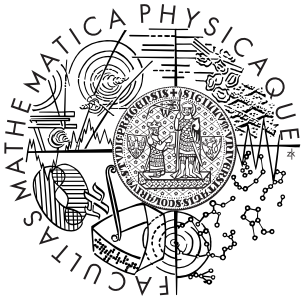
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Michal Zajac

Generování Delaunayho triangulací

Katedra numerické matematiky

Vedoucí bakalářské práce: Doc. Mgr. Petr Knobloch, Dr.
Studijní program: matematika, obecná matematika

2006

Na tomto místě bych chtěl poděkovat především vedoucímu mojí bakalářské práce, Doc. Mgr. Petru Knoblochovi, Dr., za volbu zajímavého tématu, připomínky a rady, ochotu ke konzultacím a pomoc s problémy, které se během psaní práce vyskytly.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 30.5.2006 Michal Zajac

Obsah

Název práce: Generování Delaunayho triangulací
Autor: Michal Zajac
Katedra: Katedra numerické matematiky
Vedoucí bakalářské práce: Doc. Mgr. Petr Knobloch, Dr.
e-mail vedoucího: Petr.Knobloch@mff.cuni.cz

Abstrakt: Cílem této bakalářské práce je vytvořit software na generování Delaunayho triangulací. Vychází z algoritmu uvedeného v článku [2]. Program je implementovaný pro všechny uživatelem zadané konvexní oblasti, pro kruhy, dále jsou implementované kruhové výřezy daných oblastí. Výstup tvoří dva soubory snadno použitelné v zdarma dostupném programu Gnuplot. Jeden obsahuje trojúhelníky tvořící Delaunayho triangulaci, druhý úsečky tvořící Voronoiho mnohoúhelníky. Práce obsahuje grafy několika testovacích oblastí. Jsou zde popsané problémy, které se během implementace objevily.

Klíčová slova: Delaunay, triangulace, Voronoi, teselace

Title: Generation of Delaunay triangulations
Author: Michal Zajac
Department: Department of Numerical Mathematics
Supervisor: Doc. Mgr. Petr Knobloch, Dr.
Supervisor's e-mail address: Petr.Knobloch@mff.cuni.cz

Abstract: The aim of this work is to create software for generating Delaunay triangulations. The algorithm is described in the article [2]. The program is implemented for any convex or circle domain given by user. Domains with circle holes are also implemented. The output consists of two files, which can be easily used in Gnuplot program. One contains triangles of Delaunay triangulation and the other contains segments of Voronoi polygons. There are also graphs for some test domains included. Problems which appeared during implementation are also further described.

Keywords: Delaunay, triangulation, Voronoi, tesselation

Kapitola 1

Delaunayho triangulace

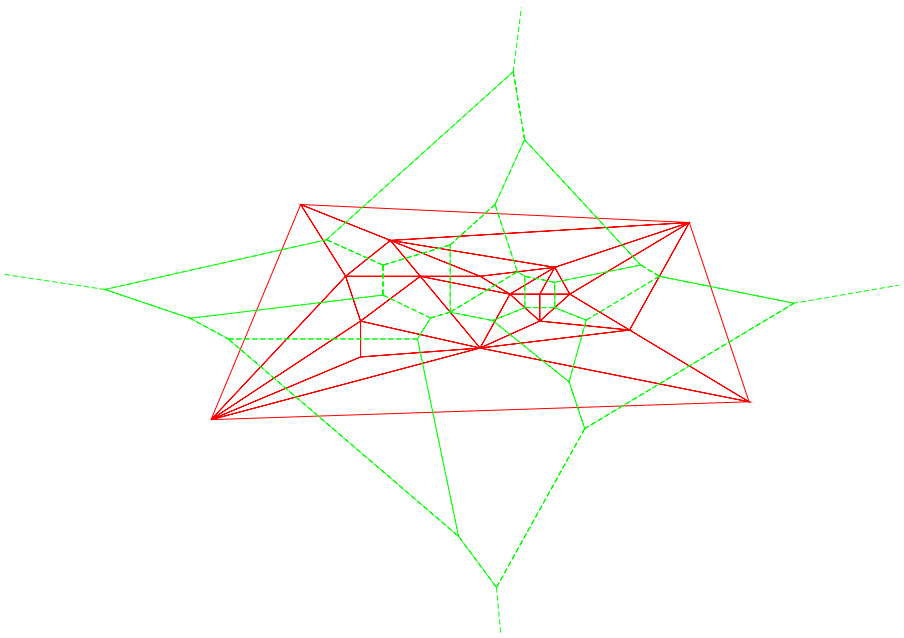
Při hledání numerického řešení různých problémů je potřebné vybrat konečný počet bodů v oblasti a spojit je tak, aby definovaly síť. Je mnoho způsobů jak takovou síť definovat. Jedním z nich je tzv. Delaunayho triangulace.

