšky a vybraného výškového – vrstevnicového intervalu. Pro každou konkrétní hodnotu výšky vrstevnice algoritmus vyhledá nejdříve všechny spojnice vrcholů – hrany, které tato protíná. Provádí se to porovnáním výšky vrstevnice s výškami vrcholů všech spojnic v soustavě. Pro snadnější pochopení viz obr. 3.1. Pro výšku 100 v příkladu to bude 9 z existujících 19 spojnic vyznačených hrubou čarou. V dalším kroku se vypočítá poloha x, y bodů na spojnicích, ve kterých je vrstevnice přetíná – zde se uplatňuje lineární interpolace na základě výškového rozdílu a vzdálenosti na spojnici. Dalším krokem je pospojování bodů na hranách do vrstevnice. [Tuček J., 1998].



Obr.3.1: Generování vrstevnice

Tímto postupem je možné generovat všechny vrstevnice procházející jedním trojúhelníkem. Výhoda algoritmu spočívá v tom, že zpracujeme každý trojúhelník pouze jedenkrát, nevýhodou však zůstává setřídění a pospojování vypočtených bodů do vrstevnic.

Druhý možný způsob nalezení izolinií vychází z podstaty již zmiňované datové struktury pro uložení TIN. S úspěchem lze využít toho, že pro každý trojúhelník sítě jsou známy jeho sousední trojúhelníky. Při prvním kroku musíme opět stanovit výšku izolinie jako v předchozím případě. Algoritmus poté vyhledá jeden trojúhelník, kterým daná vrstevnice prochází, a určí polohu x, y průsečíků hran trojúhelníku s touto vrstevnicí. Při výpočtu průsečíků se též uplatňuje lineární interpolace. Ze znalosti sousedních trojúhelníků lze jednoznačně odvodit další směr pokračování vrstevnice (následující trojúhelník pro zpracování). Tento způsob se opakuje do té doby, než narazíme na konečný trojúhelník sítě (nelze dále pokračovat v generování vrstevnice), nebo se vrátíme zpět do prvního zpracovaného trojúhelníku (vrstevnice se uzavírá). Tento způsob má výhodu oproti předchozímu v tom, že dostaneme body vrstevnice seřazené tak, jak jdou za sebou a nemusíme je složitě třídit.

Známým problémem při těchto přístupech je lomený průběh vrstevnice. Tento problém lze odstranit buď použitím vyhlazovacího algoritmu (jen mírného, neboť jinak snižuje polohovou věrnost odvozené vrstevnice), anebo použitím nelineární interpolační funkce již při tvorbě.

3.2 Vyhlazovací algoritmy

Vyhlazování se v digitální kartografii používá hlavě z toho důvodu, aby se odstranil nežádoucí efekt lomového průběhu linie či na eliminaci z hlediska měřítka zanedbatelných detailů. Můžeme říci, že jde v podstatě o estetickou úpravu dané digitální linie.

McMaster, Shea (1992) rozdělují vyhlazovací algoritmy podle způsobu a kritéria vyhlazení do tří základních skupin:

1. **Váhové průměrování** (Weighted averages)
   - S krokem tří bodů (three-point weighted moving average)
   - S krokem pěti bodů (five-point weighted moving average)
   - Jiné krokové techniky (other moving average techniques)
   - Vzdálenostní váhové průměrování (distance weighted averaging)
   - Průměrování s posunem (slide averaging)

2. **Filtrování pomocí epsilon okolí** (Epsilon filtering)

3. **Matematické aproximace** (Mathematical approximation)
   - Lokální zpracování (Local processing): kubický spline
   - Rozšířené lokální zpracování (Extended local processing): B-spline
   - Globální zpracování (Global processing): Bézierova křivka

3.2.1 Váhové průměrování

Uvedeme zde pouze dva algoritmy prezentované McMasterem, a to McMasterův algoritmus průměrování s posunem (McMaster´s slide averaging algorithm) a algoritmus vzdálenostně-váhového průměrování (distance-weighting algorithm).

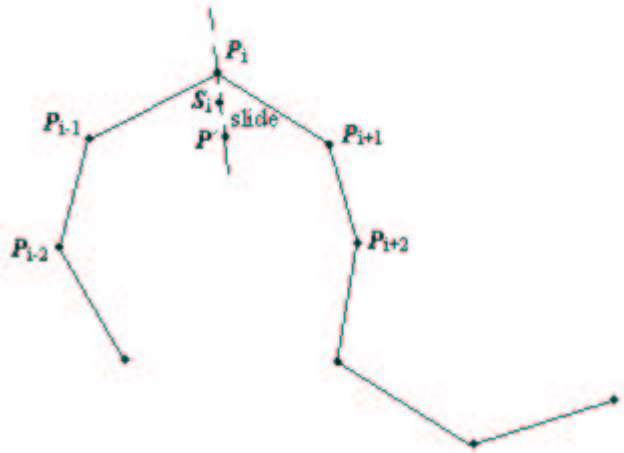
První algoritmus používá pět bodů původní linie k vyhlazení prostředního, třetího bodu. V prvním kroku vypočítáme bod $\boldsymbol{P}'$ aritmetickým průměrem z pěti daných bodů $\boldsymbol{P}_{i-2}$, $\boldsymbol{P}_{i-1}$, $\boldsymbol{P}_i$, $\boldsymbol{P}_{i+1}$, $\boldsymbol{P}_{i+2}$. Tento vypočtený bod $\boldsymbol{P}'$ je poté posunut (slide) směrem k původnímu bodu $\boldsymbol{P}_i$. Toto se může provést průměrováním bodů $\boldsymbol{P}'$ a $\boldsymbol{P}_i$. Tento postup se aplikuje na všechny body dané linie. Pomocí této techniky se posouvají body původní linie, a tím dochází k jejímu vyhlazení (viz obr. 3.2). Bod $\boldsymbol{P}'$ vypočítáme aritmetickým průměrem z pěti bodů takto:

$$\boldsymbol{P}' = \frac{\boldsymbol{P}_{i-2} + \boldsymbol{P}_{i-1} + \boldsymbol{P}_i + \boldsymbol{P}_{i+1} + \boldsymbol{P}_{i+2}}{5} \tag{3.1}$$

a bod vyhlazené linie vypočteme podle vztahu:

$$\boldsymbol{S}_i = slide \cdot \boldsymbol{P}' + (1 - slide) \cdot \boldsymbol{P}_i \tag{3.2}$$

Čím větší bude konstanta *slide*, tím se vypočtený bod linie $\boldsymbol{S}_i$ bude blížit k bodu $\boldsymbol{P}'$.



Obr. 3.2: McMasterův algoritmus průměrování s posunem

Druhá technika používá vzdálenostně-váhovou metodu, která daným bodům přiřazuje váhy, a tím dochází k intenzivnějšímu posunu bodů směrem k centrálnímu bodu. I v této metodě se vypočítá bod $\boldsymbol{P}'$ pomocí pěti bodů. Váhami v tomto případě jsou převrácené hodnoty vzdálenosti bodů od prostředního, třetího, bodu. Pro názornost vyjdeme ze stejného obrázku jako v předchozím případě (obr. 3.2). První bod $\boldsymbol{P}_{i-2}$ leží ve vzdálenosti d_1 od třetího bodu $\boldsymbol{P}_i$, druhý bod $\boldsymbol{P}_{i-1}$ leží ve vzdálenost d_2 od třetího bodu, atd. Je samozřejmé, že třetí bod má vzdálenost rovnu nule. Ze znalosti daných vzdáleností určíme polohu bodu $\boldsymbol{P}'$ podle vzorce:

$$\boldsymbol{P}' = \frac{\frac{1}{d_1}\boldsymbol{P}_{i-2} + \frac{1}{d_2}\boldsymbol{P}_{i-1} + \frac{1}{d_4}\boldsymbol{P}_{i+1} + \frac{1}{d_5}\boldsymbol{P}_{i+2}}{\frac{1}{d_1} + \frac{1}{d_2} + \frac{1}{d_4} + \frac{1}{d_5}} \tag{3.3}$$

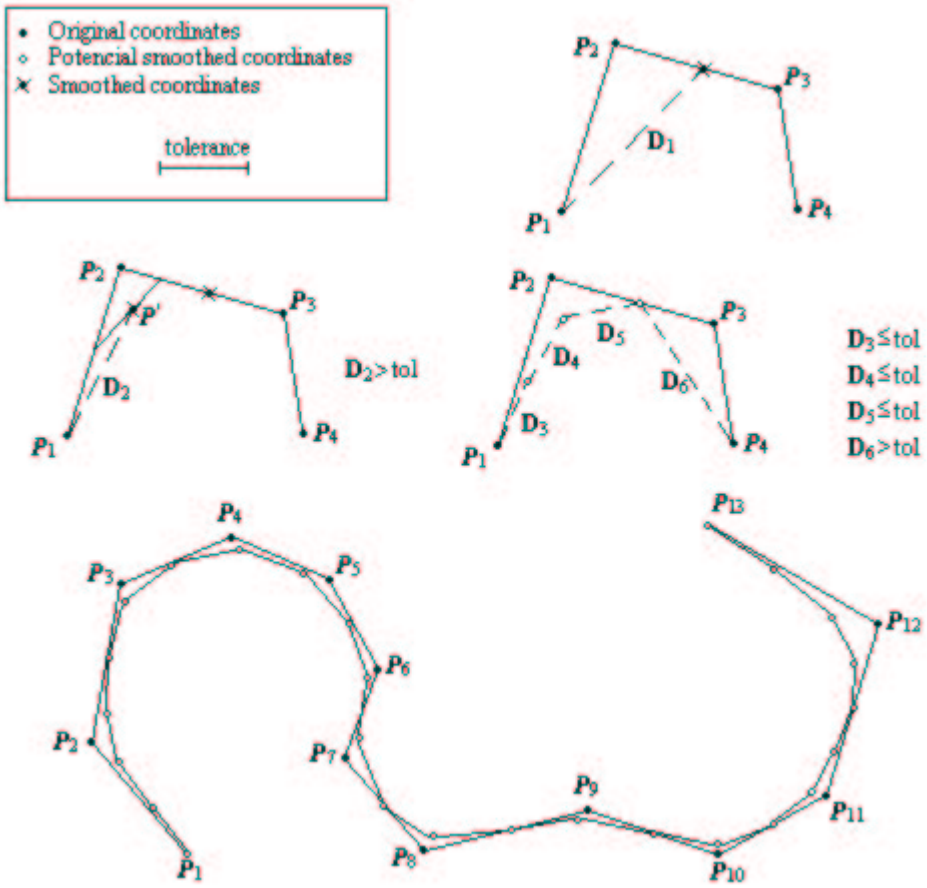
a bod linie $\boldsymbol{S}_i$ se určí obdobným způsobem jako v předchozím případě:

$$\boldsymbol{S}_i = slide \cdot \boldsymbol{P}' + (1 - slide) \cdot \boldsymbol{P}_i \tag{3.4}$$

[McMASTER, R.B., Shea, K.S., 1992]

3.2.2 Filtrování pomocí epsilon okolí

Zde uvedeme pouze jednu metodu – Chaikensův vyhlazovací algoritmus (Chaiken´s smoothing algorithm) (viz obr. 3.3). V prvním kroku se vypočítá vektor mezi bodem $\boldsymbol{P}_1$ a středním bodem úsečky $\boldsymbol{P}_2$ a $\boldsymbol{P}_3$ nebo bodem $\boldsymbol{P}'$. Tento vektor je označen $\boldsymbol{D}_1$. Když je velikost vektoru $\boldsymbol{D}_1$ větší něž tolerance *tol*, musí se vypočítat nový bod $\boldsymbol{P}'$, který nyní leží na středu úsečky $\boldsymbol{P}_1$ a $\boldsymbol{P}_2$. Velikost vektoru $\boldsymbol{D}_2$, který leží mezi body $\boldsymbol{P}_1$ a $\boldsymbol{P}'$, je opět poměřována vzhledem k toleranci *tol*. V případě, že je velikost vektoru stále větší než tolerance, vektor $\boldsymbol{D}_2$ se rozdělí. Třetí sekvence na obrázku ukazuje vytvoření vyhlazovacích bodů mezi bodem $\boldsymbol{P}_1$ a středním bodem úsečky $\boldsymbol{P}_2$ a $\boldsymbol{P}_3$. Ve spodní části obrázku je ukázáno vyhlazení celé linie touto metodou. Čím je tolerance, *tol* menší, tím samozřejmě dochází k lepší aproximaci původní křivky. Riesenfeld (1975) dokázal, že když je tolerance nekonečně malá, aproximovaná křivka je shodná s B-splinem.
[McMASTER, R.B., Shea, K.S., 1992]



Obr. 3.3: Chaikensův vyhlazovací algoritmus

3.2.3 Matematické aproximace

Matematických aproximací existuje celá řada. My se zde omezíme na již zmiňované křivky (spline, Bézierova křivka a B-spline)

Fergusonova kubika

Nechť jsou dány body $\boldsymbol{P}_i$ a $\boldsymbol{P}_{i+1}$. Označme dále $\boldsymbol{P}'_i$ a $\boldsymbol{P}'_{i+1}$ tečné vektory v těchto bodech. Nechť uvedeným bodům odpovídají hodnoty t_i a t_{i+1} parametru. Furgusonovu kubiku určíme pomocí rovnice

$$\boldsymbol{P}(i) = H_0\,(t-t_i)\boldsymbol{P}_i + H_1(t-t_i)\boldsymbol{P}_{i+1} + H_2(t-t_i)\boldsymbol{P}'_i + H_3(t-t_i)\boldsymbol{P}'_{i+1} \tag{3.5}$$

kde $H_i(s)$ jsou polynomy třetího stupně.

Z podmínek $\boldsymbol{P}(t_i) = \boldsymbol{P}_i$, $\boldsymbol{P}(t_{i+1}) = \boldsymbol{P}_{i+1}$, $\boldsymbol{P}'(t_i) = \boldsymbol{P}'_i$, $\boldsymbol{P}'(t_{i+1}) = \boldsymbol{P}'_{i+1}$ určíme vektory koeficientů u jednotlivých mocnin výrazu t - t_i a dojdeme při označení ${}^ik = t_{i+1} - t_i$, $s = t - t_i$ k těmto vztahům:

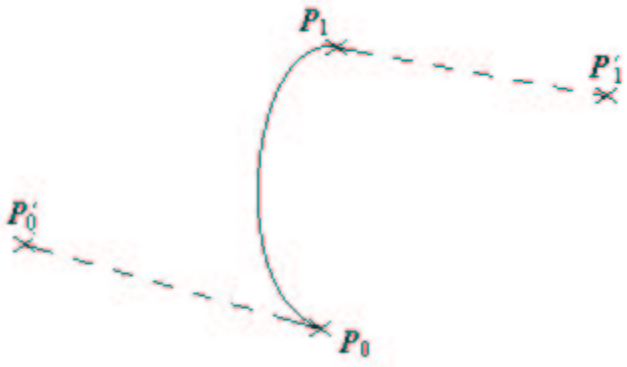
$$\begin{aligned}
H_0(s) &= \frac{2}{{}^ik^3}s^3 - \frac{3}{{}^ik^2}s^2 + 1 \\
H_1(s) &= -\frac{2}{{}^ik^3}s^3 + \frac{3}{{}^ik^2}s \\
H_2(s) &= \frac{1}{{}^ik^2}s^3 - \frac{2}{{}^ik}s^2 + s \\
H_3(s) &= \frac{1}{{}^ik^2}s^3 - \frac{1}{{}^ik}s
\end{aligned} \tag{3.6}$$

Pro uniformní parametrizaci $t_i = 0$, $t_{i+1} = 1$, tj. ${}^ik = 1$ získáme z předchozích vztahů funkce F_i , které hrají důležitou roli u kubické spline křivky:

$$\begin{aligned}
F_0(t) &= 2t^3 - 3t^2 + 1 \\
F_1(t) &= -2t^3 + 3t^2 \\
F_2(t) &= t^3 - 2t^2 + t \\
F_3(t) &= t^3 - t^2
\end{aligned} \tag{3.7}$$

Tyto křivky (příklad viz obr. 3.4) použil prvně v počítačové grafice J.C.Ferguson. Je třeba zdůraznit, že pro tvar Fergusonovy kubiky má značný význam délka tečných vektorů.

[Ježek F., 2000]



Obr. 3.4: Fergusonova kubika

Kubické interpolační spline křivky

Spline křivky patří k často užívaným interpolačním křivkám v technické praxi. Jsou totiž matematickým modelem chování pružného laťového křivítka, které v minulosti používali konstruktéři trupů lodí, když z několika významných profilů trupu lodi odvozovali (interpolovali) další profily. Z hlediska praxe mají zásadní význam kubické spline křivky. Výklad proto omezíme pouze na tuto skupinu spline křivek. Nejprve budeme definovat pojem **kubické spline funkce**: Nechť je dána posloupnost $x_0 < x_1 < \ldots < x_n$ a funkční hodnoty $y_0, y_1, \ldots, y_n$. Kubickou spline funkcí $f(x)$ rozumíme interpolační funkci, tj. platí $f(x_i) = y_i$, která je na každém z intervalů $\langle x_i, x_{i+1} \rangle$ polynomem třetího stupně, a funkce f má spojitou první a druhou derivaci v intervalu (x_0, x_n).

Zadáním opěrných bodů však není kubická spline funkce určena jednoznačně, neboť kubické polynomy (je jich n) mají celkem $4n$ koeficientů a k dispozici máme $4n - 2$ podmínek:

- krajní body jednotlivých úseků … $2n$ podmínek,
- spojitost první derivace … $n - 1$ podmínek
- spojitost druhé derivace … $n - 1$ podmínek

Je zřejmé, že k určení kubické spline funkce jsou třeba ještě dvě další podmínky. Zpravidla se zadávají hodnoty první derivace v počátečním a koncovém bodě křivky nebo

hodnoty druhé derivace v těchto bodech (speciálně se často volí „podmínka volného konce“, tj. nulová druhá derivace v krajních bodech).

Spline křivkou (kubickou) pro dané opěrné body $\boldsymbol{P}_0$, $\boldsymbol{P}_1$, …, $\boldsymbol{P}_n$ a dané hodnoty parametru $t_0 < t_1 < t_n$ rozumíme křivku $\boldsymbol{P} = \boldsymbol{P}(t)$, $t \in \langle t_0, t_n \rangle$, pro níž každá ze složek vektorové funkce $\boldsymbol{P}(t)$ je kubickou spline funkcí. Zadání hodnot parametru pro opěrné body se většinou realizuje jistým algoritmem. Např. se volí uniformní parametrizace $t_i = i$.

Výpočet kubické spline křivky je možné provést tak, že nejprve určíme tečné vektory křivky v opěrných bodech a pak jednotlivé oblouky spline křivky vypočteme jako Fergusonovy kubiky podle vztahů uvedených výše.

Označme $\boldsymbol{P}'_i$, $i = 0, \ldots, n$ hledané tečné vektory kubické spline křivky v opěrných bodech. Vzhledem k požadavku spojitosti druhé derivace získáme vztah:

$$\frac{1}{{}_{i}k}\boldsymbol{P}'_i + \left(\frac{2}{{}_{i}k} + \frac{2}{{}_{i+1}k}\right)\boldsymbol{P}'_{i+1} + \frac{1}{{}_{i+1}k}\boldsymbol{P}'_{i+2} = \frac{3}{{}_{i+1}k^2}\boldsymbol{P}_{i+2} + \left(\frac{3}{{}_{i}k^2} - \frac{3}{{}_{i+1}k^2}\right)\boldsymbol{P}_{i+1} - \frac{3}{{}_{i}k^2}\boldsymbol{P}_i \tag{3.8}$$

a pro uniformní parametrizaci:

$$\boldsymbol{P}'_i + 4\boldsymbol{P}'_{i+1} + \boldsymbol{P}'_{i+2} = 3\boldsymbol{P}_{i+2} - 3\boldsymbol{P}_i \tag{3.9}$$

kde $i = 0, \ldots, n-2$.

Další dvě rovnice pro výpočet tečných vektorů se zpravidla odvodí z některé z následujících podmínek.

a) Je dáno $\boldsymbol{P}'_0$ a $\boldsymbol{P}'_n$, tj. je dán tečný vektor v počátečním a koncovém bodě křivky. Pak má soustava již jediné řešení (viz obr. 3.5).

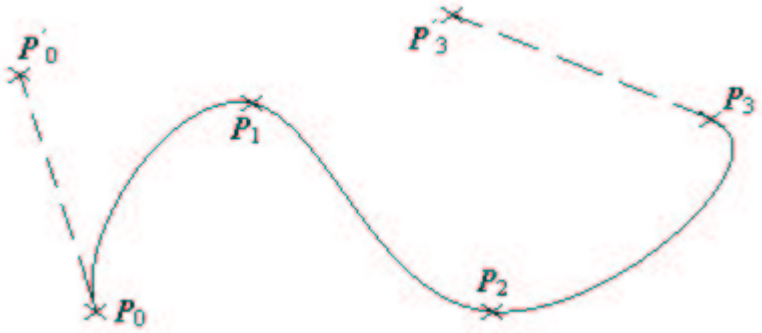
b) Jsou dány vektory $\boldsymbol{A}$ a $\boldsymbol{B}$ druhých derivací v počátečním a koncovém bodě křivky. V tomto případě doplníme soustavu o rovnice:

$$\begin{aligned} 2\boldsymbol{P}'_0 + \boldsymbol{P}'_1 &= \frac{3}{{}_{0}k}(\boldsymbol{P}_1 - \boldsymbol{P}_0) - \frac{{}_{0}k}{2}\boldsymbol{A} \\ \boldsymbol{P}'_{n-1} + 2\boldsymbol{P}'_n &= \frac{3}{{}_{n-1}k}(\boldsymbol{P}_n - \boldsymbol{P}_{n-1}) + \frac{{}_{n-1}k}{2}\boldsymbol{B} \end{aligned} \tag{3.10}$$

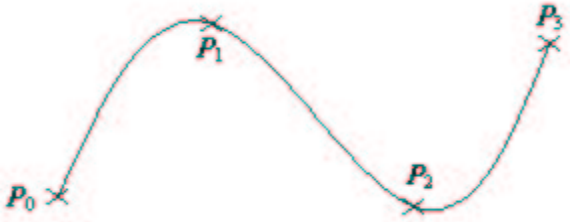
Speciálně pro $\boldsymbol{A} = \boldsymbol{B} = \mathbf{o}$ dostáváme tzv.kubický přirozený spline (viz obr. 3.6)

c) Podmínka uzavřenosti křivky vede k doplnění soustavy o další dvě rovnice, které získáme použitím formule (3.8) $i = n - 1$ a $i = n$ s tím, že od indexů větších než n odečteme $n + 1$.

