Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

Bakalářská práce

Měření přesnosti interpolace na trojúhelníkové síti pro různá rozložení bodů

Plzeň, 2015 Jan Schröpfer

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně, výhradně s použitím citovaných pramenů.

V Plzni dne _____ _____

Abstrakt

Hlavním cílem této práce je prozkoumat možnost využití centroidního Voroného diagramu pro modelování terénu. Původní body *(x, y)* jsou touto metodou přemístěny tak, že výsledná teselace na těchto bodech má výhodnější vlastnosti. Přemístění bodů však vyžaduje přepočítat souřadnici *z*, čímž dochází k určité chybě. Práce otestovala velikost chyby pro různá vstupní data.

Dalším dílčím úkolem bylo zjistit, jak se centroidní Voroného diagram chová na reálných datech (tedy jak se změní původní vzhled terénu), kde měření chyby není možné.

Posledním úkolem bylo vytvoření pomocných modulů, například generátoru zkušebních bodů, které pomohou s testováním.

Abstract

The main objective of this paper is to explore the possibility of using centroidal Voronoi tessellations for modeling of the terrain. The original points *(x,y)* are relocated by this method so that the resulting tessellation of those points has more favorable properties. Relocation of the points requires recalculation of *z* coordinate, which creates error. We tested the error for different input data.

Another partial task was to determine how centroidal Voronoi tesselletaion behaves on real data (ie how it changes the original appearance of the terrain), where measuring error is not possible.

The last task was to create auxiliary modules, such as a generator of test points.

Obsah

1. Úvod

Pro digitální reprezentaci terénů je potřeba velkého množství naměřených bodů. Z těchto bodů se následně vytvaruje trojúhelníková sít. Pro co možná nejpřesnější modely je však důležité se zaměřit také na rozložení bodů. Pro přesun bodů existuje mnoho různých interpolačních algoritmů, které je vhodné použít podle časové či výkonnostní náročnosti. Běžně se pro tento druh problému používají centroidní Voroného diagramy, které vhodně rozdělují prostor na menší podprostory tvořené množinami bodů.

Centroidní Voroného diagramy (dále CVT) jsou způsobem, jak rozdělit prostor k dané diskrétní množině objektů v prostoru, například diskrétní množině bodů. CVT se velmi často používá například pro úpravu povrchů triangulovaných modelů. V této práci nás tedy zajímá, jak lze využít CVT pro digitální modely terénu.

Úkolem této práce je ověřit, zda při přemístění bodů dle CVT dostaneme vhodnější model terénu, než byl původní. Toho docílíme porovnáním chyby interpolace na původním a upraveném modelu na bodech vygenerovaných z funkce $z = \mathrm{f}(x, y)$. Dále ukážeme chování na reálných bodech naměřených v terénu, kde přímé porovnání není možné. Součástí práce jsou další pomocné moduly, například generátor zkušebních bodů $(x, y, \mathrm{f}(x, y))$ pro různé funkce.

Práce obsahuje následující části:

- Kapitola 2 - Teoretická část – podává stručné informace o interpolacích, trojúhelníkových sítích, Voroného diagramech (obzvláště o Centroidním Voroného diagramu), vhodné metody používané pro interpolaci bodů a vysvětlení dalších důležitých pojmů.
- Kapitola 3 - Realizace řešení – věnuje se návaznosti modulů, jednotlivým modulům bakalářské práce, jejich návaznosti a vhodnému způsobu ukládání dat do souboru pro další použití.
- Kapitola 4 - Experimenty – věnuje se průběhu testování, popisuje experimenty a vyhodnocuje jejich výsledky.
- Kapitola 5 – Závěr – rekapituluje stanovené cíle a shrnuje dosažené výsledky.

2. Teoretická část

2.1. Potřebné pojmy

Afinní prostor

Afinní prostor je v geometrii prostor, na kterém je definováno sčítání bodů a vektorů. Jedná se o zobecnění eukleidovského prostoru [2].

Afinní prostor je množina A spolu se zobrazením

$$+: V \times A \to A,\ (v, a) \mapsto v + a$$

kde V je vektorový prostor, který má následující vlastnosti [13]:

1. Pro každé a v A platí $0 + a = a$, kde $0 \in V$ je nulový vektor
2. Pro každé v, w ve V a a **v** A platí $v + (w + a) = (v + w) + a$
3. Pro každé a v A, zobrazení $V \to A$, $v \mapsto v + a$ je bijekce.

Volbou počátku $a \in A$ je možné identifikovat A s vektorovým prostorem V zobrazením $a + v \mapsto v$. Naopak, každý vektorový prostor V je afinní prostor nad sebou samým.

Barycentrické souřadnice

Tento souřadnicový systém byl vytvořen Augustem Ferdinandem Möbiem v roce 1827 [4]. Obecnou definici barycentrických souřadnic lze zapsat takto:

Nechť $x_1,\ldots,x_n$ jsou vrcholy simplexu v afinním prostoru A. Pokud pro jakýkoliv bod p v A platí:

$$(a_1 + \cdots + a_n)p = a_1x_1 + \cdots + a_nx_n$$

a zároveň alespoň jedno z $a_1, \ldots, a_n$ je nenulové, můžeme říct, že koeficienty ($a_1, \ldots, a_n$) jsou barycentrické souřadnice bodu p s ohledem na $x_1, \ldots, x_n$.

Barycentrické souřadnice jako takové nejsou unikátní – pro každé b, které není nulové, platí, že ($ba_1, \ldots, ba_n$) jsou rovněž barycentrickými souřadnicemi bodu p. Z tohoto důvodu se v některých případech používá podmínka:

$$\sum a_i = 1$$

která dělá souřadnice unikátními. Pokud tato podmínka platí, barycentrické souřadnice již nejsou homogenní souřadnice, ale afinní souřadnice.

Mějme trojúhelník T definovaný jeho třemi vrcholy r_1, r_2 a r_3. Každý bod p uvnitř trojúhelníku pak lze zapsat jako unikátní konvexní kombinaci těchto tří vrcholů. To znamená, že pro každé p existuje unikátní kombinace čísel $\lambda_1, \lambda_2, \lambda_3 \geq 0$, pro které platí:

$$\lambda_1 + \lambda_2 + \lambda_3 = 1$$

pak každý bod p lze definovat jako:

$$p = \lambda_1 r_1 + \lambda_2 r_2 + \lambda_3 r_3$$

čísla λ_1, λ_2, λ_3 jsou pak barycentrické souřadnice bodu p s ohledem na trojúhelník, ve kterém se p nachází.

Jedním z nejefektivnějších způsobů zjištění barycentrických souřadnic je použití Cramerova pravidla pro řešení lineárních systémů. Pokud známe umístění vrcholů A, B, C trojúhelníku a bod p, u kterého chceme zjistit barycentrické souřadnice, spočítáme vektory $v_0 = C - A$, $v_1 = B - A$, $v_2 = p - A$, poté jejich skalární součiny $dot_{00} = v_0 \times v_0$, $dot_{01} = v_0 \times v_1$, $dot_{02} = v_0 \times v_2$ $dot_{11} = v_1 \times v_1$ a $dot_{12} = v_1 \times v_2$ a dostaneme tuto soustavu rovnic:

$$\lambda_1 = \frac{dot_{01}\, dot_{02} - dot_{01}\, dot_{12}}{dot_{00}\, dot_{11} - dot_{01} dot_{01}}$$

$$\lambda_2 = \frac{dot_{00}\, dot_{12} - dot_{01} dot_{02}}{dot_{00}\, dot_{11} - dot_{01} dot_{01}}$$

$$\lambda_3 = 1 - \lambda_1 - \lambda_2$$

pro které platí:

$$(dot_{00}\, dot_{11} - dot_{01} dot_{01}) \neq 0$$

Centroid

Centroid (geomterické těžiště, geometrický střed) dvoudimenzionální oblasti je aritmetický průměr pozic všech bodů v simplexu. V konvexních tvarech obecně platí, že centroid se nachází uvnitř konvexního obalu. V trojúhelníku leží centroid na místě, kde dochází k průniku všech těžnic trojúhelníku. Leží tedy na Eulerově přímce trojúhelníku [1].

V trojúhelníku lze centroid popsat pomocí kartézských souřadnic. Máme-li trojúhelník s vrcholy $a = (x_a, y_a)$, $b = (x_b, y_b)$, $c = (x_c, y_c)$, lze centroid C vyjádřit pomocí vzorce:

$$C = \frac{1}{3}(a + b + c) = (\frac{1}{3}(x_a + x_b + x_c), \frac{1}{3}(y_a + y_b + y_c))$$

Pro obecný výpočet centroidu musíme znát body oblasti a její velikost. Velikost oblasti $\boldsymbol{A}$ tvořené $\boldsymbol{N}$ body můžeme spočítat pomocí rovnice [3]:

$$A = \frac{1}{2}\sum_{i=0}^{N-1}(x_i y_{i+1} - x_{i+1} y_i)$$

a následně lze vypočítat centroid C pomocí soustavy rovnic [3]:

$$C_x = \frac{1}{6A}\sum_{i=0}^{N-1}(x_i + x_{i+1})(x_i y_{i+1} - x_{i+1} y_i)$$

$$C_y = \frac{1}{6A}\sum_{i=0}^{N-1}(y_i + y_{i+1})(x_i y_{i+1} - x_{i+1} y_i)$$

Interpolace

Existuje několik běžně používaných matematických definic interpolací. Jednotlivé definice se od sebe liší pouze tím, pro jakou matematickou činnost je autor používá nebo s jakými daty pracuje. Interpolace (z latinského inter-polare, vylepšit vkládáním) mají v numerické matematice následující definici:

„Interpolace je metoda pro nalezení přibližné či přesné hodnoty za pomoci známých hodnot.“ [15]

Jedná se tedy o způsob, jak nalézt hodnoty bodů funkce za pomoci jiných spočtených nebo naměřených hodnot.

Nyní definujme interpolace z matematického hlediska. Mějme funkci f(x), jejíž hodnota je známa v bodech f(x_0), f(x_1), ... f(x_n). V tomto případě interpolace znamená nalezení funkční hodnoty f(x), platí-li, že $x_0 < x < x_n$ – tedy známe-li body x_0 a x_n, můžeme pomocí interpolace nalézt bod x (například polynomem).

2.2 Trojúhelníková síť

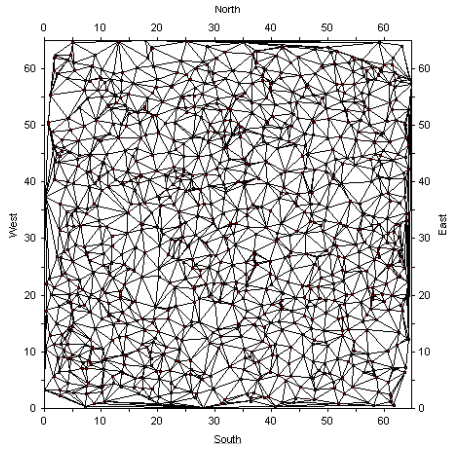
Mějme množinu bodů v terénu, kterou nazýváme vrcholy. Tyto vrcholy pak budou vstupem do triangulačního algoritmu. Hranou nazvěme spojnici mezi dvěma vrcholy v síti, jež neprotíná žádnou z jiných hran. Z toho vyplývá, že hrany se mezi sebou neprotínají nikde jinde než ve vrcholech sítě. Trojice hran pak tvoří jeden trojúhelník v síti a každá vnitřní hrana (tedy neokrajová) je součástí dvou trojúhelníků v síti. Celkový počet hran a trojúhelníků sítě v 2D nezávisí na použitém triangulačním algoritmu.

Samotnou trojúhelníkovou síť [14] pak tvoří množina sousedících a vzájemně se nepřekrývajících trojúhelníků interpolující zadané body. K jejich popisu se obvykle používá množina všech vrcholů a hran.

Pro výpočet trojúhelníkové sítě je potřeba vstupní množina bodů. Běžně jsou rozšířené dva možné druhy vstupních dat – pravidelné a nepravidelné vzorky dat. Pravidelné vzorky se běžně v literatuře nazývají pojmem GRID a z názvu vyplývá, že se jedná o síť pravidelně rozmístěných vrcholů. Zpravidla se pro pravidelné sítě využívá čtvercové rozmístění bodů v síti, ale občas se lze setkat i s šestiúhelníkovým rozmístěním.

Výhodou tohoto druhu sítě je jednoduchost samotné triangulace bodů a pravidelnost výsledku. Hlavní nevýhodou je malá flexibilita – nemožnost reagovat na nerovnosti v terénu, které se kvůli pravidelnosti ve výsledcích ztratí. Tím lze ztratit určité detaily terénu a zároveň tím dochází k nedokonalosti interpolace.

Nepravidelné sítě se běžně označují jako Triangulated Irregular Network (TIN, viz obr. 2.1), z názvu lze poznat, že rozmístění vrcholů postrádá pravidelnost. Hlavní výhodou tohoto modelu je schopnost reprezentace dat s proměnlivou přesností. Triangulace takovéto sítě je sice obtížnější než u pravidelné, ale má své výhody – např. na rovině není potřeba velkého množství vrcholů pro přesnou reprezentaci, zatímco na nerovných terénech je rozmístěno více vrcholů. Tím se zvyšuje přesnost interpolace v takovýchto terénech. Další výhodou je vyšší efektivita ukládání dat, protože není třeba konstantní přesnosti na celé síti pro podobnou přesnost, jakou má hustá pravidelná síť. Nevýhodou je však obtížnější manipulace, triangulace a pozdější úpravy.



Obr. 2. 1 Příklad TIN pro 256 bodů

Nepravidelné trojúhelníkové sítě se z množiny bodů získají procesem, který nazýváme triangulací. Existují různé algoritmy triangulace podle kritérií, která má výsledná síť splňovat. Kritéria jsou lokální a globální.

Lokální kritéria lze rozdělit na dvě hlavní skupiny – kritéria zabývající se úhly v trojúhelníku (úhlová kritéria) a kritéria zabývající se délkou hran (hranová kritéria). Obecně jsou v praxi více rozšířena úhlová kritéria, a to hlavně z důvodu výsledků, které se vyhýbají příliš malým, velkým či „úzkým“ trojúhelníkům. Úhlová kritéria se zaměřují na maximalizování minimálního úhlu trojúhelníku (Delaunayho triangulace, dále DT), minimalizování maximálního úhlu (minmax triangulace) a minimalizování maximální excentricity. Hranová kritéria se pak zaměřují na minimalizování sumy délek hran (triangulace s minimální vahou), její místní aproximace, složené z nejkratších možných hran (greedy triangulace, dále GT), minimalizování maximální délky hrany.

