1 Úvod

Triangulace oblasti má dnes využití například v počítačové grafice nebo numerické matematice, kde základní algoritmy pro výpočet parciálních diferenciálních rovnic vyžadují rozdělení zadané souvislé oblasti na množinu jednoduchých podoblastí, na nichž se parciální diferenciální rovnice nahradí rovnicemi algebraickými. V našem případě budou danými jednoduchými podoblastmi trojúhelníky (je možno využívat i jiné konvexní n-úhelníky).

Mezi nejčastěji používané algoritmy patří tzv. metoda level–setů a algoritmus vrstvení. Samotný algoritmus vrstvení probíhá následujícím způsobem: Vstupem je uspořádána množina křivek, které oblast ohraničují. Tyto okrajové křivky se postupně posouvají ve směru jejich normály dovnitř zadané oblasti. V každé iteraci vrstvení se vypočtou průsečíky posunutých křivek a mezi nimi se poté vytvoří nová vrstva. Nakonec se vzniklý pruh mezi minulou a současnou vrstvou vyplní sítí trojúhelníků. Na výstupu pak bude množina trojúhelníku – indexy vrcholů a jejich souřadnice.

Oblast může být ohraničena jakýmkoliv druhem křivek; v práci se dále zaměříme pouze na úsečky a Bézierovy křivky, kterými se dá popsat celá řada oblastí. Od množiny křivek se očekává, že budou zadány v pravotočivém směru.

Implementace algoritmu, kterou budeme v práci popisovat, je zjednodušením metody vrstvení. Od základního algoritmu se liší v tom, že neposouváme původní hladké křivky a nepočítáme jejich průsečíky, což vede na řešení pomocí Newtonovy metody, viz [5], ale posouváme uzly diskretizace ležící na křivkách směrem dovnitř oblasti. Z výsledných bodů pak souborem několika pravidel, např. eliminujeme pseudoostré úhly, vytvářím jednotlivé trojúhelníky. Z nově přidaných vrcholů vytvoříme další vrstvu, na kterou pak iterativně aplikujeme opět stejný postup. Celý proces bude ukončen v momentě, kdy už nemá smysl pokračovat, např. počet bodů klesne pod předem stanovenou hodnotu.

Metoda level–setů je založena na implicitním popisu hranice oblasti, které je obecnější než parametrické. Další vrstvy jsou dány řešením evoluční diferenciální rovnice viz [6].

Text bude rozdělen do následujících částí: V kapitole 2 více rozebereme jednotlivé křivky, jakým způsobem se zadávají, jejich základní vlastnosti a více se budeme také věnovat způsobu, jak spočítat jejich délku. V kapitole 3 zavedeme některé pojmy, důležité pro popis sítě a jednotlivých vrstev, hlavní náplní bude ovšem popis metody vrstvení a to ve 2D. 4. kapitola pak bude věnována konkrétní implementaci algoritmu. Jedná se hlavně o jednoduchý popis a návrh datových struktur, které jsou použity, a dále detailnější rozbor některých důležitých procedur. V poslední 5. kapitole pak uvedeme pár příkladů oblastí, na kterých byla metoda použita a některé problémy, kterým je třeba v budoucnu věnovat větší pozornost.

2 Křivky a operace s nimi

2.1 Základní křivky

V této kapitole se budeme věnovat podrobnějšímu rozboru jednotlivých geometrických tvarů, které mohou ohraničovat oblast. Mohou být libovolné; ovšem pro účely této práce byly zvoleny 2 druhy – úsečka a Bézierova křivka.

2.1.1 Úsečka

Úsečka je definována jako podmnožina $\boldsymbol{U}_{A\vec{u}}$ prostoru $\mathbb{R}^n$, jejíž parametrický předpis je následující:

$$\boldsymbol{U}_{A\vec{v}} = \{\boldsymbol{A} + \vec{\boldsymbol{v}} \cdot t \in \mathbb{R}^n : \quad \boldsymbol{A}, \vec{\boldsymbol{v}} \in \mathbb{R}^n, \quad t \in \langle 0,1 \rangle\},$$

kde $\vec{\boldsymbol{v}}$ je směrový vektor úsečky definovaný:

$$\vec{\boldsymbol{v}} \in \mathbb{R}^n; \quad \vec{\boldsymbol{v}} = \boldsymbol{B} - \boldsymbol{A}.$$

Body $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^n$ jsou po řadě počáteční a koncové body úsečky; *t* je parametr:

Celý algoritmus vrstvení je založený na určování vzájemných poloh bodů a úseček, případně úhlů, které svírají. Proto zde uvedu několik základních definic, viz [3], které se pro určování vzájemných poloh používají.

Norma vektoru je zobrazení $\|.\|: \mathbb{R}^n \to \mathbb{R}$, pro které platí následující 4 podmínky:

i) $\|\vec{\boldsymbol{v}}\| = 0 \Leftrightarrow \vec{\boldsymbol{v}} = \vec{\boldsymbol{0}}; \quad \forall \vec{v} \in \mathbb{R}^n; \quad \vec{\boldsymbol{0}} = \{0,0,\ldots,0\}$

ii) $\|\vec{\boldsymbol{v}}\| \geq 0; \quad \forall \vec{\boldsymbol{v}} \in \mathbb{R}^n$

iii) $\|\alpha \vec{\boldsymbol{v}}\| = \alpha \|\vec{\boldsymbol{v}}\|; \quad \forall \alpha \in \mathbb{R}; \quad \forall \vec{\boldsymbol{v}} \in \mathbb{R}^n$

iv) $\|\vec{\boldsymbol{u}} + \vec{\boldsymbol{v}}\| \leq \|\vec{\boldsymbol{u}}\| + \|\vec{\boldsymbol{v}}\|; \quad \forall \vec{\boldsymbol{u}}, \vec{\boldsymbol{v}} \in \mathbb{R}^n$

Dále budeme výhradně používat tzv. *Eukleidovskou normu*, která je pro vektor $\vec{\boldsymbol{u}}$ definována:

$$\vec{\boldsymbol{u}} \in \mathbb{R}^n; \quad \|\vec{\boldsymbol{u}}\| = \sqrt{u_1^2 + u_2^2 + \ldots + u_n^2}.$$

Skalární součin je zobrazení $(.,.): \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$, které dané dvojici vektorů $\vec{\boldsymbol{u}}$ a $\vec{\boldsymbol{v}}$ přiřadí reálné číslo. Musí pro něj platit tyto podmínky:

i) $(\vec{\boldsymbol{u}} + \vec{\boldsymbol{v}}, \vec{\boldsymbol{w}}) = (\vec{\boldsymbol{u}}, \vec{\boldsymbol{w}}) + (\vec{\boldsymbol{v}}, \vec{\boldsymbol{w}}); \quad \forall \vec{\boldsymbol{u}}, \vec{\boldsymbol{v}}, \vec{\boldsymbol{w}} \in \mathbb{R}^n$

ii) $(\alpha \vec{\boldsymbol{u}}, \vec{\boldsymbol{v}}) = \alpha (\vec{\boldsymbol{u}}, \vec{\boldsymbol{v}}); \quad \forall \alpha \in \mathbb{R}; \quad \forall \vec{\boldsymbol{u}}, \vec{\boldsymbol{v}} \in \mathbb{R}^n$

iii) $(\vec{\boldsymbol{u}}, \vec{\boldsymbol{v}}) = (\vec{\boldsymbol{v}}, \vec{\boldsymbol{u}}); \quad \forall \vec{\boldsymbol{u}}, \vec{\boldsymbol{v}} \in \mathbb{R}^n$

iv) $(\vec{\boldsymbol{v}}, \vec{\boldsymbol{v}}) \geq 0; \quad (\vec{\boldsymbol{v}}, \vec{\boldsymbol{v}}) = 0 \Leftrightarrow \vec{\boldsymbol{v}} = \vec{\boldsymbol{0}}; \quad \forall \vec{\boldsymbol{v}} \in \mathbb{R}^n; \quad \vec{\boldsymbol{0}} = \{0,0,\ldots,0\}$

Dále budeme výhradně používat Euklidův *skalární součin* vektorů $\vec{\boldsymbol{u}}$ a $\vec{\boldsymbol{v}}$ definovaný následujícím způsobem:

$$\vec{\boldsymbol{u}}, \vec{\boldsymbol{v}} \in \mathbb{R}^n; \quad (\vec{\boldsymbol{u}}, \vec{\boldsymbol{v}}) = u_1 v_1 + u_2 v_2 + \ldots + u_n v_n.$$