Soustavu n rovnic o n neznámých vektorech řešíme pomocí Gaussovy eliminace. Matice soustavy je pro všechny složky vektorů stejná, proto lze provést přímý chod Gaussovy eliminace najednou pro různé pravé strany. Matice soustavy je diagonálně dominantní a pro případ a) a b) je navíc tato matice třídiagonální.
[Ježek F., 2000]

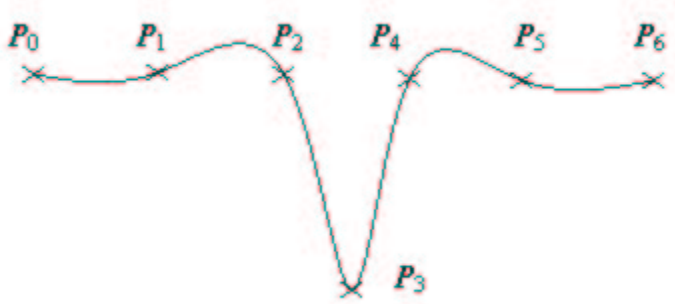


Obr. 3.5: Spline se zadanými koncovými tečnami

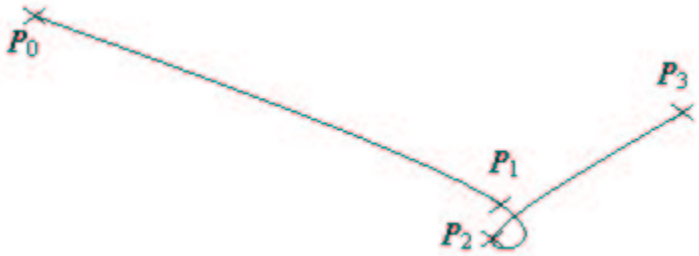


Obr. 3.6: Přirozený spline

Spline křivka je velice silný nástroj v CAD systémech a počítačové grafice, avšak při nevhodném rozmístění řídících bodů dochází k nežádoucímu chování této křivky. Jak vidíme na obrázku 3.7, křivka kmitá v těch místech, kde řídící body jsou v přímce. V krajních případech může protínat sama sebe, viz příklad na obrázku 3.8.



Obr. 3.7: Kmitání spline křivky



Obr. 3.8: Protnutí spline křivky

Bézierovy křivky

Uvažujme lomenou čáru (řídící polygon) o vrcholech $\boldsymbol{P}_0, \ldots, \boldsymbol{P}_n$ ($n \geq 2$). Bézierovou křivkou pro tento řídící polygon rozumíme křivku:

$$\boldsymbol{P}(t) = \sum_{i=0}^{n} \boldsymbol{P}_i B_i^n(t), t \in \langle 0,1 \rangle \tag{3.11}$$

kde $B_i^n(t)$ jsou Bernsteinovy polynomy (přesněji báze prostoru Bernsteinových polynomů), tj.

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i} \tag{3.12}$$

$i = 0, \ldots, n$, kde definitoricky položíme $\binom{0}{0} = 1$ a $\binom{n}{0} = 1$.

Položíme-li ve vztahu (3.11) $t = 0$, je $\boldsymbol{P}(0) = \boldsymbol{P}_0$. Analogicky pro $t = 1$ je $\boldsymbol{P}(1) = \boldsymbol{P}_n$. Křivka tedy prochází krajními body řídicího polygonu.
Po derivaci $\boldsymbol{P}(t)$ dostaneme:

$$\boldsymbol{P}'(t) = \sum_{i=0}^{n} \boldsymbol{P}_i B_i'^{n}(t) \tag{3.13}$$

Dosazením za $t = 0$ a $t = 1$ vyplyne, že

$$\begin{aligned} \boldsymbol{P}'(0) &= n(\boldsymbol{P}_0 - \boldsymbol{P}_1) \\ \boldsymbol{P}'(1) &= n(\boldsymbol{P}_{n-1} - \boldsymbol{P}_n) \end{aligned} \tag{3.14}$$

Tedy tečné vektory mají směr spojnice vždy dvojice krajních bodů a velikost n-násobku jejich velikosti (n je stupeň aproximační křivky).

Vlastnosti Bernsteinových polynomů:

- $\forall i, n \in N \cup \{0\}$ a $t \in \langle 0,1 \rangle$ je $B_i^n(t) \geq 0$
- $\sum_{i=1}^{n} B_i^n(t) = 1$ pro $t \in \langle 0,1 \rangle$
- $B_i^n(t) = (1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(t)$

První vztah zaručuje nezápornost Bernsteinových polynomů. Z druhé vlastnosti vyplývá, že celá křivka bude ležet v tzv. konvexní obálce řídícího polygonu. Třetí vlastnost využil P. de Casteljau pro tvorbu rekurzivního algoritmu tvorby Bézierovy křivky.
[Alexandr L., 2001]

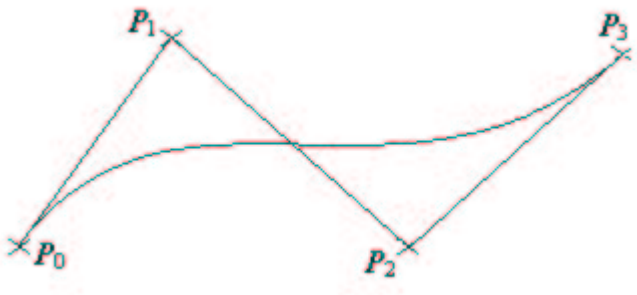
Bézierova kubika

Pro $n = 3$ jsou Bernsteinovy polynomy tvaru (horní index 3 zde vynecháváme):

$$\begin{aligned} B_0(t) &= (1-t)^3 \\ B_1(t) &= 3t(1-t)^2 \\ B_2(t) &= 3t^2(1-t) \\ B_3(t) &= t^3 \end{aligned} \tag{3.15}$$

Křivka je v tomto případě kubikou a mluvíme o Bézierově kubice. Je určena řídícím polygonem $\boldsymbol{P}_0$, $\boldsymbol{P}_1$, $\boldsymbol{P}_2$, $\boldsymbol{P}_3$ (viz obr. 3.9) a má rovnici:

$$\boldsymbol{P}(t) = \sum_{i=0}^{3} \boldsymbol{P}_i B_i(t), t \in \langle 0,1 \rangle \tag{3.16}$$

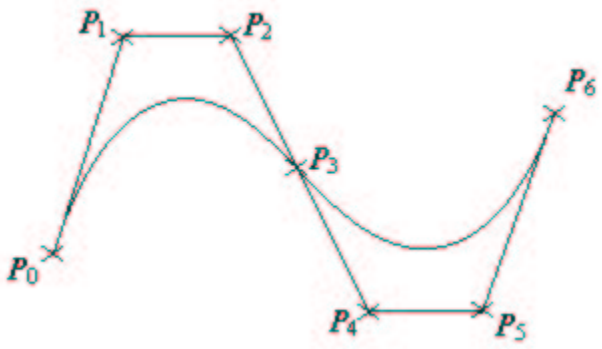


Obr. 3.9: Bézierova kubika

[Ježek F., 2000] [Alexandr L., 2001]

Napojení Bézierových kubik

Dva Bézierovy oblouky budou spojeny hladce, bude-li zaručena jejich spojitost (tj. poslední bod předchozího oblouku je identický s prvním následujícího, nebo křivka prochází posledním a prvním bodem) a pokud budou identické tečné vektory. Z toho jednoznačně plyne, že druhý bod následující křivky je určen posledními dvěma body křivky předchozí (viz obr. 3.10)



Obr. 3.10: Napojení Bézierových kubik

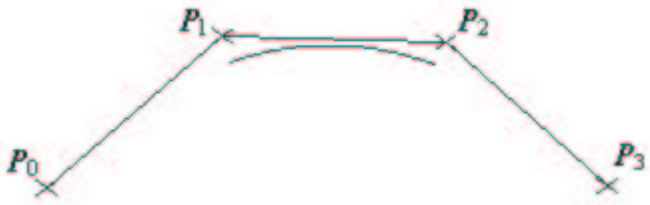
Coonsův kubický B-spline

Tato kubika má s Bézierovou kubikou podobný způsob zadávání. Rovněž pro Coonsovu kubiku zadáváme čtyři body $\boldsymbol{P}_0$, $\boldsymbol{P}_1$, $\boldsymbol{P}_2$, $\boldsymbol{P}_3$, které tvoří charakteristický polygon. Coonsova kubika (viz obr. 3.11) má rovnici:

$$\boldsymbol{P}(t) = \frac{1}{6}\sum_{i=0}^{3} \boldsymbol{P}_i C_i(t), t \in \langle 0,1 \rangle \tag{3.17}$$

kde

$$\begin{aligned} C_0(t) &= -t^3 + 3t^2 - 3t + 1 \\ C_1(t) &= 3t^3 - 6t^2 + 4 \\ C_2(t) &= -3t^3 + 3t^2 + 3t + 1 \\ C_3(t) &= t^3 \end{aligned} \tag{3.18}$$



Obr. 3.11: Coonsova kubika

Vlastnosti Coonsovy kubiky

Po dosazení $t = 0$ a $t = 1$ do vztahů (3.17) a (3.18) dostáváme:

$$\begin{aligned} \boldsymbol{P}(0) &= \frac{1}{6}(\boldsymbol{P}_0 + 4\boldsymbol{P}_1 + \boldsymbol{P}_2) \\ \boldsymbol{P}(1) &= \frac{1}{6}(\boldsymbol{P}_1 + 4\boldsymbol{P}_2 + \boldsymbol{P}_3) \end{aligned} \tag{3.19}$$

krajní body křivky tedy nevychází z řídicích bodů, ale z antitěžiště trojúhelníků $\boldsymbol{P}_0\boldsymbol{P}_1\boldsymbol{P}_2$ a $\boldsymbol{P}_1\boldsymbol{P}_2\boldsymbol{P}_3$. Antitěžiště trojúhelníka $\boldsymbol{P}_0\boldsymbol{P}_1\boldsymbol{P}_2$ je bod, ležící na těžnici procházející vrcholem $\boldsymbol{P}_1$ a vzdálenost od tohoto vrcholu je rovná $^1/_3$ délky těžnice. To samé platí pro trojúhelník $\boldsymbol{P}_1\boldsymbol{P}_2\boldsymbol{P}_3$ a bod $\boldsymbol{P}_2$.

Derivací vztahu (3.17) obdržíme:

$$\boldsymbol{P}'(t) = \frac{1}{6}\sum_{i=0}^{3} \boldsymbol{P}_i C_i'(t) \tag{3.19}$$

po dosazení parametru $t = 0$ a $t = 1$ dostáváme:

$$\begin{aligned} \boldsymbol{P}'(0) &= \frac{1}{6}\sum_{i=0}^{3} \boldsymbol{P}_i C_i'(0) = \frac{1}{6}(-3\boldsymbol{P}_0 + 3\boldsymbol{P}_2) = \frac{1}{2}(\boldsymbol{P}_2 - \boldsymbol{P}_0) \\ \boldsymbol{P}'(1) &= \frac{1}{6}\sum_{i=0}^{3} \boldsymbol{P}_i C_i'(1) = \frac{1}{6}(-3\boldsymbol{P}_1 + 3\boldsymbol{P}_3) = \frac{1}{2}(\boldsymbol{P}_3 - \boldsymbol{P}_1) \end{aligned} \tag{3.20}$$

tečna Coonsovy kubiky v bodě $\boldsymbol{P}(0)$ je rovnoběžná s přímkou $\boldsymbol{P}_0\boldsymbol{P}_2$ a je rovna polovině její délky. To samé platí pro bod $\boldsymbol{P}(1)$ a přímku $\boldsymbol{P}_1\boldsymbol{P}_3$.

- Je-1i bod $\boldsymbol{P}_1$ středem úsečky $\boldsymbol{P}_0\boldsymbol{P}_2$, vychází kubika z bodu $\boldsymbol{P}_1$. Analogicky pro body $\boldsymbol{P}_1$, $\boldsymbol{P}_2$ a $\boldsymbol{P}_3$
- Je-li $\boldsymbol{P}_0 = \boldsymbol{P}_1$, říkáme, že se jedná o dvojnásobný bod, a kubika vychází z bodu

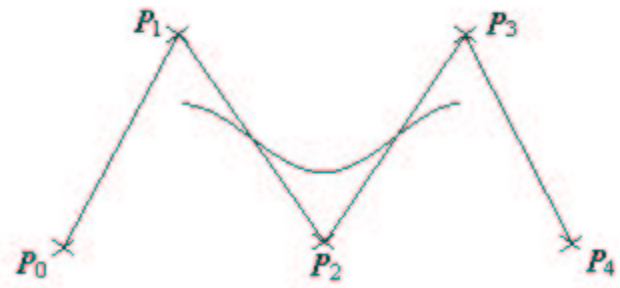
$$\frac{\boldsymbol{P}_0\boldsymbol{P}_2}{6} = \frac{\boldsymbol{P}_1\boldsymbol{P}_2}{6} \tag{3.21}$$

- Je-li $\boldsymbol{P}_0 = \boldsymbol{P}_1 = \boldsymbol{P}_2$, říkáme, že se jedná o trojnásobný bod a křivka vychází z něho

Body Coonsovy kubiky leží v konvexním obalu množiny $\mathrm{M} = \{\boldsymbol{P}_0, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{P}_3\}$. Důkaz uvedené vlastnosti plyne z toho, že

$$\frac{1}{6}\sum_{i=0}^{3} C_i(t) = 1 \text{ pro každé } t \tag{3.22}$$

Největší výhoda Coonsových kubik se stane zřejmou teprve v okamžiku, kdy je použijeme pro skládání aproximačních křivek. Uvažujme řídicí polygon složený z bodů $\boldsymbol{P}_0$, $\boldsymbol{P}_1$, ..., $\boldsymbol{P}_n$. Budeme-li výslednou křivku skládat z Coonsových oblouků vždy tak, že pro jeden oblouk použijeme vrcholy $\boldsymbol{P}_0\boldsymbol{P}_1\boldsymbol{P}_2\boldsymbol{P}_3$, pro další $\boldsymbol{P}_1\boldsymbol{P}_2\boldsymbol{P}_3\boldsymbol{P}_4$ atd., získáme křivku, která se nazývá B-spline (viz obr. 3.12).



Obr. 3.12: Napojení Coonsových kubik

[Ježek F.,2000][Alexandr L.,2001]

Doposud jsme se věnovali metodám, které v prvním kroku vypočítají z trojúhelníkové sítě lineární interpolací vrstevnici a v druhém kroku ji vyhlazují, aby nedocházelo k jejímu nežádoucímu lomenému průběhu. Existují však i metody, které spočítají hladkou izolinii přímo z trojúhelníkové sítě bez následného zpracování. Tento problém dobře řeší nelineární interpolace.

3.3 Algoritmy pro nelineární interpolaci

Techniky pro nelineární interpolaci v žádném případě nejsou triviální a vyžadují poměrně složitý matematický aparát. Při těchto metodách se rovinné trojúhelníky sítě nahrazují obecně nerovinnými, zaoblenými trojúhelníky, které lépe aproximují daný terén. Z takto aproximované trojúhelníkové sítě se odvozují vrstevnice, které jsou již dostatečně vyhlazené už při prvotním výpočtu a nedochází k nežádoucímu lomenému průběhu vrstevnice. Velikost zaoblení trojúhelníku samozřejmě závisí na jeho sousedech. Pro tyto metody se v praxi ve značné míře uplatňují Bézierovy trojúhelníkové pláty a pláty na bázi B-spline a NURBS (non-uniform rational B-spline). V této práci naznačíme postup pro aproximování trojúhelníkové sítě pomocí Bézierových trojúhelníkových plátů.

Bézierův trojúhelníkový plát

Nejprve objasníme pojem **barycentrické souřadnice**, které hrají významnou roli u zmíněného trojúhelníkového plátu. Barycentrické souřadnice (x_1, x_2) bodu X na úsečce s krajními body $\boldsymbol{P}_1\boldsymbol{P}_2$ definujeme přepisem:

$$\boldsymbol{X} = x_1\boldsymbol{P}_1 + x_2\boldsymbol{P}_2 \tag{3.23}$$

a podmínkou $x_1 + x_2 = 1$ ($x_1 \geq 0$, $x_2 \geq 0$).

Podobně pro rovinu, v níž uvažujeme trojúhelník $\boldsymbol{P}_1\boldsymbol{P}_2\boldsymbol{P}_3$. Barycentrické souřadnice (x_1, x_2, x_3) bodu $\boldsymbol{X}$ splňují vztahy:

$$\boldsymbol{X} = x_1\boldsymbol{P}_1 + x_2\boldsymbol{P}_2 + x_3\boldsymbol{P}_3 \ , \quad x_1 + x_2 + x_3 = 1 \tag{3.24}$$

Pokud $x_i \geq 0$, $i = 1, 2, 3$, náleží bod trojúhelníku $\boldsymbol{P}_1\boldsymbol{P}_2\boldsymbol{P}_3$.
Uvedeme tvar Bernsteinových polynomů pro případ barycentrických souřadnic. Platí:

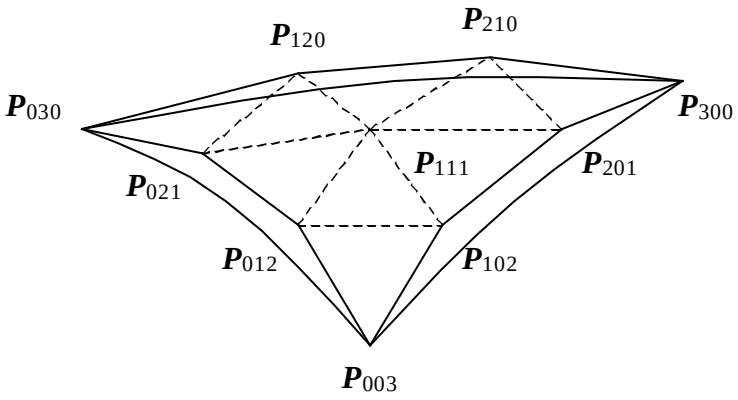
$$(x_1 + x_2 + x_3)^n = \sum_{\substack{i+j+k=n \\ i,j,k\geq 0}} \frac{n!}{i!\,j!\,k!} x_1^i x_2^j x_3^k \tag{3.25}$$

Zobecněný Bernsteinův polynom definujeme vztahem:

$$B_{i,j,k}^n(x_1, x_2, x_3) = \frac{n!}{i!\,j!\,k!} x_1^i x_2^j x_3^k \ , \quad i + j + k = n \ , \quad x_1 + x_2 + x_3 = 1 \tag{3.26}$$

Z předchozích vztahů vyplývá, že hraniční křivky plátu jsou též Bézierovy křivky.

Podobně, jako tomu bylo u Bézierovy křivky, i zde geometrii plátu definují řídící body $\boldsymbol{P}_{i,j}$. Tvoří řídící polyedr plátu. Pro trojúhelníkový plát je třeba 10 řídících bodů, z nichž $\boldsymbol{P}_{030}$, $\boldsymbol{P}_{003}$, $\boldsymbol{P}_{300}$ jsou vrcholy rovinného trojúhelníka (viz obr. 3.13). Ostatní řídící body se mohou určit například takto: v každém vrcholu se určí společná tečná rovina nějakou vhodnou metodou. Je například vhodné určit průměrný normálový vektor z normál rovin, které s vrcholem incidují. Zvolíme-li pak řídící body v příslušných tečných rovinách, je hladkost napojení plátů zaručena. Problém naopak nastává u bodu $\boldsymbol{P}_{111}$, kde se nenabízí tak jednoduché řešení.
[Ježek F., 2000][Urban J., 1991]



Obr. 3.13: Řídící body trojúhelníkového Bézierova plátu

V této kapitole jsme uvedli a naznačili několik možných metod a postupů na výpočet izolinií z trojúhelníkové sítě. Samozřejmě, že existuje ještě celá řada dalších metod,o kterých jsme se zde nezmiňovali. Např. NURBS křivky a pláty, které našly široké využití v modelování a počítačové grafice.

Pro realizaci programové části práce jsme vycházeli víceméně z uvedených skutečností. Bližším matematickým popisem a vysvětlením užitých algoritmů se zabývá následující kapitola.

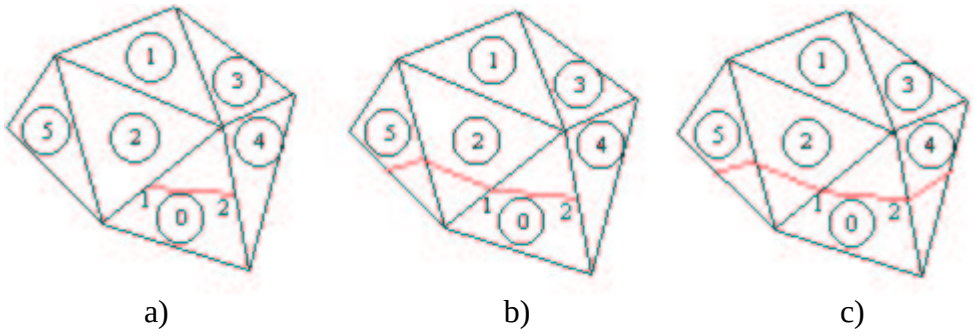
4. Metody a algoritmy použité v programu

V této kapitole se pokusíme čtenáři blíže přiblížit algoritmy a metody, které jsme implementovali v programu. Pokud nebude uvedeno jinak, postupy a metody byly poskytnuty vedoucím diplomové práce v algoritmické podobě. To umožnilo jejich snadnější naprogramování, avšak znesnadnilo jejich matematický popis a vysvětlení.

