SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
Stavebná fakulta

Numerické riešenie úloh advekcie v prostredí DUNE

Diplomová práca

SvF-5343-39302

Študijný program:	Matematicko-počítačové modelovanie
Študijný odbor:	9.1.9 aplikovaná matematika
Katedra:	Katedra matematiky a deskriptívnej geometrie
Vedúci diplomovej práce:	RNDr. Peter Frolkovič, PhD.

2011 **Bc. Maroš Bohunčák**

Čestné prehlásenie

Vyhlasujem, ze som diplomovú prácu vypracoval samostatne s použitím uvedenej odbornej literatúry.

Bratislava 20. 5. 2011 .....

Vlastnoručný podpis

Poďakovanie

Ďakujem môjmu konzultantovi RNDr. Petrovi Frolkovičovi, PhD. za cenné rady a podnety, ktoré mi poskytoval počas tvorby tejto práce.

Bratislava 20. 5. 2011 .....

Vlastnoručný podpis

Abstrakt:
V práci sa zaoberáme aplikáciou numerickej level set metódy na riešenie matematického modelu rovnice advekcie pre šírenie sa pozemných požiarov. Na modelovanie hranice pozemného požiaru používame krivku, ktorá je implicitne vyjadrená pomocou izočiary level set funkcie. Takéto implicitné vyjadrenie nám umožňuje na popis pohybu hranice požiaru použiť rovnicu advekcie. Samotný pohyb hranice pozemného požiaru je opísaný rýchlostným poľom, ktoré zahŕňa zložku rýchlosti v smere vonkajšej normály k hranici požiaru, rýchlosť vetra a taktiež aj vplyv kvality paliva na rýchlosť šírenia sa požiaru. Do modelu zavádzame vplyv krivosti, ktorá je štandardným nástrojom pri práci s pohybujúcimi sa krivkami a umožňuje nám ovplyvňovať výsledný tvar krivky. Z numerického hľadiska pôsobí krivosť ako regularizačný faktor. Výslednú advekčno-difúznu rovnicu riešime metódou konečných objemov explicitnou metódou v čase pre advekčnú časť a semi-implicitnu metódou v čase pre difúznu časť. Aplikáciu implementujeme s využitím softvérovej knižnice Dune, pričom používame lokálne zjemňovanie výpočtovej siete.

Abstract:
This thesis deals with an application of a numerical level set method for solving a mathematical model of advection equation of a spread of surface fires. To model a fire front we use a curve, which is implicitly expressed by isoline of the level set function. Such implicit expression allows us to describe the movement of the fire front by the advection equation. The fire front movement of ground fire is caused by the velocity field, which includes component of the velocity in the direction of outer normal to the fire front, wind speed and also the impact of fuel quality on the speed of propagation of fire. We include the influence of curvature which is a standard tool when working with moving curves and allows us to control the final shape of the curve. From the numerical point of view, curvature acts as a regularization factor. We solve the resulting advection-diffusion equation using finite volume method that is explicit in time for the advection part and semi-implicit in time for the diffusion part. Our application is implemented using of software library Dune and we apply local refinement of the computational grid.

Obsah

Úvod

Spolu s rozvojom výpočtovej techniky sa v posledných desaťročiach do popredia postupne dostávali aj numerické metódy na riešenie diferenciálnych rovníc. Jednou zo známych rovníc je tzv. rovnica advekcie, alebo advektívna rovnica, ktorá popisuje prenos látky v nejakom rýchlostnom poli. Túto rovnicu použili v roku 1988 James Sethian a Stanley Osher na sledovanie vývoja krivky v čase, čo viedlo následne k masívnemu rozvoju level set metódy.

Sledovanie vývoja rovinných kriviek v čase, alebo ináč povedané evolúcie rovinných kriviek, má dnes v praxi mnoho aplikácií. Môžeme tu zaradiť modelovanie pohybu morských vĺn, hranice vyhorenej oblasti pri požiaroch, či rozhrania v materiáloch rôznych skupenstiev (napr. bublina vo vode). Významné aplikácie možno nájsť aj v spracovaní obrazu, kde krivky môžu popisovať hranice objektu, ktorý potrebujeme z obrazu segmentovať.

Nie vždy sa však krivky používajú na modelovanie nejakej formy hranice. Pri filtrácii obrazu spájajú miesta s rovnakými farebnými odtieňmi, teda pri ich zhlaďovaní dochádza k zhladeniu farebných prechodov (tu sa zas môžeme stretnúť s pomenovaním izočiary intenzity). V geodézii a robotike sa evolúcia kriviek používa na nájdenie najkratšej, resp. optimálnej trajektórie na povrchoch.

V našej diplomovej práci sa zaoberáme numerickým riešením evolúcie kriviek pomocou level set metódy na riešenie parciálnych diferenciálnych rovníc s dominantnou advektívnou časťou. Aj keď vyvinuté metódy, včítane ich implementácie v softvéri Dune, je možné použiť na viaceré uvedené aplikácie, v tejto práci sa podrobnejšie zameriame na matematický model šírenia sa pozemných požiarov.