Euklidova norma je indukována Euklidovým skalárním součinem následovně:

$$\|\vec{\boldsymbol{u}}\| = \sqrt{(\vec{\boldsymbol{u}}, \vec{\boldsymbol{u}})}.$$

Úhel $\alpha \in \mathbb{R}$ svírající dva vektory $\vec{\boldsymbol{u}}$ a $\vec{\boldsymbol{v}}$ je definován takto:

$$\cos(\alpha) = \frac{(\vec{\boldsymbol{u}}, \vec{\boldsymbol{v}})}{\|\vec{\boldsymbol{u}}\| \cdot \|\vec{\boldsymbol{v}}\|}.$$

Pomocí těchto zobrazení a definic jsme schopni spočítat úhel $\alpha \in \mathbb{R}$, který svírají dvě úsečky $\boldsymbol{U}_{A\vec{u}}$ a $\boldsymbol{V}_{B\vec{v}}$, které mají aspoň jeden společný bod. Do předchozího vzorce dosadíme směrové vektory úseček.

$$\boldsymbol{U}_{A\vec{u}} = \{\boldsymbol{A} + \vec{\boldsymbol{u}} \cdot t \in \mathbb{R}^n : \ \boldsymbol{A}, \vec{\boldsymbol{u}} \in \mathbb{R}^n, \ t \in \langle 0,1 \rangle\},$$
$$\boldsymbol{V}_{B\vec{v}} = \{\boldsymbol{B} + \vec{\boldsymbol{v}} \cdot s \in \mathbb{R}^n : \ \boldsymbol{B}, \vec{\boldsymbol{v}} \in \mathbb{R}^n, \ s \in \langle 0,1 \rangle\},$$
$$\alpha = \arccos\left(\frac{(\vec{\boldsymbol{u}}, \vec{\boldsymbol{v}})}{\|\vec{\boldsymbol{u}}\| \cdot \|\vec{\boldsymbol{v}}\|} \right).$$

Mějme vektor $\vec{\boldsymbol{v}} \in \mathbb{R}^2 : \ \vec{\boldsymbol{v}} = (x, y)$, pak *vektor* $\vec{\boldsymbol{u}}$, *kolmý na* $\vec{\boldsymbol{v}}$ *v pravotočivém smyslu* definujeme takto:

$$\vec{\boldsymbol{u}} = (-y, x).$$

2.1.2 Bézierova křivka

Bézierova křivka je množina $\boldsymbol{P}(t) \subset \mathbb{R}^n$, která je určena *řídícím polygonem*, což je konečná posloupnost *n* bodů v $\boldsymbol{P}_0, \boldsymbol{P}_1, \ldots, \boldsymbol{P}_n \in \mathbb{R}^n$ prostoru. Body na Bézierově křivce jsou dány tímto výrazem, viz [4]:

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

kde $B_i^n(t)$ jsou Bernsteinovy polynomy, konstruované následujícím způsobem:

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}; \ \ i = 0, \ldots, n \ ,$$

kde

$$\binom{n}{i} = \frac{n!}{(n-i)! \cdot i!}; \ \ n \in \mathbb{N}_0, \ \ i \in \{0,1,\ldots,n\},$$

kde $! : \ \mathbb{N}_0 \to \mathbb{N}$ je *faktoriál* a je definovaný tímto předpisem:

$$0! = 1,$$
$$i+1! = i! \cdot (i+1), \ \ i \in \mathbb{N}.$$

Pro ilustraci mějme Bézierovu křivku zadanou 4 body, pak její Bernsteinovy polynomy vypadají následujícím způsobem:

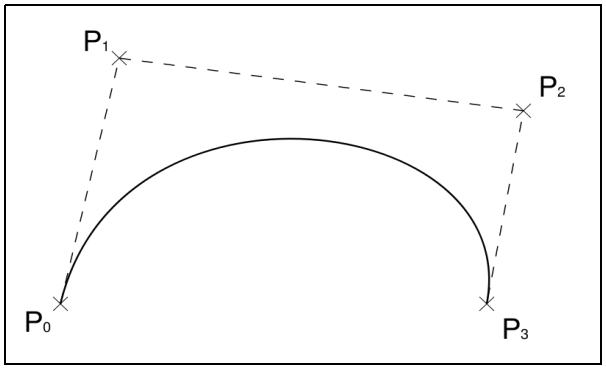
1. $B_0^4(t) = (1-t)^3$,

2. $B_1^4(t)=t(1-t)^2$,
3. $B_2^4(t)=t^2(1-t)$,
4. $B_3^4(t)=t^3$

a výsledný tvar křivky je:

$$\boldsymbol{P}(t)=\boldsymbol{P_0}\cdot(1-t)^3+\boldsymbol{P_1}\cdot t(1-t)^2+\boldsymbol{P_2}\cdot t^2(1-t)+\boldsymbol{P_3}\cdot t^3.$$

Z definice vyplývá, že Bézierova křivka vychází z prvního bodu řídícího polygonu, jelikož pro $t=0$: $\boldsymbol{P}(t)=\boldsymbol{P_0}$, a končí v posledním bodě polygonu $t=1$: $\boldsymbol{P}(t)=\boldsymbol{P_n}$. Další důležitou vlastností křivky je to, že její tečna v počátku je spojnicí prvního a druhého bodu řídícího polygonu a v koncovém bodu je její tečna ke spojnicí posledního a předposledního bodu polygonu. Tyto vlastnosti jsou ilustrovány na Obrázku 2.1.



Obrázek 2.1: Bézierova křivka

2.2 Výpočet délky křivky

Pro obecný výpočet délky křivky je nejprve třeba definovat několik dalších pojmů. *Derivace funkce,* viz [1]: Mějme funkci $f(x):\mathbb{R}\to\mathbb{R}$, pak její derivace je funkce $f'(x):\mathbb{R}\to\mathbb{R}$, která v bodech svého definiční oboru nabývá hodnot, existuje-li pravá strana rovnosti:

$$f'(x)=\lim_{h\to 0}\frac{f(x+h)-f(x)}{h};\quad x\in\mathbb{R}.$$

Jednostranná derivace funkce zprava je definována (opět za předpokladu existence pravé strany):

$$f'_{+}(x)=\lim_{h\to 0_+}\frac{f(x+h)-f(x)}{h};\quad x\in\mathbb{R}.$$

Obdobně platí vztah i pro derivaci zleva.

Mějme funkci $\boldsymbol{f}(x_1, x_2, \ldots, x_n): \mathbb{R}^n \to \mathbb{R}^m$, pak *parciální derivace* $\boldsymbol{f}$ podle x_i je definovaná vztahem:

$$\frac{\partial \boldsymbol{f}(x_1, \ldots, x_n)}{\partial x_i} = \lim_{h \to 0} \frac{\boldsymbol{f}(x_1, \ldots, x_{i-1}, x_i + h, x_{i+1}, \ldots, x_n) - \boldsymbol{f}(x_1, \ldots, x_n)}{h}, \quad i = 1, 2, \ldots, n.$$

Je dána funkce $f(x): \mathbb{R} \to \mathbb{R}$, pak aby byla funkce *diferencovatelná na intervalu* $I \subset \mathbb{R}: \; I = \langle a, b \rangle$, musí být splněny následující tři podmínky:

1. $x \in (a, b) \Rightarrow f'(x) \in \mathbb{R}$
2. $f'_+(a) \in \mathbb{R}$
3. $f'_-(b) \in \mathbb{R}$

Opačným procesem k derivaci je *integrace*. Potřebujeme k ní definici primitivní funkce: Funkce $F(x): \; \mathbb{R} \to \mathbb{R}$ se nazývá *primitivní funkcí* k funkci $f(x): \; \mathbb{R} \to \mathbb{R}$ na otevřeném intervalu *I*, když:

$$\forall x \in I: \; F'(x) = f(x).$$

Primitivní funkce $F(x): \; \mathbb{R} \to \mathbb{R}$ k zadané funkci $f(x): \; \mathbb{R} \to \mathbb{R}$ je jednoznačná až na konstantu a značíme ji:

$$\int f(x)\,dx = F(x) + c; \quad c \in \mathbb{R}.$$

Levou stranu této rovnice nazveme *neurčitým integrálem*. Abychom integrál vůbec mohli najít, vyžaduje se spojitost funkce *f(x)* na intervalu *I*.