Jednou z nejčastějších metod triangulace sítě využívající úhlové kritérium je DT. Její největší výhodou je produkce „dobrých“ trojúhelníků (jde o maximální minimalizaci úhlů).

Hlavní myšlenkou GT je pak vždy použít nejkratší hranu. To znamená, že se vygenerují všechny možné hrany, seřadí se a následně se v cyklu testuje hrana po hraně v pořadí rostoucí délky. Proces skončí až ve chvíli, kdy se ukončí triangulace. GT je sice složitější na implementaci než DT, přesto má pouze srovnatelné výsledky s jednodušší DT. Proto se častěji volí DT před GT.

2.3 Konvexní obálka

Ve výpočetní geometrii je konvexní obálka (někdy také označovaná jako konvexní obal) jednou ze základních struktur. Výpočet konvexního obalu je jedním ze základních geometrických problémů studovaných v tomto oboru a používá se v mnoha algoritmech. Ve výpočetní geometrii je konvexní obálka Q označován $H(Q)$, resp. jako $CH(Q)$.

Konvexní obálku lze definovat takto: „Konvexní obal Q je množinou všech konvexních kombinací bodů v dané množině Q." [11]

Z algebraického hlediska je konvexní obálka:

„Množina x_y je množinou všech bodů ve tvaru $\propto x + \beta y$ kde $\propto \geq 0$, $\beta \geq 0$, a $\alpha + \beta = 1$. α a β jsou reálná čísla, zatímco x a y jsou body nebo vektory. Konvexní kombinace bodů $x_1, \ldots x_k$ je sumou všech $\alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_k x_k$, kde $\alpha_i \geq 0$ pro všechna i a $\alpha_1 + \ldots \alpha_k = 1$." [11]

Z toho vyplývá, že konvexní obálku úsečky tvoří konvexní kombinace vrcholů úsečky, nebo je trojúhelník tvořen všemi konvexními kombinacemi svých tří vrcholů.

Algoritmus balení dárku (gift wrapping)

Existuje mnoho různých algoritmů pro výpočet konvexní obálky, mimo jiné Grahamův algoritmus (R. Graham 1972), Quickhull algoritmus (W.Eddy 1977 a A. Bykat 1978), Andrewův algoritmus (A. M. Andrew 1979) nebo algoritmus balení dárku (Chand a Kapur 1970 a R. A. Jarvis 1973). Protože jsem ve své práci využil zmiňovaný algoritmus balení dárku, budu se jím nyní zabývat podrobněji.

Algoritmus balení dárku (také známý jako Jarvisův pochod) je algoritmus s časovou náročností $O(nh)$, kde n je počet bodů a h je počet bodů konvexní obálky. Tento algoritmus je vhodný pro malá n a h, nebo pokud očekáváme malé h [6].

Algoritmus začíná s $i = 0$ a bodem p_1, o kterém víme, že se nachází na konvexní obálce (jedná se o bod nejvíce vlevo). Dále vybereme bod p_{i+1}, který tvoří s bodem p_i přímku, od kterého se všechny ostatní body nacházejí vpravo. Tento bod lze najít v čase $O(n)$. Poté změníme i na $i+1$ a opakujeme, dokud $p_h \neq p_0$. Tím získáme konvexní obálku v h krocích. V pseudokódu lze tento proces popsat následovně:

```
Gift_wrapping(S)
    bodObalu = bod nejvíce vlevo v S
    i = 0
    opakuj
        P[i] = bodObalu
        koncovýBod = S[0]
        for j from 1 to |S|
            if(koncovýBod == bodObalu) or (S[j] je vlevo od P[i] a koncovéhoBodu)
            koncovýBod = S[j]                    //nalezen bod nejvíce vlevo
        i = i+1
        bodObalu = koncovýBod
    dokud koncovýBod == P[0]                     //jsme zpět na prvním bodě obalu
```

2.4 Voroného diagram

Nechť E^n je n-rozměrný euklidovský prostor a S konečná nespojitá množina K bodů z_i na tomto prostoru (někdy se body z_i také nazývají generátory Voroného diagramu) [12]. Pro každé dva body $p, q, Q \in S; p \neq q$ definujeme dělící nadrovinu $P(p,q) \equiv P(q,p)$ a dva poloprostory *D(p, q), D(q, p)*:

$$\forall x \in P(p,q); |p\ x| = |q\ x|$$

$$\forall x \in D(p,q); |p\ x| < |q\ x|$$

$$\forall x \in D(q,p); |p\ x| > |q\ x|$$

kde */p x/* a */q x/* je vzdálenost bodu p resp. q od bodu x. Potom Voroného buňka *v(P,S)* bodu p z množiny S je definována průnikem:

$$v(P,S) = \bigcap_{Q \in S \setminus P} D(p,q)$$

a Voroného diagram *Vor(S)* definujeme jako sjednocení hranic všech těchto buněk na množině *S*. Toto lze zapsat:

$$Vor(S) = E^n\left(\bigcup_{P \in S} v(P,S)\right)$$

Z definice lze poznat, že Voroného diagram rozděluje prostor E^n na m buněk. Každá z nich pak obsahuje veškeré body z_i, jejichž poloha je nejblíže k jednomu bodu z_j z množiny *S*. Pokud máme vzdálenostní funkci *d(p, q)* pro body p, q, můžeme definovat Voroného buňku V_j jako:

$$V_j = \{Q \in S \mid d(Q, z_j) < d(Q, z_i), i = 1, \dots, K, i \neq j\}$$

Všechny tyto Voroného buňky jsou pak konvexními mnohoúhelníky. Některé buňky jsou neuzavřené – korespondují s body konvexní obálky.

Centroidní Voroného diagram

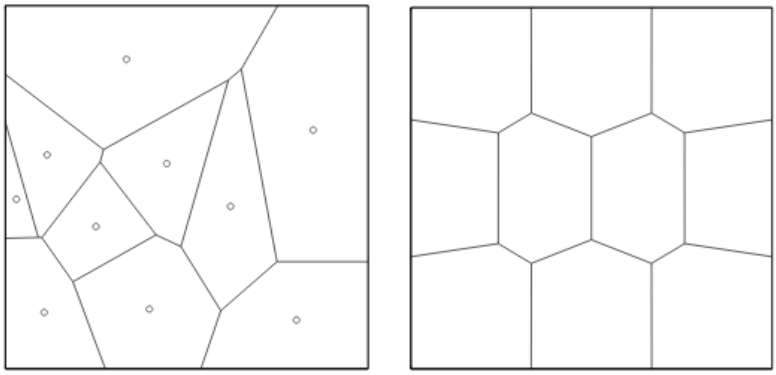
Mějme K generátorů z_i, množinu Voroného buněk V_i a množinu K centroidů Voroného buněk z_i^*. Běžně platí, že:

$$z_i \neq z_i^*, \qquad i = 1, \dots, K$$

V případě, že pro všechny generátory $\mathbf{z}_i$ platí:

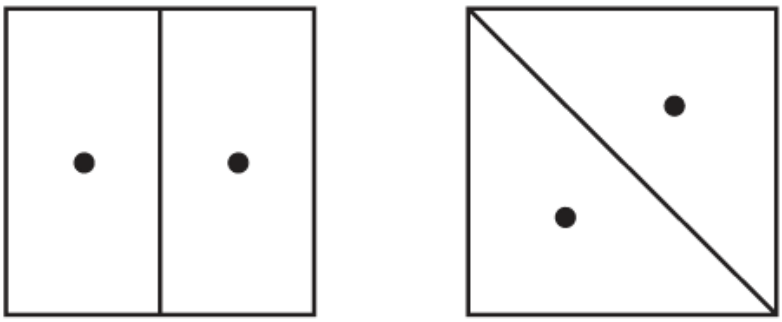
$$z_i = z_i^*, \qquad i = 1, \dots, K$$

nazýváme Voroného diagram centroidním (CVT) [5] – všechny generátory Voroného buněk jsou zároveň jejich geometrickým těžištěm – centroidem (viz obr. 2.2).



Obr. 2. 2 Nalevo: Voroného buňky a jejich centroidy pro 10 bodů v čtverci; generátory a centroidy jsou různé. Napravo: Voroného buňky pro 10 bodů v čtverci; generátory a centroidy jsou totožné

Za normálních okolností je šance, že množina bodů $\{z_i\}_{i=1}^{K}$ bude mít vlastnosti CVT (generátory Voroného diagramu jsou zároveň centroidy), nulová. Proto je třeba CVT sestrojit pomocí jiných metod. Sestrojení CVT je však samo o sobě problematické z důvodu, že žádné rozložení vrcholů nemá unikátní CVT (viz obr. 2.3).



Obr. 2. 3 Dva různé dvoubodové Voroného diagramy stejného čtverce

CVT lze využít i pro jiné diskrétní množiny dat (například vektory v R^n). Pro tyto diskrétní množiny dat lze na CVT nahlížet jako na shlukovací algoritmus – způsob

rozdělování množin na podmnožiny, které mají podobné vlastnosti (například vzdálenost od generátoru Voroného buňky nebo od geografického středu). Lze tedy v případě shlukování označit CVT za zobecněnou formu k-means shlukovacího algoritmu.

Nejdůležitější vlastnost CVT lze popsat pomocí Gershovy domněnky prokázané ve dvou dimenzích. Ta říká, že pro každou množinu bodů, při dostatečně velkém množství bodů, je distribuce bodů v CVT lokálně uniformní – tedy vezmeme-li dostatečně malý vzorek bodů z množiny, bude distribuce bodů CVT vypadat uniformně nehledě na uniformnost bodů v celé množině [5]. Ve dvou dimenzích to tedy znamená, že CVT Voroného buněk budou vždy pravidelné rovnostranné šestiúhelníky – lokální body jsou vrcholy rovnostranných trojúhelníků. Taková data jsou výhodná pro Delaunayho triangulace, neboť u nich chceme docílit co nejvíce rovnostranných trojúhelníků. Pro více než dvě dimenze však Gershova domněnka není prozkoumána a vzhled CVT není známý.

CVT se využívá v mnoha různých oborech. Lze jej zejména využít pro shlukování, optimální rozdělení prostředků (například problém umístění poštovních schránek), generování sítí, distribuci bodů a vytváření sítí na různém povrchu, pro numerické řešení parciálních diferenciálních rovnic, pro kompresi dat či segmentaci obrázků.

Nyní popíšeme dva základní algoritmy výpočtu CVT.

Lloydova metoda

Lloydova metoda, také známá jako Voroného iterace, je algoritmus pojmenovaný po svém autorovi Stuartu P. Lloydovi a slouží k nalezení rovnoměrně rozložených množin bodů v podprostoru Euklidovských prostorů a k rozdělení těchto podmnožin na konvexní buňky uniformní velikosti [9]. Slouží tedy k vytváření centroidních Voroného diagramů.

Stejně jako některé k-means shlukovací algoritmy postupně nalézá centroidy daných množin bodů v oblasti a následně předělá tyto oblasti bodů podle toho, kde se centroidy nalézají.

Od ostatních k-means algoritmů se Lloydova metoda liší hlavně tím, že používá Voroného diagramy místo toho, aby určila nejbližší střed pro každý bod z konečné množiny bodů.

Postup Lloydovy metody pro vstupní množinu K bodů $\{z_i\}_{i=1}^{K}$ lze zapsat takto:

1. Sestrojíme Voroného diagram $\{V_i\}_{i=1}^{K}$ oblasti, kterou tvoří body $\{z_i\}_{i=1}^{K}$
2. Sestrojíme centroidy Voroného buněk $\{V_i\}_{i=1}^{K}$ nalezených v kroku 2; těmito centroidy nahradíme původní množinu bodů $\{z_i\}_{i=1}^{K}$
3. Opakujeme postup od kroku 2, nebo ukončíme algoritmus, pokud jsme již s výsledkem spokojení.

McQueenova metoda

McQueenova metoda je metodou náhodného vzorkování a zaokrouhlování. Jedná se o typický k-means shlukovací algoritmus [10]. Postup McQueenovy metody pro vstupní množinu K bodů $\{z_i\}_{i=1}^{K}$ lze zapsat takto:

1. Vybereme náhodný bod w v oblasti, kterou tvoří množina bodů $\{z_i\}_{i=1}^{K}$
2. Zjistíme, které ze z_i je nejblíže k bodu w
3. Najdeme střed na přímce mezi $\boldsymbol{w}$ a nejbližším z_i
4. Vyměníme bod z_i za střed

Proces opakujeme od bodu 2, ale:

- Pamatujeme si, kolikrát jsme přesunuli bod ze své původní pozice na novou
- Pokud hledáme střed, dbáme na to, kolikrát byl bod v minulosti přesunut

Uveďme příklad pro lepší představu:

Mějme bod z_2, který byl již přesunut 12krát (původní pozici považujeme za první přesun), pak:

- Místo toho, aby se počítalo: $z_2 \leftarrow \frac{w+z_2}{2}$, vezme se $z_2 \leftarrow \frac{w+12z_2}{13}$

Z toho lze tedy určit vzorec pro nové z_i:

$$z_i = \frac{w + počet\ přesunů * z_i}{počet\ přesunů + 1}$$

Výhodou McQueenovy metody oproti Lloydově je, že nevyžaduje sestrojení Voroného buněk či nalezení centroidů. Přesto ale platí, že body vzniklé McQueenovou metodou konvergují k centroidům centroidního Voroného diagramu.

Konvergence McQueenovy metody je velmi pomalá a pro získání množiny bodů centroidního Voroného diagramu je třeba mnoho (až miliony) iterací. Důvodem je, že vzorkujeme vždy jen jeden bod před přemístěním.

Pro zrychlení McQueenovy metody lze zavést úpravy, které jsou vhodné pro paralelní výpočet:

- Místo jednoho bodu vezmeme mnoho (někdy i tisíce) bodů a rozdělíme je do oblastí podle jejich nejbližšího z_i
- Nalezneme nejbližší bod k bodu z_i pro každou oblast
- Přesuneme každé ze z_i, stejně jako v původní metodě, a zapamatujeme si, že jsme bod přesunuli

3. Navržené řešení

V této kapitole se budu zabývat samotným řešením problému. Konkrétně si řekneme, jakým způsobem jsme vytvářeli CVT (přemisťovali body), zda námi vybraná McQueenova metoda opravdu vytváří rozložení CVT a jakým způsobem jsme porovnávali chyby interpolace mezi body původního a nového modelu. Postupně si popíšeme hierarchii modulů, jejich vstupní a výstupní data a změny, ke kterým dojde po použití dat v jednotlivých modulech.