Kapitola 1

Evolúcia rovinných kriviek

V tejto kapitole sa venujeme problematike šírenia sa pozemných požiarov a v súvislosti s nimi uvedieme kľúčové detaily, ktoré sa týkajú tejto konkrétnej aplikácie. Popisujeme explicitné a implicitné vyjadrenie krivky a jej pohybu, pričom dôraz kladieme na druhý spôsob, ktorý tvorí základ pre advektívnu level set rovnicu, ktorú následne odvádzame. Nakoniec definujeme znamienkovú funkciu vzdialenosti, ktorú neskôr využijeme.

1.1 Pozemné požiare

Našou hlavnou motiváciou je aplikovanie evolúcie kriviek na numerické modelovanie šírenia sa pozemných požiarov.

Každoročne sa na našej planéte vyskytnú tisíce obrovských požiarov, ktoré zanechávajú po sebe veľké materiálne škody a nanešťastie mnohokrát aj ľudské obete. Či už je to vinou človeka alebo nie, faktom je, že oheň je živel, ktorý sa veľmi ľahko vymkne spod kontroly. Preto podobne ako pri iných živelných pohromách, je v rámci bezpečnosti obyvateľov veľkou výhodou byť vopred pripravený. Samozrejme, vždy ak je to možné, je vhodnejší prístup namiesto riešenia prípadných následkov hľadať spôsob ako zamedziť príčine. Ešte donedávna lesníci a požiarnici reagovali na katastrofy zapríčinené požiarmi logicky tým, že sa snažili vznik požiarov obmedzovať všetkými možnými prostriedkami. Po rokoch štúdií sa však ukázalo,

že týmto spôsobom síce predídu menším požiarom, z dlhodobého hľadiska však všetko hralo v prospech oveľa väčším a ničivejším požiarom. Prišli teda s paradoxným riešením, ktorý by sme mohli nazvať „ohňom proti ohňu“. Dôvod je jednoduchý. Kontrolované požiare napríklad v lesoch, les zbavujú starého, odumretého a vyschnutého porastu, ktorý sa by sa ľahšie vznietil ako zdravý porast. Počítačový model môže byť v tomto prípade veľmi nápomocný pri predpovedaní šírenia sa požiaru a prípadnému vyhnutiu sa neželaným následkom.

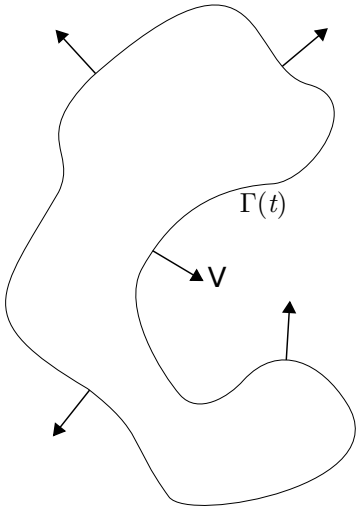
Šírenie sa pozemného požiaru závisí od viacerých faktorov. V našej aplikácií prirodzene predpokladáme, že požiar sa len rozširuje, teda čo už raz pri požiari vyhorelo, nemôže horieť opäť. Ako vyplýva z vyššie spomenutého, rýchlosť šírenia sa požiaru bude závisieť aj od kvality paliva. Nemenej dôležitú úlohu hrá aj rýchlosť a smer vetra. Ak si predstavíme hranicu požiaru v tvare písmena U, pričom vnútro je nevyhorená oblasť, môžeme predpokladať, že v tejto oblasti je vyššia teplota, kedže je zohrievaná z oboch strán. Teda aj tvar hranice požiaru ovplyvňuje jeho rýchlosť horenia. Všetky spomenuté faktory sme zahrnuli do nášho matematického modelu.

1.2 Explicitné vyjadrenie krivky a jej pohybu

Hranicu pozemného požiaru budeme modelovať pomocou *jednoduchej uzavretej krivky* Γ (obr. 1.1) v priestore $\Re^2$, ktorá ho rozdeľuje na uzavretú oblasť Ω^- pričom $\partial\Omega^- = \Gamma$ a otvorenú oblasť $\Omega^+ = \Re^2 \backslash \Omega^-$. Symboly znamienok vysvetlíme neskôr.

Jeden zo spôsobov ako vyjadriť krivku Γ je, že *explicitne* popíšeme všetky body, ktoré ležia na nej. Krivku musíme teda parametrizovať pre nejaký parameter $s \in [s_d, s_u]$. Vo všeobecnosti majú krivky zložité tvary a teda analytické vyjadrenie parametrizácie nemusí byť k dispozícií. Zaužívaný spôsob aproximovania explicitného vyjadrenia je diskretizácia intervalu $[s_d, s_u]$ na konečný počet bodov $s_d < s_1 < \cdots < s_i < s_{i+1} < \cdots < s_u$, kde podintervaly $[s_i, s_{i+1}]$ nemusia byť nutne rovnakej dĺžky. Pre každý bod $s_i \in [s_d, s_u]$ dostávame jeden bod krivky $\mathbf{x}(s_i)$ a teda pre jemnejšie delenie intervalu dostávame presnejšiu aproximáciu krivky. Body krivky, ktoré nie sú zahrnuté v intervale

$[s_d, s_u]$ musíme vyjadriť pomocou interpolácie.



Obr. 1.1: Jednoduchá uzavretá krivka.

Predstavme si teraz jednoduchý prípad, keď sa všetky body $\mathbf{x}(s_i) = \mathbf{x}(s_i, t)$ krivky $\Gamma = \Gamma(t)$ pohybujú (obr. 1.1). Pri explicitne vyjadrenej krivke to znamená, že musíme z nejakých známych pozícií bodov vypočítať jej nové pozície. Takýto prístup k modelovaniu pohybu krivky sa nazýva *Lagrangeovský*.