4.1 Výpočet vrstevnice z TIN

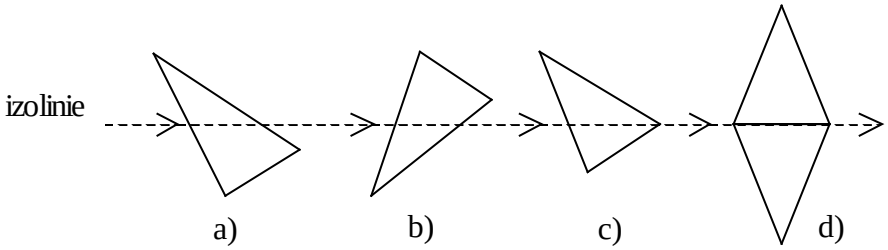
Pro nalezení a výpočet vrstevnice jsme použili algoritmus, využívající sousednost trojúhelníků, který jsme zmiňovali v podkapitole 3.1. Při znalosti výšky izolinie algoritmus sekvenčně prohledává tabulku trojúhelníků, dokud nenajde trojúhelník, jímž vrstevnice prochází. Poté se pomocí lineární interpolace vypočítají průsečíky s hranami tohoto trojúhelníku, které se uloží do datové struktury (popis datových struktur viz níže). Program si tento první nalezený trojúhelník zapamatuje a označí si ho jako *již zpracovaný* (obr. 4.1a). Dále pak je nalezen sousední trojúhelník, na jehož společné hraně s prvním trojúhelníkem byl spočítán jeden ze dvou průsečíků. Tento soused se označí jako *zpracovaný* a opět je spočítán průsečík na hraně, která je společná s dosud nezpracovaným trojúhelníkem. Tento postup se opakuje, dokud se nevrátíme do prvního zpracovávaného trojúhelníku (pak je vrstevnice uzavřená) nebo dokud nenarazíme na trojúhelník, který nemá souseda (obr. 4.1b). V tomto případě otočíme pořadí již vypočtených průsečíků a vracíme se do prvního trojúhelníka, kde prohledáváme síť v ještě nezpracovaném směru. Když i v tomto směru narazíme na okrajový trojúhelník, pak jeden segment vrstevnice dané výšky je kompletně spočítán (obr. 4.1c). Poté pokračujeme v sekvenčním prohledávání tabulky trojúhelníků a hledáme další vyhovující trojúhelník. Tabulku prohledáme celou, přičemž přeskakujeme ty trojúhelníky, které již byly zpracovány. Přeskakování zpracovaných trojúhelníků je logické, protože izolinie konstantní výšky může jeden trojúhelník procházet maximálně jedenkrát. Tímto způsobem nalezneme všechny segmenty vrstevnice dané výšky. Před dalším procházením tabulky je nutno vymazat příznak u těch trojúhelníků, které byly zpracovány (prochází jimi vrstevnice).

Na obr. 4.1 je graficky znázorněn postup algoritmu. Čísla v kroužku označují identifikátor trojúhelníku a čísla u světle kreslené izolinie značí první a druhý směr výpočtu.



Obr. 4.1: Postup při generování vrstevnice

Možnost průniku vrstevnice trojúhelníkem ukazuje obrázek 4.2:



Obr. 4.2: Možnosti průniku vrstevnice trojúhelníkem

Je zřejmé, že u příkladů a) a b) není problém určit sousední trojúhelník, do kterého povede izolinie. Směr postupu výpočtu je naznačený šipkami. Oproti tomu v případě c) a d) není jednoznačně dáno, jakým sousedním trojúhelníkem povede další výpočet průsečíků, neboť izolinie prochází vrcholem, resp. hranou trojúhelníku. V těchto případech se *z*-ová souřadnice daného vrcholu (vrcholů) trojúhelníku nepatrně zvětší o ε, které je blízké nule. Takže vrstevnice nebude procházet bodem, resp. hranou, ale dvěma hranami trojúhelníka. Můžeme tedy určit souseda trojúhelníku, tzn. jednoznačné pokračování dalšího postupu výpočtu. Při tomto postupu musíme dbát na to, aby ε nebylo příliš veliké a neovlivnilo tak přesnost výpočtu, zároveň však nesmí být příliš malé, aby omezená aritmetika počítače byla schopna vypočítat průsečík izolinie s hranou trojúhelníka. V programu je ε nastaveno na hodnotu $1 \cdot 10^{-4}$.

Tento algoritmus má nesrovnatelnou výhodu v tom, že průsečíky vrstevnice se uspořádají již při její tvorbě a není nutno je později zatřiďovat. Nevýhodou však zůstává

neustálé procházení celé tabulky s trojúhelníky, kterou projdeme tolikrát, kolik je výšek vrstevnic.

Složitost algoritmu je $O(NM)$, kde N je počet trojúhelníků a M počet vodorovných rovnoběžných rovin, které protínají danou trojúhelníkovou síť (počet výšek vrstevnic).

4.2 Použité vyhlazovací algoritmy

V programu jsme aplikovali dva druhy různých vyhlazovacích metod. První metoda je kombinací matematického a váhového vyhlazování, druhá metoda je čistě matematická.

4.2.1 První metoda (Normal Smooth)

Vyhlazování lomené čáry probíhá ve dvou krocích.

1. krok: Původní lomená čára se rovnoměrně proloží (zhustí) přidáním a přemístěním bodů tak, aby úseky mezi jednotlivými body byly stejné. Pokud jsou body v původní lomené čáře rozložené dostatečně hustě, mohou se i zředit. Při prokládání bodů se zároveň použije kubická interpolace na jejich částečné vyhlazení. Interpolace se provádí s ohledem na délku tečny. Nyní vysvětlíme matematickou podstatu tohoto prvního kroku metody. Nový bod interpolované křivky se vypočítá podle vztahu:

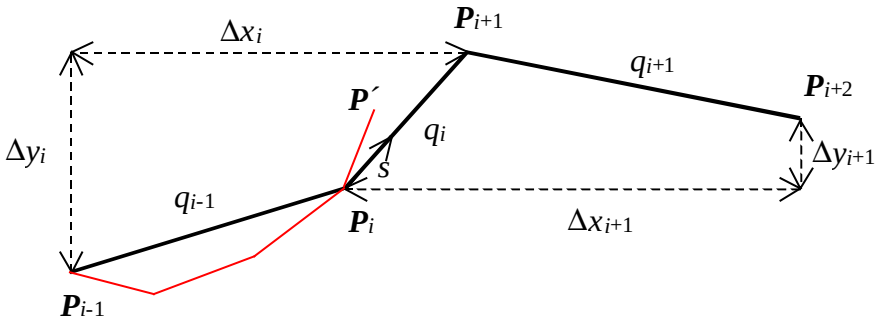
$$\begin{aligned} P'(x) &= u_1 \cdot P_i(x) + u_2 \cdot P_{i+1}(x) + v_1 \cdot sx_1 + v_2 \cdot sx_2 \\ P'(y) &= u_1 \cdot P_i(y) + u_2 \cdot P_{i+1}(y) + v_1 \cdot sy_1 + v_2 \cdot sy_2 \end{aligned} \quad , \quad i = 0, \ldots, n-1 \qquad (4.1)$$

kde $\boldsymbol{P}_i$ jsou vrcholy původní lomené čáry a n jejich počet. Koeficienty sx_1, sx_2, sy_1, sy_2 jsou rovny:

$$\begin{aligned} sx_1 &= \frac{q_i}{q_{i-1} + q_i} \cdot \Delta x_i & sy_1 &= \frac{q_i}{q_{i-1} + q_i} \cdot \Delta y_i \\ sx_2 &= \frac{q_i}{q_i + q_{i+1}} \cdot \Delta x_{i+1} & sy_2 &= \frac{q_i}{q_i + q_{i+1}} \cdot \Delta y_{i+1} \end{aligned} \qquad (4.2)$$

kde jednotlivé symboly jsou zřejmé z obrázku 4.3. Na obrázku je znázorněn silnou čarou původní polygon a červenou interpolační zhuštěná křivka. Symboly q_{i-1}, q_i a q_{i+1} jsou délky

jednotlivých úseků původní lomené čáry. ***P´*** je označen vypočítávaný bod interpolační křivky.

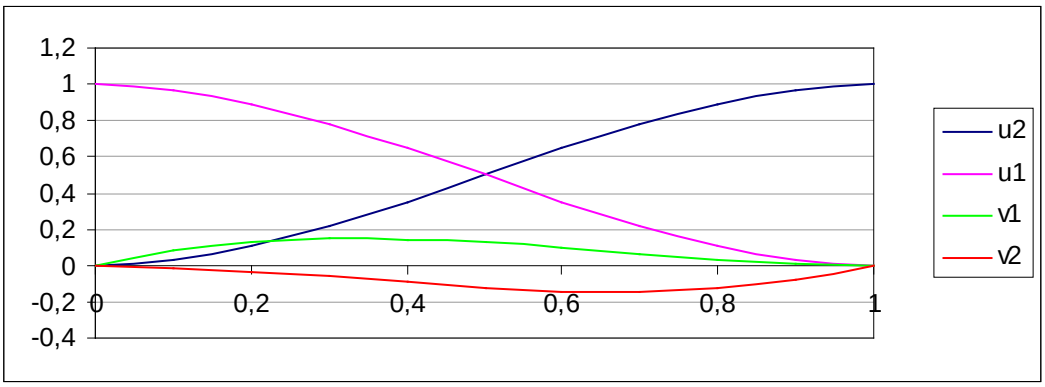


Obr. 4.3: První krok vyhlazovací metody Normal smooth

Koeficienty uváděné v původním vztahu u_1, u_2, v_1, v_2 se rovnají:

$$\begin{aligned} u_1 &= 1-u_2 & \qquad v_1 &= s(1-s)^2 \\ u_2 &= s^2(3-2s) & \qquad v_2 &= s^2(s-1) \end{aligned} \tag{4.3}$$

Jedná se o kubické polynomy parametru *s*. Jejich grafický průběh na intervalu $s \in \langle 0,1 \rangle$ ukazuje obrázek 4.4.



Obr. 4.4: Průběh kubických polynomů parametru *s*

Nyní objasníme význam *s*. Pro parametr *s* platí následující formule:

$$s = 1 - \frac{r-w}{q_i} \tag{4.4}$$

kde $r = \sum_{j=0}^{i} q_j$, neboli délka původní lomené čáry od bodu $\boldsymbol{P}_0$ až po bod $\boldsymbol{P}_{i+1}$ a $w = m \cdot ds$,

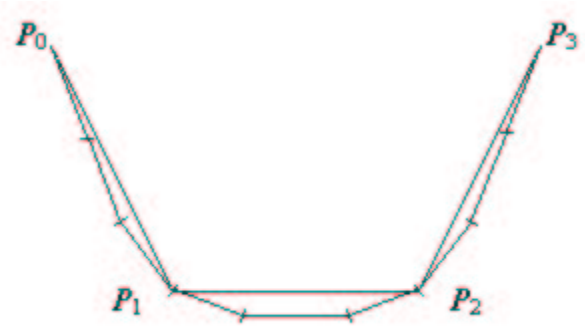
kde m je počet úseků dosud vypočtené interpolované křivky a ds je průměrná délka jednoho úseku původní lomené čáry.

$$ds = \frac{\sum_{j=0}^{n-1} \left| \boldsymbol{P}_j \boldsymbol{P}_{j+1} \right|}{n-1} \tag{4.5}$$

Jinými slovy, w je délka interpolované křivky od jejího začátku po předposlední bod. Poslední úsek končící bodem $\boldsymbol{P}'$ se nepočítá.

Míru zhušťování/ředění bodů ovlivňuje parametr p, kterým je před výpočtem interpolační křivky dělena konstanta ds – průměrná délka jednoho úseku původní lomené čáry. Tím se tato konstanta zmenší/zvětší a dochází se k žádanému výsledku. Je-li $p > 1$, pak se body zhušťují, naopak je-li p z intervalu (0, 1), body se ředí. Pro jednoduchost lze říci, že výsledný počet bodů linie se přibližně rovná $p \cdot (n - 1)$, kde n je počet bodů původní lomené čáry. Výsledek prvního kroku metody pro parametr p = 2,5 lze vidět na obrázku 4.5. Hodnotu parametru p musíme volit s ohledem na velikost trojúhelníkové sítě. Doporučené hodnoty viz podkapitola 5.2.

(V programu je možné nastavení parametru p a to v okně **Interpol_CL** v kolonce **Factor of Density)**.



Obr. 4.5: Výsledek prvního kroku metody normal smooth

Složitost prvního kroku algoritmu je $O(N)$, kde N je počet segmentů vrstevnic.

2. krok: Takto připravená a zčásti vyhlazená křivka se ještě dohlazuje za pomoci váhového průměrování z pěti bodů (viz obr. 4.6). Vypočítáme vektor posunu $\boldsymbol{V}(xs, ys)$ bodu $\boldsymbol{P}_i$ takto:

$$V(xs) = \frac{-P_{i-2}(x) + 4\left[P_{i-1}(x) + P_{i+1}(x)\right] - P_{i+2}(x)}{6} - P_i(x)$$
$$V(ys) = \frac{-P_{i-2}(y) + 4\left[P_{i-1}(y) + P_{i+1}(y)\right] - P_{i+2}(y)}{6} - P_i(y) \qquad (4.6)$$

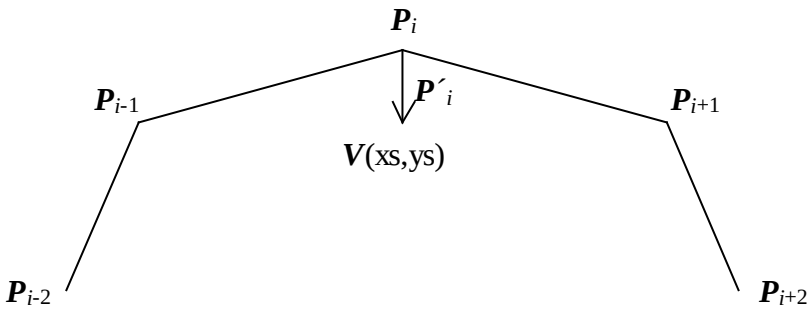
Ze znalosti velikosti úsečky $\boldsymbol{P}_i\boldsymbol{P}_{i+1}$ a hodnoty σ, která se volí, vypočítáme multiplikátor *d* takto:

$$d = \frac{\left|\boldsymbol{P}_i\boldsymbol{P}_{i+1}\right|\sigma}{\left|\boldsymbol{P}_i\boldsymbol{P}_{i+1}\right|\sigma + \left|xs\right|\left|ys\right|} \qquad (4.7)$$

Z této rovnice je zřejmé, že hodnotu *d* do značné míry ovlivňuje velikost úsečky $\boldsymbol{P}_i\boldsymbol{P}_{i+1}$. Proto delší úseky mají větší vliv na posun prostředního bodu $\boldsymbol{P}_i$. Nově spočítaný bod $\boldsymbol{P}'_i$ získáme ze vztahu:

$$P'_i(x) = P_i(x) + xs \cdot d$$
$$P'_i(y) = P_i(y) + ys \cdot d \qquad (4.8)$$

Volbou σ lze měnit míru vyhlazení. Pro $\sigma = 0$ nedochází k žádné změně původních a nově vypočtených bodů, neboť $d = 0$. Naopak, čím dosahuje σ větší hodnoty, tím dochází k většímu posunu bodu $\boldsymbol{P}_i$ ve směru vektoru $\boldsymbol{V}$. Uzavřené vrstevnice se vyhlazují tímto způsobem cyklicky přes počáteční a koncový bod.



Obr. 4.6: Schéma váhového průměrování pro druhý krok metody normal smooth

Složitost druhého kroku algoritmu je *O*(*N*), kde *N* značí počet segmentů vrstevnic.

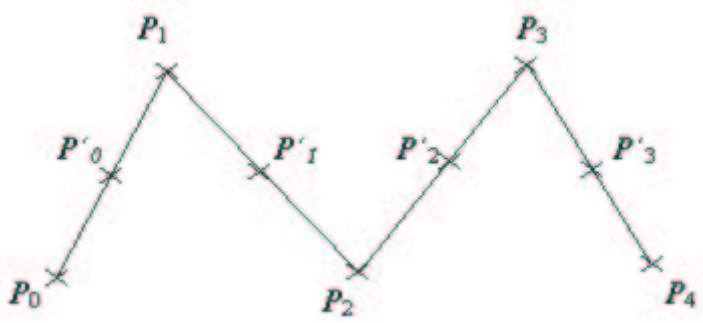
(Parametr sigma σ je v programu nastavitelný v okně **Interpol_CL** v kolonce **Factor of smooth)**.

Metoda Normal smooth je navržena pro vyhlazování polygonů s větším počtem bodů (řádově desítky a sta), proto je vhodná pro vyhlazení vrstevnic počítaných z větších trojúhelníkových sítí. Naopak pro polygony s velmi malým počtem bodů (řádově jednotky) se metoda nechová příliš korektně. Celkové složitost algoritmu je $O(N)$, N je počet segmentů vrstevnic.

4.2.2 Druhá metoda (B-spline smooth)

Při implementaci druhé vyhlazovací metody jsme testovali matematické křivky uváděné v podkapitole 3.2.3. Nakonec jsme se rozhodli pro metodu založenou na B-spline. Algoritmus této metody nebyl poskytnut vedoucím diplomové práce a je dílem autora.

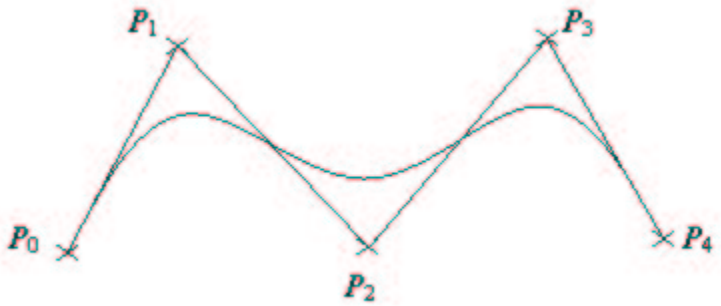
K této metodě nás vedly dobré vlastnosti aproximační křivky B-spline (viz podkapitola 3.2.3). Zvláště pak snadné a efektivní naprogramování daného předpisu této křivky a snadné napojování kubických oblouků. Ve prospěch této metody hovoří i Chaikensův vyhlazovací algoritmus (viz podkapitola 3.2.2), jenž se k této aproximační křivce blíží. Původní lomenou čáru jsme vzali jako základ pro řídící polygon pro B-spline. Protože však tato křivka je aproximační, nikoli interpolační, a dochází ke spíše větším odchylkám od původního polygonu, snažili jsme se o částečné zpřesnění průběhu hladké křivky. Zpřesnění spočívá v tom, že jsme původní polygon opět zhustili body. Na rozdíl od předchozí metody tyto nové body leží přímo na lomené čáře, vždy ve středu úseček mezi body původního polygonu (viz obr. 4.7). Na obrázku jsou tyto přidávané body značeny čárkovaně. Z vlastností uvedených v odstavci 3.2.3 věnovanému B-spline vyplývá, že aproximační křivka prochází nově přidanými (čárkovanými) body (středy úseků polygonu). Tím se tato křivka znatelně přiblížila původnímu polygonu, avšak ani teď neprochází jejími vrcholy. Připomínáme opět, že první a poslední body polygonu musí být tzv. trojnásobné body, aby aproximační křivka začínala, resp. končila v nich.



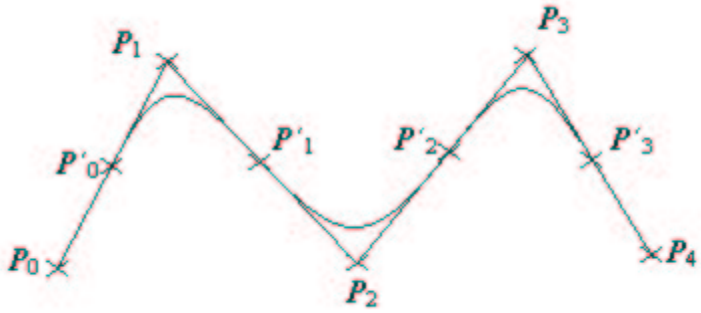
Obr. 4.7: Polygon proložený zhušťovacími body

Obrázek 4.8 ukazuje aproximaci původní lomené čáry pomocí B-spline s počátečním a koncovým trojnásobným bodem. Další obrázek (obr. 4.9) znázorňuje stejný polygon s přidanými body.

Aproximační křivka je u této metody naprogramována s neuniformní parametrizací s ohledem na jednotlivé délky úseků polygonu. Je možné volit počet bodů připadající na jeden oblouk kubické B-spline křivky a tím regulovat hustotu bodů na celkové vyhlazené křivce. (V programu se tento parametr volí v okně **Interpol_CL** v položce Value of step). Čím je parametr větší, tím jsou body na vyhlazené křivce umisťovány hustěji a křivka se jeví hladší.



Obr. 4.8: Aproximace nezhuštěného polygonu



Obr. 4.9: Aproximace zhuštěného polynomu

Složitost algoritmu je $O(N)$, kde N je počet segmentů vrstevnic.