Dalším důležitým pojmem pro výpočet délky křivky je určitý integrál. Buď funkce f omezená v intervalu $\langle a, b \rangle$; tzn., že existují čísla $m, M \in \mathbb{R}$ taková, že:

$$m \le f(x) \le M; \; \forall x \in \langle a, b \rangle.$$

Pro každé dělení D intervalu $\langle a, b \rangle$ označme:

$$s(D) = \sum_{k=1}^{n} \left(\inf_{x \in \langle x_{k-1}, x_k \rangle} f(x) \right) \cdot (x_k - x_{k-1}) \quad \text{a} \quad S(D) = \sum_{k=1}^{n} \left(\sup_{x \in \langle x_{k-1}, x_k \rangle} f(x) \right) \cdot (x_k - x_{k-1}).$$

Dělení intervalu $\langle a, b \rangle$ je definováno takto:

$$D: \; a = x_0 < x_1 < \ldots < x_{n-1} < x_n = b.$$

Dolním (resp. *horním*) *Riemannovým integrálem funkce f od a do b*, rozumíme číslo:

$$\underline{\int_a^b} f(x)\,dx = \sup \{ s(D): \; D \text{ je dělení } \langle a, b \rangle \},$$

resp.:

$$\overline{\int_a^b} f(x)\,dx = \inf \{ S(D): \; D \text{ je dělení } \langle a, b \rangle \}.$$

Platí-li:

$$\int_{\underline{a}}^{b} f(x)\,dx = \int_{a}^{\underline{b}} f(x)\,dx,$$

pak nazýváme toto číslo *Riemannovým integrálem funkce f od a do b* a značíme:

$$\int_{a}^{b} f(x)\,dx.$$

Máme-li graf nezáporné funkce *f(x)*, pak hodnota Riemannova integrálu odpovídá obsahu plochy mezi funkcí *f(x)* a osou x. Pokud funkce leží pod osou x, je hodnota integrálu záporná.

Nyní jsme schopni spočítat délku libovolné křivky na intervalu $I \in \langle a, b \rangle$. Mějme funkci *f(x)* diferencovatelnou na $\langle a, b \rangle$, Její délka pak odpovídá následujícímu integrálu:

$$\int_{a}^{b} \sqrt{1 + [f'(x)]^2}.$$

Pokud máme křivku $\boldsymbol{\varphi}: I \to \mathbb{R}^m$ zadanou parametricky následujícím způsobem:

$$\boldsymbol{\varphi} = (\varphi_1(t), \varphi_2(t), \ldots, \varphi_m(t)), \quad t \in \langle a, b \rangle,$$

kde $\varphi_i(t)$ jsou diferencovatelné na $\langle a, b \rangle$, pak její délka na intervalu $I \in \langle a, b \rangle$ je rovna tomuto integrálu:

$$\int_{a}^{b} \sqrt{[\varphi'_1(t)]^2 + [\varphi'_2(t)]^2 + \ldots + [\varphi'_m(t)]^2}\,dt.$$

Speciálně délka úsečky $\boldsymbol{U}_{A\vec{u}}$ je dána normou jejího směrového vektoru $\|\vec{v}\|$. Avšak pro obecnou Bézierovu křivku neznáme obecný analytický předpis, jak vypočítat tento integrál.

Mějme pro náš případ Bézierovu křivkou v 2D zadanou *n* body. Její parametrická rovnice pak vede na tvar:

$$(x(t), y(t)) = \boldsymbol{P}(t) = \sum_{i=0}^{n} \boldsymbol{P}_i B_i^n(t); \quad t \in \langle 0, 1 \rangle,$$

kde *x(t)* a *y(t)* jsou polynomy stupně *n*. Jejich derivace bude tedy polynom stupně maximálně *n – 1*. Pokud tento polynom dosadíme do vzorce pro výpočet délky křivek, dostáváme pod odmocninou polynom stupně maximálně *2n – 2*:

$$\int_{0}^{1} \sqrt{k_0 + k_1 t + k_2 t^2 + \ldots + k_{2n-3} t^{2n-3} + k_{2n-2} t^{2n-2}}\,dt = \ ?\ ;\ k_0, k_1, \ldots, k_{2n-2} \in \mathbb{R}.$$

A zde vzniká problém, obecně tento integrál totiž analyticky řešit nelze. Jsme tedy nuceni řešit danou úlohu numericky.

Jedna z možností je aproximovat Bézierovu křivku pomocí krátkých úseček, kdy délku

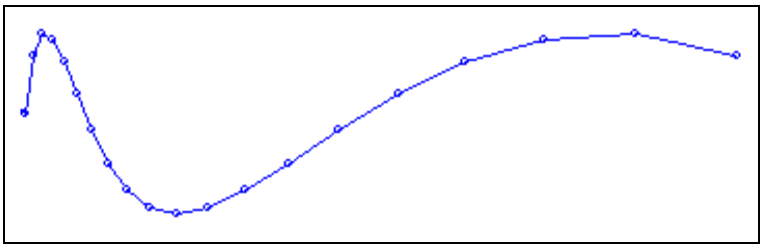
odhadneme jako součet délek těchto úseček. Pokud délka těchto diskretizovaných úseček půjde k nule, pak se limitně blížíme k délce křivky. Křivku diskretizujeme za pomocí parametru *t*. Jelikož $t \in \langle 0,1 \rangle$ stačí zvolit diskretizační krok Δt, se kterým budeme interval dělit.
Buď tedy φ^D diskretizací definičního oboru Bézierovy křivky.

$$\varphi^D = (t_0, t_1, \ldots, t_n): \quad t_i = i \cdot \Delta t, \quad i = 0,1,\ldots,n$$

Pak platí:

$$k \to 0 \;\Rightarrow\; \sum_{i=1}^{n} \left| \boldsymbol{P}(t_i) - \boldsymbol{P}(t_{i-1}) \right| \to \int_0^1 \sqrt{k_0 + k_1 t + k_2 t^2 + \ldots + k_{2n-3} t^{2n-3} + k_{2n-2} t^{2n-2}}\, dt\,.$$

U Bézierovy křivky je nutno počítat s dostatečně malým Δt, jelikož pokud interval $\langle 0,1 \rangle$ dělíme na stejné díly, neznamená to, že jsme schopni stejnoměrně rozdělit i křivku viz Obrázek 2.2, kde $\Delta t = 0{,}05$.



Obrázek 2.2: Diskretizace Bézierovy křivky pomocí parametru *t*

3 Metoda vrstvení

V této kapitole popíšeme jednu z nejpoužívanějších metod na triangulaci 2D oblastí. Nejprve je ovšem třeba definovat několik pojmů, které popisují vlastnosti oblasti, kterou chceme diskretizovat, a vlastnosti celé sítě. Definice jsou převzaty z [2].

3.1 Definice

Vektorovou funkcí z rozumíme zobrazení z $\mathbb{R}^n \to \mathbb{R}^m$ do $\mathbb{R}^n$.

Křivkou zadanou parametricky v $\mathbb{R}^m$, dále jen křivka, rozumíme každou spojitou vektorovou funkci:

$$\varphi: \quad I \to \mathbb{R}^m,$$

kde $I = D_\varphi \subset \mathbb{R}$ je interval.

Množinu $\langle \varphi \rangle = \varphi(I) = \{ \varphi(t) : t \in I \} \subset \mathbb{R}^m$ pak nazýváme *geometrickým obrazem křivky* φ.

Křivku φ nazýváme:

1. *jednoduchou,* je-li φ prosté;
2. *uzavřenou*, je-li $I=\langle a,b\rangle (a,b\in\mathbb{R};\ a<b)$ a navíc $\varphi(a)=\varphi(b)$;
3. *jednoduchou uzavřenou*, je-li φ uzavřená a platí:

$$t_1,t_2\in\langle a,b\rangle,\ 0<|t_1-t_2|<b-a\Rightarrow\varphi(t_1)\neq\varphi(t_2).$$

Diskretizací φ^D *definičního oboru křivky* $\varphi: I\to\mathbb{R}^m, I=\langle a,b\rangle$ nazveme tuto konečnou uspořádanou množinu bodů:

$$\varphi^D=(t_1,t_2,\ldots,t_n)$$

kde $t_i\in I,\ i=1,2,\ldots,n;\ a=t_1<t_2<\ldots<t_n=b.$

Množinu $\Omega\in\mathbb{R}^n$ nazýváme *oblastí*, je-li současně :

1. Ω je *otevřená*: $(\forall x\in\Omega)(\exists U(x)):\ x\in U(x)\subset U$
2. Ω je *souvislá*: Každé dva body Ω lze spojit křivkou v Ω;
   přesněji pro každé dva body $\alpha,\beta\in\Omega$ existuje křivka $\varphi:\ \langle a,b\rangle\to\Omega\subset\mathbb{R}^n$ taková, že $\varphi(a)=\alpha$ a $\varphi(b)=\beta$.