Zmenu pozície bodu $\mathbf{x}(s_i, t)$ v čase môžeme vyjadriť ako

$$\frac{\partial \mathbf{x}(s_i, t)}{\partial t} = \mathbf{V}(\mathbf{x}, t), \tag{1.1}$$

kde $\mathbf{V}(\mathbf{x}, t)$ je rýchlosť bodu $\mathbf{x}(s_i, t)$ a môže vo všeobecnosti závisieť od lokálnych alebo globálnych vlastností krivky alebo od iných (externých) vplyvov. Podrobnejšie sa opisu rýchlosti $\mathbf{V}(\mathbf{x}, t)$ venujeme v kapitole 2.

Venujme sa ďalej Langrangeovskému popisu pohybu krivky. Našim cieľom je zostrojiť matematický model, ktorý bude definovať rýchlosť $\mathbf{V}(\mathbf{x}, t)$ a pomocou ktorého budeme predpovedať pohyb našej krivky (teda bodov $\mathbf{x}(s_i, t)$). Situácia sa začne výrazne komplikovať napríklad v prípade viacerých expandujúcich kriviek, ktoré by sa po istom čase mali dotknúť, resp. spojiť do jednej. Problémom je totiž vystihnutie okamihu, v ktorom sú dva body krivky už dostatočne blízko na to, aby sme mohli prehlásiť, že sa krivky dotkli.

Analogicky to platí aj pre prípad jednej krivky, ktorá sa má rozdeliť na viac kriviek.

Takisto je v tomto prístupe potrebné riešiť problém redistribúcie bodov a teda okrem normálovej rýchlosti zaviesť aj tangenciálnu zložku celkovej rýchlosti. Pre detaily týkajúce sa modelovania pohybu krivky Lagrangeovským prístupom pozri [6], pre použitie Lagrangeovského prístupu na modelovanie šírenia sa pozemného požiaru viď [7].

Všetkým spomínaným ťažkostiam je možné sa vyhnúť pri implicitnom vyjadrení krivky a použití *level set metódy*, ktorú sme použili aj v našej práci.

1.3 Implicitné vyjadrenie pohybu krivky a level set metóda

Implicitné vyjadrenie krivky znamená, že ju reprezentujeme ako izočiaru nejakej funkcie. Postupujeme tak, že zostrojíme funkciu $\phi : \Re^2 \to \Re$, ku ktorej bude naša krivka napríklad nulovou izočiarou, ktorá je definovaná následovne:

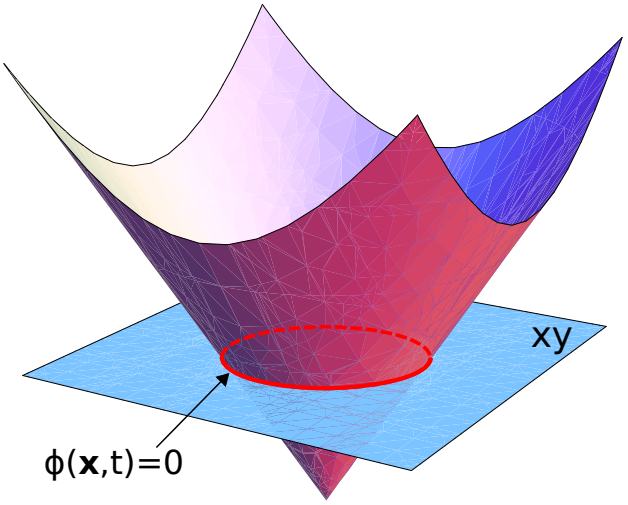
$$\Gamma = \{x \in \Re^2, \phi(x) = 0\} \tag{1.2}$$

Na výbere izočiary nezáleží, pretože pre každú izočiaru vieme funkciu $\phi(\mathbf{x})$ „posunúť“ tak, aby zodpovedala našej krivke.

Funkcia $\phi(\mathbf{x})$ sa niekedy nazýva *level set* alebo *úrovňová funkcia* a môže byť napríklad v tvare

$$\phi(\mathbf{x}) = |\mathbf{x} - \mathbf{a}| - r \tag{1.3}$$

kde $\mathbf{a} \in \Re^2$, $r \in \Re$ a $|\cdot|$ je euklidovská norma, pozri obrázok 1.2. Na zachytenie tvaru počiatočnej hranice požiaru budeme používať práve funkciu definovanú v (1.3).



Obr. 1.2: Graf level set funkcie (1.3) pre $t = 0$ so znázornenou nulovou izočiarou, ktorá predstavuje krivku, ktorej pohyb chceme sledovať. Pre jednoduchý prípad expandujúcej (sťahujúcej sa) krivky v tvare kružnice, jednotkovou rýchlosťou v smere normály, sa bude teda level set funkcia pohybovať nadol (nahor).