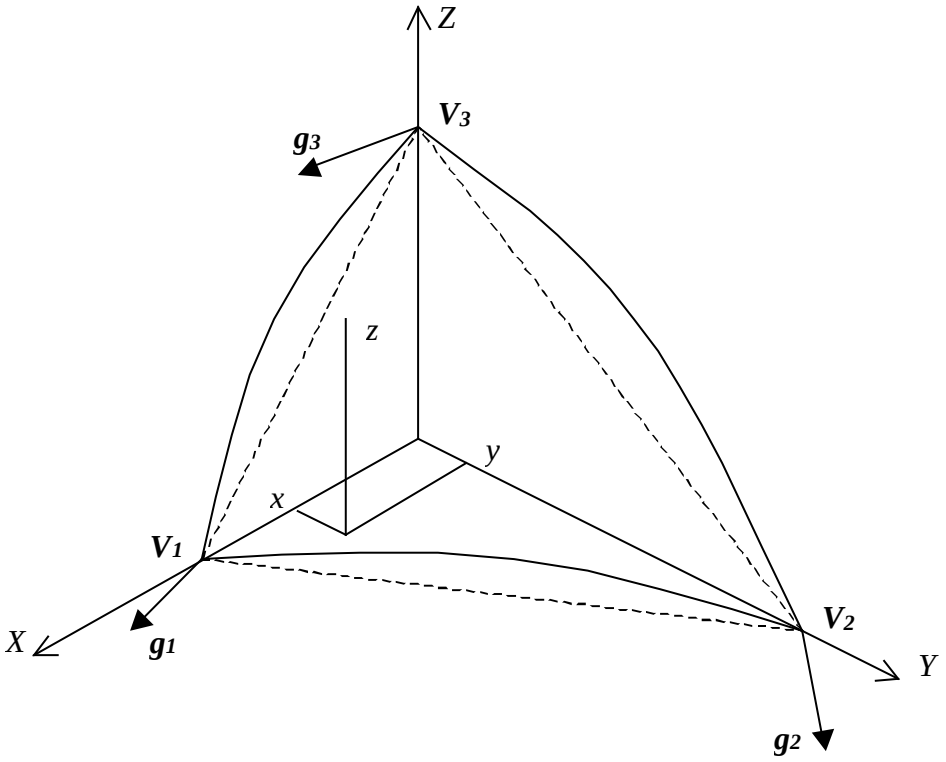
Zde možná vzniká otázka, proč nebyly použity racionální křivky (např. NURBS), které dokáží přiblížit křivku za pomoci vah k polygonu samy, bez zhušťovacích bodů. Kdybychom však použili tyto křivky a konstantně zvýšili váhy u všech bodů původního polygonu (bez zhušťovacích bodů), nemělo by to žádný efekt, protože každý bod polygonu by opět ovlivňoval křivku stejnou měrou. Křivka by měla stejný tvar jako pro váhy 1.

Abychom docílili lepšího přiblížení křivky k polygonu, museli bychom původní polygon nějakým způsobem zhustit pomocí bodů a váhy u jednotlivých bodů nastavit nerovnoměrně. Navíc tyto křivky jsou náročnější na výpočet, tedy i na čas potřebný ke zpracování.

4.3 Nelineární interpolace

Jak jsme uvedli v podkapitole 3.3, tento problém je dosti náročný a ne zcela matematicky triviální. Nejprve jsme chtěli v programu implementovat Bézierovy trojúhelníkové pláty, avšak zde narážíme na problém určení řídících bodů plátu, zvláště pak prostředního $\boldsymbol{P}_{111}$. Nakonec jsme se rozhodli zvolit Zienkiewiczovu metodu nelineární interpolace. Vše, co je uvedeno k této metodě, je získáno z algoritmu poskytnutého vedoucím diplomové práce.

Zienkiewiczova metoda nelineární interpolace trojúhelníků v trojúhelníkové síti spočívá v určení výšky jakéhokoliv bodu ležícího uvnitř trojúhelníka pomocí kubické interpolace. K tomu, aby metoda pracovala korektně, musíme znát polohu vrcholů trojúhelníka, gradienty v těchto vrcholech a samozřejmě *x*-ovou a *y*-ovou souřadnici vypočítávaného bodu (viz obr.4.10).



Obr. 4.10: Vyklenutý trojúhelník pro Zienkiewiczovu metodu

Platí:

$$z = F(\boldsymbol{V}_1, \boldsymbol{V}_2, \boldsymbol{V}_3, \boldsymbol{g}_1, \boldsymbol{g}_2, \boldsymbol{g}_3, x, y) \tag{4.9}$$

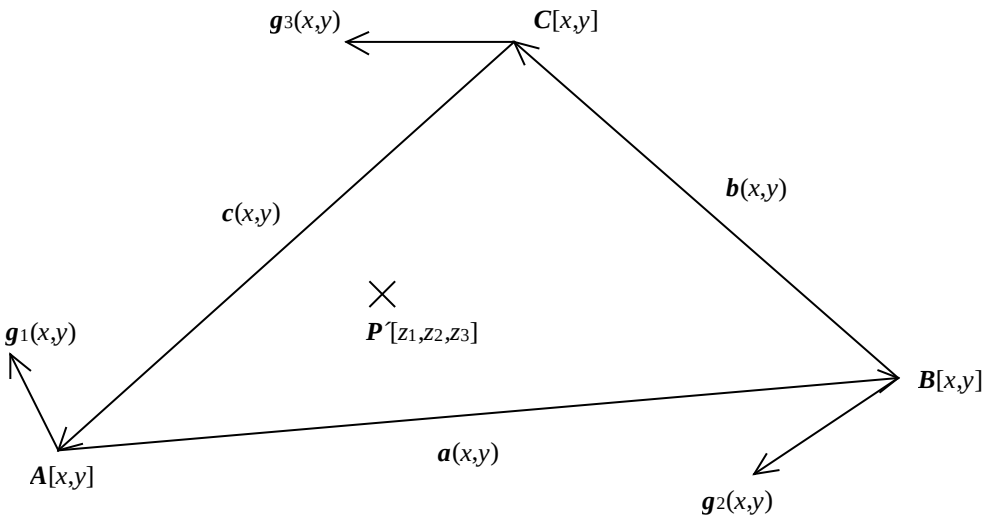
kde $\boldsymbol{V}_1$, $\boldsymbol{V}_2$, $\boldsymbol{V}_3$ jsou vrcholy trojúhelníka, $\boldsymbol{g}_1$, $\boldsymbol{g}_2$ a $\boldsymbol{g}_3$ gradienty v nich a *x*, *y* jsou souřadnice, ve kterých požadujeme vypočítat výšku. Gradienty ve vrcholech trojúhelníka závisí na okolních trojúhelnících, které incidují s tímto vrcholem. Tím metoda zohledňuje trend terénu.

Gradientem rozumíme maximální poměr změny výšky. Má dvě složky – poměr změny výšky v ose x $\frac{\delta z}{\delta x}$ a poměr změny výšky v ose y $\frac{\delta z}{\delta y}$. V programu je počítán z normálového vektoru plochy trojúhelníka. Celkový gradient ve vrcholu, jak už jsme naznačili, je vypočítán váženým průměrem z gradientů trojúhelníků incidujících s vrcholem. Jako váhy jsou použity obsahy jednotlivých trojúhelníků. Tím je zaručeno to, že větší trojúhelníky mají větší vliv na směr celkového gradientu ve vrcholu. Celkový gradient v jednom vrcholu se tedy vypočítá podle vzorce:

$$\boldsymbol{g} = \frac{\sum_{i=1}^{n} S_i \cdot \boldsymbol{g}_i}{\sum_{i=1}^{n} S_i} \tag{4.10}$$

kde S_i je obsah trojúhelníka *i*, $\boldsymbol{g}_i$ je vektor jeho gradientu a *n* je počet trojúhelníků incidujících s vrcholem.

Pro představu dalšího výkladu uvádíme obrázek 4.11, který představuje zpracovávaný trojúhelník promítnutý do roviny *xy*. Kvůli lepší orientaci v textu jsme upustili od označení vrcholů pomocí indexů a označujeme je ***ABC***.



Obr. 4.11: Zpracovávaný trojúhelník promítnutý do roviny *xy*

Na obrázku 4.11 pro dvourozměrné vektory $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$ platí:

$$\begin{aligned} \boldsymbol{a} &= \boldsymbol{B} - \boldsymbol{A} \\ \boldsymbol{b} &= \boldsymbol{C} - \boldsymbol{B} \\ \boldsymbol{c} &= \boldsymbol{A} - \boldsymbol{C} \end{aligned} \tag{4.11}$$

Výšku z daného bodu vypočítáme podle vzorce:

$$\begin{aligned} z = {} & u_1\left[z_1^2(3-2z_1)+k_1\right]+u_2(z_1^2\cdot z_2+k_2)-u_3(z_1\cdot z_2^2+k_2)+ \\ & u_4\left[z_2^2(3-2z_2)+k_1\right]+u_5(z_2^2\cdot z_3+k_2)-u_6(z_2\cdot z_3^2+k_2)+ \\ & u_7\left[z_3^2(3-2z_3)+k_1\right]+u_8(z_3^2\cdot z_1+k_2)-u_9(z_3\cdot z_1^2+k_2) \end{aligned} \tag{4.12}$$

kde z_1, z_2, z_3 jsou barycentrické souřadnice počítaného bodu. Dále platí:

$$\begin{aligned} k_1 &= 2z_1z_2z_3 \\ k_2 &= \frac{z_1z_2z_3}{2} \end{aligned} \tag{4.13}$$

a koeficienty $u_1 \ldots u_9$ se určí podle vzorců:

$$\begin{aligned} u_1 &= A_z & u_4 &= B_z & u_7 &= C_z \\ u_2 &= \boldsymbol{a}\cdot\boldsymbol{g_1} & u_5 &= \boldsymbol{b}\cdot\boldsymbol{g}_2 & u_8 &= \boldsymbol{c}\cdot\boldsymbol{g}_3 \\ u_š &= \boldsymbol{a}\cdot\boldsymbol{g}_2 & u_6 &= \boldsymbol{b}\cdot\boldsymbol{g}_3 & u_9 &= \boldsymbol{c}\cdot\boldsymbol{g_1} \end{aligned} \tag{4.15}$$

Tímto způsobem tedy dokážeme vypočítat *z*-ovou souřadnici v jakémkoliv vnitřním bodě trojúhelníka. Pomineme-li vektory vrcholů a gradientů, dostáváme funkční závislost :

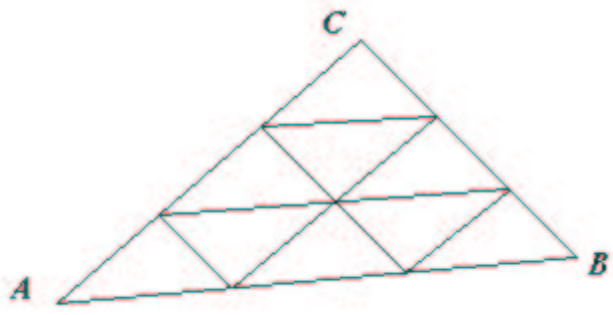
$$z = f(x, y) \tag{4.16}$$

Pro výpočet vrstevnice však požadujeme inverzní funkci. Potřebujeme získat *x*-ovou a *y*-ovou souřadnici při znalosti výšky *z*.

$$[x, y] = f^{-1}(z) \tag{4.17}$$

Tento problém však nelze vyřešit exaktním funkčním předpisem, proto jsme byli nuceni použít interpolační metodu (metoda byla odvozena díky konzultaci s RNDr. J. Semančíkem, studentem doktorandského studia na KIV). Uvažovali jsme o dvou způsobech realizace. V obou uvažovaných metodách jde o to, rozdělit nějakým způsobem zpracovávaný trojúhelník na menší oblasti. Tím získáme *x*-ové a *y*-ové souřadnice vnitřních bodů trojúhelníka. V těchto vnitřních bodech se spočítají výše uvedenou Zienkiewiczovou metodou výšky, které se lineárně pospojují úsečkami. Tímto způsobem vznikne nad trojúhelníkem lineární síť, ve které již dokážeme lineární interpolací spočítat průsečíky na hranách v požadované výšce. Čím budou oblasti uvnitř trojúhelníka menší, tím získáme hustší síť bodů a aproximace bude přesnější.

1. Zpracovávaný trojúhelník rozdělíme na pravidelné oblasti trojúhelníkového tvaru podle obrázku 4.12. Oblasti trojúhelníkového tvaru jsou v tomto případě výhodné, neboť pro výpočet vrstevnice je možno použít již známých a popsaných metod.



Obr. 4.12: Dělení trojúhelníka na pravidelné oblasti

Když jsme však tuto metodu testovali, objevily se nečekané problémy, které vyvstávají z podstaty této metody. Protože trojúhelník ***ABC*** je vyklenutý, může jedna vrstevnice konstantní výšky tento trojúhelník protínat více než jednou. Tím jsme se dostávali do problémů a docházelo k tzv. „zacyklení“ programu. K vyřešení tohoto problému by bylo vhodné zhuštění celé TIN struktury takto popsaným způsobem, avšak to by znamenalo značný nárůst dat a modifikování tabulek trojúhelníků, vrcholů i sousedů. Proto jsme od této metody upustili a použili jsme druhou, jednodušší.

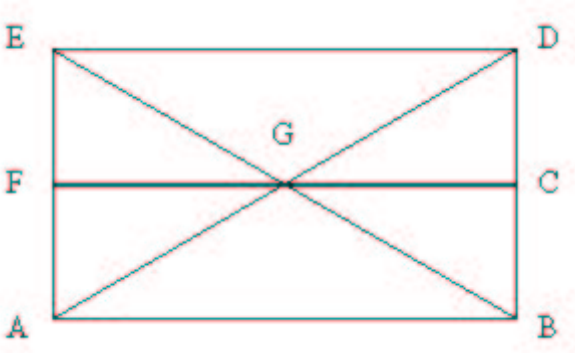
2. V této interpolaci jsme dělili a vypočítávali výšky Zienkiewiczovou metodu pouze na okrajových hranách trojúhelníku ***ABC***, jak ukazuje obrázek 4.13. Vypočtené výšky jsme lineárně pospojovali a vypočítali lineární interpolací průsečík vrstevnice s příslušným úsekem. Tímto způsobem jsme získali průsečíky na těch hranách trojúhelníka ***ABC***, které daná vrstevnice protíná. Tyto dva průsečíky jsme proložili úsečkou (vrstevnicí).

Obr. 4.13: Dělení hrany trojúhelníka

Z popsaného postupu je zřejmé, že průsečíky na hranách byly získány z vyklenutého trojúhelníka ***ABC***, kdežto průběh vrstevnice uvnitř trojúhelníka má lineární charakter.

Tento způsob interpolace zajisté nedává tak přesné výsledky, jak by tomu bylo u prvního způsobu, avšak je přesnější než pouhá lineární interpolace. Navíc je zde splněna zásada, že daná vrstevnice může jeden trojúhelník protínat nejvíce jedenkrát.

V následujících příkladech ukážeme, jak se vybraná metoda chová v konkrétní trojúhelníkové síti. Síť je znázorněná na obrázku 4.14, kde je silně vyznačen zkoumaný profilový řez. U bodů sítě jsme modifikovali *z*-ové souřadnice a sledovali chování zkoumaného profilu.



Obr. 4.14: Testovací síť pro Zienkiewiczovu metodu

Rovinné souřadnice bodů sítě jsou uvedeny v tabulce tab. 4.1. Výškové souřadnice jsou uváděny u jednotlivých příkladů. Body na profilových křivkách v jednotlivých příkladech znázorňují vypočtené hodnoty Zienkiewiczovou metodou. Tyto body jsou pak lineárně pospojovány do profilové křivky.

	A	B	C	D	E	F	G
X	-1.0	1.0	1.0	1.0	-1.0	-1.0	0.0
Y	-1.0	-1.0	0.0	1.0	1.0	0.0	0.0

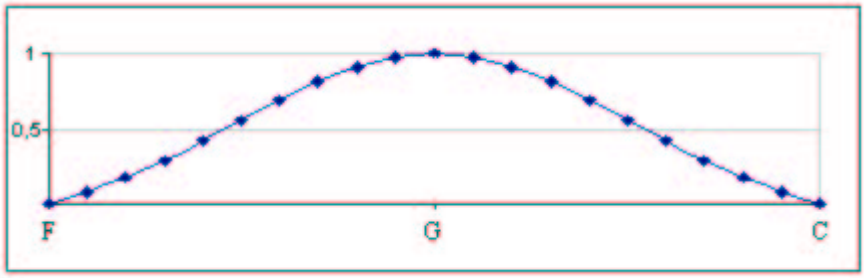
Tab. 4.1

Příklad 1.

V tomto příkladě jsme sledovali jak se daná metoda chová v konvexních případech, tj. interpolace vrcholku kopce. U prostředního bodu sítě (G) jsem volili výšku rovnu 1, u ostatních bodů pak byla výška rovna 0.

	A	B	C	D	E	F	G
Z	0.0	0.0	0.0	0.0	0.0	0.0	1.0

Tab. 4.2



Obr. 4.15: Profil 1

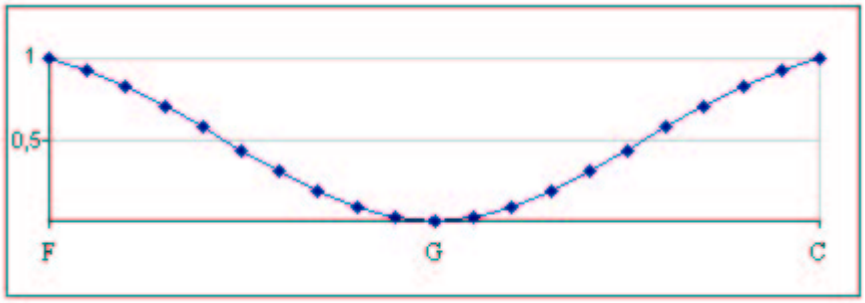
Z obrázku 4.15 lze vidět, že napojení dvou oblouků ve vrchole (bod G) je hladké a že se metoda v těchto situacích chová očekávaným způsobem.

Příklad 2.

V druhém příkladě jsme naopak sledovali chování metody v oblastech konkávních, tj. interpolace dna údolí. Pro tento příklad jsme volili výšku prostředního bodu (G) rovnu 0 a výšky všech ostatních bodů rovny 1.

	A	B	C	D	E	F	G
Z	1.0	1.0	1.0	1.0	1.0	1.0	0.0

Tab. 4.3



Obr. 4.16: Profil 2

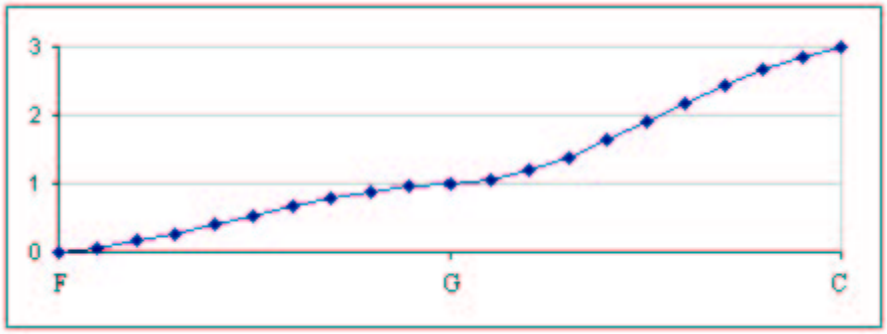
I v tomto případě je napojení oblouků v bodě G hladké a výsledek metody se jeví korektní.

Příklad 3.

V tomto příkladě jsme sledovali chování metody v situacích, kdy se mění skon terénu. Mezi body sítě F a G je sklon terénu roven 45°, zatímco mezi body G a C je sklon roven 63,4°.

	A	B	C	D	E	F	G
Z	0.0	0.0	3.0	0.0	0.0	0.0	1.0

Tab. 4.4



Obr. 4.17: Profil 3

Na obrázku 4.17 lze vidět, že v místě změny spádu, tj. v bodě G, dochází k hladkému napojení, avšak vzniká zde singulární bod.

Z příkladů lze vidět, že Zienkiewizcova interpolační metoda se chová očekávaným způsobem a že dobře vystihuje trend daného terénu.
Složitost algoritmu je $O(N)$, kde N je počet trojúhelníků.

4.4 Popis vrstevnice – kótování

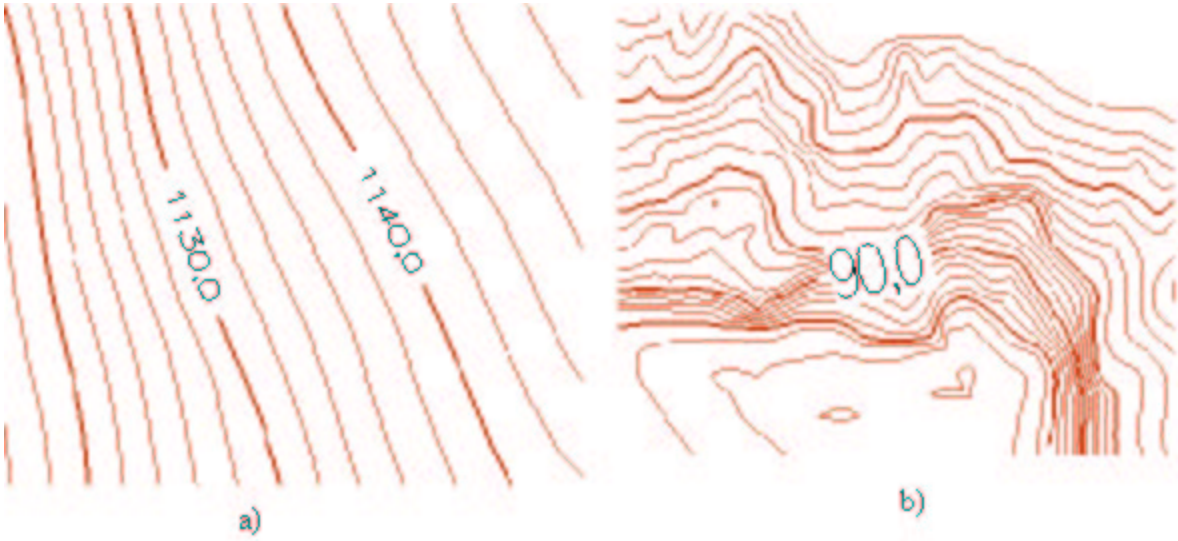
Kótování vrstevnic v kartografii je důležité, abychom mohli zjistit výšku vrstevnice. Zásady z klasické kartografie říkají, že se většinou kótují vrstevnice základní, obvykle s výškou odpovídající pětinásobku základního intervalu (každá „pátá“ vrstevnice). Pro umístění popisu vrstevnice platí několik zásad:

- Popis vrstevnic se umisťuje rovnoměrně a nepravidelně v rámci daného území
- Popis vrstevnic se vždy umisťuje tak, aby byl čitelný při pohledu ve směru stoupání (do svahu)
- Popis vrstevnic se umisťuje tam, kde nenarušuje důležitý průběh vrstevnice
- Popis se neumisťuje v místech příliš velkého spádu terénu, kde jsou vrstevnice příliš blízko sebe, aby je svým textem nenarušil

Jak lze vidět z uvedeného, umístit dobře popis na vrstevnici není zcela jednoduchá záležitost a vyžaduje to praxi zkušeného kartografa. O to hůře se umisťují popisy vrstevnic automaticky, kdy o vhodném místu rozhoduje pouze počítač na základě nějaké váhové funkce. Praxe ukázala, že je to v podstatě neřešitelný problém, a většina profesionálních softwarů umožňuje uživateli editaci automaticky umístěného popisu.