Předpokládejme, že máme n různých bodů v rovině. Každému z nich můžeme přiřadit teritorium, které je definováno jako množina těch bodů, které jsou bližší k danému bodu než k libovolnému jinému bodu. Takto vytvoříme teritoria pokrývající celou oblast. Tato konstrukce je nazvána Dirichletova teselace. Na Obr. 1.1 je ukázka teselace (zelená barva) pro malý počet bodů. Body, které leží na hranici konvexního obalu dané množiny bodů, mají nekonečná teritoria, ostatní mají konečná. Z výše zmíněné definice vyplývá, že úsečky, které tvoří hranice jednotlivých teritorií, musí ležet přesně v polovině spojnice dvou bodů, jejichž teritoria pomáhají vymezit. Jestliže spojíme všechny dvojice, které mají společnou nějakou úsečku hranice, dostaneme triangulaci daných n bodů známou jako Delaunayho triangulace (na Obr. 1.1 červená barva).

Podle toho jak jsou spojeny body sítě, můžeme rozlišit dvě základní třídy sítí. *Strukturované* jsou takové, u kterých jsou vnitřní body spojeny nezávisle na jejich pozici. Jestliže se způsob propojení mění v závislosti na pozici bodu, nazýváme síť *nestrukturovanou*. Numerické řešení na strukturovaných sítích je levnější než na nestrukturovaných. Na druhé straně ale nestrukturované sítě poskytují větší geometrickou flexibilitu.

Metody generací nestrukturovaných sítí založené na Delaunayho triangulaci nám dovolují přidat nové body do oblasti bez toho, abychom museli přepočítat propojení celé sítě.

Algoritmus použitý v této práci plně využívá sekvenčnosti konstrukce



Obrázek 1.1: Delaunayho triangulace

Delaunayho triangulace. V každém kroku se spočítá pozice nového bodu a pak se pomocí Bowyerova-Watsonova algoritmu (články [1], [3]) spočítá propojení bodů sítě.

Implementovány jsou dvě metody počítání pozice nového bodu. Jedna je jednodušší na implementaci, zatímco druhá dává pravidelnější trojúhelníky.

Kapitola 2

Bowyerův-Watsonův algoritmus

2.1 Implementace

Uložení dat

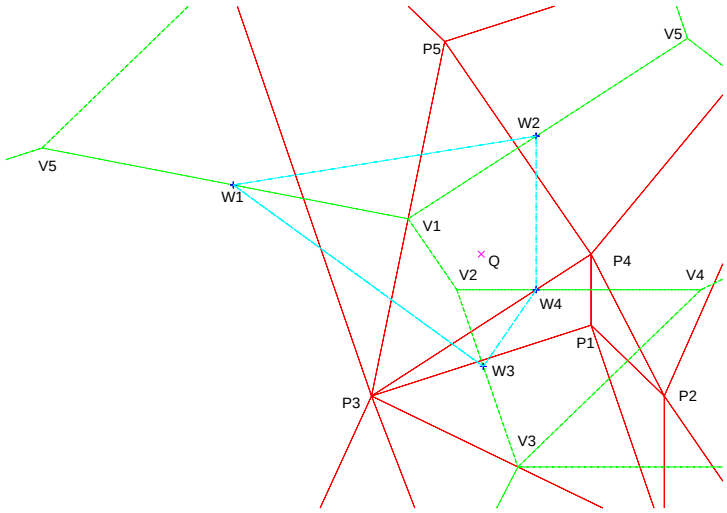
Data jsou uložena tak, jak je navrhnuto v článku [1]. To znamená, že všechny body triangulace a všechny vrcholy teselace jsou uloženy do pole. Každý bod triangulace si pamatuje indexy všech svých sousedů a každý vrchol teselace si pamatuje indexy svých tří sousedů, indexy tří bodů tvořících trojúhelník, jehož střed kružnice opsané je daný vrchol, a typ trojúhelníku (vzhledem k hlavnímu algoritmu).

Algoritmus

Nechť máme uložených n bodů ve výše definované struktuře. Nový bod (Q na Obr. 2.1) musí být přidán do konvexního obalu daných n bodů. Toto musí být ošetřeno v hlavním algoritmu. Vkládání bodu můžeme rozdělit do šesti kroků:

1. Nalezení prvního vrcholu teselace, který bude novým bodem vymazán. Takový vrchol je definován jako ten, který je blíž k novému bodu než k bodům, které ho tvoří (např. V1) .
2. Nalezení všech dalších vrcholů, které budou vymazány. V našem případě: (V1, V2)
3. Určení sousedů nového bodu. (P1, P3, P4, P5)

4. Vymazání již neplatných sousedských vztahů. (P4 - P3)
5. Spočtení nových vrcholů a určení bodů, které je tvoří, a jejich sousedů. (W1, W2, W3, W4)
6. Vymazání vrcholů, které mají být vymazány.



Obrázek 2.1: Vkládání nového bodu Q do triangulace

Krok jedna je obecně závislý na aktuálním počtu vrcholů. První mazaný vrchol totiž lze jednoduše získat tak, že procházíme všechny vrcholy a najdeme ten z mazaných, který má nejnižší index. Jenomže obě metody výpočtu hlavního algoritmu poskytují index prvního mazaného vrcholu prakticky zdarma. Tímto se budu zabývat v kapitole 3.