Implicitné vyjadrenie krivky so sebou prináša niekoľko výhod, ktoré si ilustrujeme pomocou funkcie definovanej v 1.3. Napríklad ak máme krivku popísanú nulovou izočiarou funkcie ϕ, vieme veľmi ľahko zistiť, na ktorej strane krivky sa nachádza nejaký bod $\mathbf{x} \notin \Gamma$ tak, že sa jednoducho pozrieme na znamienko hodnoty funkcie ϕ. Z tohto dôvodu sme zaviedli indexy $+/-$ pre oblasti Ω^- a Ω^+. Budeme uvažovať len prípady, kedy platí:

$$\begin{aligned} \phi(\mathbf{x}) &> 0 \quad \forall x \in \Omega^+ \\ \phi(\mathbf{x}) &< 0 \quad \forall x \in \Omega^- \\ \phi(\mathbf{x}) &= 0 \quad \forall \in \Gamma(t) \end{aligned}$$

Na rozdiel od toho, určenie pozície bodu $\mathbf{x}$ bodu vzhľadom na explicitne vyjadrenú krivku môže byť náročné. Štandardne by sme postupovali tak, že by sme z bodu $\mathbf{x}$ vyslali lúč a spočítali počet priesečníkov lúča s krivkou. Nepárny počet priesečníkov znamená, že bod leží mimo oblasti uzavretej krivkou, párny počet priesečníkov znamená, že bod leží vnútri oblasti uzavretej krivkou. Samozrejme, že popísaný postup je, v porovnaní s vyhodnotením

znamienka funkcie, omnoho komplikovanejší na implementáciu.

Funkciu ϕ využijeme aj na definície ďalších nástrojov, ktoré využijeme neskôr v matematických modeloch:

Gradient $\nabla\phi = \left(\frac{\partial\phi}{\partial x}, \frac{\partial\phi}{\partial y}\right)$ implicitne danej funkcie je kolmý na jej izočiary. Teda ak $\mathbf{x}_0$ je bod patriaci nulovej izočiare, gradient v tomto bode je vektor v smere jednotkovej vonkajšej normály $\mathbf{N}$ k našej krivke.

Jednotkovú vonkajšiu normálu ku krivke ukazujúcu z Ω^- do Ω^+ môžeme teda vyjadriť ako:

$$\mathbf{N} = \frac{\nabla\phi}{|\nabla\phi|}. \tag{1.4}$$

Krivosť κ definujeme ako divergenciu vonkajšej normály:

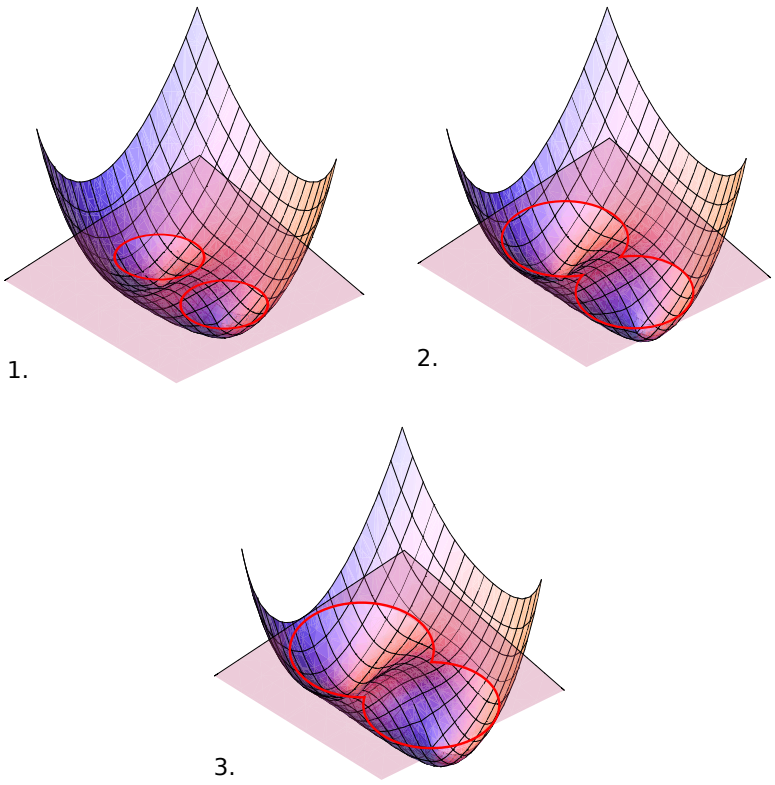
$$\kappa = \nabla \cdot \mathbf{N} = \nabla \cdot \left(\frac{\nabla\phi}{|\nabla\phi|}\right). \tag{1.5}$$

Pohyb (obrázok 1.3) implicitne vyjadrenej krivky popíšeme rovnicou advekcie, čo je hlavnou myšlienkou našej level set metódy na sledovanie evolúcie rovinných kriviek.

Level set metóda je numerická technika založená na tzv. *Eulerovskom prístupe*. Vo všeobecnosti to znamená, že problém riešime na ohraničenej podoblasti $D \subset \Re^n$, ktorú diskretizujeme na sieť s fixnými súradnicami jej bodov, pričom v každom bode počítame príslušnú reprezentačnú hodnotu funkcie ϕ.

Môže sa stať, a v praxi to tak väčšinou býva, že po diskretizácii oblasti na sieť s konečným počtom bodov nevieme, kde sa presne naša nulová izočiara nachádza. Preto, ak sme nijako špeciálne nezvolili tvar krivky, tak na vyjadrenie hodnôt mimo bodov siete je potrebné použiť interpoláciu zo známych hodnôt bodov siete.