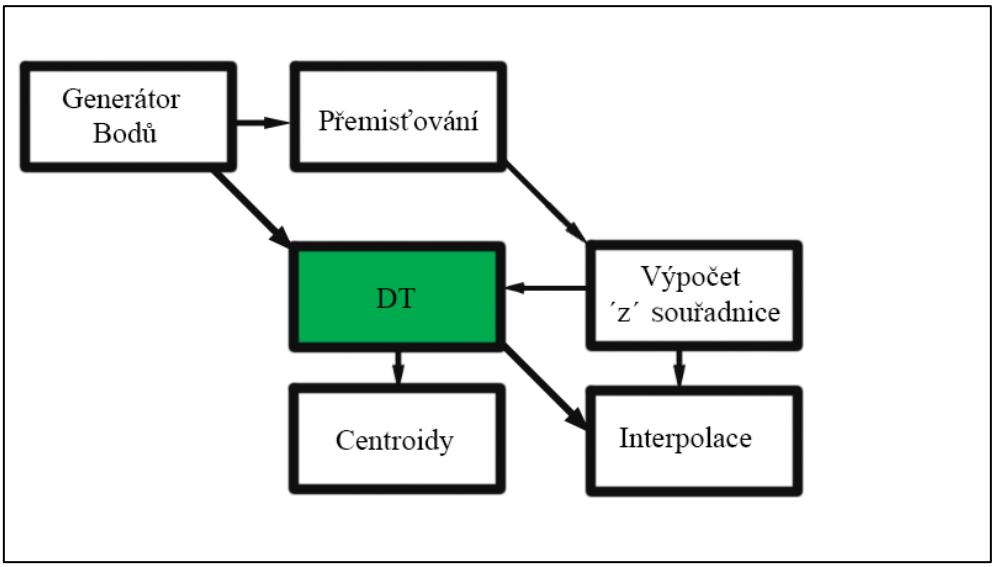
V řešení nejprve potřebujeme vytvořit vstupní data, na kterých lze měřit chybu interpolace. Jakmile máme vstupní data, musíme z nich vytvořit trojúhelníkovou síť pro pozdější použití. Body, které jsme si vytvořili, přemístíme pomocí McQueenovy metody tak, aby jejich rozložení bylo CVT. Kvůli nepřesnostem vzniklým při přesunech pomocí trojúhelníkové sítě přepočítáme *z* souřadnici přemístěných bodů. Nakonec spočteme relativní chybu a střední kvadratickou odchylku vstupních dat. Těmito problémy se zabývají jednotlivé moduly řešení (viz obr. 3.1).

3.1 Moduly

Na začátku popíšeme samotné rozložení modulů realizovaného řešení. Protože jednotlivé dílčí problémy může být zapotřebí řešit samostatně, jsou jednotlivé dílčí moduly v samostatných programech. Celé řešení je rozděleno na tyto moduly:

- Modul pro generování bodů, který vytváří vstupní data
- Modul pro přemisťování bodů, který za pomoci McQueenovy metody přemisťuje body a tím vytváří CVT
- Modul pro Delaunayho triangulace (DT), který vytváří trojúhelníkovou síť (modul DT byl dodán vedoucím práce)
- Modul pro výpočet *z* souřadnice, který pomocí barycentrických souřadnic a trojúhelníkové sítě spočítá přesnější *z* souřadnici vstupních bodů
- Modul pro výpočet odchylky centroidů, který zjišťuje odchylku bodů od centroidů
- Modul pro měření přesnosti interpolace bodů, který z množiny bodů a trojúhelníkové sítě zjišťuje střední kvadratickou odchylku a relativní chybu

Použití výstupů jednotlivých modulů jako vstupy jiných modulů je popsáno na obrázku 3.1.



Obr. 3. 1 Moduly řešení a jejich předávání výstupů do dalších modulů

3.2 Vstupní a výstupní soubory

Z důvodu kompatibility dat, která jednotlivé moduly vytvářejí, s tvarem vstupních dat pro modul DT, jsou omezeny vstupní a výstupní data na dva typy: soubor s trojúhelníky a soubor s body.

Soubor s body

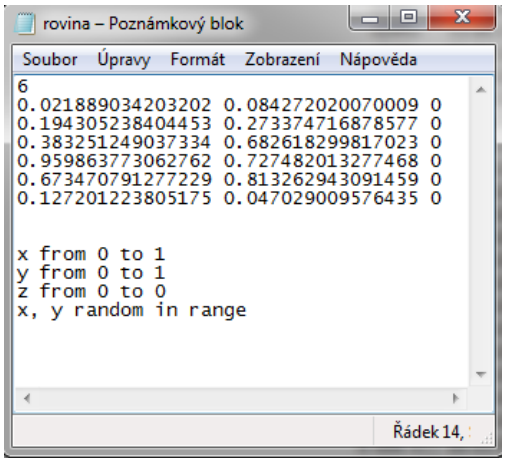
Aby mohl být soubor s body použit jako vstupní soubor pro jednotlivé moduly, musí mít následující formát:

- Počet bodů
- Jednotlivé body ve tvaru $x\,y\,z$

Pokud byl soubor vygenerován modulem pro generování bodů, může dále na konci obsahovat:

- Název generovaného tvaru (plochy)
- Rozmezí x, y a z

Příklad souboru s body je na obrázku 3.2.



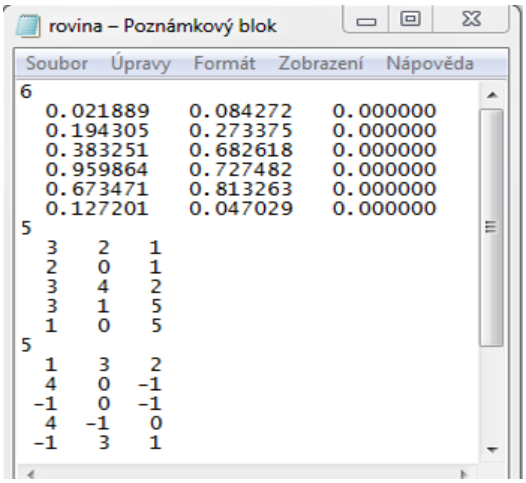
Obr. 3. 2 Příklad souboru s body

Soubor s trojúhelníky

Soubor s trojúhelníky je vytvářen v modulu DT. Má následující tvar:

- Počet vrcholů
- *x y z* souřadnice jednotlivých vrcholů
- Počet trojúhelníků
- Vrcholy jednotlivých trojúhelníků
- Počet sousedů
- Sousedi jednotlivých trojúhelníků (není-li soused, zapisuje se -1)

Příklad souboru s trojúhelníky je na obrázku 3.3:



Obr. 3. 3 Příklad souboru s trojúhelníky

3.3 Vizualizátory

K zobrazení bodů a trojúhelníkových sítí byly použity dva programy dodané vedoucí práce – Viewer a ShowDT.

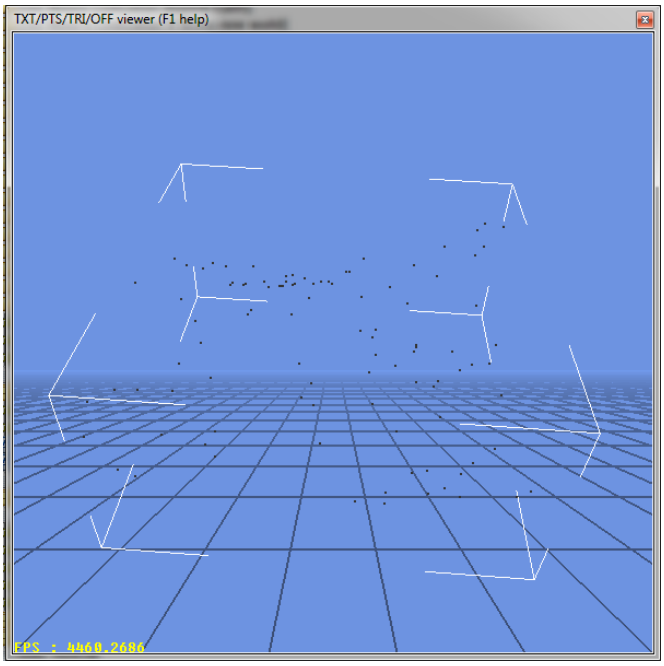
Viewer

Viewer je program pro grafické zobrazení bodů ze souborů s body. Jeho autorem je Ing. M. Varnuška, Ph.D..

Pro spuštění programu Viewer a zobrazení bodů:

- spustíme příkazový řádek a přepneme do adresáře, v němž se nachází program Viewer
- zobrazíme body ze souboru s body pomocí Vieweru přes příkaz: viewer cesta_k_souboru_s_body

Pokud bylo vše zadáno správně, spustí se program Viewer a zobrazí body stejně jako na příkladu na obrázku 3.4.



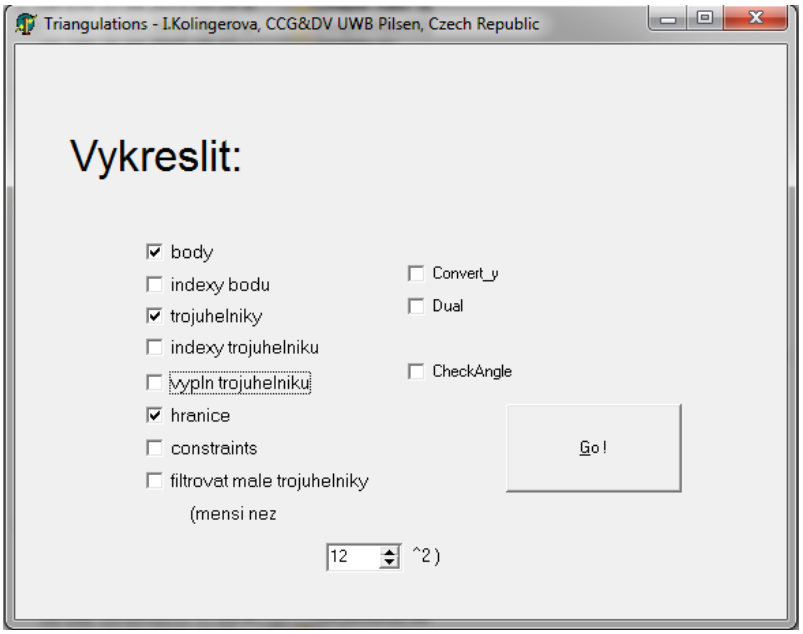
Obr. 3. 4 program Viewer zobrazující funkci "hladké údolí"

ShowDT

Program ShowDT slouží k zobrazování trojúhelníkových sítí a bodů ze souboru s trojúhelníky (viz obr. 3.5). Autorem programu je Prof. Dr. Ing. I. Kolingerová.

Pro zobrazení trojúhelníků ze souboru s trojúhelníky:

- po zapnutí ShowDT odškrtneme možnost trojúhelníky
- stiskneme tlačítko „Go!“ a vybereme soubor s trojúhelníky, které chceme zobrazit

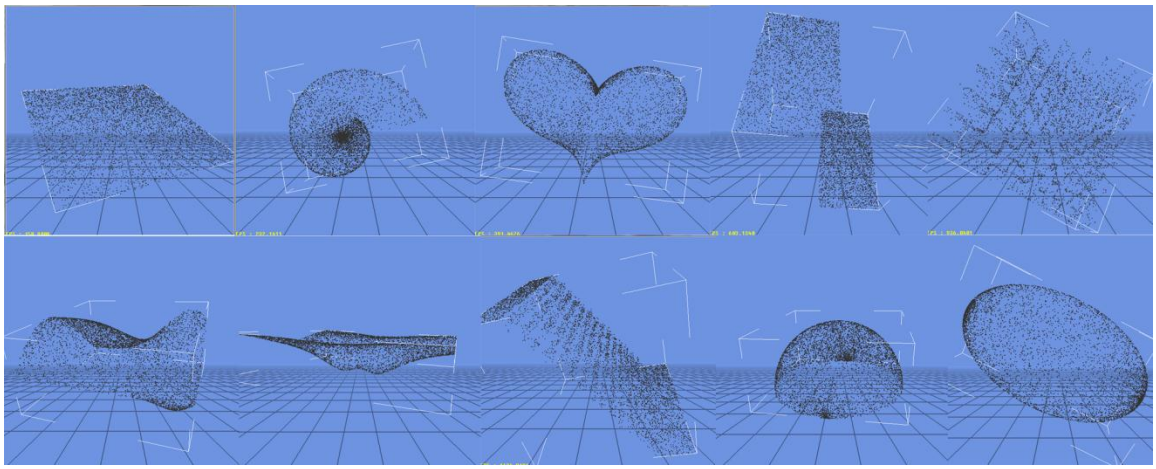


Obr. 3. 5 Grafické uživatelské rozhraní ShowDT

3.4 Modul pro generování bodů

Kromě dat naměřených v terénu je potřeba mít i data, u kterých můžeme spočítat přesné hodnoty souřadnice z, která se změní interpolací. Z tohoto důvodu byl vytvořen a přidán modul pro generování bodů. Tento modul slouží ke generování bodů různých matematických funkcí $z = f(x, y)$ pro x, y generované s rovnoměrným náhodným rozložením na uživatelem zadaném intervalu x a y nebo funkcí $f(x, y, z) = 0$.

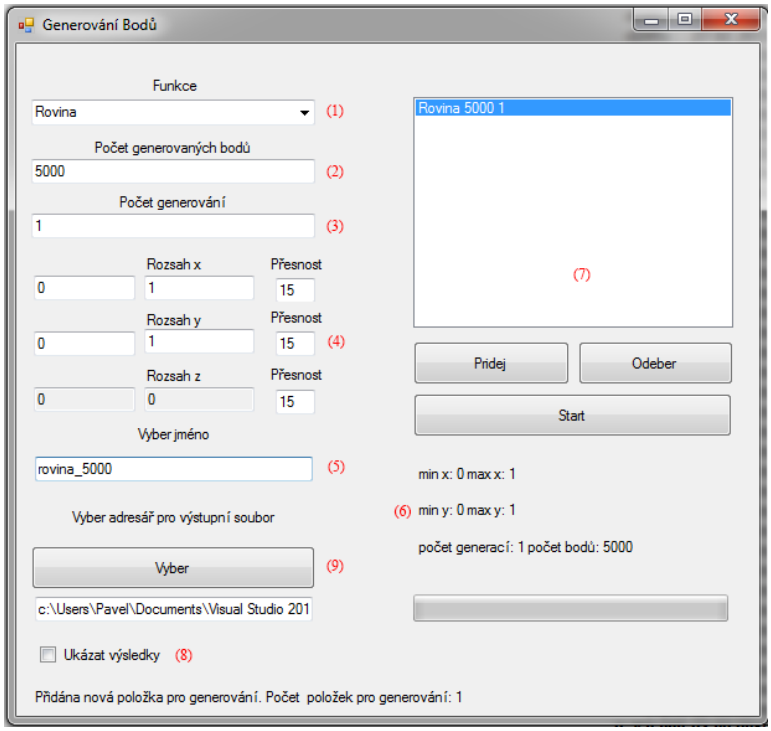
Modul má velmi jednoduchou implementaci. Podle vstupních údajů vybere ze seznamu správnou funkci, kterou má vygenerovat, a poté body uloží do souboru. Modul umí vytvářet 10 rozložení bodů: rovinu, „ulitu“, „srdce“, „schod“, „nové hory“, „hladký kopec“, „dvě údolí“, „divoký schod“, „hladké údolí“ a „elipsoid“. Jejich reprezentace body je na obrázku 3.6.