Další mazané vrcholy najdeme tak, že rekurzivně projdeme sousedy prvního mazaného vrcholu, jež mají být také mazané. Kvůli numerickým nepřesnostem je pravidlo pro určení, jestli je daný vrchol vymazán, upraveno tak, aby počítalo s určitou tolerancí. Neztratíme nic, jestliže nějaký vrchol zařadíme do seznamu mazaných. Jestliže tam být neměl, pak se opět vytvoří. Problém nastane, pokud do tohoto seznamu zařazený není a měl být. V takovém případě se struktura naruší a algoritmus nebude fungovat. Proto jsou do seznamu zařazeny i vrcholy, které mají vzdálenost od nového vrcholu i o ϵ větší než od formujícího vrcholu.

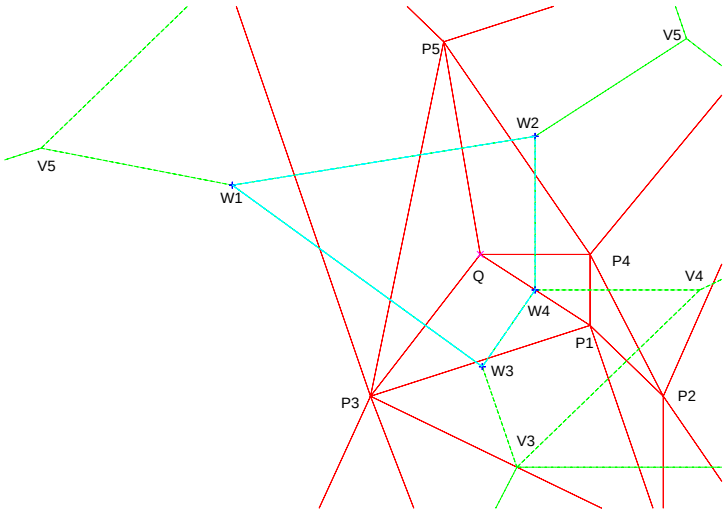
Sousedi nového bodu jsou body formující všechny mazané vrcholy.

Krok 4 znamená, že projdeme všechny mazané vrcholy a hledáme společné dvojice formujících bodů. Jestliže takovou dvojici najdeme, sousedství mezi těmito vrcholy se vymaže.

Nejobtížnější je krok 5. Nové body počítáme následovně. Procházíme všechny sousedy nového bodu. Pokud existuje sousedství mezi dvojicí z nich (např. P1, P3), pak nový vrchol a tato dvojice tvoří trojúhelník (Q, P1, P3). Tento trojúhelník ale může být takový, že uvnitř něj se nacházejí nějaké další body. Toto je nutné zkontrolovat. Jestliže se v něm další bod už nenachází, pak střed opsané kružnice tohoto trojúhelníku tvoří nový vrchol teselace (W3). Vrcholy trojúhelníku jsou formující body vrcholu teselace, tudíž je hned uložíme. Po spočtení všech nových vrcholů a jejich formujících bodů vypočteme jejich sousedy. Nejdříve ze sousedů mazaných vrcholů (V3) a následně určíme sousedství mezi novými vrcholy (W1, W4). Vždy využijeme toho, že sousedi mají dva stejné formující vrcholy.

Nové vrcholy již máme uložené v poli. Vrcholy, které mají být mazané, vymažeme. Implementované to je jednoduše tak, že index vrcholu, který má být vymazán, se prohlásí za neplatný a nic se ze struktury nemaže. Při tvoření nových vrcholů se totiž tyto vrcholy ukládají na místa, která jsou neplatná, a přepisují ty vrcholy, které tam byly kdysi uloženy.

Obr. 2.2 znázorňuje danou triangulaci po vložení bodu Q.



Obrázek 2.2: Triangulace po vložení bodu Q

2.2 Problémy

Problematický je jenom krok 5. Článek [1] neobsahuje návod jak spočítat nové body. To, že může nastat situace, že trojúhelník sestávající z nového bodu a dvojici jeho sousedů může obsahovat nějaký bod (a tím pádem se nový bod nevytvoří), jsem zjistil až při testovacích výpočtech.

Je nutné myslet na to, že nový bod musí být vložen do konvexního obalu již vytvořených bodů. Je také důležité, že algoritmus je definován pro n různých bodů a $(n+1)$-ní bod musí být od těchto bodů také různý. Při porušení těchto podmínek algoritmus nefunguje.

Kapitola 3

Hlavní algoritmus

3.1 Inicializace

Obě metody vkládání nového bodu využívají počáteční triangulaci a počáteční rozdělení na vnitřní a vnější trojúhelníky. Vzhledem k tomu, že bod, který vkládáme pomocí Bowyerova algoritmu, musí být v konvexním obalu už triangulovaných bodů, potřebujeme konvexní obal, který se během celého počítání nemění. Tím pádem vznikají trojúhelníky, které jsou vně oblasti, kterou máme triangulovat.