Hranice oblasti Ω značíme $\partial\Omega$.

Uzávěr oblasti Ω je množina všech bodů prostoru $\mathbb{R}^m$, jejichž libovolné okolí U má s oblastí Ω neprázdný průnik:

$$\overline{\Omega}=\{x\in\mathbb{R}^m:\ \forall U(x), U(x)\cap\Omega\neq\emptyset\}$$

platí, že $\overline{\Omega}=\Omega\cup\partial\Omega$.

Nechť φ je jednoduchá uzavřená křivka v $\mathbb{R}^2$. Pak existují oblasti $\Omega_1,\Omega_2\subset\mathbb{R}^2$ takové, že:

1. $\mathbb{R}^2\setminus\langle\varphi\rangle=\Omega_1\cup\Omega_2$,
2. $\Omega_1\cap\Omega_2=\emptyset$,
3. $\partial\Omega_1=\partial\Omega_2=\langle\varphi\rangle$,
4. Ω_1 je omezená a Ω_2 je neomezená v $\mathbb{R}^2$.

Oblast $\Omega_1=int\,\varphi$ nazýváme *vnitřkem křivky* φ, oblast $\Omega_2=ext\,\varphi$ se nazývá *vnějšek křivky* φ.

Oblast nazveme *jednoduše souvislou*, pokud platí pro každou uzavřenou křivku φ : $\varphi:\ \langle a,b\rangle\to\Omega\subset\mathbb{R}^2\Rightarrow int\,\varphi\subset\Omega$.

Vrstvou v síti označíme posloupnost bodů z $\mathbb{R}^2$.

Element (prvek) sítě označme K_i, v našem případě půjde o trojúhelník.

Diskretizační síť τ definujeme:

$$\tau = \{K_i : \ i = 0,1,\ldots,n\}$$

kde K_i jsou trojúhelníkové elementy sítě, které pokrývají diskretizovanou oblast:

$$dom\,\tau = \sum_{i=0}^{n} K_{i.}$$

a každé dva různé K_i, K_j mají buď právě jednu společnou hranu, nebo právě 1 společný vrchol nebo $K_i \cap K_j = \emptyset$. Snažíme se o to, aby $dom\,\tau \approx \Omega$, což nelze např. pro nepolygonální Ω.

3.2 Vrstvení ve 2D

Na vstupu algoritmu máme zadanou uspořádanou množinu Φ křivek φ_i zadaných parametricky v pravotočivém smyslu, které ohraničují zvolenou konvexní jednoduše souvislou oblast $\Omega \subset \mathbb{R}^2$:

$$\Phi = (\varphi_1, \varphi_2, \ldots, \varphi_n)$$

kde $\varphi_i : I_i \to \mathbb{R}^2$ jsou křivky.

Pro množinu Φ musí platit toto pravidlo: Nechť $I_i = \langle a_i, b_i \rangle : \ \forall i \in \{1,\ldots,n\}$, pak

$$\forall i \in \{1,\ldots,n-1\} \ \ \varphi_i(b_i) = \varphi_{i+1}(a_{i+1}) \wedge \varphi_n(b_n) = \varphi_1(a_1)$$

Jinými slovy křivka musí vycházet z posledního bodu předcházející křivky a poslední křivka musí končit v počátečním bodu první křivky.

3.2.1 Krok 1: Vytvoření základní vrstvy

Dalším krokem v našem algoritmu je vytvoření základní vrstvy V_0 z množiny *n* křivek. Máme zadanou V_0: délku úseček, na kterou chceme zadané křivky diskretizovat. Základní vrstvu lze popsat následujícím způsobem: Nechť $\varphi_i^D = (t_1^i, \ldots, t_{m_i}^i)$ je diskretizací definičního oboru křivky $\varphi_i : \ I_i \to \mathbb{R}^2$, pak platí pro vrstvu:

$$V_0 = \{(\varphi_1(t_1^1), \varphi_1(t_2^1), \ldots, \varphi_1(t_{m_1}^1), \ \varphi_2(t_2^2), \ldots, \varphi_2(t_{m_2}^2), \ \ldots, \varphi_n(t_2^n), \ldots, \varphi_n(t_{m_n-1}^n))\}.$$

Jak je vidět, tak z první křivky načteme všechny body, z dalších pak vynecháme jejich počátky a z poslední křivky vynecháme navíc i její koncový bod, to vše kvůli tomu, že křivky na sebe navazují.

Body $v_1^0 = \varphi_1(t_1^1)$ až $v_k^0 = \varphi_n(t_{m_n-1}^n)$ označíme jako *vrcholy ve vrstvě* V_0. Mějme tedy vrchol v_j^0 pak symboly v_{j-1}^0 a v_{j+1}^0 jsou myšleny předcházející a následující vrcholy ve stejné vrstvě.

Tímto jsme si ze zadaných vstupních křivek vytvořili základní vrstvu, z ní pak začneme

postupně vytvářet další vrstvy, přičemž použijeme níže popsaný soubor pravidel.

3.2.2 Krok 2: Výpočet úhlů

Mějme tedy vrstvu V_i s vrcholy $\boldsymbol{v}_1^i, \ldots, \boldsymbol{v}_{k_i}^i, k_i \in \mathbb{N}$, jako další spočítáme úhly u jednotlivých vrcholů. Tzn. u vrcholu $\boldsymbol{v}_j^i$ úhel, který svírají směrový vektor úsečky určené body $\boldsymbol{v}_{j-1}^i$ a $\boldsymbol{v}_j^i$ a směrový vektor úsečky daný body $\boldsymbol{v}_j^i$ a $\boldsymbol{v}_{j+1}^i$. Postup je podrobněji popsán v kapitole 2.

Abychom generovali téměř rovnostranné trojúhelníky, zkušenost ukázala, že je vhodné rozlišovat následující úhly:

i) *pseudoostrý úhel* α: $0 \le \alpha < \frac{2\pi}{5}$,

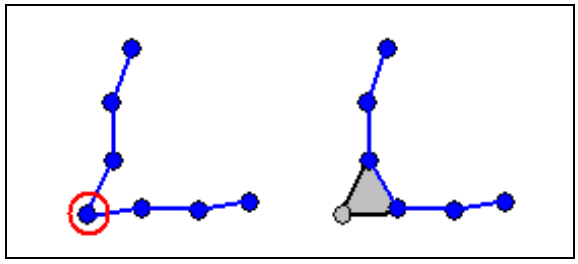
ii) *pseudotupý úhel* α: $\frac{2\pi}{5} < \alpha \le \frac{4\pi}{5}$,

iii) *pseudopřímý úhel* α: $\frac{4\pi}{5} < \alpha \le \frac{6\pi}{5}$

Díky předpokládané konvexnosti oblasti není třeba uvažovat další možné hodnoty úhlů.

3.2.3 Krok 3: Rušení nevhodných vrcholů

Nejprve z vrstvy vyškrtneme všechny vrcholy, u kterých jsou ostré úhly (viz Obrázek 3.1). Předpokládejme, že ve vrstvě V_i je ostrý úhel u vrcholu $\boldsymbol{v}_j^i$, pak do diskretizační sítě přidáme element K určený vrcholy $\boldsymbol{v}_{j-1}^i, \boldsymbol{v}_j^i, \boldsymbol{v}_{j+1}^i$, vyškrtneme vrchol $\boldsymbol{v}_j^i$ z V_i vrstvy a dále pak přepočítáme úhly u vrcholů $\boldsymbol{v}_{j-1}^i$ a $\boldsymbol{v}_{j+1}^i$.

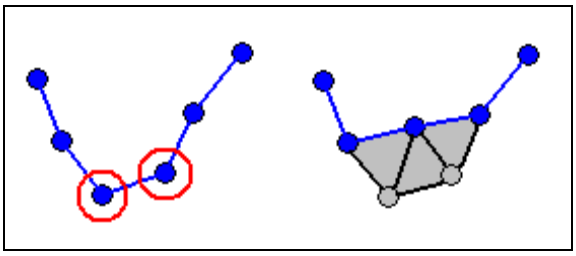


Obrázek 3.1: Zrušení pseudoostrého úhlu ve vrstvě

Pak z vrstvy vyškrtneme takovou dvojici po sobě jdoucích vrcholů, kde u obou vrcholů je tupý úhel. Mějme tedy takové dva vrcholy $\boldsymbol{v}_j^i, \boldsymbol{v}_{j+1}^i$ ve vrstvě V_i. Mezi body $\boldsymbol{v}_{j-1}^i, \boldsymbol{v}_{j+2}^i$ vytvoříme úsečku a její střed $\boldsymbol{S}$ přidáme do vrstvy. Z vrstvy vyškrtneme body $\boldsymbol{v}_j^i, \boldsymbol{v}_{j+1}^i$ a přepočítáme nové úhly. Do diskretizační sítě pak přidáme tyto tři elementy:

K_1 určený vrcholy $\boldsymbol{v}_{j-1}^i, \boldsymbol{v}_j^i, \boldsymbol{S}$,
K_2 určený vrcholy $\boldsymbol{v}_j^i, \boldsymbol{v}_{j+1}^i, \boldsymbol{S}$,
K_3 určený vrcholy $\boldsymbol{v}_{j+1}^i, \boldsymbol{v}_{j+2}^i, \boldsymbol{S}$.