Obr. 3. 6 Různá rozložení bodů vygenerovaná modulem pro generování bodů. Horní řada zleva: rovina, ulita, srdce, schod, nové hory. Spodní řada zleva: hladké údolí, dvě údolí, divoký schod, hladký kopec, elipsoid

Data jsou generována ve formátu double z důvodu vyšší přesnosti.

Uživatelská dokumentace



Obr. 3. 7 Grafické uživatelské rozhraní modulu pro generování bodů

Uživatelské rozhraní (viz obr. 3.7) modulu pro generování bodů umožňuje uživateli snadno vybrat požadované rozložení bodů. Pro vygenerování bodů:

1. Vybereme požadovanou funkci ze seznamu funkcí.
2. Vybereme požadovaný počet bodů. Minimální počet je 100 bodů.
3. Vybereme počet generování zadané funkce o požadovaném množství bodů.
4. Určíme rozsahy souřadnic *x*, *y* a *z*, pokud je to pro danou funkci možné. Lze také nastavit maximální přesnost za desetinnou čárkou (implicitně 6 čísel)
5. Vybereme jméno výstupního souboru. V případě více generování modul generuje soubory podle vzoru jmenoSouboru_cisloGenerace.
6. Pokud nám vyhovují zadané údaje, stisknutím tlačítka „Přidej“ přidáme funkci do seznamu. V případě, že chceme funkci odebrat, označíme funkci v seznamu a stiskneme tlačítko „Odeber“
7. Po stisknutí tlačítka „Přidej“ se funkce zobrazí v seznamu funkcí pro generování v pravém horním rohu.
8. Pokud chceme vygenerované výsledky zobrazit, zaškrtneme „Ukázat výsledky“. (Tato funkce funguje pouze v případě, že ve stejném adresáři, ve kterém se modul nachází, se vyskytuje program viewer.exe. Tato možnost není doporučovaná pro generování velkého množství rozložení bodů)
9. Vybereme místo, kam se zadané soubory vygenerují. Implicitně je nastaven adresář, ve kterém se modul vyskytuje.
10. Generování nakonec spustíme stisknutím tlačítka Start.

Po stisknutí tlačítka „Start“ se pak vygenerují body a uloží do souborů.

3.5 Modul pro přemisťování bodů

Tento modul slouží pro přemisťování bodů pomocí McQueenovy metody. Tak lze dosáhnout rovnoměrnějšího rozložení bodů funkce po ploše, protože metoda postupně mění rozložení na centroidní Voroného diagramy. Tím lze například dosáhnout menšího počtu artefaktů při vytváření modelu nebo lepších dat pro tvorbu trojúhelníkových sítí (trojúhelníky konvergují k rovnostranným trojúhelníkům).

Algoritmus

Jak již bylo zmíněno, modul přemisťuje body pomocí McQueenovy metody (viz 2.4). Ta byla v práci zvolena především z důvodu jednoduché implementace a rychlosti přemisťování pro menší až středně velké počty bodů (to znamená tisíce až desítky milionů). Byl tedy vybrán hlavně z důvodu optimálního poměru náročnosti ku rychlosti vykonávání.

Algoritmus probíhá tímto způsobem:

- Načteme vstupní množinu bodů P
- Najdeme konvexní obálku množiny P pomocí algoritmu balení dárků
- Postupně vybíráme náhodné body w uvnitř konvexního obalu
- Vždy, když se vybere bod w, nalezneme nejbližší bod z ze vstupní množiny P
- Přesuneme bod z podle pravidel McQueenovy metody a zapamatujeme si, že byl bod přesunut
- Opakujeme výběry w a přesuny z podle počtu vstupních iterací

Omezení přemisťování bodů

Algoritmus můžeme dále vylepšit vzorkováním velkého množství bodů w nezávisle na sobě a nalezením jejich nejbližšího z. Algoritmus by tedy mohl paralelně vykonávat McQueenovu metodu na několika bodech, aby snížil časovou náročnost. K tomu je zapotřebí rozdělit množinu P na podmnožiny z důvodu bezpečnosti – mohlo by totiž dojít k situaci, kdy by dva body w byly stejně vzdálené od jednoho z a chtěly by jej přesunout dle pravidel metody. Tím by docházelo k chybě. Z tohoto důvodu modul vzorkuje w vždy po jednom bodu.

Přesnost

Z důvodu velkého množství přesunů bodů je nutné upozornit na fakt, že souřadnice z má vzhledem k mnoha přepočtům sníženou přesnost, což znamená, že každá iterace zavádí při výpočtu malou chybu. Jelikož je zapotřebí velkého množství iterací, chyba se neustále zvyšuje. Proto je vhodnější výsledné body předat do modulu pro výpočet souřadnice z a souřadnice z pro ně spočítat z původních bodů interpolací na trojúhelníkové síti.

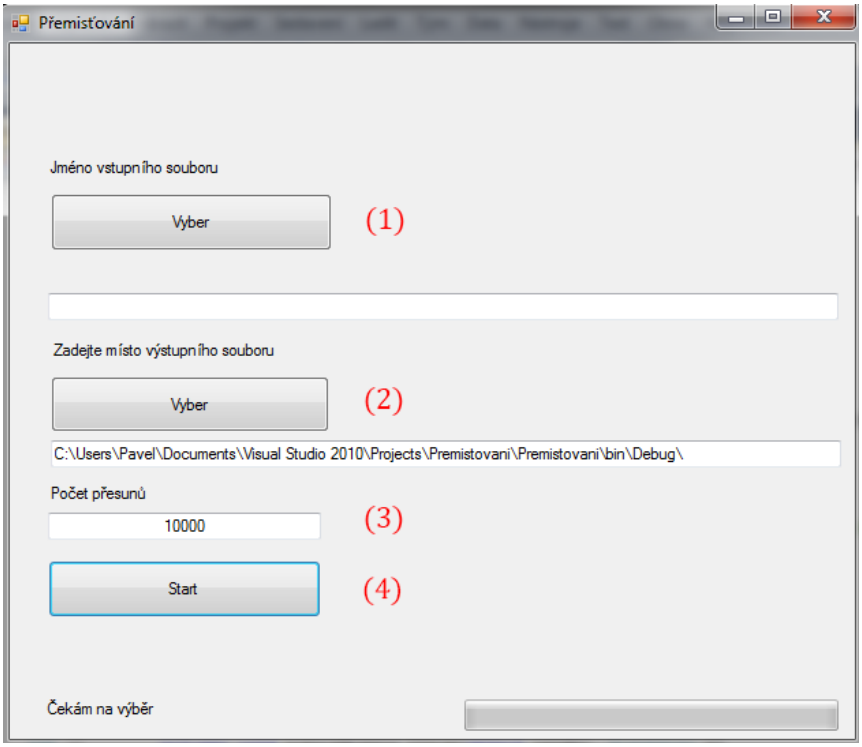
Uživatelská dokumentace

Okno aplikace pro přemisťování bodů je zobrazeno na obrázku 3.8.

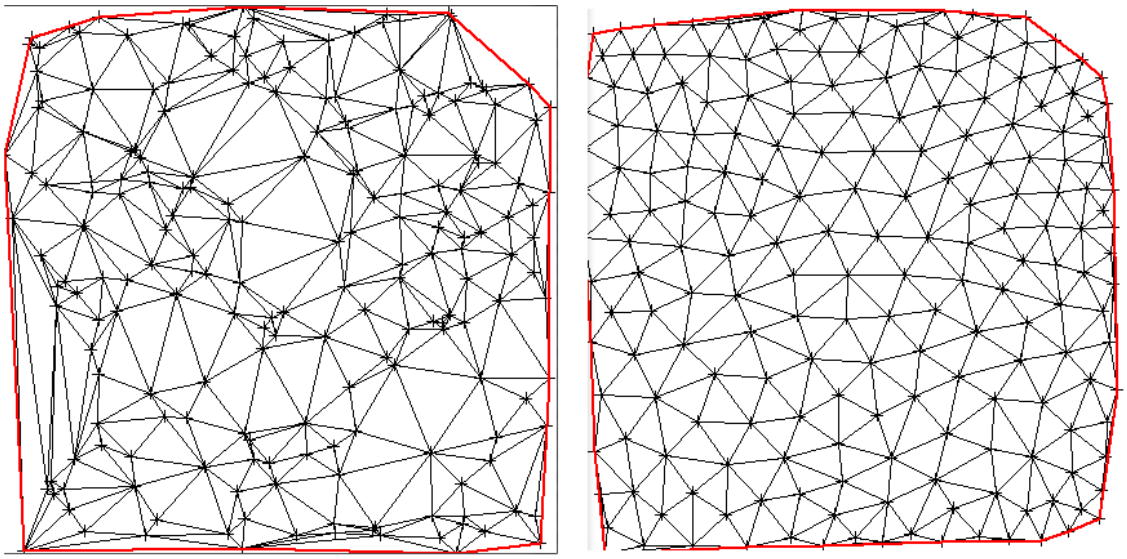
Pro přemístění bodů v GUI modulu pro přemisťování bodů:

1. Zadejte jméno vstupního souboru
2. Zvolte jméno výstupního souboru
3. Zvolte počet přesunů. Je třeba dbát na to, že pokud chceme dosáhnout výsledku co nejbližšímu centroidnímu Voroného diagramu, je zapotřebí mnoha (až milionů) iterací, počet iterací je třeba nadále zvyšovat v závislosti na počtu vstupních bodů. Minimální doporučený počet je 10000 přesunů, ale vhodnější je vybrat vyšší počet.

Nakonec stiskneme tlačítko „Start“. Příklad výsledku lze vidět na obrázku 3.9.

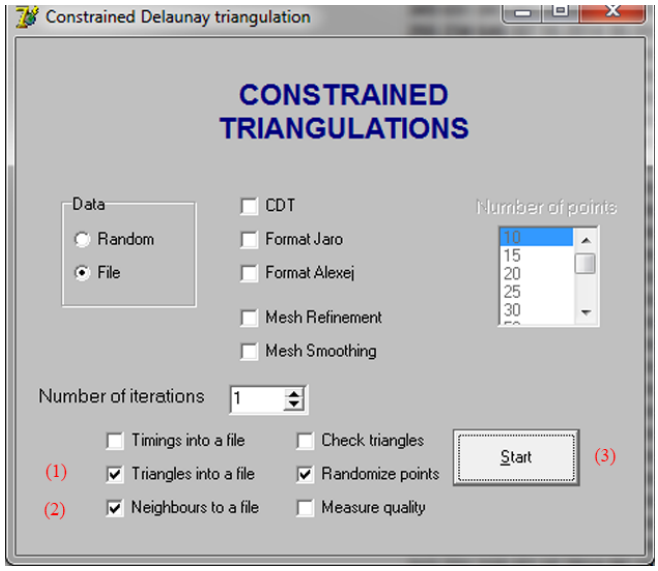


Obr. 3. 8 Grafické uživatelské rozhraní modulu pro přemisťování bodů



Obr. 3. 9 Triangulace 200 bodů před přemístěním bodů (vlevo) a po 100000 přemístěních (vpravo)

3.6 Modul pro Delaunayho triangulace (DT)



Obr. 3. 10 Grafické uživatelské rozhraní modulu pro Delaunayho triangulace

Tento modul slouží pro vytváření Delaunayho triangulací (dále DT) ze souborů s body (viz obr. 3.10). Program byl dodán vedoucí práce a pro naše účely je třeba dbát na zapnutí voleb tak, aby:

1. do výsledného souboru byly vygenerovány trojúhelníky
2. do výsledného souboru byli vygenerováni sousedi trojúhelníků

Poté už jen zvolíme vstupní soubor a místo a název výstupního souboru (v dialogových oknech, která se objeví po stisknutí tlačítka „Start“).

Autorem modulu je Prof. Dr. Ing. Ivana Kolingerová.

3.7 Modul pro výpočet ‚z' souřadnice

Tento modul slouží pro výpočet souřadnice z v souborech s body, které byly zpracovány modulem pro přemístění bodů. Modul ke svému chodu vyžaduje soubor s body, které byly přemístěny v modulu pro přemisťování bodů, a soubor s trojúhelníky vytvořený z původních bodů před přemístěním v modelu pro Delaunayho triangulace. Z nich poté modul vytvoří pomocí barycentrických souřadnic soubor s přesnějšími souřadnicemi z.

Tento modul potřebujeme, protože v modulu po přemisťování bodů dochází k vytváření chyby v souřadnici z. K ní dochází kvůli velkému množství přepočtů souřadnic jednotlivých bodů. Proto je třeba souřadnici z přepočítat. Tento krok úmyslně není součástí modulu pro přemisťování bodů, aby bylo možné tento modul využít i pro jiná vstupní data než jsou body přepočtené CVT.

Algoritmus

Jak již bylo zmíněno dříve, modul vypočítává přesnější z souřadnice pomocí barycentrických souřadnic. Algoritmus postupuje následovně:

- Načte se vstupní množina bodů p a trojúhelníková síť T
- Pro každý bod p zjistíme, do kterého trojúhelníku sítě T náleží.
- Spočteme barycentrické souřadnice bodů p pomocí Cramerova pravidla (viz kapitola 2.1 barycentrické souřadnice)
- Z barycentrických souřadnic přepočteme z souřadnici bodů p

Je dobré upozornit na fakt, že modul je schopný tímto způsobem dopočítávat z bodů p, pokud je neznáme – na nalezení barycentrických souřadnic nám stačí pouze souřadnice x a y u bodů p.