Problém s konvexním obalem jsem vyřešil tak, že uživatel zadá čtyři body tvořící konvexní čtyřúhelník, který bude obalem celé triangulované oblasti. Z těchto čtyř bodů snadno vytvořím počáteční triangulaci v definované datové struktuře. Úkolem uživatele je ohlídat, že tyto čtyři body budou opravdu konvexním obalem pro všechny hraniční body.

Dále musím spočítat body na hranici. Ty jsou pak postupně vkládány do triangulace pomocí Bowyerova algoritmu. Úsečky, které takto na hranicí vytvořím, nebudou nikdy smazány, aby byla hranice zachována. Proto nesmí být příliš daleko od sebe vzhledem k velikosti opsané kružnice v dané oblasti. Tato velikost je zadaná funkcí $f = f(x, y)$.

Pro kružnice jsem vymyslel algoritmus, který vypočítá hraniční body následovně. Mám kružnici k tvořící hranici zadanou parametricky. Spočtu si bod P_1, který odpovídá parametru $t_1 = 0$. Nechť máme spočítaných i bodů. $(i + 1)$-ní bod P_{i+1} hranice spočtu jako ten průnik kružnice k a kružnice $k_i = k_i(P_i, c.f(P_i))$, který odpovídá parametru vyššímu než je parametr t_i. Konstantu c jsem volil jako poměr délky hrany a poloměru opsané kružnice rovnostranného trojúhelníku ($c = \sqrt{3}$). Počítání zastavím, pokud parametr $t_{i+1} > 2\pi$.

Tímto způsobem mohu vytriangulovat libovolnou kružnici (vnitřní nebo vnější).

Pro konvexní oblasti je snadné rozdělit trojúhelníky na vnitřní a vnější. Vnější jsou ty, které tvoří alespoň jeden ze čtyř bodů konvexního obalu - to se jednoduše identifikuje. Problém nastává u nekonvexních oblastí, kde nelze snadno trojúhelníky rozlišit.

Naimplementoval jsem ale proceduru, která určí vnější trojúhelníky u oblastí, jejichž nekonvexita je způsobená kruhovým výřezem. Vnější trojúhelníky jsou pak ty, u nichž středy všech stran mají vzdálenost od středu kružnice menší než poloměr (počítáno s tolerancí).

3.2 Metoda vkládání nového bodu do vrcholu teselace

Tato metoda může být rozepsaná do 9 kroků:

1. Triangulace všech hraničních bodů pomocí Bowyerova-Watsonova algoritmu.
2. Kontrola, zda triangulace obsahuje všechny hraniční úsečky.
3. Rozdělení trojúhelníků na vnitřní a vnější.
4. Výpočet koeficientu α_k pro každý vnitřní trojúhelník T_k.
5. Seřazení vnitřních trojúhelníků podle α_k.
6. Vložení nového bodu do středu opsané kružnice trojúhelníku na vrchu seřazeného seznamu.
7. Spuštení Bowyerova-Watsonova algoritmu s tím, že body, které by způsobily vymazání vnějšího trojúhelníku, nejsou brány v úvahu.
8. Zařazení nových trojúhelníků do seznamu trojúhelníků.
9. Jestliže $\max_k \alpha_k > 1$ jdi na krok 6, jinak konec.

Zde α_k je definováno jako podíl $\alpha_k = \frac{\rho_k}{f(\mathbf{x}_k)}$, kde ρ_k je poloměr opsané kružnice daného trojúhelníku a $\mathbf{x}_k$ je pozice středu opsané kružnice.

Jediná potíž tohoto algoritmu je, že střed opsané kružnice i u vnitřních trojúhelníků může ležet mimo konvexní obal. Toto musí být ošetřeno a takové body jsou zapomenuty.

Při hledaní prvního mazaného vrcholu v Bowyerově algoritmu je jako první brán v úvahu vrchol, do kterého vkládám nový bod. Ten je totiž vždy vymazán. Tím pádem je Bowyerův algoritmus jen lokálního charakteru.

3.3 Metoda vkládání nového bodu na úsečku teselace

Tato metoda rovněž může být rozepsána do 9 kroků:

1. Triangulace všech hraničních bodů pomocí Bowyerova-Watsonova algoritmu.
2. Kontrola, zda triangulace obsahuje všechny hraniční úsečky.
3. Rozdělení trojúhelníků na vnitřní a vnější. Rozdělení vnitřních trojúhelníků na akceptované a neakceptované. Rozdělení neakceptovaných na aktivní a čekající.
4. Seřazení aktivních trojúhelníků podle poloměru opsané kružnice.
5. Výpočet pozice nového bodu, uvažujíc trojúhelník z vrcholu seznamu aktivních trojúhelníků.
6. Spuštení Bowyerova-Watsonova algoritmu s tím, že body, které by způsobily vymazání vnějšího trojúhelníku, nejsou brány v úvahu.
7. Rozdělení nových trojúhelníků na akceptované a neakceptované. Rozdělení neakceptovaných na aktivní a čekající.
8. Rozdělení neakceptovaných trojúhelníků sousedících s novými trojúhelníky na aktivní a čekající.
9. Jestliže je seznam aktivních trojúhelníků prázdný, konec, jinak jdi na krok 5.