V této práci řeší problém umístění popisů na vrstevnice algoritmus, který se snaží zohlednit některé výše uvedené zásady. Algoritmus je sestaven tak, aby vyhledával

potencionální místo pro popis v určité vzdálenosti od okraje daného území. Dále algoritmus zajišťuje, aby popis nezakrýval důležité charakteristiky vrstevnice a byl umístěn v určité vzdálenosti od nich a aby byl správně orientován a natočen. Stává se, že v některých případech program umisťuje popisy tam, kde by je zkušený kartograf nikdy neumístil. Například program nezohledňuje sklon terénu a umisťuje kóty do míst, kde jsou vrstevnice kresleny hustě vedle sebe a jsou pak popisem ořezávány. Na obrázku 4.15a) je popis vrstevnice algoritmem umístěn správně, zatímco na obrázku 4.15b) je popis vrstevnice umístěn nevhodně.



Obr. 4.15: Umístění popisu na vrstevnici a) správně, b) nevhodně

Popis vrstevnice je v programu definován svým referenčním bodem, tj. bodem umístěným na boxu, který ohraničuje text popisu, a natočením φ vůči *x*-ové ose (viz obr. 4.16). Referenční bod leží na přední hraně boxu ve směru čtení popisu uprostřed. Tímto boxem se také ořezávají okolní vrstevnice, jsou-li příliš blízko sebe a zasahují do boxu. V programu se popisy vrstevnic zaokrouhlují na jedno desetinné místo, protože jsou používána i syntetická data blízká nule.

Obr. 4.16: Umístění a orientace popisu

Řešením pro nesprávně umístěné popisy vrstevnic by bylo vytvoření editačního programu, který by umožňoval rušení nebo posun takto nevhodně umístěných popisů. To však je nad rámec této práce.

5. Implementace

Programové produkty této práce byly vytvořeny a odladěny v objektově orientovaném programovacím jazyce PASCAL ve vývojovém prostředí Borland Delphi verze 5.0. Jedním z důvodů byla autorova znalost tohoto jazyka, dalším důvodem byl požadavek o začlenění produktů do MVE, který pracuje výhradně na platformě Microsoft Windows. Pro vykreslení a zobrazení dat bylo použito knihovny OpenGL. Dané algoritmy byly realizovány jako knihovny DLL pro systém MVE [viz MVE].

5.1 Integrace do MVE

MVE je zkratka pro Modular Visualization Environment [viz MVE]. Jedná se o systém, který je vyvinut a neustále udržován v centru počítačové grafiky a vizualizace dat na katedře informatiky a výpočetní techniky na Západočeské univerzitě v Plzni. Jak již název napovídá, jedná se o systém zahrnující vizualizačně orientované knihovny nejrůznějších modulů a jednoduchý runtime editor, který umožňuje uživatelům interaktivní spojování a spouštění těchto modulů. Modulem se rozumí „obyčejný“ program, který produkuje nějakou činnost. Modul například generuje či nahrává data ze souboru, nebo je naopak ukládá, výpočetní moduly zpracovávají vstupní data atd. V nejjednodušším případě pouhým pospojováním výstupů jednoho modulu se vstupy jiných modulů vytvoří uživatel „aplikaci“ řešící daný problém. Programátor nastupuje teprve, když je třeba implementovat nový algoritmus.

Systém programátorům poskytuje možnost jednoduchého zabudování vlastních částí mezi již existující celky. Z programátorského hlediska je MVE řešeno jako systém dynamických knihoven pod operačním systémem Microsoft Windows. V jedné knihovně může být umístěno i několik modulů, přičemž každý modul je vytvořen pro svoji konkrétní činnost. Knihovny komunikují s editorem pomocí jednoduchého rozhraní několika jasně definovaných funkcí. Implementace algoritmu spočívá tedy ve vytvoření jednoho či více modulů, které jsou umístěny v jedné nebo více DLL knihovnách.

Realizovali jsme celkem čtyři moduly, které jsou napsány ve třech knihovnách. Knihovna **Contour_Line_Loader** obsahuje modul **Loader_CL**, který načítá data ze souboru a ukládá je do datové struktury *Triangle*. V knihovně **Contour_Line** se nacházejí moduly **Calcul_CL** a **Interpol_CL**. Modul **Calcul_CL** realizuje výpočet vrstevnic z trojúhelníkové sítě lineární interpolací a Zienkiewczovou metodou a ukládá je do datové struktury *TContour*. Tato datová struktura je speciálně navržená na uložení vrstevnic.

Modul **Interpol_CL** je naprogramován na vyhlazování lomového průběhu vrstevnice zmiňovanými postupy. Vstupem do modulu jsou vypočtené vrstevnice (*TContour*), výstupem je shodná avšak modifikovaná datová struktura. Poslední knihovna **Contour_Line_Render** obsahuje modul **Render_CL** realizující vykreslení a vizualizaci vrstevnic. Vstupem do tohoto modulu je datová struktura obsahující vrstevnice.

5.2 Popis datových struktur a výměnných formátů

Datová struktura *Triangle*:

V této struktuře jsou uloženy informace o trojúhelníkové síti a je jí používáno ke komunikaci mezi moduly **Loader_CL** a **Calcul_CL** nebo **DTlib** a **Calcul_CL**. Obsahuje tyto informace: minimální a maximální *x*-ovou, *y*-ovou a *z*-ovou souřadnici daného území, tabulku souřadnic vrcholů, tabulku seznamu vrcholů a tabulku seznamu sousedních trojúhelníků, velikosti těchto tabulek, tabulku s normálovými vektory jednotlivých trojúhelníků a další potřebné informace. I když ve struktuře mohou být zakódovány informace o jednotlivých trojúhelníkách a jejich hranách (soft a hard lines), této skutečnosti nevyužíváme z důvodu nedostupnosti testovacích dat. Programová definice struktury je uvedena v příloze B.

Datová struktura *TContour*:

Tato struktura je navržena a realizována pro uložení vrstevnic vypočtených z TIN. Tato struktura slouží k přenosu informací mezi moduly **Calcul_CL**, **Interpol_CL** a **Render_CL**. Aby nedocházelo ke ztrátě důležitých informací o síti, obsahuje stejné údaje jako datový typ *Triangle* a navíc uchovává data o vrstevnicích. U každé samostatné linie (vrstevnice) se uchovává výška, minimální a maximální *x*-ová a *y*-ová souřadnice, počet bodů, které linii tvoří a jejich rovinné souřadnice, informace o popisku a příznaky linie. Příznaky nesou informaci zda je vrstevnice hlavní, je-li popisována a je-li uzavřená nebo otevřená. Informace o popisku obsahují souřadnice referenčního bodu, text popisku, úhel natočení, indexy bodů na vrstevnici, mezi kterými popis leží a souřadnice rohů boxu, v kterém popis leží. Programová definice struktury je uvedena v příloze B.

Popis vstupního souboru pro modul **DTlib**:

Soubor je textový a obsahuje počet bodů, jejich třírozměrné souřadnice, počet trojúhelníků a seznam vrcholů. Počet trojúhelníků a seznam vrcholů však není povinný a nemusí být součástí souboru. Příklad je uveden v příloze B.

Popis vstupního souboru pro modul **Loader_CL**:

Soubor je textový a obsahuje počet bodů, jejich souřadnice, počet trojúhelníků a seznam vrcholů a hran. Seznam hran je oddělen od seznamu vrcholů opětovným zápisem počtu trojúhelníků. Jestliže hrana trojúhelníku není sdílena s žádným jiným trojúhelníkem, je tato hrana v seznamu hran označována konstantou –1. Příklad je uveden v příloze B.

Popis výstupního souboru z modulů **Caclul_CL** a **Interpol_CL**.

Výstupní soubor z těchto modulů má shodnou syntaxi. Na prvním řádku se udává konstanta XYZ. Následuje blok souřadnic jednotlivých bodů vrstevnice, který začíná identifikátorem vrstevnice a končí konstantou END. Souřadnice jsou oddělovány čárkou. Je-li vrstevnice uzavřená, opakuje se první bod linie též na konci bloku. Za posledním blokem souřadnic se opět vyskytuje konstanta END. Příklad souboru je uveden v příloze B.

K zvolení tohoto formátu nás vedl fakt, že takto strukturovaný soubor může být použit jako vstupní do softwaru ArcView firmy ESRI. V programu ArcView je třeba pouze nahrát příslušný modul, který to dovede . Modul se v systému ArcView jmenuje Generate to Shape v5.0 a je volně přístupný na internetových stránkách firmy ESRI v souboru G2sv50 [viz ESRI]. Tento modul je též součástí přiloženého CD.

6. Testy uvedených algoritmů

V této kapitole provedeme testování jednotlivých použitých algoritmů. Budeme sledovat jejich chování při změnách v nastavení parametrů a zkoumat jejich časovou náročnost pro různě velká vstupní data. Uvedeme příklady a pokusíme se odůvodnit chování metod. Testy časové náročnosti jsou ovlivněny stejnými faktory u všech algoritmů, proto se o nich zmíníme nyní. U každé metody je uvedena pouze tabulka s naměřenými hodnotami a graf.

Časová náročnost algoritmů

Je zřejmé, že časovou náročnost algoritmů ovlivňuje několik faktorů. Prvním z nich je nepochybně velikost vstupních dat. Druhým faktorem je interval vrstevnic - čím menší interval bude, tím více času bude algoritmus potřebovat. Dalším nezanedbatelným faktorem je výšková členitost terénu. Je-li terén hodně členitý, existuje mnoho průniků terénu s danou vodorovnou rovinou řezu a počítá se mnoho segmentů vrstevnice jedné výšky. Z uvedeného vyplývá, že objektivně posoudit časovou náročnost algoritmů je velice komplikované. Nicméně jsme provedli dva testy časové náročnosti u každého algoritmu. Jeden pro syntetická data, druhý pro reálná. V testech jsme se snažili sledovat časovou závislost na velikosti vstupních dat a na intervalu výšky vrstevnice. Byl měřen pouze výpočetní čas, tzn. operace jako ukládání a čtení z disku se do časového testu nezahrnovaly. Syntetická data pro test časové náročnosti byla získána z modulu **DTlib** volbou funkce – *sum of several exp*, pro různě velký počet bodů sítě. Rozsah výšek bodů byl pro všechny sítě se syntetickými daty stejný, a to 0 – 1,21. Jako reálná data byly použity trojúhelníkové sítě modelující skutečný povrch Země. Trojúhelníkové sítě s reálnými daty měly různý počet bodů s rozdílnou členitostí a rozsahem výšek (viz Tab.6.1).

Každé měření proběhlo minimálně třikrát a výsledný čas byl vypočten aritmetickým průměrem. Hodnoty naměřených časů v níže uvedených tabulkách jsou uváděny v milisekundách. Časové testy byly prováděny na „notebooku“ firmy IBM s procesorem Pentium II o rychlosti 333MHz a pamětí 128M RAM. Tyto podmínky testování platily pro každý algoritmus stejně.

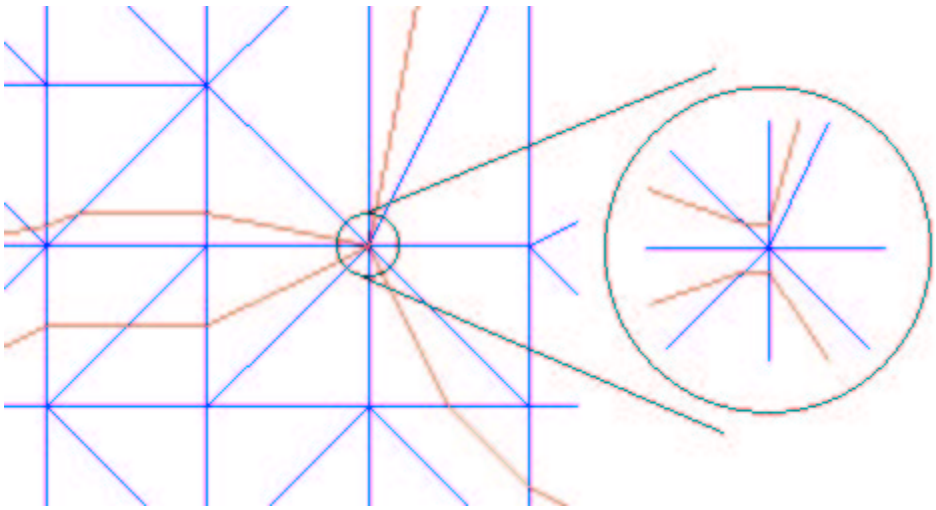
Počet bodů TIN	1 472	19 819	24 308	51 647	74 805	100 000
Min. výška TIN	1 075	800	865	865	61,32	61,32
Max výška TIN	1 155	2 360	1 250	1 250	99,12	99,12

Tab. 6.1

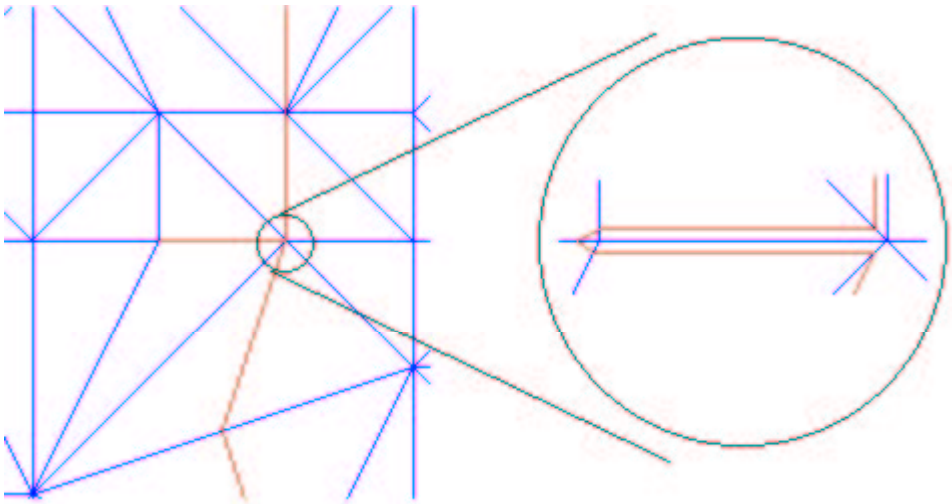
Pro všechny časové testy u všech zkoumaných algoritmů platilo, že naměřené hodnoty pro malá vstupní data (řádově tisíce) byly zatíženy značnou chybou měření. Proto tyto hodnoty jsou dosti zavádějící a nelze se o ně opírat při odvozování časové složitosti algoritmů. Nicméně jsme tyto hodnoty také uváděli v příslušných tabulkách grafech.

6.1 Test algoritmu výpočtu vrstevnic z TIN

Zde jsme testovali chování algoritmu na výpočet vrstevnic pomocí lineární interpolace bez vyhlazování a sledovali jeho časovou náročnost. Jako testovací data jsme použili jak reálná, tak syntetická data. Pro data syntetická byly výšky bodů TIN počítány matematickými funkcemi. Jak prokázaly provedené testy, algoritmus se chová bezchybně jak v rovinných oblastech sítě, tak v oblastech s většími výškovými rozdíly bodů. Z kartografického hlediska však nastávají potíže tam, kde výška bodu, resp. bodů, se rovná nebo se velmi blíží k výšce vrstevnice tj. bod/y leží v rovině řezu vrstevnice. Poté se algoritmus zachová tak, jak bylo popsáno v podkapitole 4.1. V těchto případech se zdá, že se vrstevnice dotýká sama sebe (viz obr. 6.1), nebo že vytváří „slepá ramena“ (viz obr. 6.2). Při mnohonásobném zvětšení však zjistíme, že tomu tak není. Na následujících obrázcích ukážeme vzniklé situace. Modrou barvou je znázorněna síť, hnědou vrstevnice.



Obr. 6.1: Bod ležící v rovině řezu vrstevnice, vrstevnice se zdánlivě „dotýká“ sama sebe



Obr. 6.2: Body ležící v rovině řezu vrstevnice, vytvoření „slepého ramene“

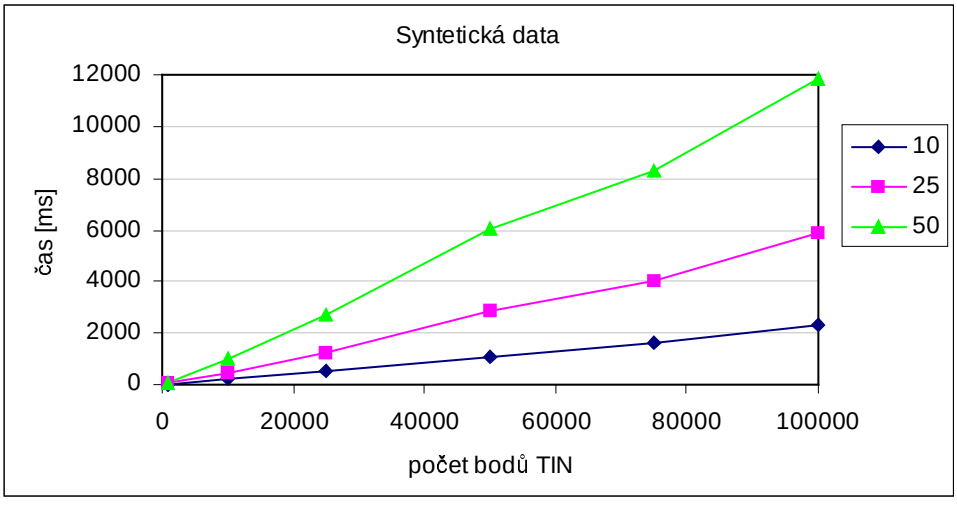
Tyto problémy by bylo možno odstranit filtrováním vypočtené vrstevnice, které by ji zbavovalo vzniklých „slepých“ ramen, nebo editačním programem, kterým by bylo možno vzniklé situace odstranit.

Časová náročnost

V tabulce 6.2 jsou hodnoty naměřených časů (uvedené v milisekundách) pro syntetická data. V tabulce 6.3 jsou uváděny hodnoty naměřených časů pro reálná data.

Syntetická data						
Počet sečných rovin	Počet bodů TIN					
	1 000	10 000	25 000	50 000	75 000	100 000
10	0	210	530	1 100	1 610	2 310
25	40	480	1 270	2 880	4 010	5 860
50	110	980	2 700	6 050	8 310	11 880

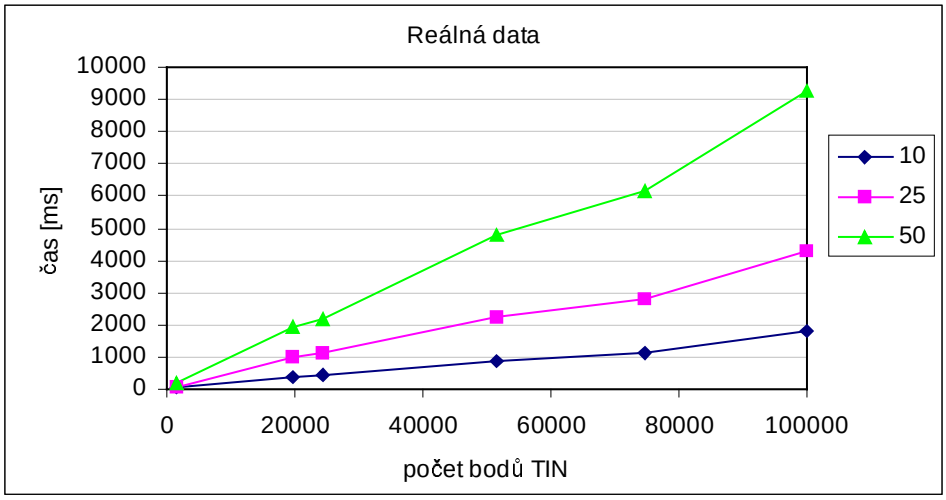
Tab. 6.2



Obr. 6.3: Časová náročnost algoritmu pro výpočet vrstevnic z TIN pro syntetická data

Reálná data						
Počet sečných rovin	Počet bodů TIN					
	1 472	19 819	24 308	51 647	74 805	100 000
10	40	380	420	870	1 130	1 830
25	50	980	1 090	2 250	2 780	4 290
50	160	1 910	2 170	4 800	6 170	9 270

Tab. 6.3



Obr. 6.4: Časová náročnost algoritmu pro výpočet vrstevnic z TIN pro reálná data

Z naměřených hodnot pro syntetická i reálná data lze usuzovat o správnosti teoretického odhadu pro lineární složitost algoritmu.

6.2 Testy vyhlazovacích algoritmů

První metoda:

Test chování metody v závislosti na volbě parametrů jsme prováděli tak, že jsme modifikovali vstupní parametry metody (hustotu bodů a míru vyhlazení) a sledovali jsme její chování pro danou trojúhelníkovou síť. Vstupní parametry spolu s metodou jsou popsány v podkapitole 4.2.1. Samozřejmě, že výstup metody značně záleží na velikosti TIN a hustotě rozmístění bodů. Pro velmi malé sítě (řádově desítky bodů) se vrstevnice vypočtené touto metodou nechovaly žádoucím způsobem a na vzniklé chyby neměla velký vliv ani volba parametrů. Testy jsme proto provedli na sítích větších, (řádově stovky, tisíce a desetitisíce bodů) znázorňujících jak syntetická, tak i reálná data. Z důvodu velkých dat a malé rozlišovací schopnosti z nich vygenerovaných obrázků jsme použili v následujících příkladech jeden testovací polygon dostatečné velikosti. V tabulkách jsou uvedeny hodnoty parametrů a barva odpovídající vygenerované linii. Na obrázku u každého příkladu je

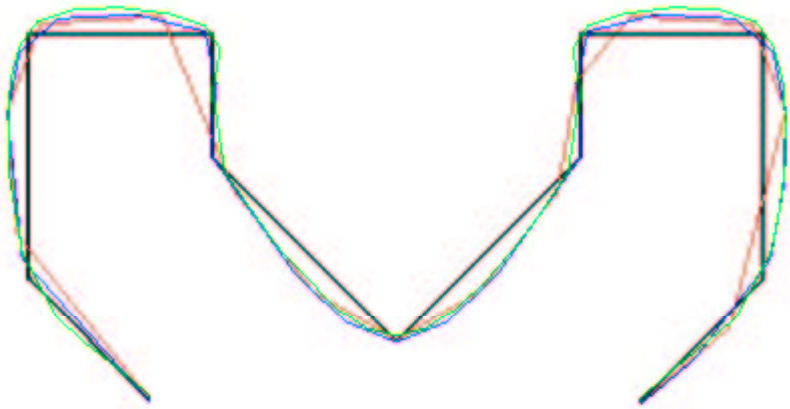
znázorněn silnou čarou původní polygon a barevně jsou značeny jednotlivé vyhlazovací křivky.