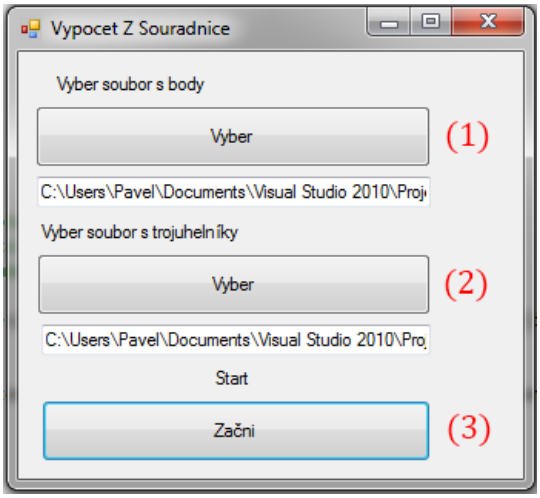
U algoritmu pro výpočet barycentrických souřadnic pomocí Cramerova pravidla lze oproti popsanému algoritmu v kapitole 2.1 dosáhnout vyšší efektivnosti v případě, že si jmenovatel vypočítáme předem. Tím v programu snížíme počet nutných elementárních operací přibližně o 20 (záleží na použitém programovacím jazyce).

Uživatelská dokumentace

Pro vytvoření souboru s přesnějšími z souřadnicemi (viz obr. 3.11):

1) Vybereme soubor, který jsme si připravili v modulu pro přemisťování bodů. Poté vás modul vyzve k zadání jména souboru, do kterého budou výsledky zpracovány
2) Vybereme soubor, který obsahuje trojúhelníky na původních nepřemístěných datech

Nakonec stiskneme tlačítko „Start“ a počkáme na výsledky. Výsledný soubor obsahuje přesněji spočtené souřadnice, a proto je vhodnější pro měření přesnosti interpolace.



Obr. 3. 11 Grafické uživatelské rozhraní modulu pro výpočet z souřadnice

3.8 Modul pro výpočet odchylky centroidů

Účelem modulu je určení odchylky centroidů v dané triangulaci oproti vypočítaným centroidům. K tomu je zapotřebí vstupního souboru s trojúhelníky. Z něj pak modul určí průměrnou a maximální odchylku, které naměřil. Ověření toho, zda jednotlivé body (centroidy) po přemístění bodů jsou s minimální odchylkou stejné jako centroidy spočítané, je důležité z důvodu ověření toho, že naše implementace McQueenovy metody vrací body, které při dostatečném množství iterací konvergují k centroidům CVT.

Algoritmus

Algoritmus funguje na jednoduchém principu, ale jeho implementace byla mírně složitá.

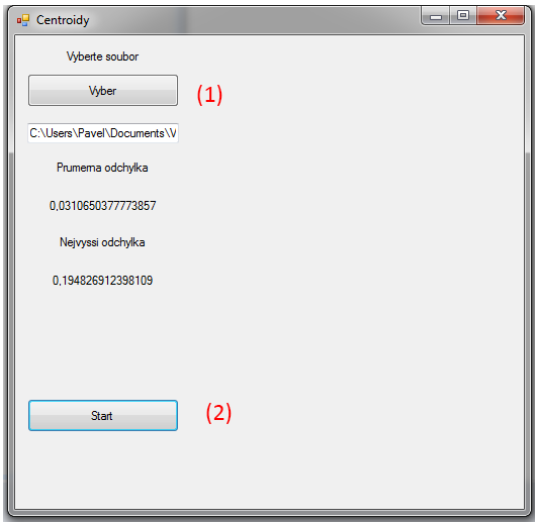
Předpokládáme, že každý bod, kromě bodů na hranici, je centroidem. Proto postupně bereme body, najdeme, ve kterých trojúhelnících je aktuální bod vrcholem, a ze zbylých vrcholů těchto trojúhelníků vytvoříme polygon S, pro který považujeme tento bod za centroid. Dále spočítáme centroid plochy S a zjistíme odchylku mezi bodem a centroidem. Postup lze tedy zapsat takto:

- Načte se vstupní množina bodů p, které tvoří trojúhelníkovou síť T
- Vytvoříme polygon z bodů a trojúhelníků S, ve kterých se p nachází
- Pokud není p součástí konvexní obálky, zjistíme, ve kterém trojúhelníku se nachází, a body q a r, které spolu s p tvoří trojúhelník, vložíme do S
- Zjistíme, který bod z q a r je dál po směru hodinových ručiček, a najdeme další bod o, který spolu s p a r (popřípadě q) tvoří trojúhelník. Bod o přidáme do množiny S. Tímto způsobem postupujeme do doby, než najdeme celou plochu,

ve které se bod p nachází (tedy do doby, než by další vybrané o mělo být prvním z bodů, který se v množině S nachází).
- Spočteme centroid C oblasti S (viz 2.1 centroid)
- Spočteme odchylku mezi C a p.

Výše popsaný algoritmus jsme vylepšili tak, že při hledání bodů o nemusíme prohledávat celou trojúhelníkovou síť T, protože známe sousední trojúhelníky trojúhelníku pqr. Hledání prvního trojúhelníku, ve kterém se bod p nachází, je sekvenční prohledávání v čase $O(N)$, protože nebyl kladen důraz na efektivitu.

Uživatelská příručka



Obr. 3. 12 Grafické uživatelské rozhraní modulu pro výpočet z souřadnice

Ovládání modulu je velice snadné (viz obr. 3.12). Je třeba pouze:

1) Vybrat soubor s trojúhelníky
2) Spustit program

Po stisknutí tlačítka „Start“ je třeba vyčkat do doby, než se spočítá průměrná odchylka a nalezne maximální odchylka.

3.9 Modul pro měření přesnosti interpolace bodů

Modul pro měření přesnosti interpolace bodů slouží k výpočtu střední kvadratické odchylky a relativní chyby mezi z souřadnicí vzniklou z interpolací bodů ku z souřadnicí spočtené z barycentrických souřadnic bodů trojúhelníku, ve kterém se nachází, nebo ku přesným z souřadnicím (známe-li funkci, ze které pochází množina vstupních bodů, nebo máme-li soubor, ve kterém správné z souřadnice dodáme). Ke svému chodu vyžaduje modul soubor s body, které byly přemístěny v modulu pro přemisťování bodů, a soubor s trojúhelníky vytvořený z původních bodů před přemístěním v modulu pro Delaunayho triangulace.

Pro střední kvadratickou odchylku byl použit vzorec:

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(přesná\ hodnota\ z_i - spočtená\ hodnota\ z_i)^2 \qquad (3.1)$$

a pro relativní chybu vzorec:

$$\delta x = \frac{1}{N}\sum_{i=1}^{N}\frac{|přesná\ hodnota\ z_i - spočtená\ hodnota\ z_i|}{\max z_i - \min z_i} * 100\ [\%] \qquad (3.2)$$

Algoritmus

Podle vybraného způsobu měření přesnosti (přes barycentrické souřadnice, přes známé funkce nebo ze souboru s hodnotami) algoritmus probíhá tímto způsobem:

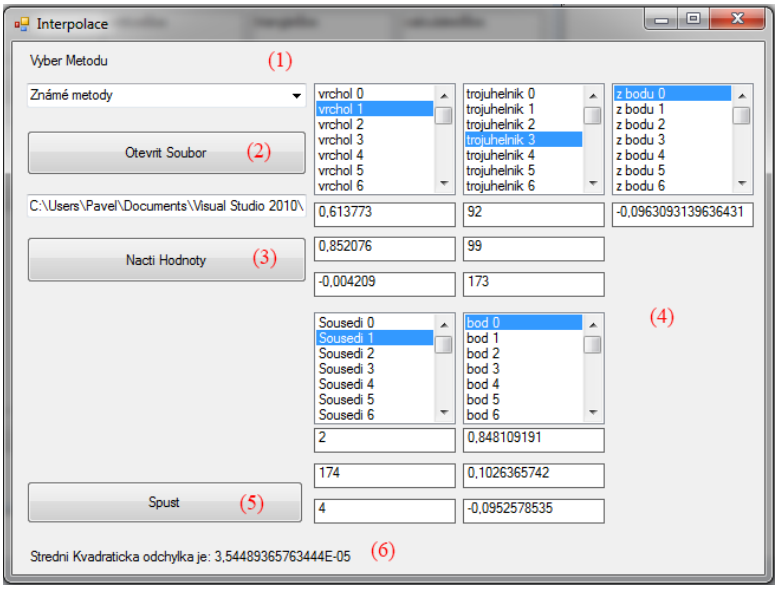
- Pokud bylo vybráno měření přes barycentrické souřadnice pro každý bod *p*, zjišťujeme, ve kterém trojúhelníku se nachází. Poté spočteme jeho barycentrické souřadnice a vypočteme z
- Pokud bylo vybráno měření přes známé funkce, algoritmus najde příslušnou funkci a spočte přesnou z souřadnici
- Pokud bylo vybráno měření přes soubor s hodnotami, algoritmus vezme hodnoty ze souboru a porovná je s hodnotami interpolovaných bodů

Po zjištění všech odchylek bodů modul spočítá střední kvadratickou odchylku.

Uživatelská příručka

Pro změření odchylky interpolace pomocí uživatelského rozhraní (viz obr 3.13):

1) Vybereme metodu měření
2) Vybereme soubor s trojúhelníky a soubor s body
3) Stiskneme tlačítko pro načtení hodnot
4) Pokud načtení proběhlo úspěšně, v pravých seznamech by se měly objevit vrcholy, trojúhelníky, seznam sousedních trojúhelníků a body
5) Stiskneme tlačítko „Spusť“. Pokud jsme vybrali způsob měření „hodnoty ze souboru“, program bude vyžadovat soubor s hodnotami
6) Po skončení výpočtu se výsledek objeví ve stavovém řádku



Obr. 3. 13 Grafické uživatelské rozhraní modulu pro měření přesnosti interpolace bodů

4. Experimenty a výsledky

Kapitolu Experimenty a výsledky jsem rozčlenil podle několika různých kritérií. Testy po dohodě s vedoucí práce probíhaly na dvou druzích triangulace – Delaunayho triangulaci (DT) a lokálně minimální triangulaci (LT).

Experimenty probíhaly s různými počty bodů - od 100 po 10000 bodů. Experimenty dále probíhaly pouze na rovnoměrně náhodně rozložených bodech, protože podle Gershovy podmínky (viz kapitola 2.4) by měly být výsledky nezávislé na rozložení bodů (body jsou po dostatečném počtu iterací vždy lokálně uniformní nehledě na počáteční rozložení).

Experimentovalo se s různým počtem iterací – od 10000 po 2000000. Iteracemi je myšlen počet přesunů bodů pomocí McQueenovy metody, která vytváří z bodů centrodní Voroného diagramy (viz kapitola 2.4).

V části 4.1 nejprve testujeme závislost velikosti chyby na počtu bodů při stejném počtu iterací. Body budou testovány na třech konfiguracích bodů – „hladké údolí“, „nové hory“ a „dvě údolí“, u kterých známe přesné hodnoty z (rozložení budou více popsána v kapitole 4.1).

V části 4.2 se budeme zabývat závislostí velikosti chyby na počtu iterací – tedy počtu přesunu bodů v terénu, na kterém se nachází.

V části 4.3 se budeme zabývat tím, jak přesuny bodů změnily vzhled terénu a případně narušily jeho celkový vzhled (Tedy zda některé zajímavé části zmizely apod.).

V části 4.4 se budeme věnovat orientačním zátěžovým testům a jejich výsledkům.

Pro měření přesnosti interpolace bodů jsme použili dva vzorce uvedené v kapitole 3 – prvním je vzorec pro střední kvadratickou odchylku 3.1. Pro měření relativní chyby byl použit vzorec 3.2.

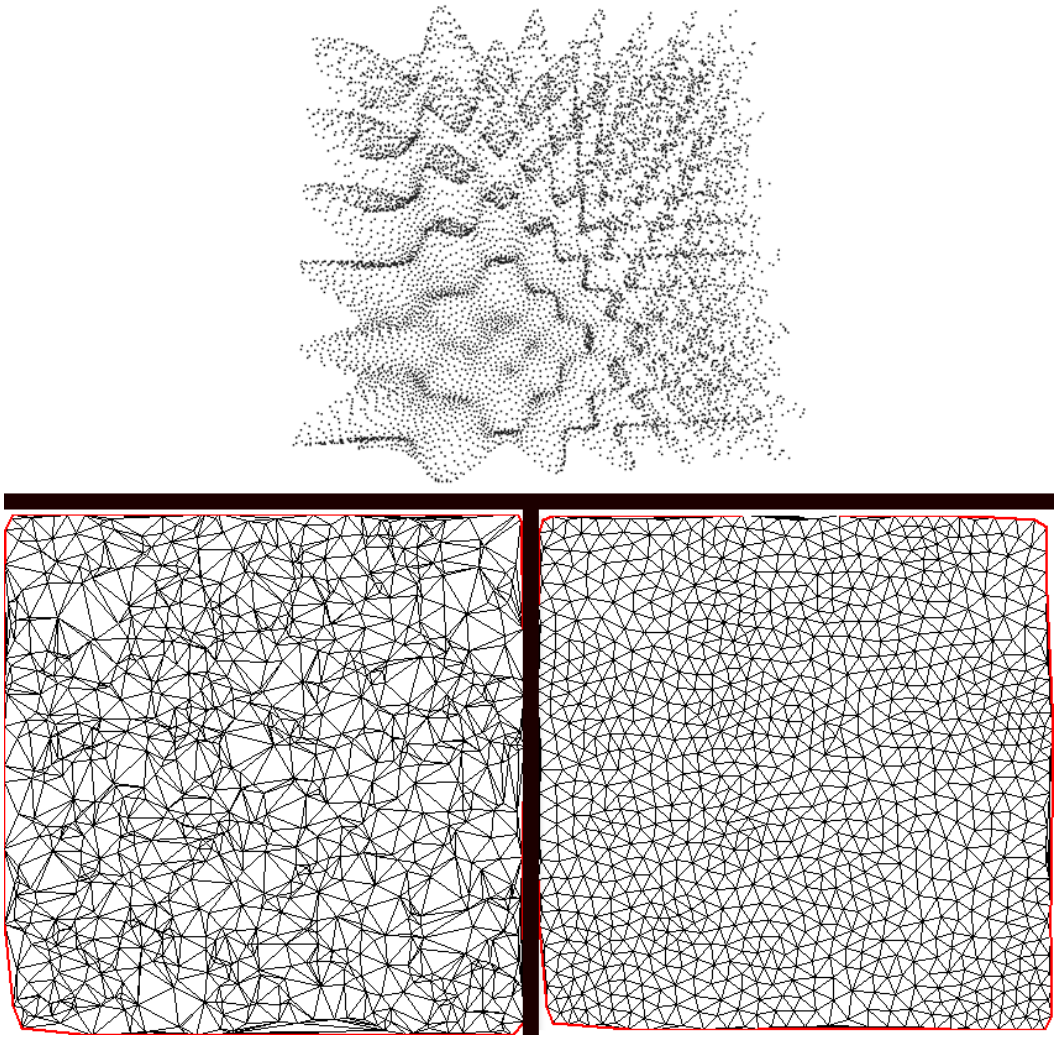
Výpočty probíhaly na čtyř-jádrovém procesoru AMD Phenom II X4 945 @3GHz.