Akceptovaný trojúhelník je definován jako ten, který má velikost předepsanou pro jeho pozici. Neakceptované trojúhelníky se dělí na aktivní a čekající. Aktivní jsou ty, které mají alespoň jednoho akceptovaného nebo vnějšího souseda. Čekající jsou zbylé neakceptované trojúhelníky.

Pozice nového bodu se spočítá následovně: Nechť C_A je střed kružnice opsané aktivnímu trojúhelníku, M střed strany společné s akceptovaným nebo vnějším trojúhelníkem (PQ) a $C_{A/E}$ střed opsané kružnice tohoto trojúhelníku. Dále nechť $\rho_M = f(\mathbf{x}_M)$ je hodnota poloměru opsané kružnice předepsané v bodě $\mathbf{x}_M$. Definujme

$$\rho'_M = min[max(\rho_M, p), \frac{p^2 + q^2}{2q}],$$

kde $p = \frac{1}{2}|PQ|$ a $q = |C_A M|$. Pozice nového bodu X je pak definovaná vztahem

$$\mathbf{x}_X = \mathbf{x}_M + d\mathbf{e},$$

kde vzdálenost $d = |XM|$ a jednotkový vektor $\mathbf{e}$ jsou

$$d = \rho'_M + \sqrt{\rho'^2_M - p^2}$$

$$\mathbf{e} = \frac{\mathbf{x}_{C_A} - \mathbf{x}_{C_{A/E}}}{|\mathbf{x}_{C_A} - \mathbf{x}_{C_{A/E}}|}$$

.

Při hledaní prvního mazaného vrcholu v Bowyerově algoritmu je jako první brán v úvahu vrchol, který je vytvořen aktivním trojúhelníkem, do něhož nový bod vkládám.

Důležitá vlastnost bodů, na kterou se musí myslet je, že musí být různé. Proto jsem zavedl funkci, která kontroluje, zda nový bod není příliš blízko některému z již triangulovaných bodů.

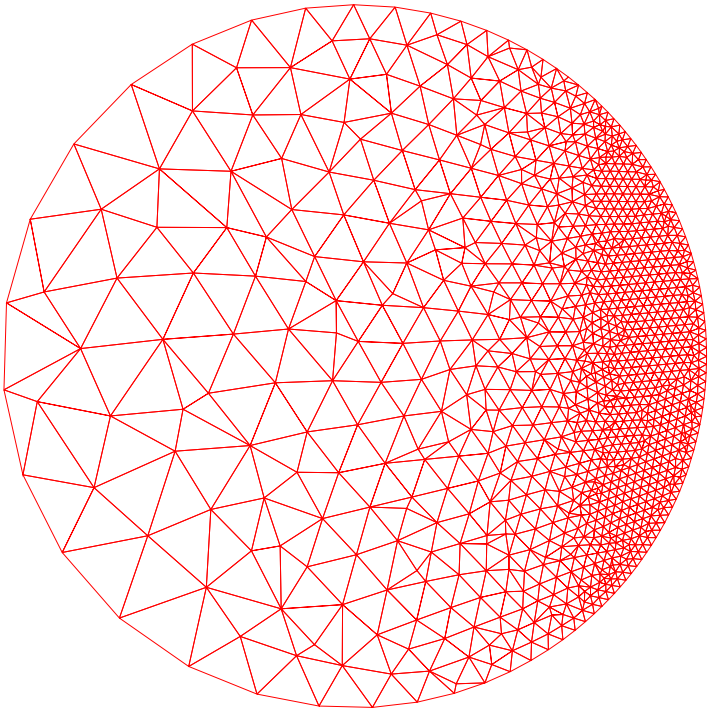
Tím pádem nastanou dva případy, kdy Bowyerův algoritmus neproběhne:

1. Nový bod je příliš blízko některému z již triangulovaných.
2. Některý z vnějších trojúhelníků by měl být smazaný.

Aktivní trojúhelník, pomocí kterého jsme pozici nového bodu spočítali, se ze seznamu aktivních trojúhelníků vymaže, protože pokud Bowyerův algorimus proběhne, trojúhelník se vymaže i z triangulace. Jesliže neproběhne, pak tento trojúhelník již uvažovat nechceme, neboť by se potom algoritmus zbytečně zpomaloval.