Celý postup je opět ilustrován na Obrázku 3.2.



Obrázek 3.2: Zrušení dvou po sobě jdoucích pseudotupých úhlů ve vrstvě

Oba kroky, při kterých rušíme vrcholy, se provádí do té doby, dokud ve vrstvě jsou přítomny vrcholy, které splňují hledané podmínky. Pokud takové vrcholy již nejsou, můžeme přikročit k tvorbě další vrstvy.

3.2.4 Krok 4: Vytvoření nové vrstvy

Vrcholy procházíme postupně od prvního až k poslednímu a podle úhlů vytváříme body v další vrstvě V_k. Mohou nastat tyto dvě možnosti:

1. Úhel α u vrcholu $\boldsymbol{v}_j^i$ ve vrstvě V_i je pseudopřímý,
2. Úhel α u vrcholu $\boldsymbol{v}_j^i$ ve vrstvě V_i je pseudotupý.

Pseudopřímý

Nechť úhel α je pseudopřímý, pak rozlišíme tři případy:

a) vrchol $\boldsymbol{v}_j^i$ je první ve vrstvě V_i,
b) vrchol $\boldsymbol{v}_j^i$ není ani první, ani poslední ve vrstvě, V_i
c) vrchol $\boldsymbol{v}_j^i$ je poslední ve vrstvě V_i.

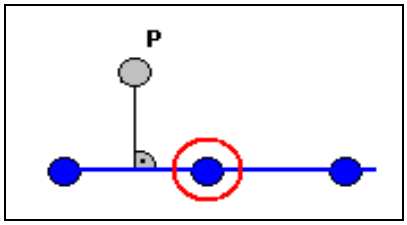
Postup přidávání bodů do vrstvy je pro všechny případy totožný, liší se pouze elementy, které poté přidáme do naší diskretizační sítě. Buď vektor $\vec{\boldsymbol{n}}$ normálovým vektorem k vektoru $\vec{\boldsymbol{u}} = \boldsymbol{v}_{j+1}^i - \boldsymbol{v}_j^i$. Postup, jak přesně vypočítat jeho souřadnice, je podrobně popsán v kapitole 2. Vektor $\vec{\boldsymbol{n}}$ přenásobíme číslem $k \in \mathbb{R}$ tak, aby platilo:

$$\|k \cdot \vec{\boldsymbol{n}}\| = \frac{\sqrt{3}}{2} \cdot d,$$

kde d je délka úsečky, na V_k které chceme síť diskretizovat. Hodnotu k počítame tak, aby vzniklý element určený $\boldsymbol{v}_j^i, \boldsymbol{v}_{j+1}^i, \boldsymbol{P}$ vrcholy byl rovnostranný. Nový vrchol $\boldsymbol{P}$ ve vrstvě tedy spočítáme:

$$\boldsymbol{P} = \boldsymbol{v}_j^i + \frac{1}{2} \cdot \vec{\boldsymbol{u}} + k \cdot \vec{\boldsymbol{n}}.$$

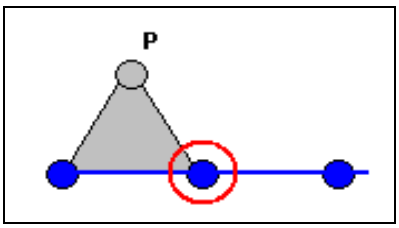
Jinými slovy, ze středu úsečky určené body $\boldsymbol{v}_j^i, \boldsymbol{v}_{j+1}^i$ vedeme kolmici dovnitř oblasti a na ní ve vzdálenosti $\|k \cdot \vec{\boldsymbol{n}}\|$ pak nalezneme bod $\boldsymbol{P}$. Postup je načrtnut na Obrázku 3.3:



Obrázek 3.3: Přidání nového vrcholu u pseudopřímého úhlu

Nyní rozebereme, které elementy přidat do naší sítě v závislosti na pozici vrcholu $\boldsymbol{v}_j^i$ ve vrstvě, viz Obrázky 3.4.a-d.

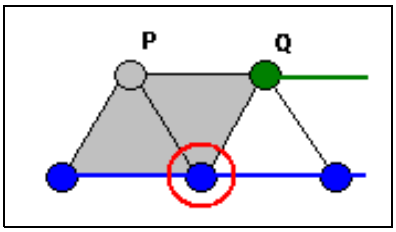
Ad a) Přidáme pouze jeden element K daný vrcholy $\boldsymbol{v}_j^i, \boldsymbol{v}_{j+1}^i, \boldsymbol{P}$.



Obrázek 3.4.a: Přidání elementu na počátku vrstvy

Ad b) Jelikož jsme uprostřed vrstvy, pak již v některém z předchozích kroků musel být přidaný nějaký vrchol do nové vrstvy V_k, označme jej $\boldsymbol{Q}$. Pak přidáme dva elementy:

K_1 určený vrcholy $\boldsymbol{v}_j^i, \boldsymbol{v}_{j+1}^i, \boldsymbol{P}$ a
K_2 určený vrcholy $\boldsymbol{v}_i^i, \boldsymbol{P}, \boldsymbol{Q}$.



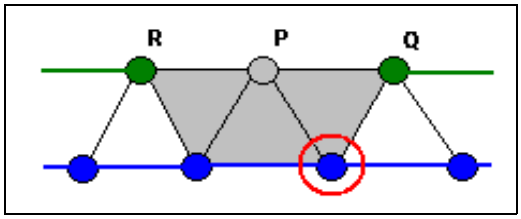
Obrázek 3.4.b: Přidání elementů uprostřed vrstvy

Ad c) Opět máme vrchol $\boldsymbol{Q}$, pak v tomto případě je třeba rozlišit dva případy:

1. Úhel u prvního vrcholu ve vrstvě V_i je pseudopřímý,
2. Úhel u prvního vrcholu ve vrstvě V_i je pseudotupý.

V prvním případě označme $\boldsymbol{R}$ první vrchol přidaný do vrstvy V_k, pak přidáme tyto 3 elementy, viz Obrázek 3.4.c:

K_1 určený vrcholy $\boldsymbol{P}, \boldsymbol{v}_j^i, \boldsymbol{Q}$,
K_2 určený vrcholy $\boldsymbol{v}_j^i, \boldsymbol{P}, \boldsymbol{v}_1^i$ a
K_3 určený vrcholy $\boldsymbol{P}, \boldsymbol{v}_1^i, \boldsymbol{R}$.

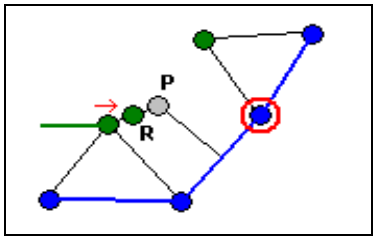


Obrázek 3.4.c: Přidání elementů na konci vrstvy (následuje pseudopřímý)

Ve druhém případě opět označíme $\boldsymbol{R}$ první vrchol přidaný co vrstvy V_k, pak abychom zaručili požadovanou rovnostrannost elementů, přepočítáme souřadnice bodu $\boldsymbol{R}$ za pomocí bodu $\boldsymbol{P}$ takto:

$$\boldsymbol{R}_{nový} = \frac{\boldsymbol{R}_{původní} + \boldsymbol{P}}{2}.$$

Nový bod $\boldsymbol{R}$ bude tedy ležet ve středu úsečky dané body $\boldsymbol{R}$ a $\boldsymbol{P}$. Bod $\boldsymbol{P}$ do nové vrstvy již nepřidáme, viz Obrázek 3.5.