Samozrejme, v porovnaní s Lagrangeovským prístupom, ide v Eulerovskom prístupe o výpočtovo náročnejšiu metódu, pretože výpočet prebieha na celej sieti, nielen „na našej krivke“. Ináč povedané - na sledovanie jednodimenzionálneho objektu potrebujeme mať definovanú funkciu v priestore $\Re^2$. Túto nevýhodu v našej práci eliminujeme pomocou lokálneho zjemňovania siete. V nasledujúcej časti odvodíme predpis, podľa ktorého sa bude level set



Obr. 1.3: Dve expandujúce krivky. Level set funkcia sa v tomto prípade pohybuje nadol. Level set metóda si ľahko poradí s problémovými situáciami popísanými v časti 1.2.

funkcia pohybovať.

1.4 Advektívna level set rovnica

Ak $\mathbf{x}(t) = [x(t), y(t)]$ je bod krivky, teda nulovej izočiary, tak aby level set funkcia popisovala pohyb tejto krivky v čase, požadujeme aby $\phi(\mathbf{x}(t), t) = 0$ v každom čase $t > 0$. Pri implicitne vyjadrenej krivke derivujeme funkciu $\phi(\mathbf{x}(t), t)$ podľa času a dostávame:

$$\frac{d\phi(\mathbf{x}(t), t)}{dt} = \frac{d\phi(x(t), y(t), t)}{dt} = \frac{\partial \phi}{\partial t} + \frac{\partial \phi}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial \phi}{\partial y}\frac{\partial y}{\partial t} = 0. \tag{1.6}$$

Rovnicu (1.6) môžeme prepísať na tvar

$$\frac{\partial \phi(\mathbf{x}(t),t)}{\partial t} + \frac{d\mathbf{x}(t)}{dt} \cdot \nabla \phi(\mathbf{x}(t),t) = 0.$$

Ak je pohyb krivky popísaný pomocou danej funkcie $\mathbf{V}$ musí platiť podobne ako v (1.1), že $\frac{d\mathbf{x}(t)}{dt} = \mathbf{V}(\mathbf{x}(t),t)$:

$$\frac{\partial \phi(\mathbf{x}(t),t)}{\partial t} + \mathbf{V}(\mathbf{x}(t),t) \cdot \nabla \phi(\mathbf{x}(t),t) = 0, \tag{1.7}$$

Rovnica (1.7) sa nazýva *advektívna level set rovnica* a jej riešenie je našou hlavnou motiváciou, preto si ju podrobnejšie popíšeme v ďalšej kapitole.

1.5 Znamienková funkcia vzdialenosti

Na implicitné zachytenie krivky existuje nekonečne veľa level set funkcií. Štandardne sa volí tzv. znamienková funkcia vzdialenosti.

Najprv si zadefinujeme *funkciu vzdialenosti* (distance function) $d(\mathbf{x})$ následovne:

$$d(\mathbf{x}) = \min_{\mathbf{x}_\Gamma \in \Gamma} |\mathbf{x} - \mathbf{x}_\Gamma|, \tag{1.8}$$

kde Γ je krivka a teda funkcia $d(\mathbf{x})$ vracia hodnotu minimálnej vzdialenosti bodu $\mathbf{x}$ od bodu $\mathbf{x}_\Gamma \in \Gamma$ a pre $\mathbf{x} \in \Gamma$ je rovná 0.

Znamienková funkcia vzdialenosti (signed distance function) je taká funkcia Φ, pre ktorú platí:

$$\Phi(\mathbf{x}) = \begin{cases} d(\mathbf{x}) & \mathbf{x} \in \Omega^+ \\ -d(\mathbf{x}) & \mathbf{x} \in \Omega^- \\ 0 & \mathbf{x} \in \Gamma \end{cases} \tag{1.9}$$

Hodnoty funkcie $\Phi(\mathbf{x})$ môžeme nájsť ako riešenie tzv. eikonalovej rovnice [6]:

$$|\nabla \Phi(\mathbf{x})| = 1 \tag{1.10}$$

s príslušnými okrajovými podmienkami.

Poznámka 1. *Funkcia 1.3, ktorú budeme používať na popísanie počiatočného tvaru požiaru, je znamienková funkcia vzdialenosti.*

Kapitola 2

Matematické modely

V nasledujúcej kapitole postupne odvodíme matematický model šírenia sa pozemného požiaru. V prvej podkapitole sa venujeme všeobecne rýchlostnému poľu pre rovnicu advekcie. V ďalšej podkapitole si povieme niečo o špeciálnom tvare rýchlostného poľa pre pohyb v smere normály a pre pohyb v závislosti od krivosti. Nakoniec tieto dva tvary použijeme na konštrukciu výsledného poľa, v ktorom sa bude pohybovať naša krivka popisujúca hranicu požiaru.

2.1 Pohyb v externom rýchlostnom poli

Ako sme už naznačili v časti 1.4, funkciu $\phi(\mathbf{x}, t)$ nebudeme používať len na implicitné vyjadrenie krivky/hranice požiaru, ale aj na jej pohyb, ktorý je riadený rovnicou advekcie (1.7). Kedže pri Eulerovskom prístupe riešime úlohu na oblasti Ω, tak aj rýchlostné pole $\mathbf{V}$ v rovnici (1.7) musíme mať dané na celej oblasti. Jedným z hlavných faktorov udávajúcich veľkosť a smer $\mathbf{V}$ v našej aplikácii bude vietor, ktorý budeme popisovať konštantným vektorom $\mathbf{q} \in \Re^2$.