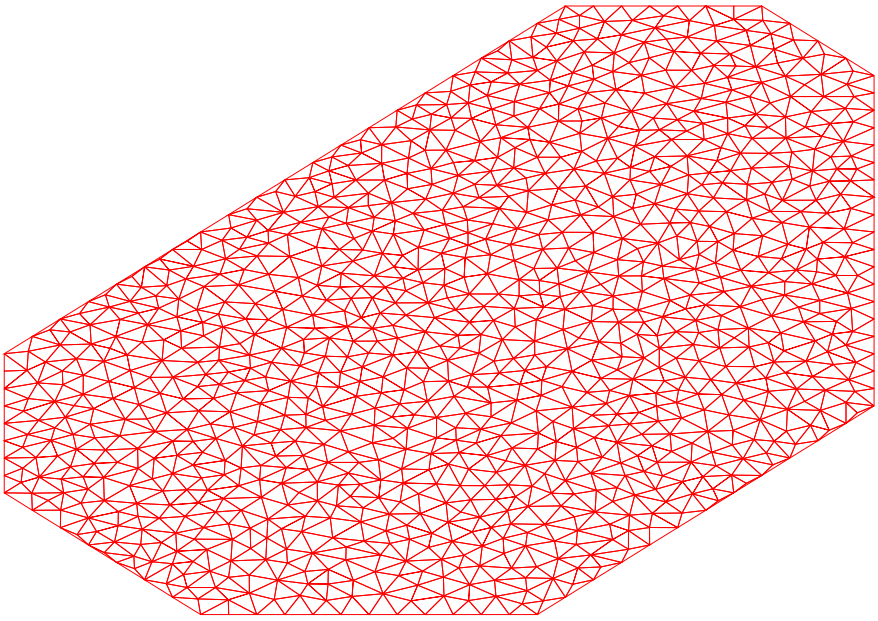
Kapitola 4

Výsledky

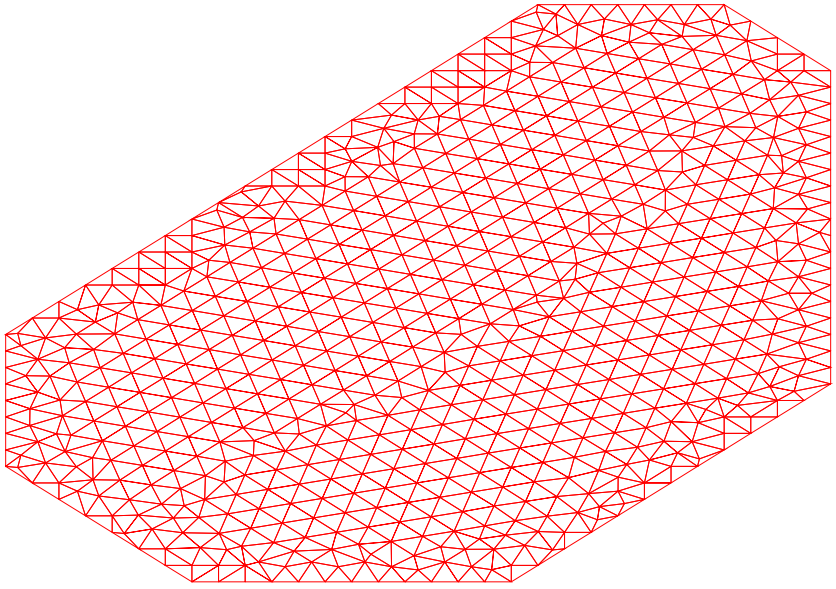


Obrázek 4.1: Kruh se středem v bodě [10,10] a poloměrem $R = 7$ s funkcí f - metoda vkládání nového bodu na úsečku teselace (dále jenom metoda 2)

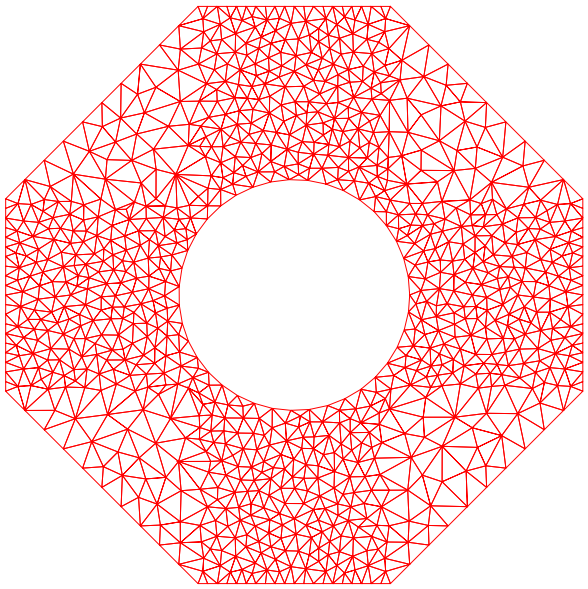
$$f(x,y) = \begin{cases} 1 & \text{je-li } x < 5, \\ \frac{29}{20} - \frac{9}{100}x & \text{je-li } 5 \leq x \leq 15, \\ 0.2 & \text{je-li } x > 15 \end{cases}$$