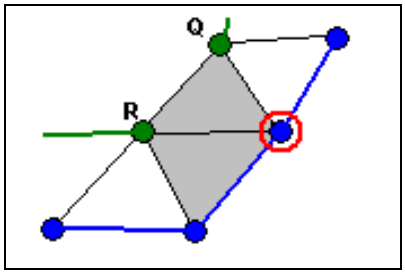


Obrázek 3.5: Přepočítání souřadnic prvního bodu vrstvy

Do sítě přidáme potom tyto dva elementy ($\boldsymbol{Q}$ je opět poslední přidaný vrchol v nové vrstvě):

K_1 určený vrcholy $\boldsymbol{Q}, \boldsymbol{v}_j^i, \boldsymbol{R}$,
K_2 určený vrcholy $\boldsymbol{v}_j^i, \boldsymbol{R}, \boldsymbol{v}_1^i$.

Celý postup opět znázorněn na Obrázku 3.4.d.



Obrázek 3.4.d: Přidání elementů na konci vrstvy (následuje pseudotupý)

Pseudotupý

Mějme úhel α pseudotupý, pak rozlišujeme tři případy:

a) vrchol $\boldsymbol{v}_j^i$ je první ve vrstvě V_i,
b) vrchol $\boldsymbol{v}_j^i$ je uvnitř vrstvy V_i ,
c) vrchol $\boldsymbol{v}_j^i$ je poslední ve vrstvě V_i.

Ad a) Vypočteme první vrchol $\boldsymbol{P}$ a zařadíme jej do nové vrstvy (nepřidáme elementy!): Mějme vektor $\vec{\boldsymbol{u}} = \boldsymbol{v}_{j+1}^i - \boldsymbol{v}_{j-1}^i$, spočítáme k němu normálový $\vec{\boldsymbol{n}}$ vektor, přenásobíme jeho souřadnice číslem $k \in \mathbb{R}$, tak aby:

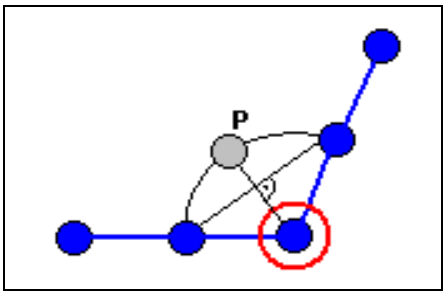
$$\|k \cdot \vec{\boldsymbol{n}}\| = d .$$

kde d, je zadaná délka diskretizované úsečky. Vektor přenásobíme, protože chceme, aby se elementy sítě co nejvíce blížily rovnostranným trojúhelníkům (ilustrováno na Obrázku 3.6).
Pro vrchol $\boldsymbol{P}$ pak platí:

$$\boldsymbol{P} = \boldsymbol{v}_j^i + k \cdot \vec{\boldsymbol{n}} .$$

Zároveň pak přibližně platí následující rovnosti:

$$\|\boldsymbol{P} - \boldsymbol{v}_j^i\| \approx \|\boldsymbol{v}_{j+1}^i - \boldsymbol{v}_j^i\| \approx \|\boldsymbol{v}_j^i - \boldsymbol{v}_{j-1}^i\|$$



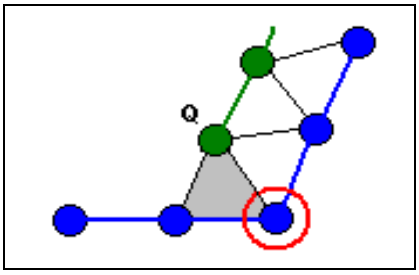
Obrázek 3.6: Výpočet souřadnic bodu P

Ad b) Obdobným postupem jako v předchozím případě spočítáme vrchol $\boldsymbol{P}$. Jelikož jsme uprostřed vrstvy, tak jsme někdy v minulosti museli do nové vrstvy přidat již nějaký

vrchol. Opět označme tedy poslední vrchol v nové vrstvě $\boldsymbol{Q}$, jeho souřadnice přepočítáme takto:

$$\boldsymbol{Q}_{nový} = \frac{\boldsymbol{Q}_{původní} + \boldsymbol{P}}{2}.$$

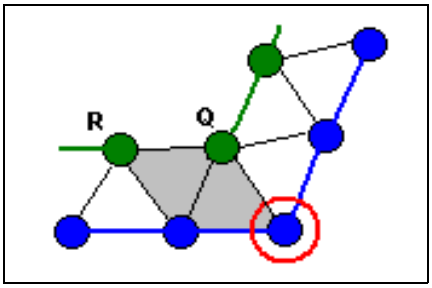
Vrchol $\boldsymbol{P}$ nepřidáme do nové vrstvy. Do sítě pak přidáme element určený vrcholy $\boldsymbol{v}_j^i, \boldsymbol{Q}, \boldsymbol{v}_{j+1}^i$, viz Obrázek 3.7.



Obrázek 3.7: Přidání elementu (pseudotupý úhel uprostřed vrstvy)

Ad c) Pokud jsme na konci vrstvy, pokračujeme stejně jako v předchozím případě. Nechť $\boldsymbol{R}$ je první bod v nové vrstvě, pak přidáme následující elementy, viz Obrázek 2.8:

K_1 určený vrcholy $\boldsymbol{v}_j^i, \boldsymbol{Q}, \boldsymbol{v}_1^i$,
K_2 určený vrcholy $\boldsymbol{Q}, \boldsymbol{v}_1^i, \boldsymbol{R}$.



Obrázek 3.8: přidání elementů (pseudotupý úhel na konci vrstvy)

Tento soubor pravidel aplikujeme na vrstvy, dokud počet bodů ve vrstvě neklesne pod 7. Pak už jednoduše vyplníme tuto malou vrstvu elementy a vrstvení ukončíme.

3.2.5 Krok 5: Dělení na podvrstvy

Během algoritmu může docházet k tomu, že se nám nově přidávaná vrstva rozdělí na dvě nebo více podvrstev. Zde tedy navrhneme, jak tento problém řešit. Algoritmus popíšeme v pseudoprogramátorském jazyce.

Mějme zásobník Z, do kterého ukládáme vrstvy. Na počátku do zásobníku vložíme vrstvu V_0. Do zásobníku budeme postupně ukládat všechny nové vrstvy, které je ještě třeba dále zpracovávat (počet vrcholů ve vrstvě je menší než 7).

Vstup: $V_0 = \left(v_{1,}^0 \ldots, v_{n_0}^0\right)$ kde n_0 je počet vrcholů ve vrstvě V_0.
Výstup: diskretizovaná síť τ.

```
k = 0;
while (Z neprázdný)
{
      nacti_vrstvu_z_vrcholu_zasobniku ( V_k )
      vrstveni ( V_k )
      k = k + 1;
}

vrstveni ( V_k )
      if ( n_k < 7 )
            return;
      else
      {
            while (!rozdel ( V_k ))
                  ;
            vytvor_novou_vrstvu( W, τ )    //pridavani elementu do τ
            vloz_vrstvu_do_zasobniku( W )
      }

rozdel ( V_k )
      for i = 1,..., n_k − 2
            for j = i + 3,..., n_k                //horní mez se může lišit
                  if ( ||v_j^k − v_i^k|| < 2d )
                  {
                        vytvor_podvrstvu( X )
                        vloz_vrstvu_do_zasobniku( X )
                        vyskrtni_body( V_k )
                        return true;
                  }
      return false;
      }
```

`vytvor_novou_vrstvu(` W, τ `)`

Zde používáme veškeré kroky, které jsou popsány v kapitole 3. Hlavní procedura, ve které se vytváří elementy a přidávají se do sítě.

`rozdel(` V_k `)`

V této části kontrolujeme, jestli se body ve vrstvě k sobě nepřiblížily natolik, že pokud bychom na ně aplikovali proceduru `vytvor_novou_vrstvu()`, tak by se nová vrstva protínala. Pokud toto nastane, pak vrstvu rozdělíme na dvě. Meze ve druhém for cyklu nastavíme vždy tak, aby se zkoumaný vrchol neporovnával s dvěma následujícími a dvěma předchozími vrcholy.