4.1 Experiment 1: Měření závislosti chyby na počtu bodů

Následující testy se zaměřují na závislost chyby vypočtených hodnot z_i oproti skutečným hodnotám na počtu bodů. Pro všechny tyto testy byl použit pevný počet iterací, a to 2 miliony. Pro první pokus byl použit model „nové hory“, který je definován funkcí:

$$f(x,y) = \frac{1}{2}\frac{\cos(40 \times x^2) \times \sin(40 \times y)}{2}$$

První testovaná data mají velmi časté změny v souřadnici z, a proto lze funkci označit jako funkci „divokou“, viz obrázek 4.1. Test probíhal pro šest množin bodů s $N = 100$ až 10000. První testy ukázaly, že střední kvadratická odchylka i relativní chyba s růstem počtu bodů pro tento druh funkce klesá.

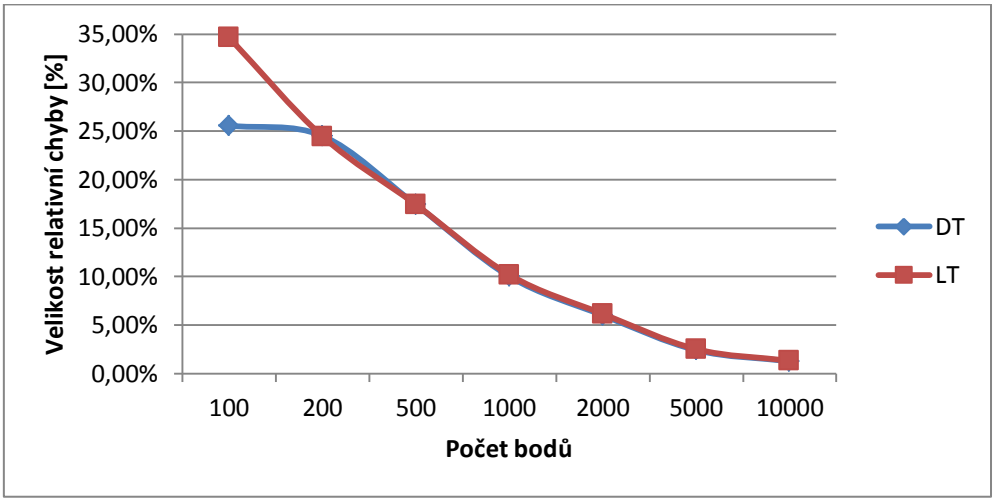


Obr. 4. 1 Funkce "nové hory"; nahoře funkce reprezentovaná 1000 body; vlevo dole původní trojúhelníková síť, vpravo trojúhelníková síť této funkce po 2000000 přesunech

Přesné výsledky experimentu jsou uvedeny v tabulce 4.1 a grafu 4.2.

		DT		LT	
Počet bodů	Počet iterací	Střední kvadratická odchylka	Relativní chyba [%]	Střední kvadratická odchylka	Relativní chyba [%]
100	2000000	0,0198	25,57%	0,0219	34,63 %
200	2000000	0,0177	24,50%	0,0176	24,44%
500	2000000	0,0107	17,39%	0,0114	17,41%
1000	2000000	0,0051	10%	0,0051	10,17%
2000	2000000	0,0004	6%	0,0021	6,15%
5000	2000000	0,0003	2,41%	0,0002	2,51%
10000	2000000	0,0001	1,26%	0,0001	1,31%

tab. 4. 1 Výsledky testů pro "nové hory"

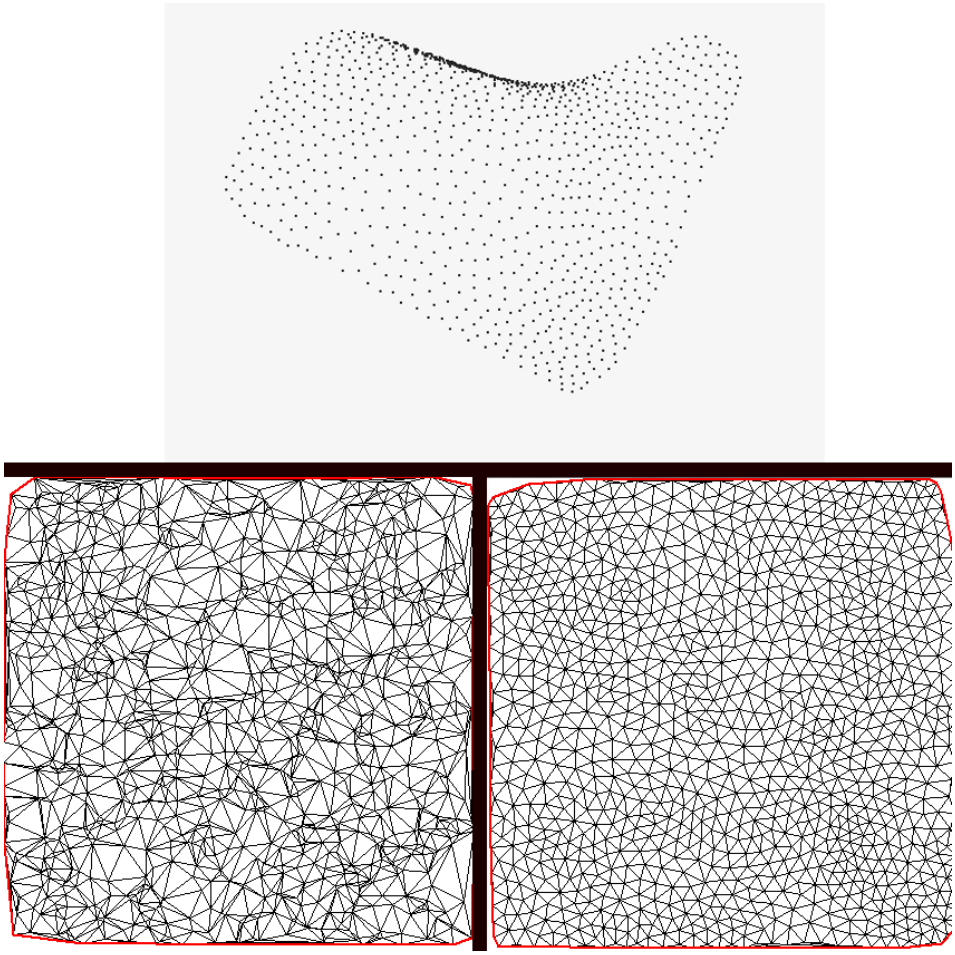


Obr. 4. 2 Graf závislosti relativní chyby [%] na počtu bodů funkce

Dalšími testovanými daty byl model „hladké údolí“ s definicí:

$$f(x,y) = \frac{1}{4}(\cos(4x^2)\sin(4y))$$

Funkce „hladké údolí“ má mírnější změny v souřadnici z oproti „novým horám“, viz obr. 4.3. Výsledky testů ukázaly, že stejně jako u „nových hor“ u „hladkého údolí“ střední kvadratická odchylka i relativní chyba klesají s rostoucím počtem bodů.

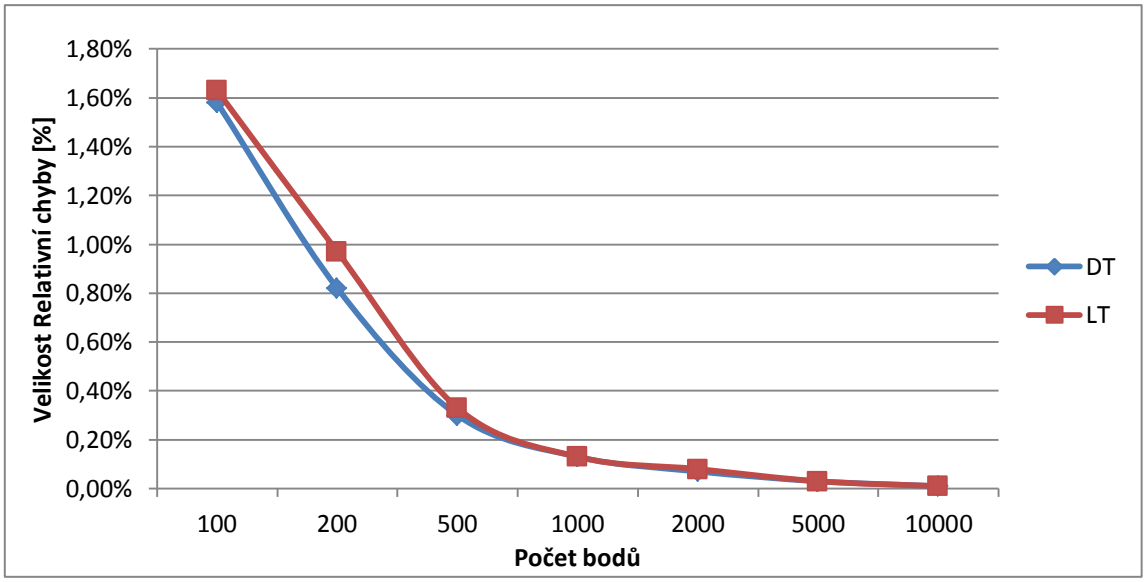


Obr. 4. 3 Funkce "Hladké údolí"; nahoře funkce reprezentovaná 1000 body; vlevo dole původní trojúhelníková síť, vpravo trojúhelníková síť této funkce po 2000000 přesunech

Přesné výsledky experimentu jsou v tabulce 4.2 a grafu v obrázku 4.4.

		DT		LT	
Počet bodů	Počet iterací	Střední kvadratická odchylka [10^{-6}]	Relativní chyba [%]	Střední kvadratická odchylka [10^{-6}]	Relativní chyba [%]
100	2000000	112,352	1,58%	122,291	1,63 %
200	2000000	452,22	0,82%	464,642	0,97%
500	2000000	6,279	0,30%	7,781	0,33%
1000	2000000	102,35	0,13%	102,484	0,13%
2000	2000000	0,395	0,07%	0,463	0,08%
5000	2000000	0,054	0,03%	0,058	0,03%
10000	2000000	0,014	0,01%	0,016	0,01%

tab. 4. 2 Výsledky testů pro "hladké údolí"

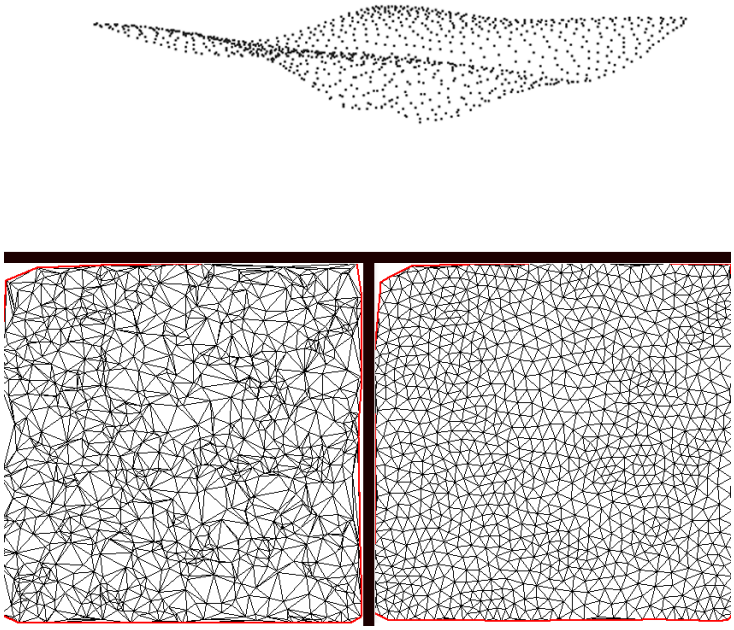


Obr. 4. 4 Graf závislosti relativní chyby [%] na počtu bodů funkce

Posledními testovanými daty byla funkce „dvě údolí“. Jedná se o funkci, která má velmi málo změn souřadnice z vzhledem k rozsahu *x*, *y* souřadnic, viz obr. 4.5. Funkce je definována jako:

$$f(x,y) = \frac{(x^2-1)(y^2-4)+x^2+y^2-5}{(x^2+y^2+1)^2}$$

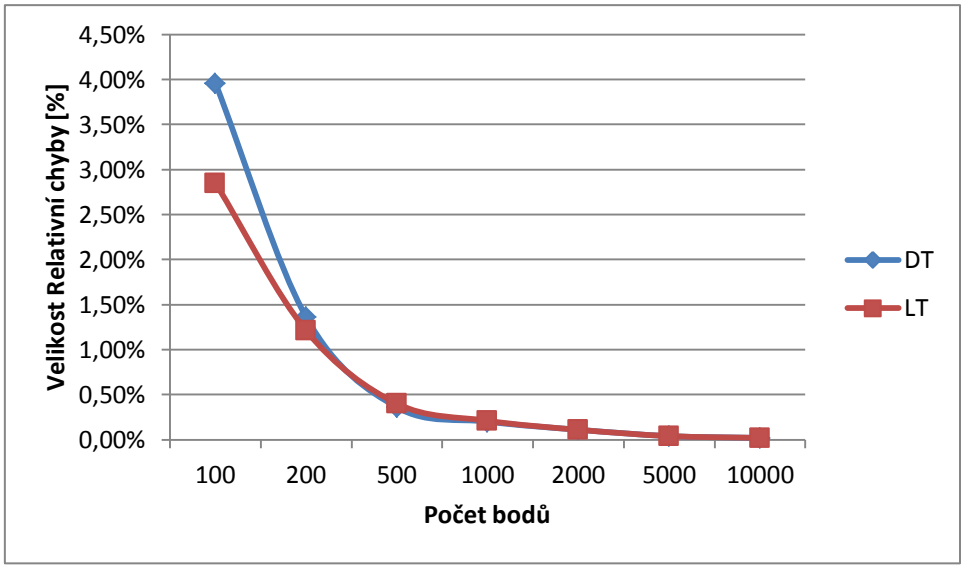
I zde výsledky ukázaly, že velikost střední kvadratické odchylky a relativní chyby klesá s vyšším počtem bodů. Přesné výsledky experimentu jsou v tabulce 4.3 a z grafu v obrázku 4.6.