Podrobnejšie sme sa modelovaním pohybu krivky v externom rýchlostnom poli zaoberali v práci [3].

2.2 Pohyb v smere normály

Rýchlostné pole nemusí byť vždy externe dané, ale môže byť závislé aj od riešenia rovnice. Je to práve ten prípad, ktorý sme už niekoľkokrát spomenuli v predchádzajúcich častiach, keď je pohyb krivky definovaný v smere svojej normály.

Kedže uvažujeme, že požiar sa môže len rozširovať, obmedzíme výber normály na vonkajšiu normálu ku krivke. V rovnici advekcie (1.7) zvolíme $\mathbf{V} = \alpha(x)\mathbf{N} = \alpha(x)\frac{\nabla\phi}{|\nabla\phi|}$, kde $\alpha(x)$ je ľubovoľná funkcia, pre našu aplikáciu má však zmysel uvažovať len $\alpha(x) \geq 0$. Po dosadení dostávame

$$\phi_t + \alpha(x)\frac{\nabla\phi}{|\nabla\phi|}\nabla\phi = 0. \tag{2.1}$$

Skombinujme teraz rýchlosť v smere normály a rýchlostné pole z predchádzajúcej časti, napríklad jednoduchým sčítaním. Výsledné pole dostaneme teda ako

$$\mathbf{V} = \alpha(x)\mathbf{N} + \mathbf{q} \tag{2.2}$$

Aby nenastal prípad keď je rýchlosť v smere normály záporná, musíme zaviesť podmienku $\alpha(x) + \mathbf{N} \cdot \mathbf{q} \geq 0$. Takto sme získali jednoduchý model rýchlostného poľa, ktorý bol použitý na modelovanie šírenia sa požiaru aj v [4].

Experimenty však ukázali, že tvar rozširujúcej sa hranice požiaru podľa tohto jednoduchého modelu príliš nezodpovedá reálnej situácii. Preto pre vplyv vetra na šírenie sa požiaru použijeme model navrhnutý v [9].

$$\mathbf{V} = V\mathbf{N} \tag{2.3}$$

$$V = e^{\lambda\mathbf{N}\cdot\mathbf{q}} f(\mathbf{x}) \tag{2.4}$$

kde $f(\mathbf{x})$ je člen udávajúci horľavosť materiálu.

2.3 Pohyb riadený krivosťou

V predchádzajúcej časti sme zaviedli rýchlostné pole, ktoré bolo generované samotnou level set funkciou, konkrétne jej gradientom, pomocou ktorého bola definovaná normála ku krivke. V mnohých aplikáciach sa ukazuje výhodné zaviesť aj závislosť rýchlosti od krivosti krivky κ, ktorá bola definovaná v (1.5).

Uvažujme teraz krivku, ktorá sa pohybuje v smere svojej vonkajšej normály, pričom veľkosť rýchlosti na krivke v nejakom bode je daná funkciou $\alpha = \alpha(\kappa(\mathbf{x}))$, kde $\kappa(\mathbf{x})$ je stredná krivosť v danom bode.

Zvolíme $\alpha(\kappa(\mathbf{x})) := \delta\kappa(\mathbf{x})$, kde δ je konštanta a teda máme:

$$\mathbf{V} = \delta\kappa\mathbf{N}. \tag{2.5}$$

Po dosadení do rovnice 1.7 dostávame

$$\phi_t - \delta\left(\nabla \cdot \frac{\nabla\phi}{|\nabla\phi|}\right)|\nabla\phi| = 0. \tag{2.6}$$

Krivosť bude v našom modele ovplyvňovať výsledný tvar pohybujúcej sa hranice požiaru. Z numerického hľadiska však krivosť zároveň pôsobí ako regularizačný faktor, keďže zhladzuje riešenie, pre detaily viď [6].

2.4 Model pohybu hranice pozemného požiaru

V tejto časti využijeme predchádzajúce úvahy na definovanie matematického modelu šírenia sa požiaru vo všeobecnom tvare.

Výsledné rýchlostné vektorové $\mathbf{V}$ pole v (1.7) zvolíme v našom prípade ako kombináciu rýchlosti v smere normály a rýchlosti závislej od krivosti v tvare [8]

$$\mathbf{V} = \beta\mathbf{N} \tag{2.7}$$

$$\beta = V(1 - \delta\kappa) \tag{2.8}$$

$$V = e^{\lambda\mathbf{N}\cdot\mathbf{q}} f(\mathbf{x}), \tag{2.9}$$

kde V je člen popisujúci rýchlosť šírenia sa požiaru v závislosti od smeru a sily vetra $\mathbf{q}$ a kvalite paliva $f(\mathbf{x})$, pozri aj (2.3).