`vytvor_podvrstvu(` X `)`

Podvrstvu vytvoříme takto: Nechť $\|\boldsymbol{v}_j^k - \boldsymbol{v}_i^k\| < 2\mathrm{d}$
Pokud $\|\boldsymbol{v}_j^k - \boldsymbol{v}_i^k\| < d$, pak do podvrstvy vložíme všechny body $\boldsymbol{v}_i^k, \ldots, \boldsymbol{v}_j^k$:

$$X = \left(\boldsymbol{v}_i^k, \boldsymbol{v}_{i+1}^k, \ldots, \boldsymbol{v}_j^k,\right).$$

A z vrstvy V_k vyškrtneme body $\boldsymbol{v}_{i+1}^k, \ldots, \boldsymbol{v}_{j-1}^k$, vrstva pak bude vypadat následovně:

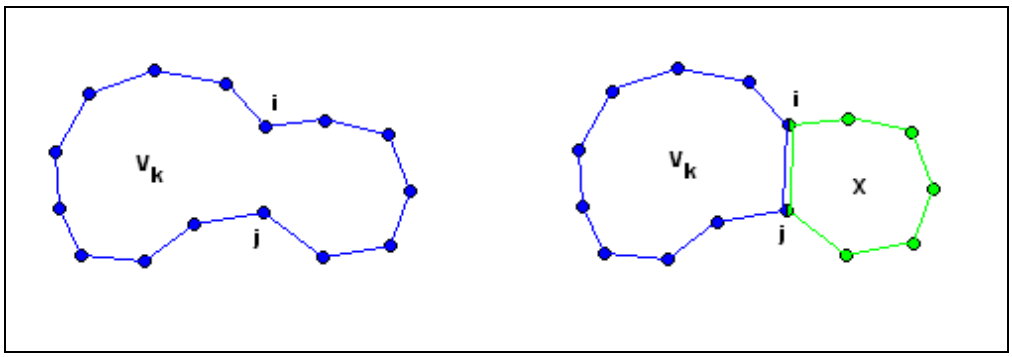
$$V_k = \left(\boldsymbol{v}_1^k, \ldots, \boldsymbol{v}_i^k, \boldsymbol{v}_j^k, \ldots, \boldsymbol{v}_{n_k}^k\right).$$

Musíme pak sesypat indexy, ať na sebe navazují.

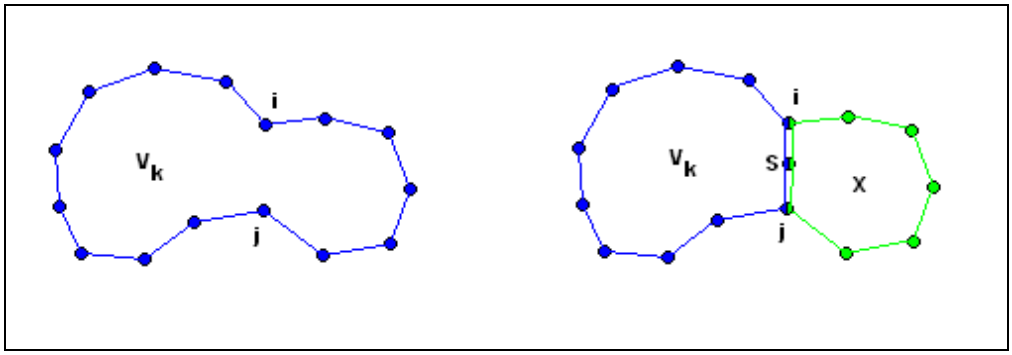
Pokud $d \le \|\boldsymbol{v}_j^k - \boldsymbol{v}_i^k\| < 2\mathrm{d}$, pak je postup obdobný, pouze budeme uvažovat navíc i střed $\boldsymbol{S}$ úsečky dané body $\boldsymbol{v}_j^k$ a $\boldsymbol{v}_i^k$. Vrstvy X a V_k pak vypadají takto:

$$X = \left(\boldsymbol{v}_i^k, \boldsymbol{v}_{i+1}^k, \ldots, \boldsymbol{v}_j^k, \boldsymbol{S}\right),$$
$$V_k = \left(\boldsymbol{v}_1^k, \boldsymbol{v}_2^k, \ldots, \boldsymbol{v}_i^k, \boldsymbol{S}, \boldsymbol{v}_j^k, \ldots, \boldsymbol{v}_{n_k}^k\right).$$

Oba dva případy jsou ilustrovány na Obrázcích 3.9 a 3.10.



Obrázek 3.9 Dělení vrstvy bez středu



Obrázek 3.10: Dělení vrstvy se středem

4 Implementace

Samotný algoritmus vrstvení není na pohled příliš složitý, ovšem při jeho implementaci může docházet k řadě komplikací a problémů, které je třeba mít na paměti. Některé z nich hlouběji probereme v této kapitole a zároveň navrhneme jejich možné řešení. Jedná se už o čistě technické věci týkající se samotného naprogramování.

4.1 Datové struktury a třídy

V této podkapitole uvedeme výčet nejdůležitějších datových struktur, které byly použity při implementaci algoritmu. U většiny struktur jsou uvedeme nejdůležitější proměnné a metody.

```
struct Bod
{double x,y;}
```

Jednoduchá struktura, k uložení x-ových a y-ových souřadnic bodů.

```
struct Body
{
      int pocet;
      Bod* data;
      double* uhly;
      bool* zrusit;
}
```

Datová struktura, která má reprezentovat jednotlivé vrstvy. Je v ní třeba uchovávat pole souřadnic všech bodů (proměnná `data`). Dále jejich aktuální počet, jelikož v průběhu algoritmu dochází situacím, kdy mohou být jednotlivé body vypouštěny z vrstvy. K rušení bodů z vrstvy slouží proměnná `zrusit`. Jedná se o pole příznaků, zda s body v dalším průběhu algoritmu ještě počítat nebo ne. Body nelze prostě jen vymazat, jelikož mohou být součástí některého z trojúhelníků, který již byl vytvořen jedné z minulých iterací vrstvení. Díky tomuto je nutné,

aby třída `Body` byla schopná řešit posun na sousední body ve vrstvě. Jinými slovy pokud mám zadaný index některého z bodu ve vrstvě, tak vrstva musí rozpoznat předchozí a následující bod a to pouze z těch, které jsou ještě platné (příznak `zrusit` je `false`). Je to potřeba hlavně kvůli počítání úhlů ve vrstvě a zároveň pro určení vzájemných vzdáleností mezi jednotlivými body. Poslední je proměnná `uhly`, která uchovává informace o úhlech v dané vrstvě. Hodnota `uhly[i]` je úhel v i-tém bodě ve vrstvě, který svírají spojnice i-tého bodu s předchozím a s následujícím bodem ve vrstvě.

```
struct Trojuhelnik
{int vrcholy[3][2]};
```

Struktura, ve které jsou uloženy informace o trojúhelníku tak, že první index i ve `vrcholy [i][j]` určuje vrchol trojúhelníku. Druhý index určuje jednak číslo vrstvy, ve které se daný bod nachází (`j = 0`) a také index bodu ve vrstvě (`j = 1`).

```
class Krivka
{
      double krok;
      double delka;
      Body body;

      virtual delka_krivky();
      virtual diskretizuj();
}
```

Základní třída pro námi uvažované typy křivek, které mohou ohraničovat zadanou oblast. Proměnná `krok` je definovaná pro všechny křivky stejně. Je načtena ze vstupního souboru a určuje délku úseček, na které se okrajové křivky mají diskretizovat; `delka` určuje celkovou délku křivky a pole `body` je pole diskretizovaných bodu křivky.

Každá odvozená třída si musí sama definovat, jakým způsobem spočítat její délku (procedura `delka_krivky`), a také proces, kterým se bude diskretizovat na jednotlivé úsečky (procedura `diskretizuj`). Přidat do programu další typ křivky znamená vytvoření nové odvozené třídy ke třídě `Krivka`.

```
class Bezier : public Krivka
{}

class Usecka : public Krivka
{}
```

4.2 Procedury

`nacti_krivky()`

Tato procedura je zde uvedena jen kvůli formátu vstupních dat. Geometrické tvary jsou

načítány ze souboru *data.txt*. Na prvním řádku bude uvedena délka úsečky, na kterou se zadané geometrické tvary mají diskretizovat. Toto číslo zároveň udává přibližnou vzdálenost dvou vrstev. Na dalším řádku je uveden počet křivek, které požadovanou postupně popisují hranici oblasti v pravotočivém směru. Na dalších řádcích jsou pak vypsány samotné křivky a to následujícím způsobem:

Jako první je uvedený počet řídících bodů, které křivku určují. Pro úsečku je tato hodnota rovna dvěma a pro Bézierovu křivku je jakákoliv větší a udává počet bodů řídícího polygonu. Dále na řádku následují x-ové a y-ové souřadnice těchto bodů (nejprve x-ová souřadnice prvního bodu, pak jeho y-ová souřadnice a pak souřadnice dalších bodů). Jsou odděleny mezerami. Na Obrázku 4.1 je uveden krátký příklad možného vstupního souboru, kde délka diskretizované úsečky je 10 a oblast je ohraničena dvěma Bézierovými křivkami a dvěma úsečkami. Oblast je svým tvarem podobná ležaté osmičce.

```
10
4
5 100 100 200 -100 300 500 400 -100 500 100
2 500 100 500 250
5 500 250 400 450 300 -150 200 450 100 250
2 100 250 100 100
```

Obrázek 4.1: Příklad vstupního souboru

`diskretizuj()`

Pro úsečku není třeba uvádět příklad její diskretizace, stačí si zvolit Δt, dosadit do rovnice úsečky a spočítat diskretizované body. Pokud máme Bézierovu křivku, tak je celá procedura komplikovanější. Algoritmus je popsán na Obrázku 4.2.