Obr. 4. 5 Funkce "Dvě údolí"; nahoře funkce reprezentovaná 1000 body; vlevo dole původní trojúhelníková síť, vpravo trojúhelníková síť této funkce po 2000000 přesunech

		DT		LT	
Počet bodů	Počet iterací	Střední kvadratická odchylka [10^{-6}]	Relativní chyba [%]	Střední kvadratická odchylka [10^{-6}]	Relativní chyba [%]
100	2000000	5001,586	3,96%	4615,023	2,85%
200	2000000	1372,878	1,36 %	942,155	1,21%
500	2000000	74,210	0,36%	106,152	0,40%
1000	2000000	25,634	0,20%	30,275	0,21%
2000	2000000	10,483	0,11%	11,047	0,11%
5000	2000000	1,567	0,04%	1,590	0,04%
10000	2000000	0,442	0,02%	0,490	0,02%

tab. 4. 3 Výsledky testů pro "dvě údolí"



Obr. 4. 6 Graf závislosti relativní chyby [%] na počtu bodů funkce

Výsledky testů ukazují, že velikost chyby klesá s počtem bodů. Dále bylo zjištěno, že LT vykazuje se stoupajícím počtem bodů vyšší nebo srovnatelnou chybu s DT, proto se v dalších testech budeme zabývat pouze testy na DT.

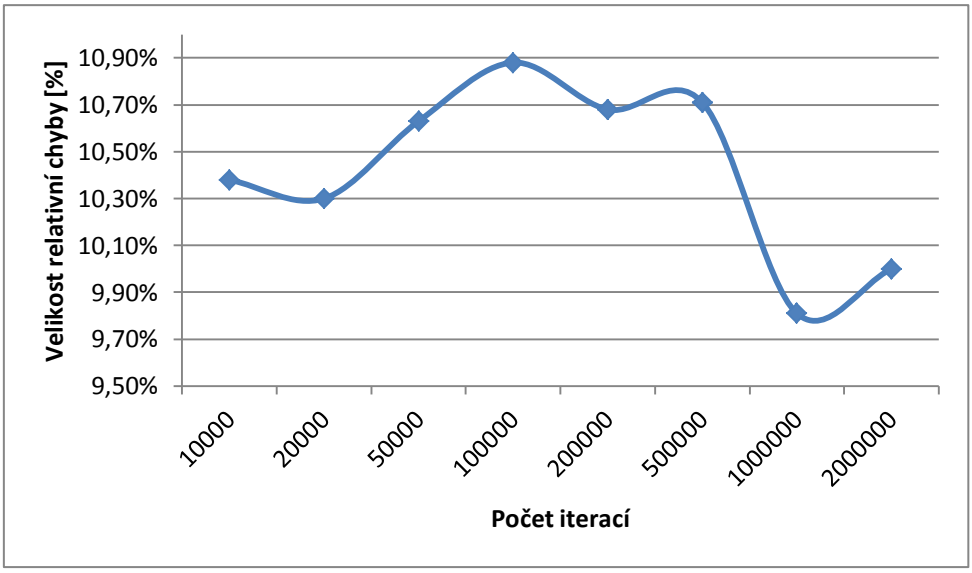
4.2 Experiment 2: Měření závislosti chyby na počtu iterací

V následujících testech jsme se zabývali závislostí chyby na počtu iterací (tedy na počtu přesunů bodů v modelu). Testy probíhaly na 1000 bodech modelu.

První test probíhal na funkci „nové hory“ (viz obr. 4.1). Výsledky lze vidět v tabulce 4.4 a na grafu v obrázku 4.7.

Počet bodů	Počet iterací	Střední kvadratická odchylka	Relativní chyba [%]
1000	10000	0,005244484	10,38%
1000	20000	0,005420370	10,30%
1000	50000	0,005344482	10,63%
1000	100000	0,005586603	10,88%
1000	200000	0,005545985	10,68%
1000	500000	0,005291574	10,71%
1000	1000000	0,004814476	9,87%
1000	2000000	0,005084607	10%

tab. 4. 4 Výsledky testů pro funkci "nové hory"



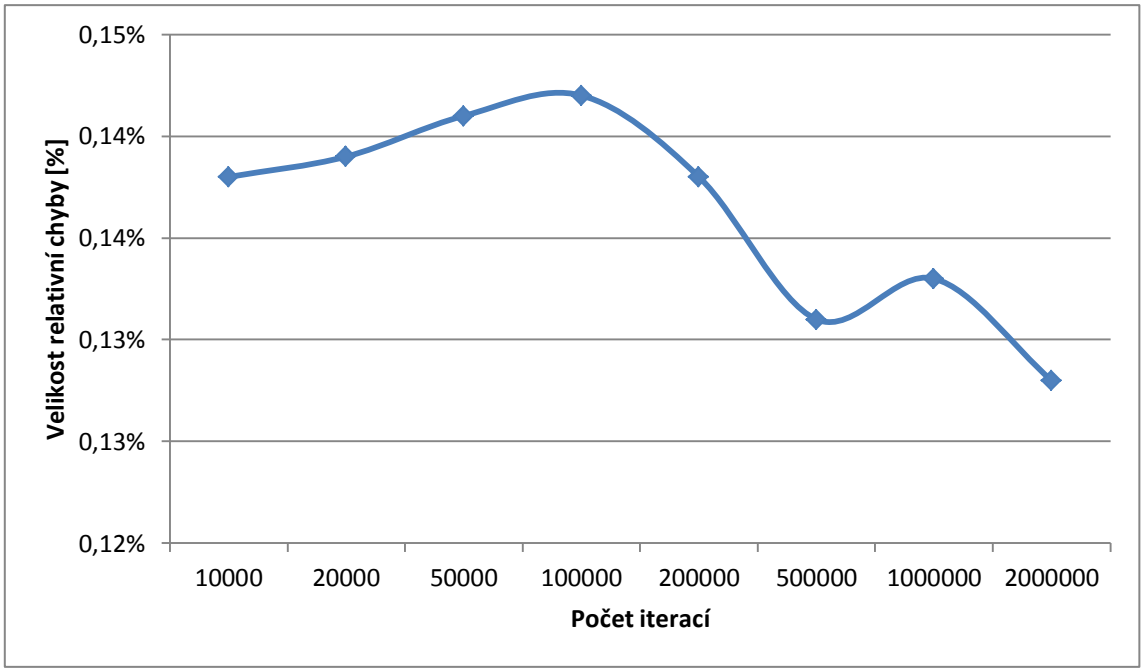
Obr. 4. 7 Graf velikosti chyby na počtu iterací pro funkci "nové hory"

Z výsledného grafu nelze přesně určit, zda se relativní chyba po určitém počtu iterací ustálí či bude nadále stoupat, proto počkáme na výsledky ostatních testů.

Druhý test probíhal na funkci „hladké údolí“ (viz obr. 4.3). Výsledky jsou v tabulce 4.5 a v grafu na obrázku 4.8.

Počet bodů	Počet iterací	Střední kvadratická odchylka [10^{-6}]	Relativní chyba [%]
1000	10000	1,10075	0,138%
1000	20000	1,11003	0,139%
1000	50000	1,17988	0,141%
1000	100000	1,20276	0,142%
1000	200000	1,13983	0,138%
1000	500000	1,02668	0,131%
1000	1000000	1,06193	0,133%
1000	2000000	1,02354	0,128%

tab. 4. 5 Výsledky testů pro funkci „hladké údolí“



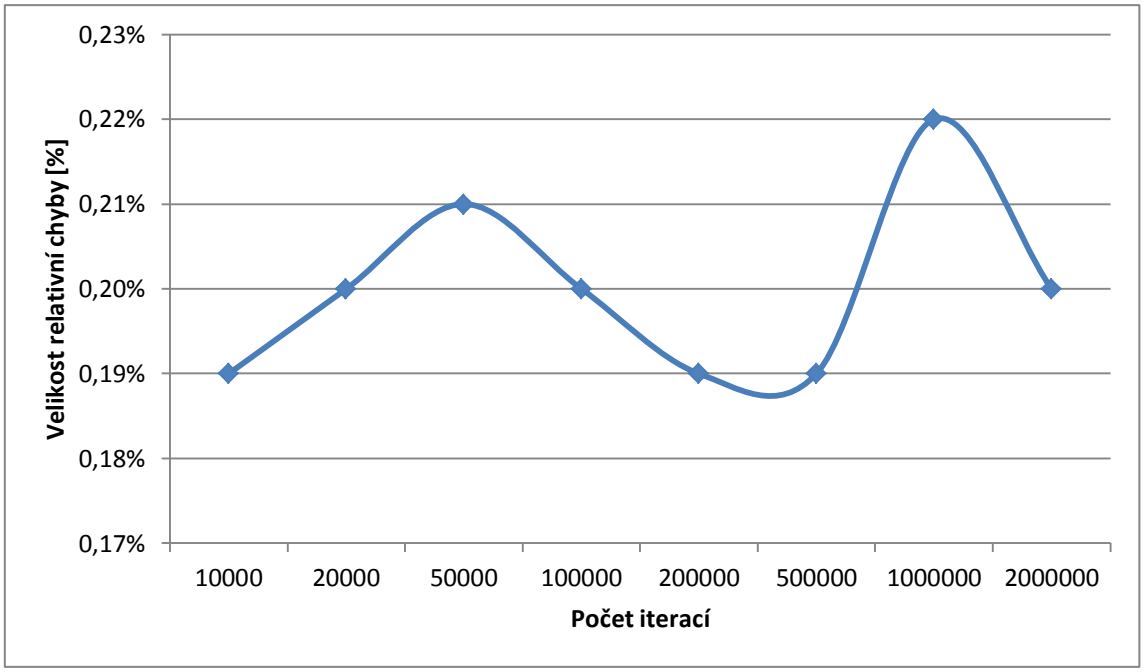
Obr. 4. 8 Graf závislosti chyby na počtu iterací pro funkci "hladké údolí"

Z výsledků druhého testu je vidět, že s růstem počtu iterací dochází k poklesu chyby.

Pro třetí test byla použita funkce „dvě údolí“ (viz obr. 4.5). Výsledky testů jsou v tabulce 4.6 a na obrázku grafu 4.9.

Počet bodů	Počet iterací	Střední kvadratická odchylka [10^{-5}]	Relativní chyba [%]
1000	10000	2,36461	0,19%
1000	20000	2,35257	0,20%
1000	50000	2,64525	0,21%
1000	100000	2,50206	0,20%
1000	200000	2,38702	0,19%
1000	500000	2,40260	0,19%
1000	1000000	2,81310	0,22%
1000	2000000	2,56342	0,20%

tab. 4. 6 výsledky testů pro funkci "dvě údolí"



Obr. 4. 9 Graf závislosti chyby na počtu iterací pro funkci "dvě údolí"

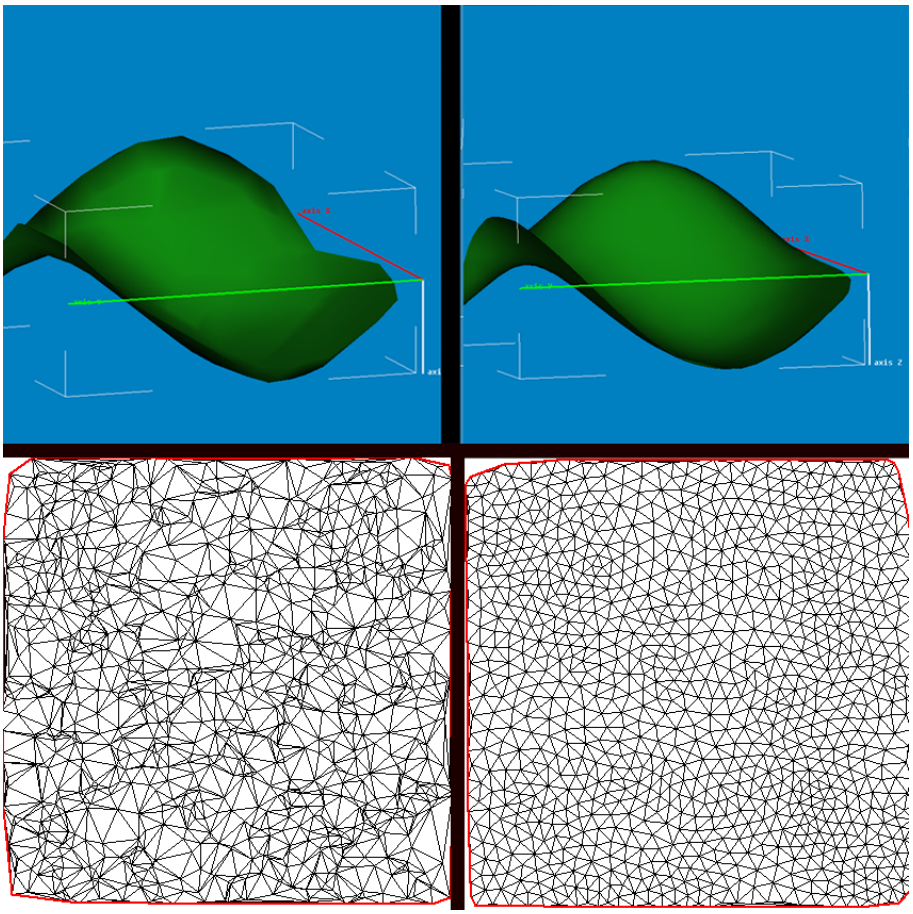
U třetího testu nelze určit zda velikost testu stoupá či klesá, neustále se mění.

Výsledky tohoto experimentu nedokázaly přesně určit, zda počet iterací zvyšuje či snižuje chybu. Lze se ale domnívat, že vhodný počet iterací bude závislý na typu modelu, a nelze tedy přesně určit vhodný počet iterací obecně.