Za predpokladu, že $\mathbf{N} = \frac{\nabla\phi}{|\nabla\phi|}$ a krivosť $\kappa = \nabla \cdot \frac{\nabla\phi}{|\nabla\phi|}$ môžeme rovnicu (1.7) prepísať na tvar

$$\frac{\partial\phi}{\partial t} + \left(V\frac{\nabla\phi}{|\nabla\phi|} - V\delta\nabla \cdot \left(\frac{\nabla\phi}{|\nabla\phi|}\right)\frac{\nabla\phi}{|\nabla\phi|}\right) \cdot \nabla\phi = 0.$$

Po úprave máme

$$\frac{\partial\phi}{\partial t} + V\frac{\nabla\phi}{|\nabla\phi|} \cdot \nabla\phi - V\delta|\nabla\phi|\nabla \cdot \left(\frac{\nabla\phi}{|\nabla\phi|}\right) = 0. \tag{2.10}$$

Rovnica (2.10) je *adkvečno-difúzna rovnica* a je to zároveň náš finálny model, ktorý sme použili na modelovanie pohybu hranice pozemného požiaru. Na jej numerické riešenie advekčnej časti použijeme metódu konečných objemov druhého rádu, explicitnú v čase použitú aj v [3], kde bola demonštrovaná aj jej presnosť a rád. Na difúznu časť použijeme semi-implicitnú diskretizáciu v čase a metódu konečných objemov podobne ako v [2].

Kapitola 3

Numerické metódy

V tejto kapitole popisujeme numerické riešenie nášho matematického modelu. V prvej podkapitole odvodíme riešenie pre advekčnú časť a v druhej podkapitole pre difúznu časť. V poslednej podkapitole obe výsledné schémy spojíme do jedného systému rovníc.

Poznámka 2. *V ďalšom budeme používať výraz „uzol siete“ aj „vrchol mriežky/elementu“, ktoré ale budú označovať tú istú entitu siete.*

3.1 Riešenie rovnice advekcie

V nasledujúcom popíšeme numerickú metódu druhého rádu na riešenie nelineárnej, hyperbolickej, parciálnej diferencálnej rovnice, ktorú dostaneme z rovnice (2.10) vynechaním člena krivosti z rýchlostného poľa (2.4):

$$\frac{\partial \phi}{\partial t} + \mathbf{V} \cdot \nabla \phi = 0 \quad \text{na} \quad \Omega \times \tau \tag{3.1}$$

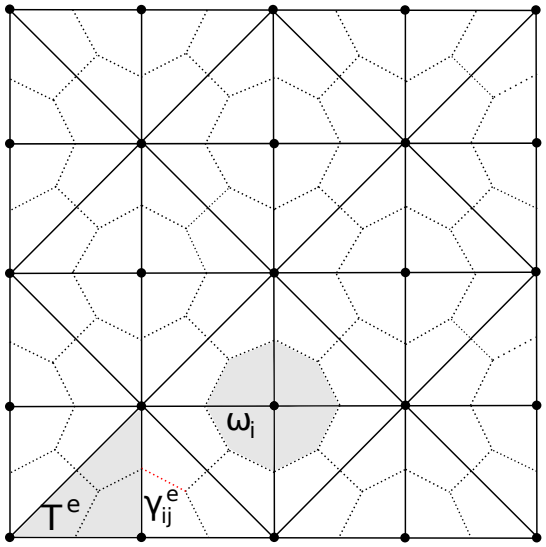
kde $\Omega \subset \Re$ je výpočtová oblasť, $\tau = (0, t^{end})$ je časový interval, $\phi : \Omega \times \tau \to \Re$ je neznáma funkcia a $\mathbf{V} : \Omega \times \tau \to \Re^2$ je rýchlostné pole závislé od riešenia. Ako počiatočnú podmienku definujeme

$$\phi(\mathbf{x}, 0) = \phi_0(\mathbf{x}) \quad \mathbf{x} \in \Omega \tag{3.2}$$

Výpočet okrajových podmienok upresníme neskôr.
Rovnicu (3.1) prepíšeme na konzervatívny tvar s pravou stranou

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\mathbf{V}\phi) = \phi \nabla \cdot \mathbf{V}. \tag{3.3}$$

Ďalej postupujeme podľa [1, 4]. Spojitú diferenciálnu rovnicu (3.3) aproximujeme diskrétnymi rovnicami pomocou metódy konečných objemov. Výpočtová sieť pozostáva z trojuholníkových elementov T^e, viď obrázok 3.1. Časový interval τ je diskretizovaný na intervaly $0 = t^0 < t^1 \cdots < t^N = t^{end}$. Ku každému uzlu $\mathbf{x}_i$, $i = 1, 2, \ldots, I$ primárnej siete skonštruujeme konečný objem ω_i tak, že uzol $\mathbf{x}_i$ bude ťažiskom konečného objemu ω_i. Všetky konečné objemy vytvoria dokopy tzv. duálnu sieť k primárnej sieti tvorenej elementami T^e. Hranica $\partial\omega_i$ konečného objemu ω_i je tvorená segmentami $\gamma_{ij}^e = T^e \cap \omega_i \cap \omega_j$, alebo $\gamma_{ij}^e = T^e \cap \partial\omega_i \cap \partial\Omega$ na hranici oblasti Ω.



Obr. 3.1: Výpočtová sieť s trojuholníkovými elementmi T^e a zostrojenou duálnou mriežkou. Každému vrcholu je priradený konečný objem ω_i. Segment hranice γ_{ij}^e kontrolného objemu ω_i zostrojíme spojením stredu hrany danej vrcholmi $\mathbf{x}_i$ a $\mathbf{x}_j$ s ťažiskom elementu.