Budeme chtít Bézierovu křivku diskretizovat na co nejmenší úseky. Buď tedy φ^D diskretizací definičního oboru Bézierovy křivky $\boldsymbol{P} : I \to \mathbb{R}^2$. Pro jednotlivé body diskretizace bude platit následující:

$$\varphi^D = (t_0, t_1, \ldots, t_n): \quad t_i = i \cdot \Delta t, \quad i = 0, 1, \ldots, n$$

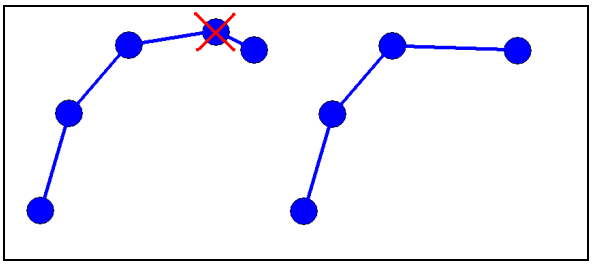
Zvolíme si tedy hodně malé Δt. V našem případě, kdy délka křivky odpovídala řadově stovkám, jsme nastavili hodnotu $\Delta t = 0{,}0001$. Po tomto rozdělení křivku spočítáme délky jednotlivých úseků, čímž dostáváme přibližný odhad celkové délky křivky.

Nyní je tedy potřeba křivku diskretizovat na posloupnost úseček, kde délka každé z nich přibližně odpovídá délce *d* načtené ze vstupního souboru. Prvním bodem v naší diskretizační síti bude počátek křivky. Od něj pak dále postupně sčítáme délky malých úseků a pokud hodnota tohoto součtu překročí zadanou délku *d*, vložíme koncový bod úseku do naší sítě. A znova začneme sčítat délky křivek. To vše uděláme pro všechny úseky a na konci do sítě vložíme koncový bod křivky.

```
Bezier::diskretizuj()
{
  j = 1;
  body.data[0] = pocatek;
  for (i = 0; i < pocetUseku; i++)
    if (suma > d)
    {
      suma = 0;
      body.data[j] = koncovyBodUseku[i];
      j++;
    }
    else
      suma = suma + delkaUseku[i];
}
```

Obrázek 4.2: Algoritmus diskretizace Bézierovy křivky

Musíme ovšem dát pozor, aby poslední úsečka nebyla moc krátká (menší než polovina vstupní délky). Pokud toto nastane, nevkládáme předposlední bod, ale pouze koncový. Bude to ilustrováno na Obrázku 4.3.



Obrázek 4.3: Diskretizace Bézierovy křivky (vlevo bod příliš blízko konci, vpravo oprava)

`vrstveni()`

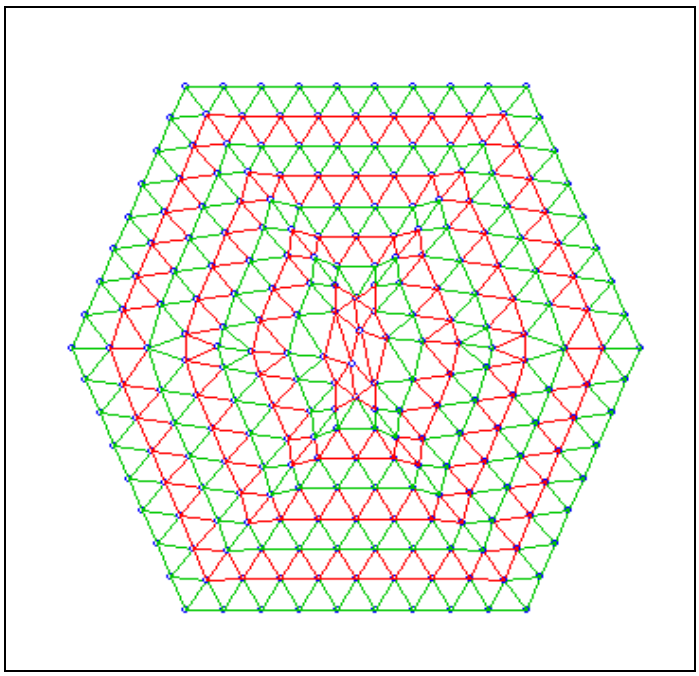
Toto je hlavní procedura, která provádí algoritmus vrstvení podrobně popsaný v 2. kapitole.

`zapis_do_souboru()`

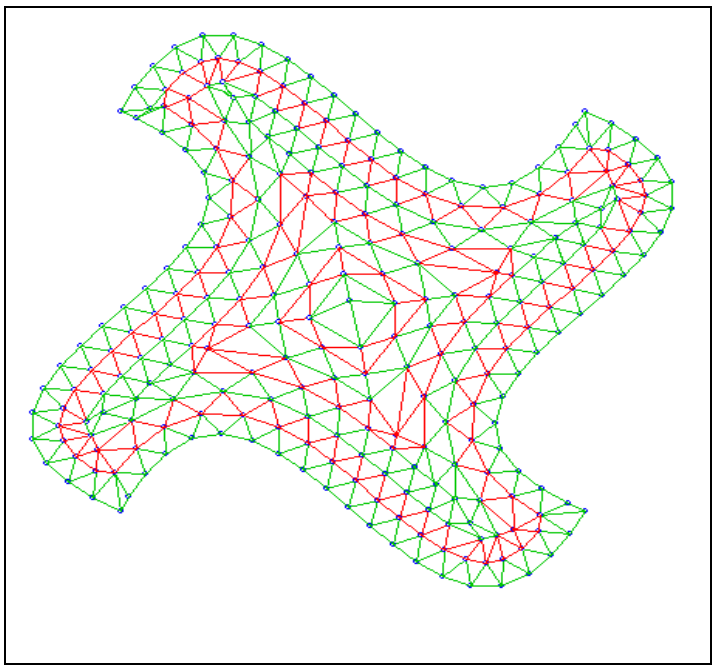
Na výstup je prováděný do souboru troj.txt, a to v tomto formátu. Zapisujeme souřadnice všech trojúhelníků, které jsme během vrstvení vytvořili. Na jednom řádku bude vždy nejprve x-ová souřadnice prvního vrchol následovaná jeho y-ovou souřadnici. Dále pak budou souřadnice druhého a třetího vrcholu. Další trojúhelníky začneme zapisovat až na další řádek.

5 Příklady

Zde uvedené příklady jsou pro oblasti, kde nedochází k dělení na podvrstvy. Samotné dělení je ještě třeba v programu doladit.



Obrázek 5.1: Diskretizace oblasti zadané úsečkami



Obrázek 5.2: Diskretizace oblasti zadané Bézierovými křivkami

6 Závěr

V této práci jsme se věnovali problematice triangulace 2-dimenzionální (2D) oblasti, která byla zadaná svými okrajovými křivkami. Vysvětlili jsme princip jednoho ze základních algoritmů, které se na triangulaci používají, algoritmu vrstvení. Popsali jsme jednu z možných implementací daného algoritmu a předestřeli řešení některých problémů, které při ní mohou vyvstat. V práci můžeme dále pokračovat a to buď vrstvením ve 3-dimenzionálním prostoru nebo se zabývat přímo metodou level-setů.

7 Zdroje:

[1] J. Bouchala - Matematická analýza 1, VŠB-TU Ostrava, 1998.

[2] J. Bouchala - Matematická analýza 3 – diferenciální a integrální počet vektorových funkcí, VŠB-TU Ostrava, 2001.

[3] Z. Dostál - Lineární algebra, VŠB-TU Ostrava, 2004.

[4] F. Ježek - Geometrické a počítačové modelování, Západočeská univerzita v Plzni, 2000.

[5] J. Schoeberl - NETGEN: An advancing front 2D/3D-mesh generator based on abstract rules, Computing and Visualization in Science 1, str. 41-52, 1997.

[6] J.A. Sethian – Level Set Methods and Fast Marching Methods, Cambridge Univerzity Press, 1996.