4.3 Experiment 3: Vliv přesunů na vzhled terénu

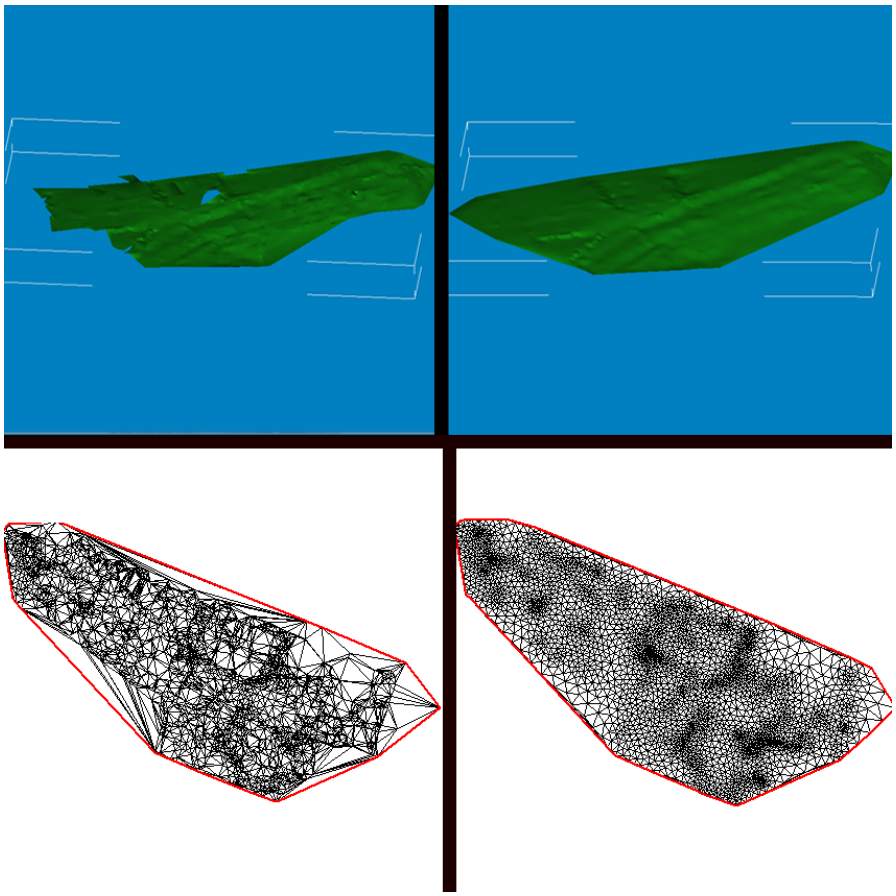
V tomto experimentu jsme se zabývali vlivem počtu přesunů na změny vzhledu původního modelu.

První experiment probíhal na funkci „hladké údolí“. Jak lze vidět na obrázku 4.10, přesuny bodů nenarušily celkový vzhled modelu, pouze jej vyhladily.



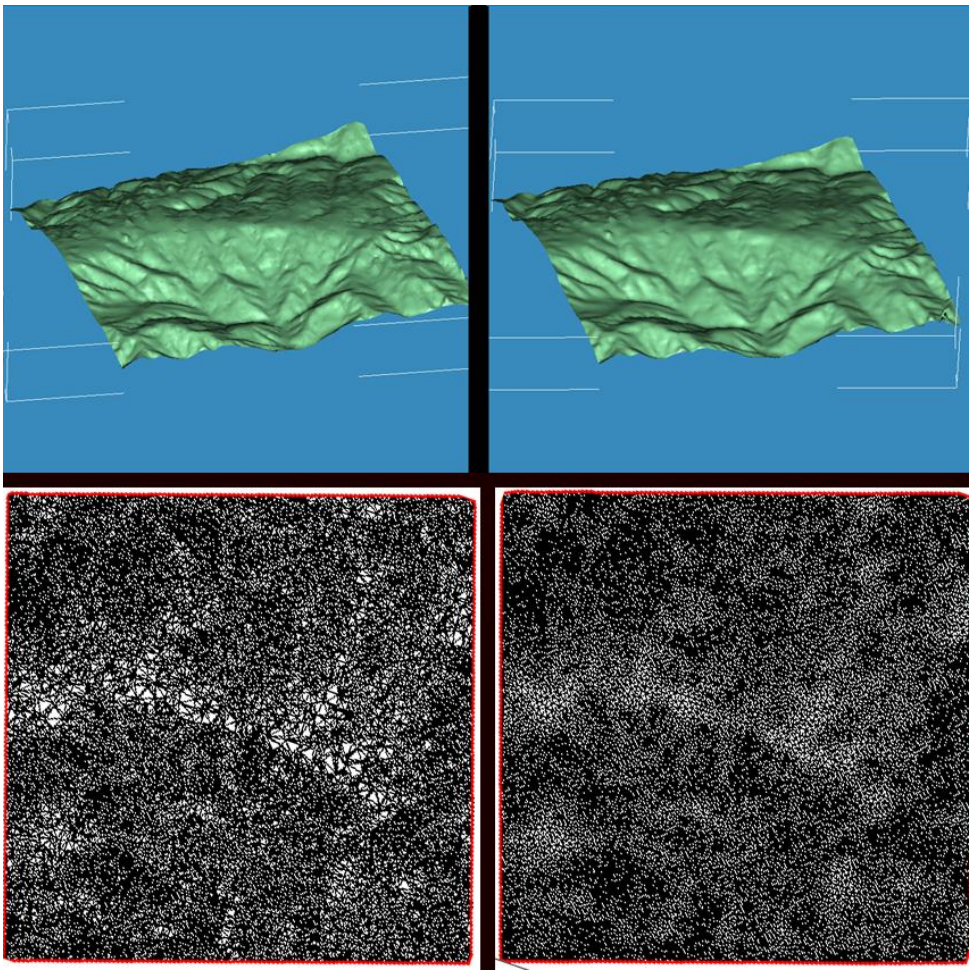
Obr. 4. 10 Model funkce "hladké údolí". Vlevo model z původního rozložení bodů, vpravo model po 2 milionech přesunů bodů

Druhý experiment probíhal na modelu botanické zahrady (data získána z PrF UK Praha). Na obrázku 4.11 je vidět, že model terénu má podobné rozložení před i po přesunech bodů. Navíc ale přesuny bodů odstranily artefakty, které vznikly při vytváření modelu v původním rozložení bodů.



Obr. 4. 11 Model botanické zahrady. Vlevo původní rozložení bodů, vpravo model vytvořený z bodů po 2 milionech přesunů

Poslední experiment probíhal na modelu hor (data získána z KMA ZČU). Na obrázku 4.12 vidíme, že přesuny nenarušily celkový vzhled modelu, zdá se pouze méně „hranatý“, tedy vyhlazený oproti původnímu rozložení bodů.



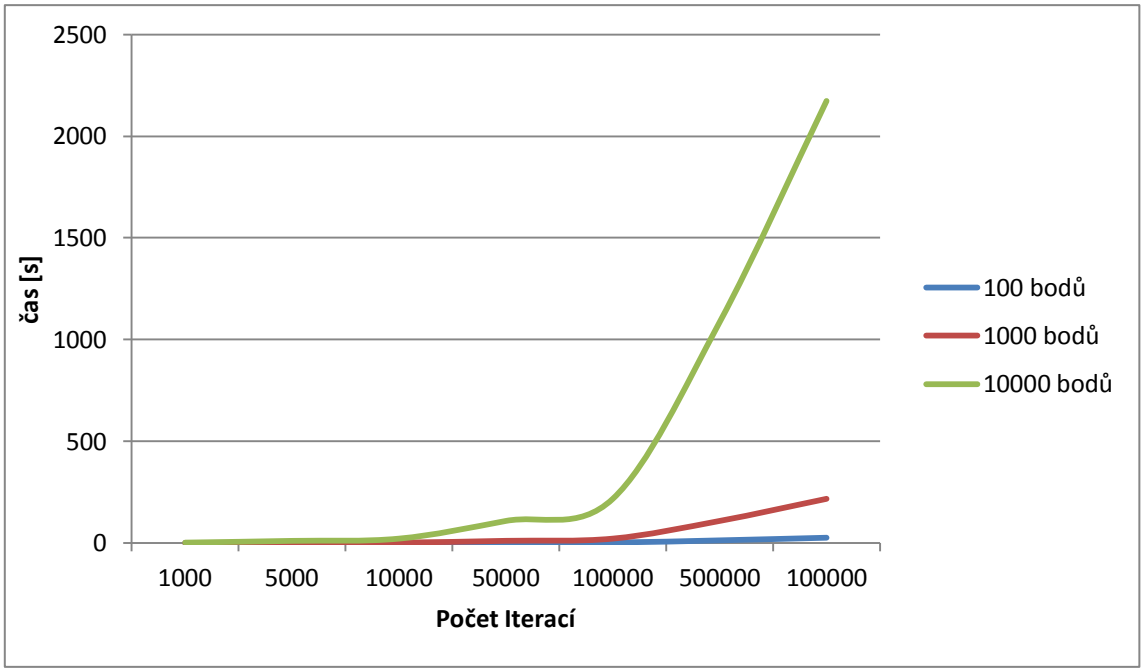
Obr. 4. 12 Model hor. Vlevo model původního rozložení bodů, vpravo model po 2 milionech přesunů bodů

Z těchto experimentů lze předpokládat, že přesuny bodů nenarušují celkový vzhled modelu. Pro potvrzení této domněnky by však bylo třeba provést více testů na větším množství testovaných dat.

4.4 Experiment 4: Orientační zátěžové testy

Tento experiment se zabývá časovou náročností programu v závislosti na počtu bodů a počtu iterací. Vstupem byla data funkce „hladké údolí“ a zabývali jsme se 100 až 10000 bodů a 1000 až 1000000 iterací.

Výsledky lze vidět v grafu na obrázku 4.12 a tabulce 4.7 a ukazují, že časová náročnost značně roste s rostoucím počtem iterací i bodů. V zásadě lze říct, že pro velká množství bodů by bylo třeba program optimalizovat. Tyto testy je proto nutno brát jako orientační.



Obr. 4. 13 Graf časové náročnosti na počtu iterací pro 100, 1000 a 10000 bodů

Počet bodů	Počet iterací	Čas [s]
100	1000	0,029
100	5000	0,132
100	10000	0,264
100	50000	1,327
100	100000	2,621
100	500000	13,244
100	1000000	26,245
1000	1000	0,221
1000	5000	1,086
1000	10000	2,200
1000	50000	10,969
1000	100000	21,723
1000	500000	108,712
1000	1000000	217,362
10000	1000	2,161
10000	5000	10,690
10000	10000	21,716
10000	50000	108,635
10000	100000	217,243
10000	500000	1085,823
10000	1000000	2173,200

tab. 4. 7 Výsledky orientačních zátěžových testů

5. Závěr

Po implementaci a provedení experimentů se mimo jiné ukázalo, že velikost chyby je závislá na typu vstupních dat. U stoupajícího množství bodů se v experimentech ukázalo, že chyba u testovaných Delaunayho triangulací vždy klesá, ale u lokálně minimálních triangulací se může občas chyba zvětšit. Celkově testy prokázaly, že Delaunayho triangulace vykazují nižší relativní chybu než lokálně minimální triangulace, a proto jsou pro interpolace vhodnější.

U počtu přesunů nelze přesně určit, zda chyba klesá či stoupá, protože experimenty ukázaly, že toto kritérium je závislé na vstupních datech – u některých typů dat chyba klesá, zatímco u jiných v není průběh jednoznačný.

Také se ukázal fakt, že McQueenova metoda je velmi časově náročná pro vyšší počty bodů, a proto by bylo vhodné algoritmus optimalizovat. Zatímco pro stovky bodů algoritmus přesouval body během sekund, pro desítky tisíc bodů již byla časová náročnost v desítkách minut až hodinách.

Reference

[1] ALTSHILLER-COURT, Nathan (1925), *College Geometry: An Introduction to the Modern Geometry of the Triangle and the Circle (2nd ed.)*, New York: Barnes & Noble, LCCN 52013504

[2] BERGER, Marcel (1984), *"Affine spaces"*, Problems in Geometry, p. 11, ISBN 9780387909714

[3] BOURKE, Paul. *Calculating The Area And Centroid Of A Polygon* (PDF) [online]. Červenec 1988 [cit. 2015-12-04]. Dostupné z <http://www.seas.upenn.edu/~sys502/extra_materials/Polygon%20Area%20and%20Centroid.pdf>

[4] COXETER, H. S. M. "*Barycentric Coordinates*." §13.7 in Introduction to Geometry, 2nd ed. New York: Wiley, pp. 216-221, 1969.

[5] DU Qiang, FABER Vance, and GUNZBURGER Max, *"Centroidal Voronoi tessellations: Applications and algorithms"*, SIAM Review 41 (1999), no. 4, pp. 637–676. MR 1722997

[6] JARVIS, R. A. (1973). "*On the identification of the convex hull of a finite set of points in the plane*". Information Processing Letters 2: 18–21. doi:10.1016/0020-0190(73)90020-3

[7] KOLINGEROVÁ Ivana.On triangulations. In Antonio Laganá, Marina L. Gavrilova, Vipin Kumar, Youngsong Mun, C.J.Kenneth Tan and Osvaldo Gervasi, editors, *Computational Science and Its Applications* ICCSA 2004, volume 3044 of *Lecture Notes in Computer Science*, pages 544-553- Springer Berlin Heindelberg, 2004

[8] LEUNG, Kam-tim. *Linear algebra and geometry*. [s.l.] : Hong Kong University Press, 1974. 309 s. ISBN 9780856561115. Kapitola 3.9, s. 96.

[9] LLOYD, Stuart P. (1982), *"Least squares quantization in PCM"*, IEEE Transactions on Information Theory 28 (2): 129–137, doi:10.1109/TIT.1982.1056489

[10] MACQUEEN, J.B., 1967. *Some methods for classification and analysis of multivariate observation*. In: In Le Cam, L.M. and NEYMAR, J., editor, 5 Berkeley Symposium on Mathematical Statistics and Probability. University of California Press.

[11] MUHAMMAD, Rashid. Convex Hull lecture [online]. Poslední revize 5. Května 2015 [cit 2015-08-04]. <http://www.personal.kent.edu/~rmuhamma/Compgeometry/MyCG/ConvexHull/convexHull.htm>

[12] OKABE A., BOOTS B. and SUGIHARA K.. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. John Wiley & Sons, Chichester, UK, 1992.

[13] TARRIDA, Agustí Reventós. *Affine Maps, Euclidean Motions and Quadrics*. [s.l.] : Springer, 2011. 458 s. Definice 1.1. ISBN 9780857297099. S. 1.

[14] WIEBEL Robert and HELLER Martin. *Digital terrain modelling*, Oxford University Press, 1993.

[15] Interpolation in numerical mathematics *Encyclopedia of Mathematics*[online]. Poslední revize 3. Ledna 2015 [cit.2015-12-04] URL: <http://www.encyclopediaofmath.org/index.php?title=Interpolation_in_numerical_mathematics&oldid=36051>.