Příklad 1.

Příklad ukazuje vyhlazení polygonu pro různé hodnoty parametru *p*, parametr σ je roven 0. Z obrázku 6.5 lze vidět, že metoda poměrně dobře interpoluje daný polygon už pro hodnotu parametru *p* rovnu 1,5.

Hustota bodů (parametr *p*)	Míra vyhlazení (parametr σ)	Barva linie na obrázku
0,9	0	
1,5	0	
2,5	0	

Tab.6.4



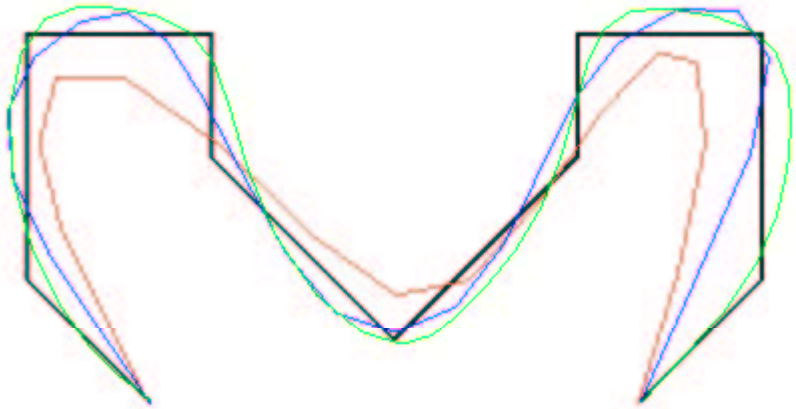
Obr. 6.5

V následujících třech příkladech modifikujeme i parametr σ, neboli míru vyhlazení křivky. Na obrázkách u příkladů je dobře vidět posun vypočtených bodů vyhlazené křivky ovlivněný okolními body (viz 2. krok metody popsaný v podkapitole 4.2.1). Též jde pozorovat, že čím menší volíme parametr *p* ovlivňující hustotu bodů na vyhlazované křivce, tím jsou vzdálenosti mezi těmito body (vypočítané v 1. kroku metody) delší a dochází k většímu posunu „dohlazovaných“ bodů vlivem druhého kroku dané metody. Obecně tedy můžeme říci, že čím menší budeme volit parametr *p*, tím bude docházet k většímu posunu bodů vyhlazené křivky vlivem druhého kroku metody, pokud parametr σ nebude roven 0.

Příklad 2.

Hustota bodů (parametr p)	Míra vyhlazení (parametr σ)	Barva linie na obrázku
0,9	0,5	
1,5	0,5	
2,5	0,5	

Tab. 6.5

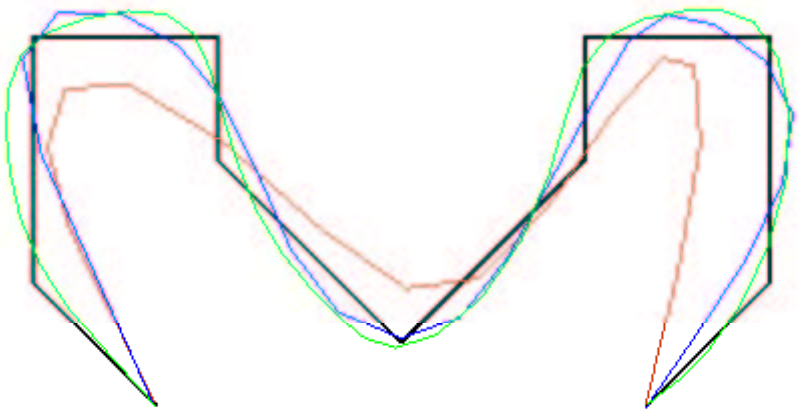


Obr. 6.6

Příklad 3.

Hustota bodů (parametr p)	Míra vyhlazení (parametr σ)	Barva linie na obrázku
0,9	1,5	
1,5	1,5	
2,5	1,5	

Tab. 6.6

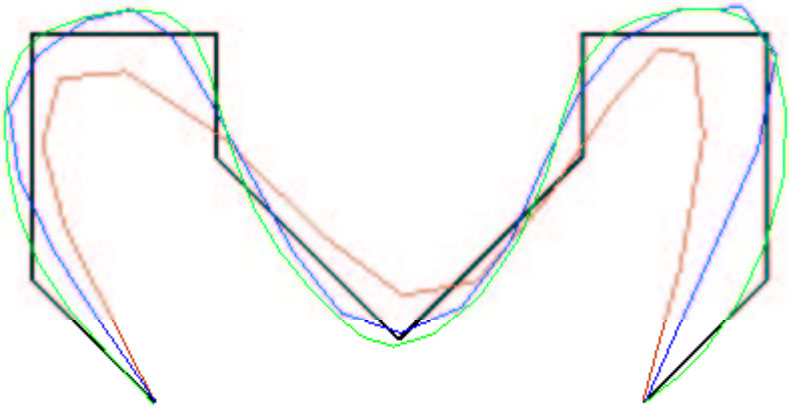


Obr. 6.7

Příklad 4.

Hustota bodů (parametr p)	Míra vyhlazení (parametr σ)	Barva linie na obrázku
0,9	2,5	
1,5	2,5	
2,5	2,5	

Tab. 6.7



Obr.6.8

Z uvedených příkladů lze snadno vyvodit pravidlo, že vygenerovaná linie je tím hladší, čím větší hustota bodů je použita, samozřejmě na úkor paměti pro ukládané body linie. Zvětšujeme-li míru vyhlazení bodů, linie se sice více vyhlazuje, ale zároveň se odchyluje od vrcholů původního polygonu. Dále lze vidět, že generovaná křivka neleží v konvexní obálce polygonu. Tím se může stát, že se jednotlivé vrstevnice mohou protínat, zadáme-li nevhodně vstupní parametry nebo jsou-li body TIN nevhodně (řídce) rozmístěny. Tato metoda se spíše hodí pro sítě s větším počtem bodů, kde vyhlazovaný polygon obsahuje body již v dostatečné hustotě z předchozího výpočtu vrstevnice.

Při volbě parametrů velmi záleží na velikosti TIN a rozmístění vrcholů v ní. Je-li síť řídká, doporučujeme volit větší hustotu bodů (parametr $p \in (2.0,4.0)$), samozřejmě s ohledem na paměť počítače. Naopak pro velké a husté sítě je zbytečné volit velkou hustotu bodů, protože už samotný polygon jich obsahuje dost. Pro tyto případy doporučujeme volit hodnotu parametru p blízkou hodnotě 1,8.

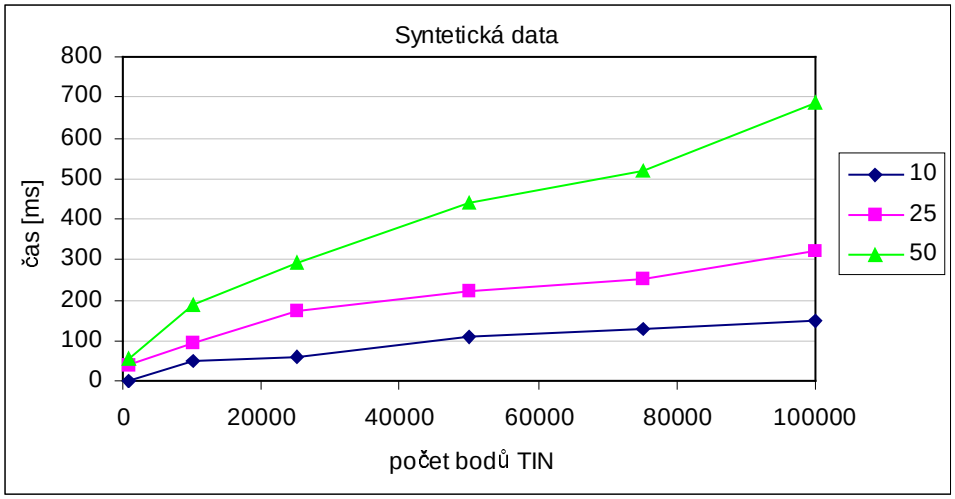
Na míře vyhlazení (parametr σ) záleží, jestli dáváme přednost přesnosti umístění vrstevnic nebo jejich estetickému vzhledu. Klademe-li důraz na přesnost, měl by být parametr míry vyhlazení z intervalu (0, 0.8). Jde-li však o estetický vzhled vrstevnice, může být tento parametr volen i větší.

Časová náročnost

V tabulce 6.8 jsou hodnoty naměřených časů (uvedené v milisekundách) pro syntetická data. V tabulce 6.9 jsou uváděny hodnoty naměřených časů pro reálná data.

Syntetická data						
Počet Sečných rovin	Počet bodů TIN					
	1 000	10 000	25 000	50 000	75 000	100 000
10	0	50	60	110	130	150
25	40	95	175	220	250	320
50	55	190	290	440	520	685

Tab. 6.8

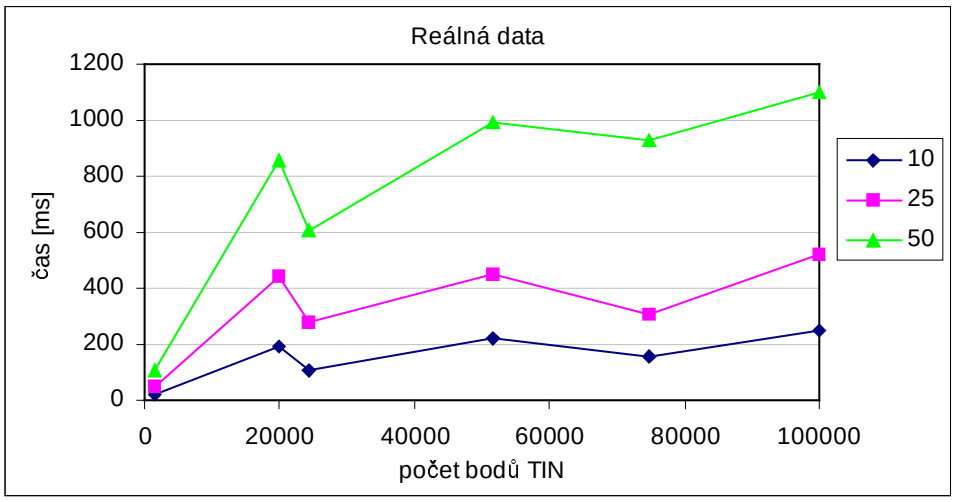


Obr. 6.9: Časová náročnost vyhlazovacího algoritmu normal smooth pro syntetická data

Jak jsme uvedli na začátku kapitoly, naměřené hodnoty pro malá vstupní data jsou zatíženy chybou a nemohou být uvažovány pro určení časové složitosti algoritmu. Pro větší vstupní data (v Tab. 6.8 od hodnoty 25 000 bodů) se algoritmus chová lineárně. To samé můžeme říci i o zkoumaných reálných datech (viz Tab. 6.9 a Obr. 6.10).

Reálná data						
Počet Sečných rovin	Počet bodů TIN					
	1 472	19 819	24 308	51 647	74 805	100 000
10	20	190	110	220	160	250
25	50	440	275	450	310	520
50	110	860	610	990	930	1 100

Tab. 6.9



Obr.6.10: Časová náročnost vyhlazovacího algoritmu normal smooth pro reálná data

Z grafu na obrázku 6.10 lze vidět, že naměřené časy pro reálná data kmitají, avšak sledují lineární trend.

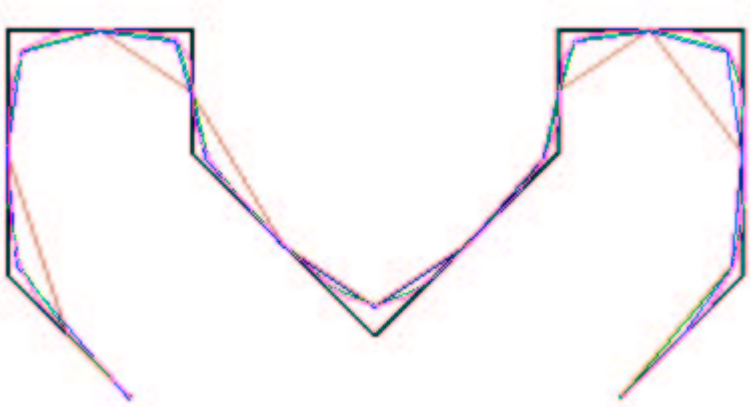
Druhá metoda:

Test druhé metody jsme opět prováděli modifikací vstupního parametru a sledovali její chování pro dané TIN. Vstupní parametr a metoda je popsána v podkapitole 4.2.2. Test byl prováděn opět pro syntetická i reálná data. Ze stejných důvodů jako u první metody je zde uvedený příklad s jedním vybraným polygonem.

Příklad 1.

Faktor vyhlazení	Barva linie
1	
2	
3	
5	

Tab.6.10



Obr.6.11

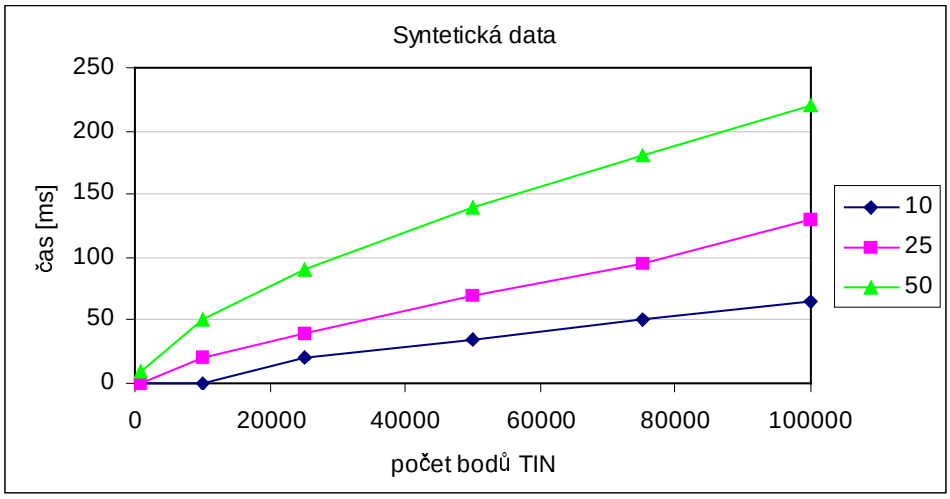
Z příkladu je patrné, že čím větší volíme faktor vyhlazení, tím dochází k lepší aproximaci původního polygonu. Tato metoda vykazuje oproti metodě předchozí lepší výsledky pro sítě řídké nebo sítě se značným sklonem.

Časová náročnost

Algoritmus jsme opět testovali pro syntetická i reálná data. V tabulce 6.11 jsou naměřené hodnoty (v milisekundách) pro data syntetická, v tabulce 6.12 pro data reálná.

Syntetická data						
Počet sečných rovin	Počet bodů TIN					
	1 000	10 000	25 000	50 000	75 000	100 000
10	0	0	20	35	50	65
25	0	20	40	70	95	130
50	10	50	90	140	180	220

Tab.6.11

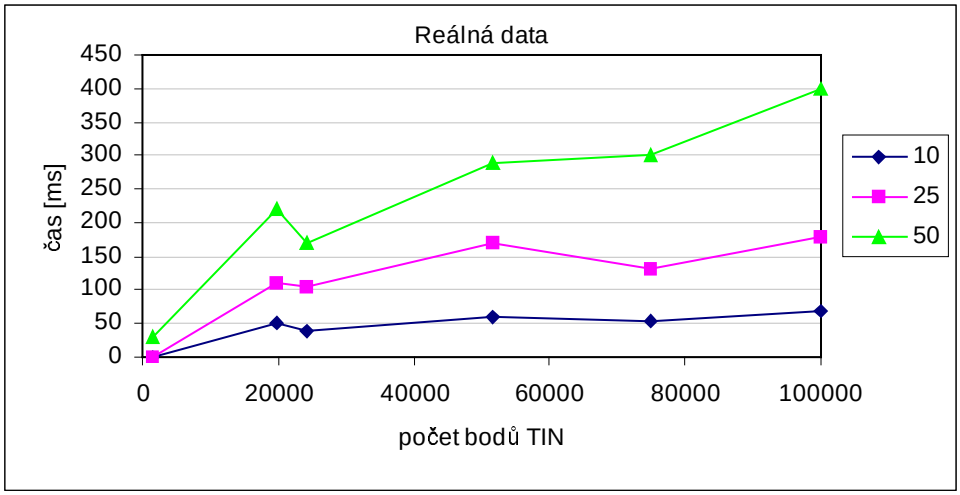


Obr. 6.12: Časová náročnost vyhlazovacího algoritmu B-spline pro syntetická data

Z tabulky 6.11 a grafu na obrázku 6.12 vyplývá, že tato metoda je dosti rychlá a pro malá vstupní data časové měřiče nezaznamenaly žádné hodnoty. Nebudeme-li brát v úvahu tyto hodnoty, které jsou zavádějící, časová složitost metody se jeví lineárně.

Reálná data						
Počet sečných rovin	Počet bodů TIN					
	1 472	19 819	24 308	51 647	74 805	100 000
10	0	50	40	60	55	70
25	0	110	105	170	130	180
50	30	220	170	290	300	400

Tab. 6.12

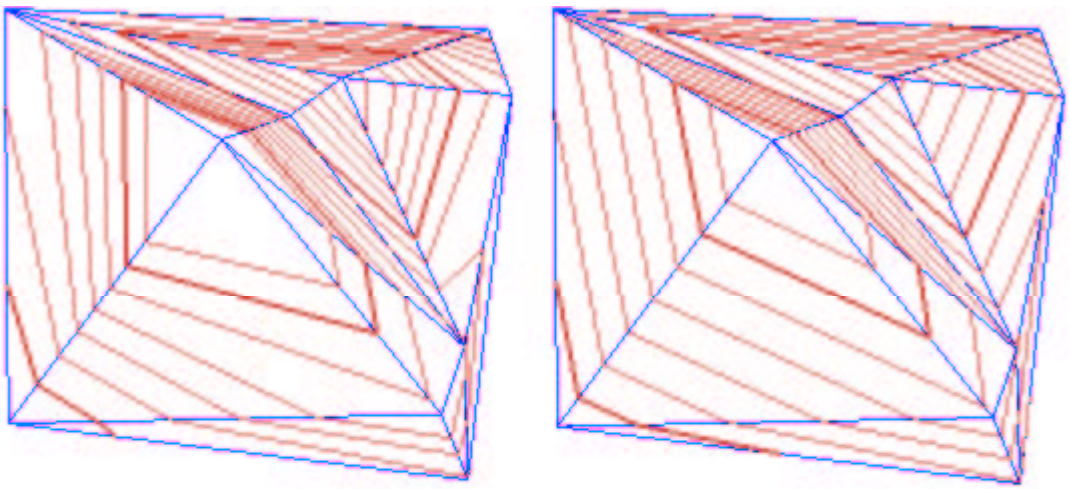


Obr. 6.13: Časová náročnost vyhlazovacího algoritmu B-spline pro reálná data

Naměřené hodnoty pro reálná data sledují lineární trend, pomineme-li hodnoty pro sítě s malým počtem bodů.

6.3 Test Zienkiewiczovy metody 3D interpolace

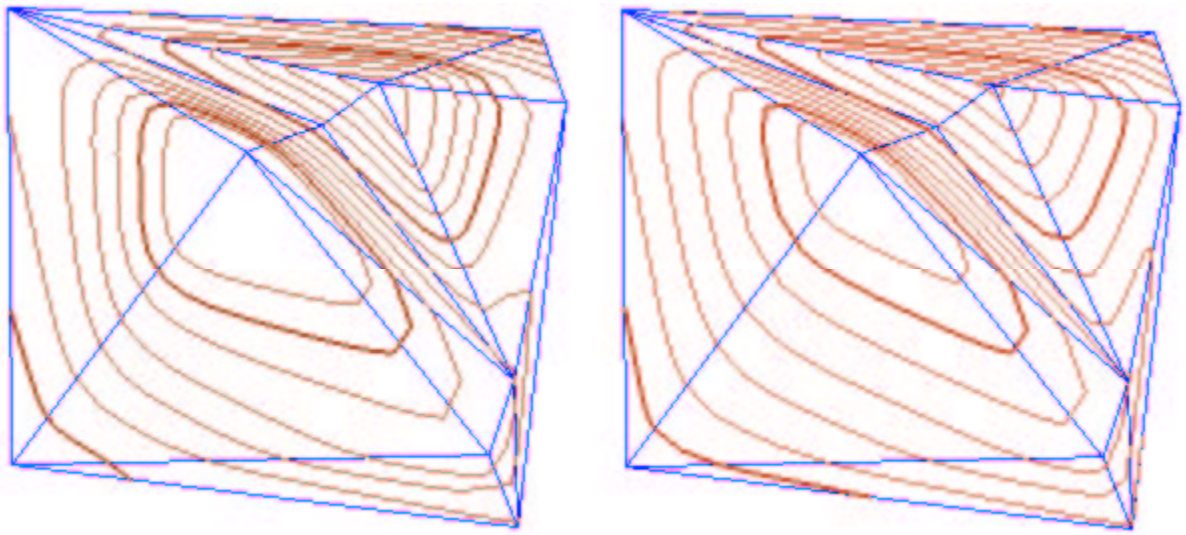
Porovnáním Zienkiewiczovy metody a metody lineární interpolace výpočtu vrstevnic dostáváme zajímavé výsledky. Pro stejnou síť jsme aplikovali oba uvedené způsoby pro stejný interval výšek vrstevnice a sledovali chování metod. Obrázek 6.14a ukazuje výpočet vrstevnic pomocí Zienkiewiczovy metody a obrázek 6.14b výpočet lineární interpolací.



a) b)

Obr. 6.14: Porovnání Zienkiewiczovy metody nelineární interpolace a) s lineární interpolací b)

Z obrázku je velice dobře patrné, že Zienkiewiczova metoda interpolace bere v úvahu zakřivení trojúhelníků a intervaly vrstevnic nejsou lineární jako v druhém případě. Na obrázku 6.15 opět porovnáváme tyto dvě metody výpočtu vrstevnic, navíc však s použitím vyhlazovacího algoritmu. Byl použit vyhlazovací algoritmus na základě B-spline s faktorem vyhlazení 5. Levý obrázek představuje vypočtené vrstevnice pomocí Zienkiewiczovy metody, pravý vrstevnice vypočtené lineární interpolací.



a) b)

Obr. 6.15: Porovnání Zienkiewiczovy metody nelineární interpolace s vyhlazením a) s lineární interpolací s vyhlazením b)

Zienkiewiczova metoda na pokusných datech dobře modelovala trend reálného povrchu, kdy spád terénu není ve všech místech trojúhelníku konstantní. Při vrcholech a úpatích je spád pozvolnější než v úbočích.