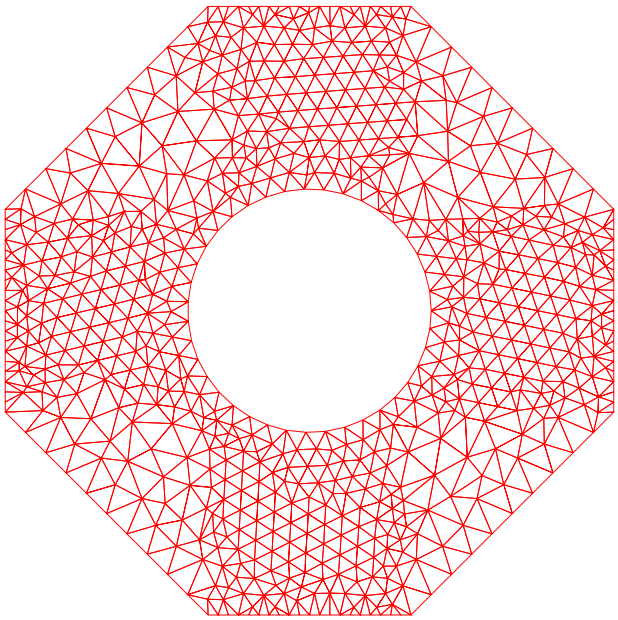
Obrázek 4.2: Konvexní mnohoúhelník - metoda vkládání nového bodu do vrcholu teselace (dále jenom metoda 1)



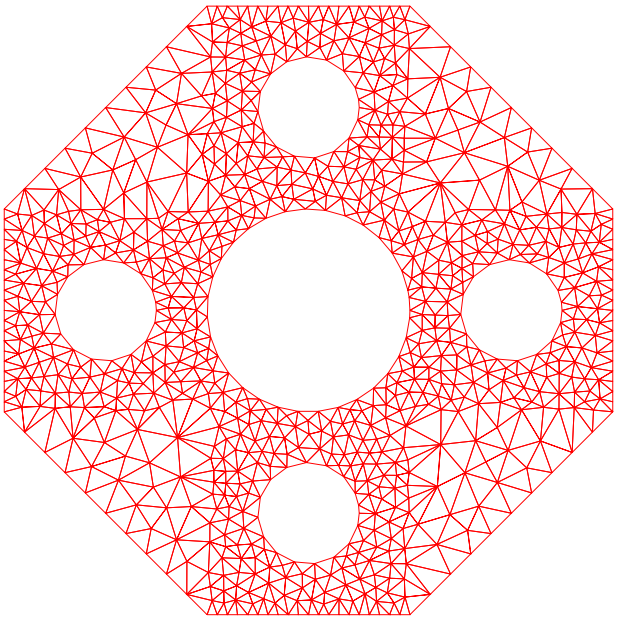
Obrázek 4.3: Konvexní mnohoúhelník - metoda 2



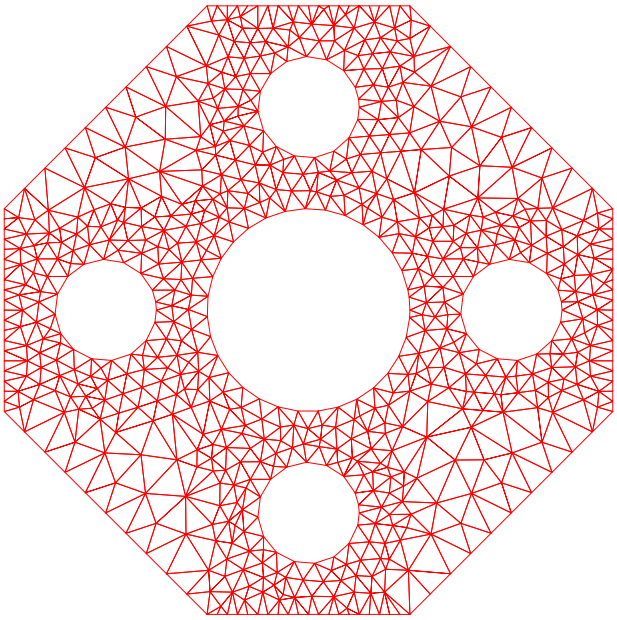
Obrázek 4.4: Osmiúhelník s kruhovým výřezem - metoda 1



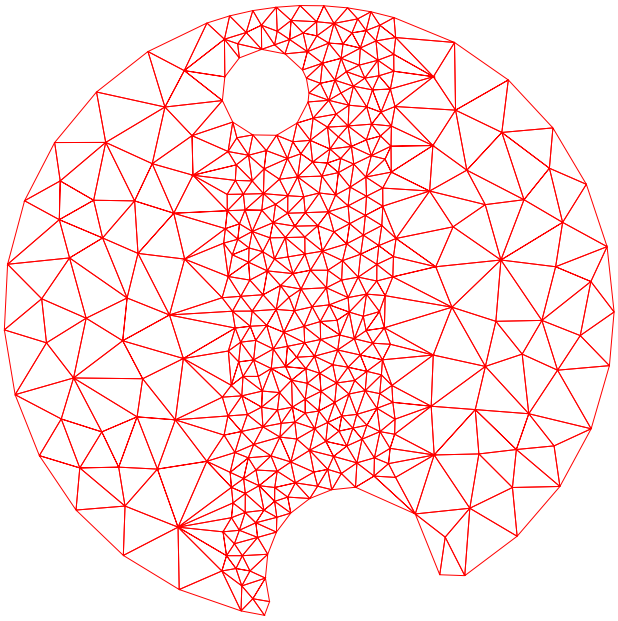
Obrázek 4.5: Osmiúhelník s kruhovým výřezem - metoda 2



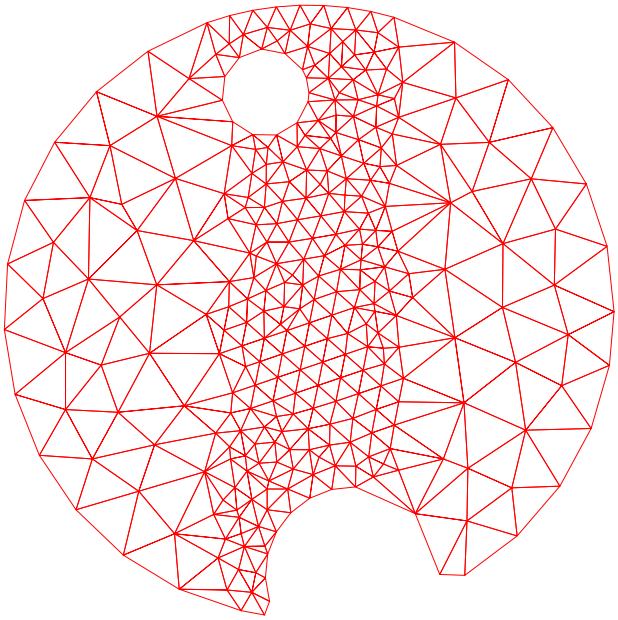
Obrázek 4.6: Osmiúhelník s 5 kruhovými výřezy - metoda 1



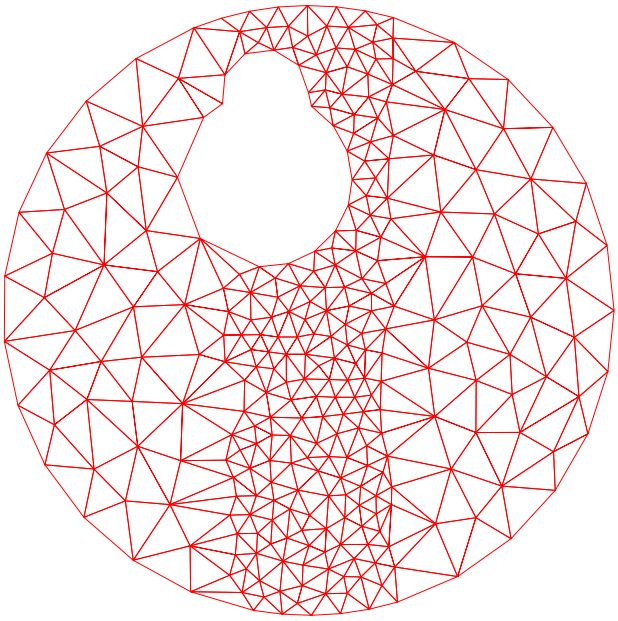
Obrázek 4.7: Osmiúhelník s 5 kruhovými výřezy - metoda 2



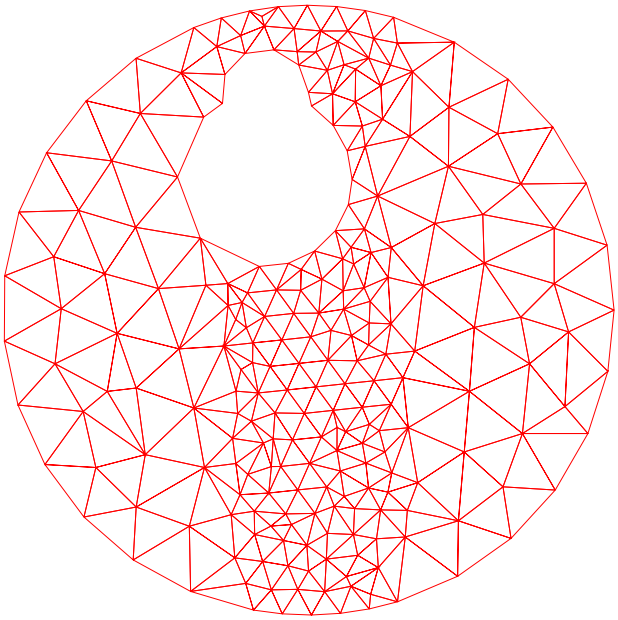
Obrázek 4.8: Kruh s 2 kruhovými výřezy - metoda 1



Obrázek 4.9: Kruh s 2 kruhovými výřezy - metoda 2



Obrázek 4.10: Kruh s 2 spojenými kruhovými výřezy - metoda 1



Obrázek 4.11: Kruh s 2 spojenými kruhovými výřezy - metoda 2

Literatura

[1] Bowyer A.: *Computing Dirichlet tessellations*, Comput. J. **24** No.2 (1981), 162-166.

[2] Rebay S.: *Efficient unstructured mesh generation by means of Delaunay triangulation and Bowyer-Watson algorithm*, J. Comput. Phys. **106** No.1 (1993), 125-138.

[3] Watson D. F.: *Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes*, Comput. J. **24** No.2 (1981), 167-172.