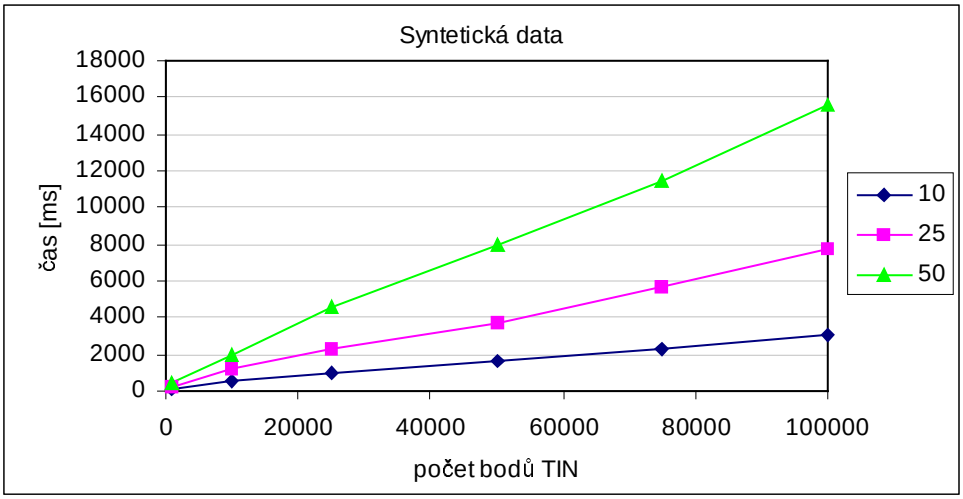
Tato metoda se lépe hodí pro sítě s menším počtem bodů. Při dostatečné hustotě bodů sítě rovinné trojúhelníky dobře reprezentují daný terén a pro výpočet vrstevnic zcela vyhovuje lineární interpolace. Navíc je tato metoda časově náročnější, jak ukážeme v následujícím odstavci.

Časová náročnost

V tabulce 6.13 jsou časové hodnoty pro syntetická data, v tabulce 6.14 pak pro data reálná. Hodnoty jsou uváděny v milisekundách.

Syntetická data						
Počet Sečných rovin	Počet bodů TIN					
	1 000	10 000	25 000	50 000	75 000	100 000
10	70	500	930	1 660	2 270	3 020
25	220	1 200	2 250	3 670	5 680	7 730
50	460	2 410	4 575	7 950	11 480	15 630

Tab. 6.13

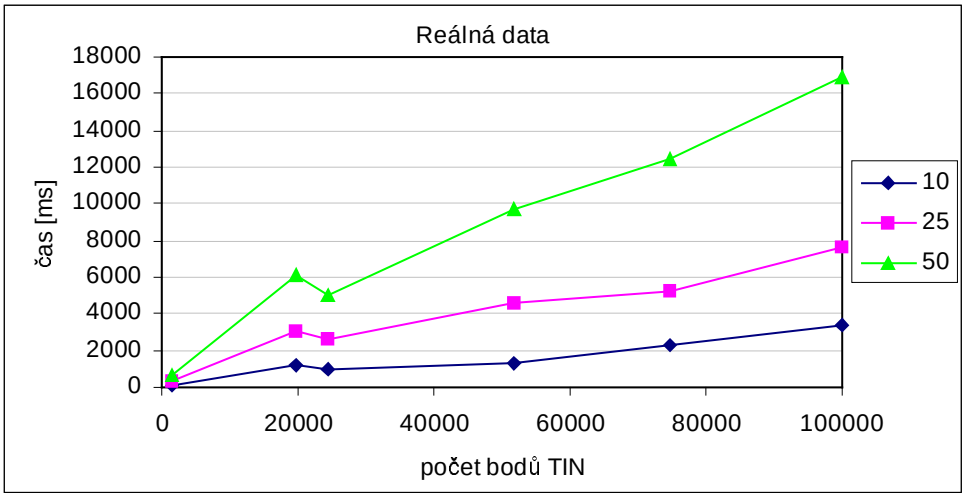


Obr. 6.16: Časová náročnost Zienkiewczova algoritmu pro syntetická data

Z naměřených hodnot lze usuzovat na lineární složitost algoritmu a tím by se potvrdil i teoretický odhad.

Reálná data						
Počet sečných rovin	Počet bodů TIN					
	1 472	19 819	24 308	51 647	74 805	100 000
10	120	1 190	1 005	1 270	2 240	3 390
25	350	3060	2 570	4 610	5 240	7 690
50	770	6 080	5 050	9 740	12 410	16 880

Tab.6.14



Obr. 6.17: Časová náročnost Zienkiewczova algoritmu pro reálná data

Nebudeme-li uvažovat hodnoty pro malá vstupní data, metoda se chová lineárně. Z tabulky však vyplývá, že tento algoritmus je poměrně časově náročný. Pro sítě s počtem bodů 100 000 se výpočetní čas na daném počítači pohyboval okolo 17 vteřin.

Závěrem celé této kapitoly bychom chtěli upozornit, že přesnost daných algoritmů a postupů do značné míry ovlivňuje nejenom velikost TIN, jejich výšková členitost a hustota bodů v nich, ale též způsob triangulace dané sítě. Ze stejných bodů můžeme vytvořit několik zcela odlišných trojúhelníkových sítí a tím samozřejmě dostáváme i rozdílné výsledky pro hledání jejich izočar.

7. Závěr

V dnešní době je velká snaha automatizovat procesy ve všech vědách a oblastech lidské činnosti. Tento trend se nevyhnul ani takovým vědním oborům, jakými jsou kartografie a geografie. Zajisté na tom má velký podíl vědní disciplína geografických informačních systémů, která se v posledních letech velice rozvinula. V této oblasti se neustále vyvíjí a zdokonalují nové programové produkty, které zpracovávají a vizualizují geografickou informaci. Jedním odvětvím této disciplíny je zobrazení digitálního modelu terénu.

Tato práce se zabývá návrhy výpočtu vrstevnic na trojúhelníkové síti představující digitální model terénu. Většina metod na výpočet byla získána v algoritmické podobě od vedoucí diplomové práce. Bylo navrženo a vytvořeno několik programů, které realizují pomocí matematických metod výpočet a vyhlazení vrstevnice. Veškeré algoritmy byly napsány a odladěny v programovém prostředí Borland Delphi 5.0 jako jednotlivé DLL knihovny systému MVE. Výstupní soubory programů jsou navrženy tak, aby je bylo možné použít jako vstupní do softwaru ArcView firmy ESRI.

Implementované řešení bylo otestováno na syntetických i reálných datech různé velikosti. Zde jsme se setkali s některými nedostatky algoritmů, zvláště algoritmu pro umístění popisu na vrstevnici. Problémy nastávají i tam, kde je ze vstupních bodů nevhodně vytvořena trojúhelníková síť. Řešením těchto problémů by bylo vytvoření editačního programu, který by umožňoval odstranění některých vzniklých chyb. To však už přesahuje rámec této diplomové práce a nabízí se tak námět pro další odbornou studii, která by na tuto navazovala.

Tato diplomová práce je pilotní prací v této oblasti a předpokládá návaznost pracemi zaměřenými hlavně na editaci vrstevnic, úpravu jejich popisu a na opravu v triangulaci bodů.

Literatura:

1. ALEXANDR, L.: Výuka počítačové grafiky cestou WWW, Diplomová práce, Vysoké učení technické, Brno, 2001
2. ANGEL, E.: Interactive computer graphics a top-down approach with OpenGL™, Addison Wesley Longman, Inc., 2000
3. BURROUGH, R. A.: Principles of GIS for land resources assement, Clarendon Press, Oxford, 1986
4. CORNELIUS, S., HEYWOOD, I.: Spatial operationns, Course notes, International distance learning GIS diploma programme, The Manchester Metropolitan University, 1994
5. JEŽEK, F.: Geometrické a počítačové modelování, Pomocný učební text, ZČU, Plzeň, 2000
6. KRCHO, J.: Reliéf ako priestorový subsystém geografickej krajiny a jako kompletný digitálny model (KDMT), Geografický časopis, ročník 31, číslo 3, 1979
7. McMASTER, R.B., SHEA, K.S.: Generalization in Digital Cartography, Association of American Geographers, Washington D.C., 1992
8. RIESENFELD, R.F.: Short note: On Chaiken´s Algorithm. Computer Graphics and Image Processing, 4:304-310, 1975
9. TUČEK, J.: Geografické informační systémy Principy a praxe, Computer Press, Praha, 1998
10. URBAN, J.: Digitální model terénu, Ediční středisko ČVUT, Praha, 1991
11. URBAN, J.: Projekt II. – Práce s grafickou informací. Digitální model terénu, Doplňková skripta, ČVUT, Praha, 1988
12. VEVERKA, B.:Topografická a tematická kartografie. Ediční středisko ČVUT, Praha, 1997

Odkazy:

1. [MVE] http://herakles.zcu.cz/research.php
2. [ESRI] www.esri.com

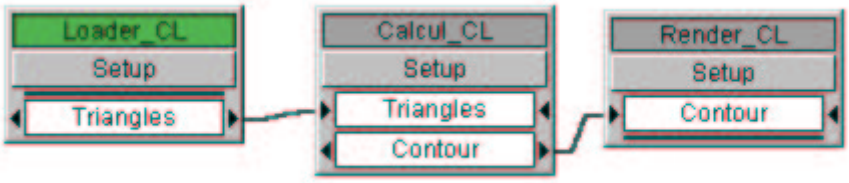
Přílohy

A. Uživatelská příručka

Snahou uživatelské příručky je popsat jednotlivé funkce programu. V této práci nemáme místo ani čas podrobně popisovat programové prostředí MVE, proto zde budou popsány pouze funkce jednotlivých vytvořených modulů.

Moduly v prostředí MVE lze libovolně spojovat, musíme jen dbát na to, aby výstupní formát modulu byl kompatibilní se vstupním formátem modulu následujícího (viz obr. A.1 a obr. A.2). Výstupy a vstupy jsou naznačeny černými šipkami po stranách u jednotlivých modulů s názvem výměnného formátu. *Triangles* je označována datová struktura *Triangle* a *Contour* struktura *TContour*.

Obr. A.1: Schéma zapojení modulů v systému MVE s modulem DTlib



Obr. A.2: Schéma zapojení modulů v systému MVE s modulem Loader_CL a bez Interpol_CL

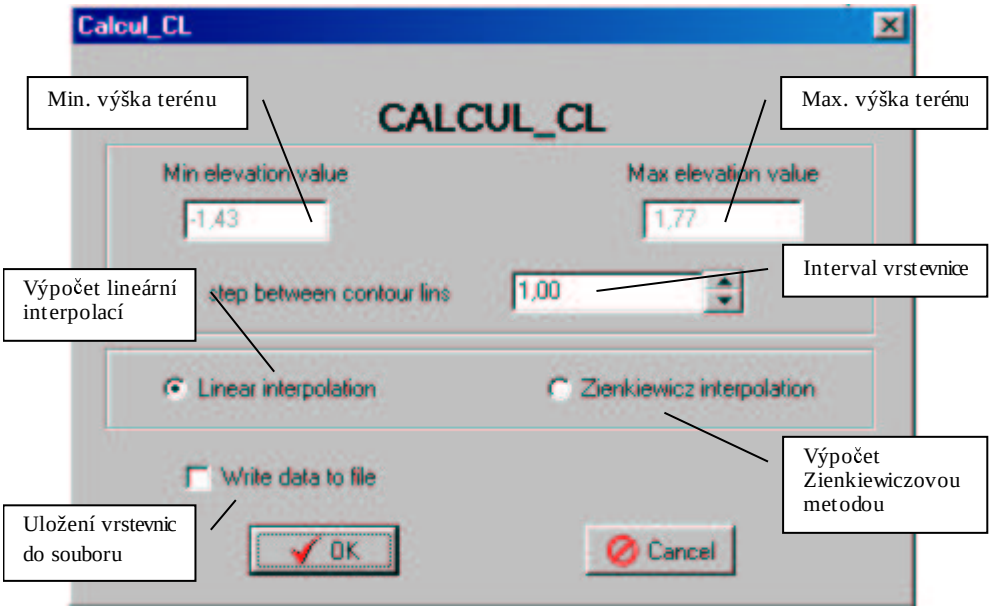
Modul **Loader_CL**:

Modul načítá data ze souboru a naplní datovou strukturu *Triangle*, která je vstupem do modulu **Calcul_CL**.

Modul **Calcul_CL**:

Tento modul počítá vrstevnice z trojúhelníkové sítě a je řazen za **DTlib** nebo za **Loader_CL.** Dojde-li program k tohoto modulu, objeví se formulář znázorněný na obrázku A.3. Obrázek obsahuje i popisky jednotlivých funkcí formuláře. Horní část formuláře obsahuje informativní údaj o minimální a maximální výše zpracovávaného terénu. Uprostřed formuláře je umístěno editační okno, kde si uživatel volí interval, po kterém se počítají vrstevnice. Dále se na tomto formuláři nachází volba druhu interpolace

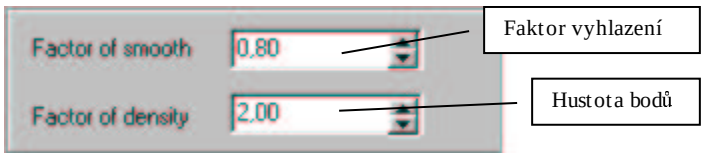
(lineární nebo Zienkiewiczova) pro výpočet vrstevnic. Ve spodní části formuláře vlevo je umístěna volba pro zápis do souboru.



Obr. A.3: Formulář modulu Calcul_CL

Modul **Interpol_CL**:

Tento modul zajišťuje vyhlazení lomového průběhu vrstevnice. Modul navazuje na **Calcul_CL**, ale nemusí být vůbec zařazován do aplikace, v tom případě vrstevnice nebudou vyhlazeny. Na formuláři odpovídajícímu tomuto modulu je možné volit jednu ze dvou metod vyhlazení. Pro metodu označovanou *Normal smooth* jsou k dispozici dva parametry (viz obr.A.4), jejichž význam jsme objasnili dříve. Pro metodu označovanou *B-spline smooth* se volí míra vyhlazení (viz obr. A.5). Ve spodní části formuláře se nachází volba pro výstup do souboru.



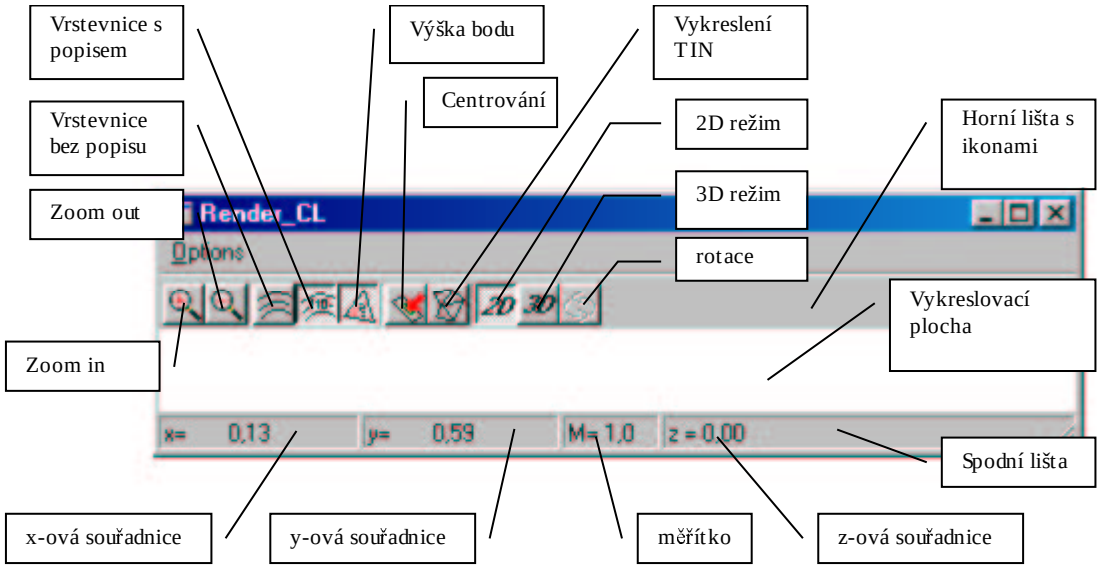
Obr. A.4: Nastavitelné parametry pro metodu Normal smooth

Obr. A.5: Nastavitelný parametr pro metodu B-spline

Objeví-li se po ukončení tohoto formuláře varovná hláška „Too many points on one segment …“, znamená to, že body na jednom ze segmentů vrstevnic jsou rozloženy příliš hustě. V tom případě doporučujeme zmenšit parametr, který ovlivňuje hustotu bodů na počítané vrstevnici.

Modul **Render_CL**:

Modul slouží jako prohlížeč vypočtených vrstevnic. Následuje po modulech **Calcul_CL** nebo **Interpol_CL** a neobsahuje už další výstup. Popis jednotlivých funkcí prohlížeče ukazuje obrázek A.6.



Obr. A.6: Formulář modulu Render_CL

Okno prohlížeče je děleno do tří částí. **Horní lišta**, kde jsou umístěny ikony a menu, **vykreslovací plocha** a **dolní lišta**, která slouží pro výpis informací. Prohlížeč pracuje ve dvou základních režimech, a to 2D a 3D vizualizace. Režimy se přepínají pomocí ikonek na horní liště okna.

Režim 2D zobrazuje vrstevnice v rovině XY. Vrstevnice se mohou vykreslit s popisy nebo bez nich a může být přidána i trojúhelníková síť. Pohybujeme-li myší se zmáčknutým levým tlačítkem ve vykreslovací ploše, posouváme obraz. Pomocí pravého tlačítka a pohybu myši měníme měřítko. Pokud chceme zjistit výšku v libovolném bodě terénu, zvolíme ikonu *Výška bodu*, najedeme na zvolené místo ve vykreslovací ploše a klikneme pravým tlačítkem myši. V dolní liště se zobrazí výška v daném bodě. Je-li aktivní funkce *Výška bodu*, nelze měnit měřítko pomocí myši.

Režim 3D umožňuje zobrazit vrstevnice a TIN ve trojrozměrném prostoru. Zobrazovaný objekt můžeme natáčet pomocí levého tlačítka a pohybu myši ve vykreslovací ploše. Při zmáčknutém pravém tlačítku myši a jejím pohybu měníme měřítko objektu. Lze využít i funkce rotace, která daným objektem rotuje kolem osy z.

Pro oba režimy jsou společné funkce změna měřítka (*Zoom in*, *Zoom out*) a vycentrování obrazu na střed (*Centrování*).

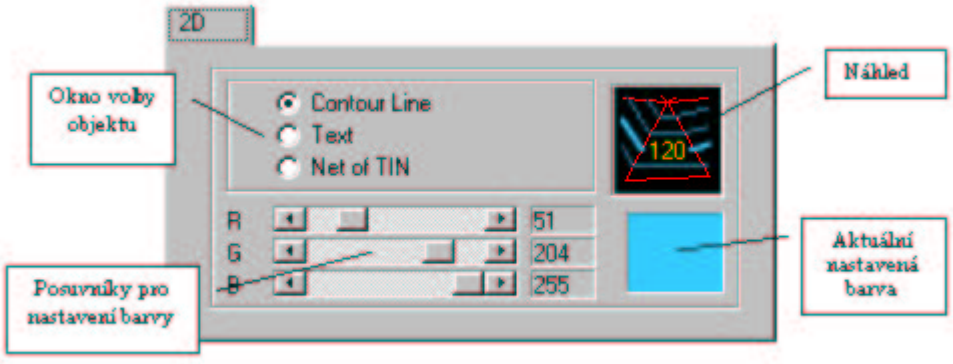
Popis a funkčnost některých kláves:

- **n,** popř. **N**: zmenšení měřítka
- **m,** popř. **M**: zvětšení měřítka
- **c,** popř. **C**: centrování obrazu
- **k,** popř. **K**: přepínání výpisu vrstevnic s popisem a bez popisu (pouze pro 2D)
- **s,** popř. **S**: zapínání/vypínání malby trojúhelníkové sítě

Popis menu

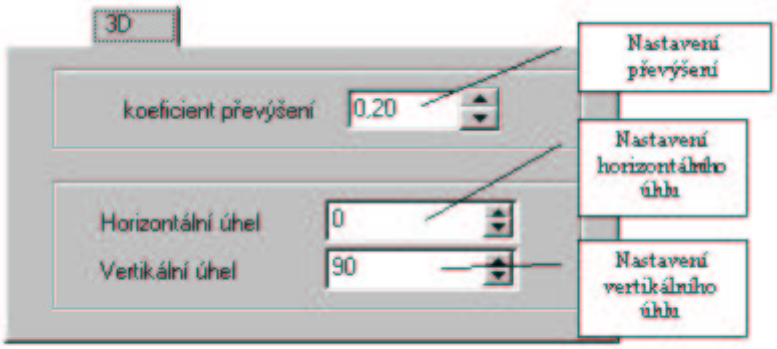
V menu je pouze jedna položka – *Options*. Touto položkou lze nastavit některé parametry pro vykreslení. Při volbě *Options* se objeví formulář s dvěma záložkami – 2D a 3D.

Záložka 2D (viz obr. A.7) umožňuje nastavení barevného prostředí pro vykreslení vrstevnic v rovině. Jednotlivé funkce jsou zřejmé z obrázku.



Obr. A.7: Záložka 2D v *Options*

Záložka 3D (viz obr A.8) nabízí nastavení parametrů pro vykreslení objektu v trojrozměrném prostoru. Nastavitelné parametry jsou převýšení a úhly natočení objektu ve vertikální a horizontální ose.



Obr. A.8: Záložka 3D v *Options*

B. Ukázky výměnných formátů a definice datových struktur

Ukázka vstupního souboru modulu **DTlib**:

```
5
0.42868 0.20106 -0.86000
0.91568 0.38039 0.04672
0.55284 0.86402 0.42103
0.93631 0.48820 1.67458
0.49317 0.46177 1.43138
4
 0  1  4
 1  3  4
 0  4  2
 4  3  2
```

Ukázka vstupního souboru modulu **Loader_CL**:

```
5
0.42868 0.20106 -0.86000
0.91568 0.38039 0.04672
0.55284 0.86402 0.42103
0.93631 0.48820 1.67458
0.49317 0.46177 1.43138
4
 0  1  4
 1  3  4
 0  4  2
 4  3  2
4
 1  2 -1
 3  0 -1
 3 -1  0
-1  2  1
```

Ukázka výstupního souboru z modulů **Calcul_CL** a **Interpol_CL**:

```
XYZ
1
0.51462,0.23270,-0.70000
0.43318,0.21926,-0.70000
0.44419,0.28386,-0.70000
END
2
0.89059,0.37115,0.00000
0.45288,0.29891,0.00000
0.51203,0.64613,0.00000
END
3
0.63818,0.78038,0.70000
0.53636,0.75295,0.70000
0.92396,0.42366,0.70000
0.47259,0.37855,0.70000
0.63818,0.78038,0.70000
END
END
```

Datová struktura *Triangle*

```
TIndex      = 0..MAX_POINTS;
Coord       = array[TIndex] of TFloat;
P_Coord     = ^Coord;
Status      = array[TIndex] of TStatus;
P_Status    = ^Status;
Index       = array[TIndex] of TCardinal;
P_Index     = ^Index;


Triangle = record   // information about the triangle mesh
             { static part }
             xmin,xmax,ymin,ymax,zmin,zmax : TFloat;  // data minmax box
             pos_x,pos_y,pos_z            : TFloat;  // position of the data set (usually (0,0,0))
             rot_x,rot_y,rot_z            : TFloat;  // rotation of the data (usually (0,0,0))
             dis_x,dis_y,dis_z            : TFloat;   // distortion, scaling (usually (1,1,1))
             NV_M,   // number of sites in the array of points
             NT_M,   // number of sites in the array of triangles
             NV,     // number of valid vertices
             NT      // number of valid triangles
             : TIndex;
             orientation : TStatus;                  // CW x CCW x INVALID
            { dynamic part }
             { vertices }
             P_VCoor     : array[0..MAX_DIM-1] of P_Coord;   // coordinates x,y,h1,...
             P_VNorm     : array[0..2] of P_Coord;          // vertex normals
             P_VStatus   : P_Status;                    // state flags of vertices
             { triangles }
             P_TV        : array[0..2] of P_Index;          // triangle vertices
             P_TT        : array[0..2] of P_Index;          // triangle neighbours
             P_TNorm     : array[0..2] of P_Coord;          // triangle normals
             P_TStatus   : P_Status;                    // triangle status
             z_valid     : Tbyte;
             end;
```

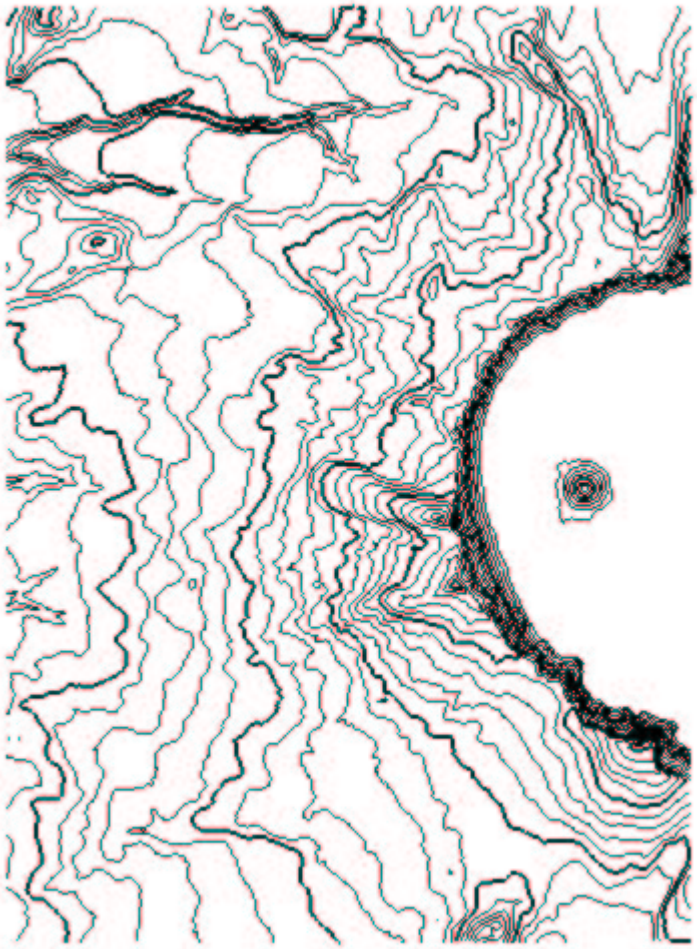
Datová struktura *TContour*

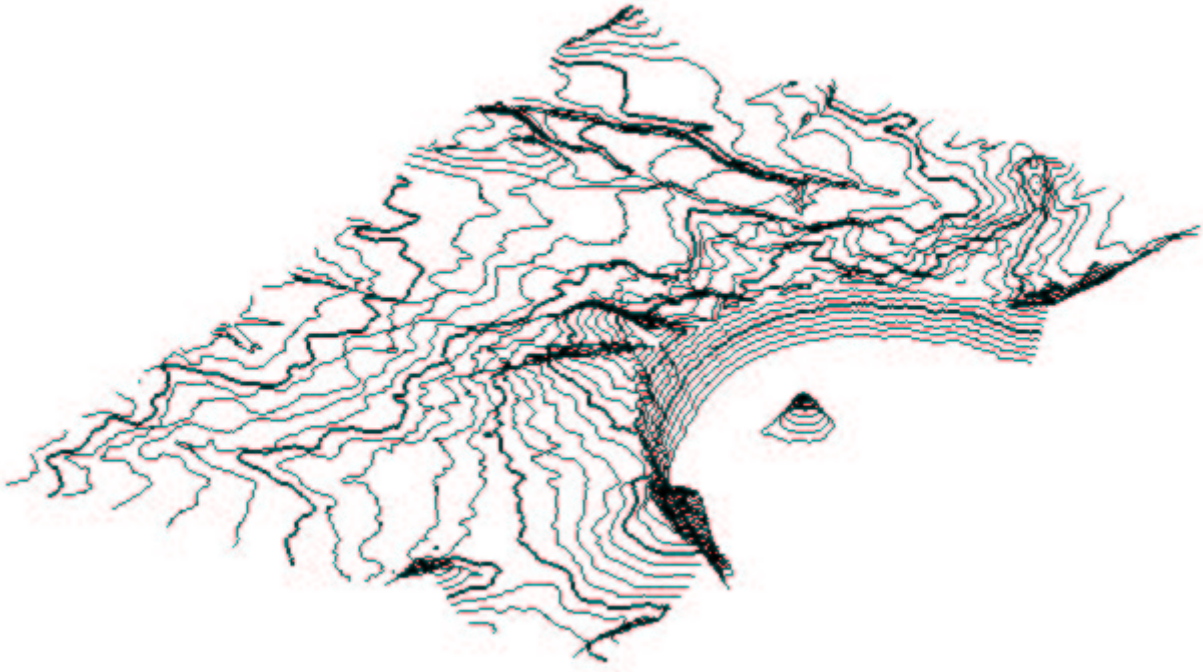
```
TConCoord = record            //contour coordinations
              x,y : TFloat;       //souradnice bodu
              etyp: TByte;        //typ hrany
            end;
ConCoord = array [TIndex] of TConCoord;
PConCoord = ^ConCoord;

TSpot = record                  //typ pro kotu
          xp,yp,fi  : TFloat;  //souradnice ref. bodu a uhel
          xl,xr,yd,yu: TFloat;  //box textu
          start,fin : TIndex;     //indexy bodu segmentu mezi nimiz je kota
          text      :string[6];     //text koty
        end;
Spot = array [0..1] of TSpot;
PSpot = ^Spot;

THead = record                  //head of one segment of contour line
          zk,                       //vyska
          xl,xr,yd,yu : TFloat;     //box segmentu
          flag,                     //stavy
          nkot        : TByte;      //pocte kot
          nv          : TIndex;     //pocet bodu tvorici segment
          P_ConCoord  : PConCoord;  //ukazatel na pole bodu
```

```
            P_Spot      : PSpot;      //ukazatel na kotu
          end;
Head = array [TIndex] of THead;
PHead = ^Head;

TContour = record                              //contour line
            {struc Contour}
            xmin,xmax,ymin,ymax,zmin,zmax : TFloat; //box of whole structure
            ns                          : TIndex; //pocet segmentu
            P_Head                      : PHead;  //ukazatel na pole segmentu

            {struc Triangle}
            pos_x,pos_y,pos_z           : TFloat;  // position of the data set (usually (0,0,0))
            rot_x,rot_y,rot_z           : TFloat;  // rotation of the data (usually (0,0,0))
            dis_x,dis_y,dis_z           : TFloat;   // distortion, scaling (usually (1,1,1))
            NV_M,   // number of sites in the array of points
            NT_M,   // number of sites in the array of triangles
            NV,     // number of valid vertices
            NT      // number of valid triangles
            : TIndex;
            orientation : TStatus;                  // CW x CCW x INVALID
           { dynamic part }
            { vertices }
            P_VCoor     : array[0..MAX_DIM-1] of P_Coord;   // coordinates x,y,h1,...
            P_VNorm     : array[0..2] of P_Coord;          // vertex normals
            P_VStatus   : P_Status;                    // state flags of vertices
            { triangles }
            P_TV        : array[0..2] of P_Index;          // triangle vertices
            P_TT        : array[0..2] of P_Index;          // triangle neighbours
            P_TNorm     : array[0..2] of P_Coord;          // triangle normals
            P_TStatus   : P_Status;                    // triangle status
            z_valid     : Tbyte;
           end;
```

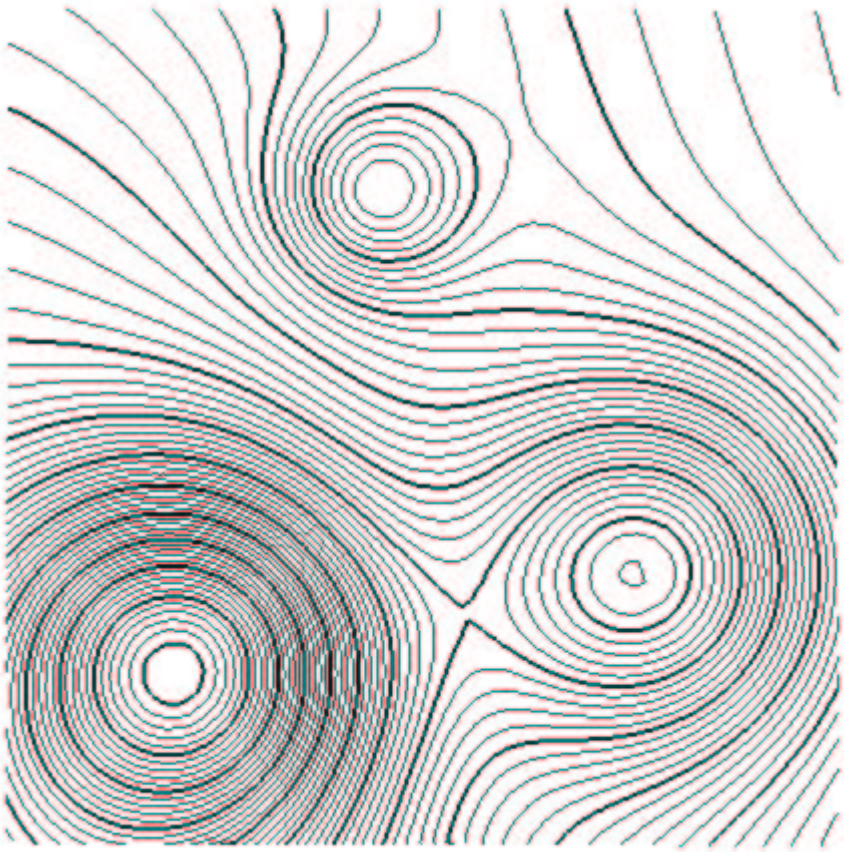
C. Ukázky grafických výstupů

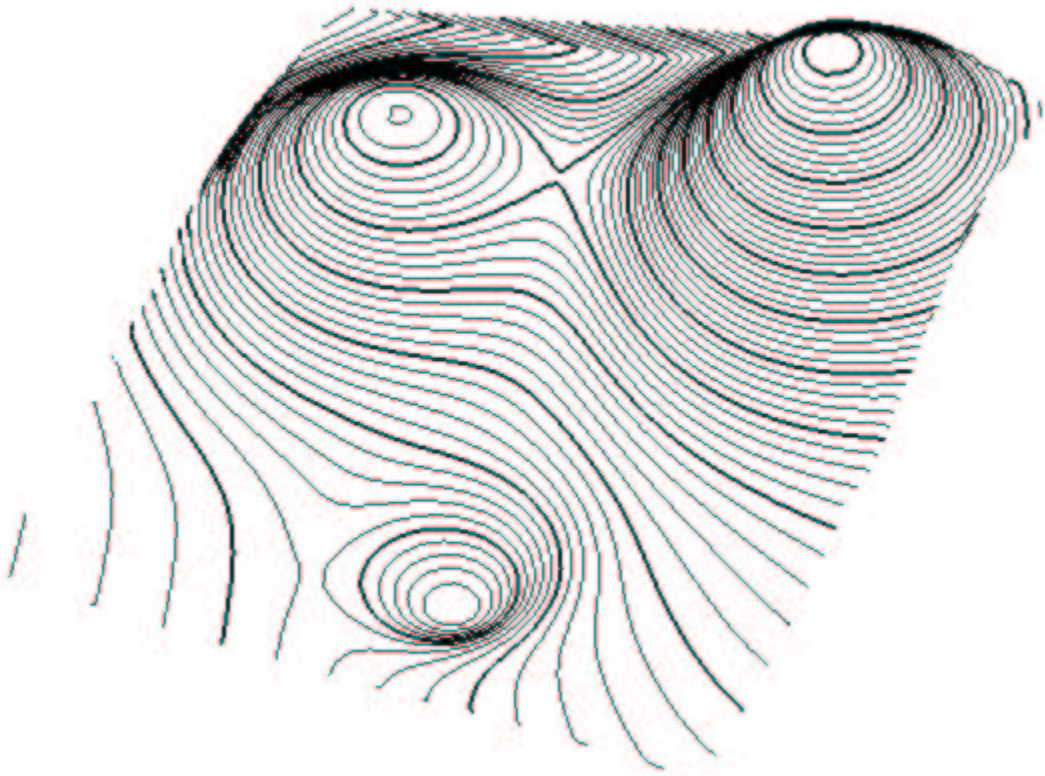
Reálná data



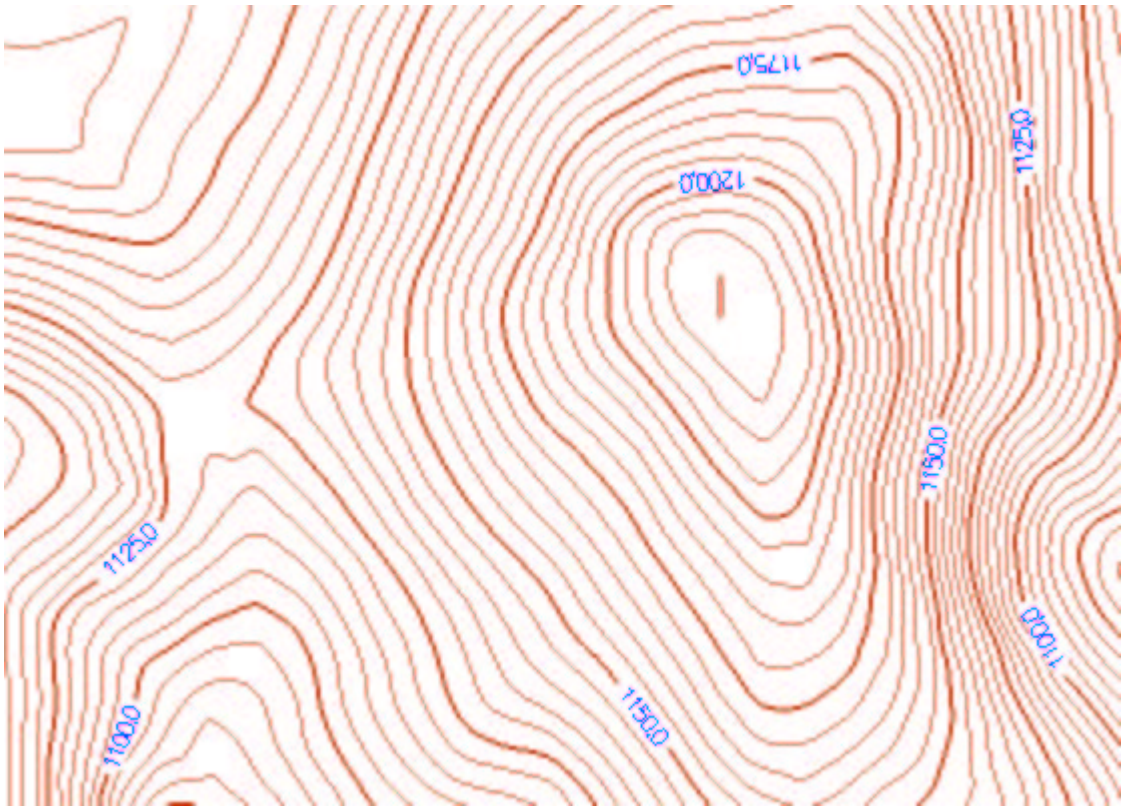


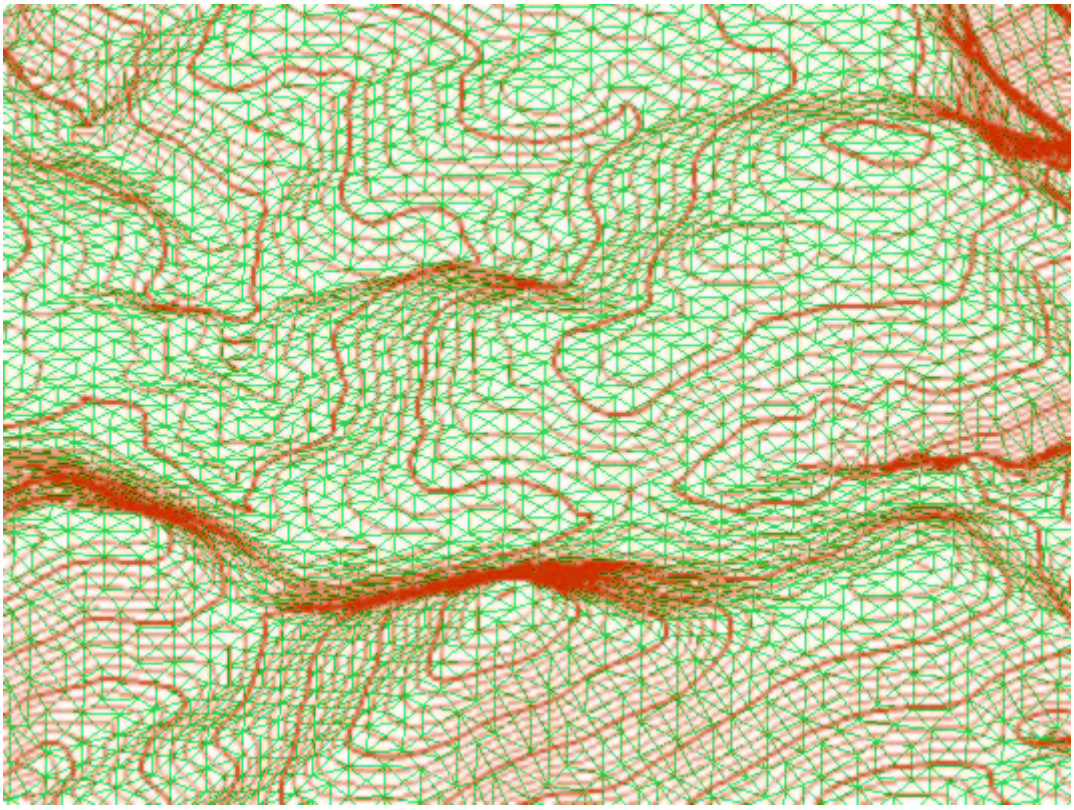
Syntetická data





Popis vrstevnic – detail, Vykreslení TIN - delail





Evidenční list

Souhlasím s tím, aby moje diplomová práce byla půjčována k prezenčnímu studiu v Univerzitní knihovně ZČU v Plzni.

Datum: Podpis:

Uživatel stvrzuje svým čitelným podpisem, že tuto diplomovou práci použil ke studijním účelům a prohlašuje, že ji uvede mezi použitými prameny.

Jméno	Fakulta/katedra	Datum	Podpis