Rovnicu (3.3) prepíšeme do integrálneho tvaru, t.j. integrujeme cez každý konečný objem ω_i siete a časový interval (t^n, t^{n+1}):

$$\int_{\omega_i}\int_{t^n}^{t^{n+1}} \frac{\partial \phi}{\partial t}\, dt\, dx + \int_{\omega_i}\int_{t^n}^{t^{n+1}} \nabla\cdot(\mathbf{V}\phi)\, dt\, dx = \int_{\omega_i}\int_{t^n}^{t^{n+1}} \phi\nabla\cdot\mathbf{V}\, dt\, dx. \tag{3.4}$$

Po čiastočnej integrácii dostávame

$$\int_{\omega_i} \phi(x,t^{n+1})\, dx - \int_{\omega_i} \phi(x,t^n)\, dx + \int_{t^n}^{t^{n+1}}\int_{\partial\omega_i} \phi\mathbf{n}\cdot\mathbf{V}\, ds\, dt = \int_{\partial\omega_i}\int_{t^n}^{t^{n+1}} \phi\nabla\cdot\mathbf{V}\, dt\, dx \tag{3.5}$$

Ďalej uvažujeme $\phi(\mathbf{x}_i, t^n) = \phi_i^n$. Rovnicu (3.3) aproximujeme diskrétnou rovnicou pre hodnoty ϕ:

$$\begin{aligned}\phi_i^{n+1}|\omega_i| - \phi_i^n|\omega_i| + \triangle t^n \sum_{j\in\Lambda_i}\sum_{e\in\Lambda_{ij}} |\gamma_{ij}^e|\tilde{\phi}_{ij}^{n+1/2,e}\,\mathbf{n}_{ij}^e\cdot\mathbf{V}_{ij}^e = \\ = \triangle t^n|\omega_i|\phi_i^{n+1/2,e}\sum_{j\in\Lambda_i}\sum_{e\in\Lambda_{ij}} |\gamma_{ij}^e|\mathbf{n}_{ij}^e\cdot\mathbf{V}_{ij}^e,\end{aligned} \tag{3.6}$$

kde $\triangle t^n = t^{n+1} - t^n$, $\mathbf{V}_{ij}^e = \mathbf{V}(\mathbf{x}_{ij}^e)$ je rýchlosť na segmente γ_{ij}^e v jeho strednom bode $\mathbf{x}_{ij}$, $\mathbf{n}_{ij}^e$ je jednotkový normálový vektor segmentu γ_{ij}^e orientovaný vždy z ω_i do ω_j. $|\gamma_{ij}^e|$ je veľkosť segmentu γ_{ij}^e. Λ_i je množina indexov konečných objemov ω_i susediacich s ω_j a Λ^{ij} je množina indexov e označujúcich všetky elementy T^e obsahujúce vrcholy $\mathbf{x}_i$ a $\mathbf{x}_j$. Hodnota $\tilde{\phi}_{ij}^{n+1/2,e}$ je definovaná pomocou upwind prinípu [1]

$$\tilde{\phi}_{ij}^{n+1/2,e} = \begin{cases} \phi_{ij}^{n+1/2,e} & ak \quad \mathbf{n}_{ij}^e\cdot\mathbf{V}_{ij}^e \geq 0 \\ \phi_{ji}^{n+1/2,e} & ak \quad \mathbf{n}_{ij}^e\cdot\mathbf{V}_{ij}^e < 0 \end{cases} \tag{3.7}$$

Pri metóde druhého rádu definujeme [1]

$$\begin{aligned}\phi_{ij}^{n+1/2,e} &= \phi_i^n + \nabla\phi_i^n\cdot(\mathbf{x}_{ij}^e - \mathbf{x}_i) - \tfrac{1}{2}\triangle t^n\nabla\phi_i^n\cdot\mathbf{V}_i \\ \phi_{ji}^{n+1/2,e} &= \phi_j^n + \nabla\phi_j^n\cdot(\mathbf{x}_{ij}^e - \mathbf{x}_j) - \tfrac{1}{2}\triangle t^n\nabla\phi_j^n\cdot\mathbf{V}_j\end{aligned} \tag{3.8}$$

kde $\mathbf{x}_i$ sú súradnice uzla siete. Pre toky cez hranicu oblasti Ω definujeme:

$$\phi_{ib}^{n+1/2} = \phi_i^n + \nabla\phi_i^n \cdot (\mathbf{x}_{ib} - \mathbf{x}_i) - \frac{\triangle t^n}{2}\nabla\phi_i^n \cdot \mathbf{V}_i \tag{3.9}$$

kde $\mathbf{x}_{ib}$ sú súradnice stredu segmentu γ_{ij}^e kontrolného objemu ω_i susediaceho s hranicou oblasti Ω.

S ohľadom na (3.7) môžeme (3.6) prepísať na výslednú schému

$$\phi_i^{n+1} = \phi_i^n - \frac{\triangle t^n}{|\omega_i|}\sum_{j\in\Lambda_i}\sum_{e\in\Lambda_{ij}} |\gamma_{ij}^e|\mathbf{n}_{ij}^e \cdot \mathbf{V}_{ij}^e(\tilde{\phi}_{ij}^{n+1/2,e} - \phi_{ij}^{n+1/2,e}) \tag{3.10}$$

Výsledná numerická schéma (3.10) je explicitná v čase a je stabilná len ak

$$\triangle t^n \leq \min_i \left(\sum_{j\in\Lambda_i}\sum_{e\in\Lambda_{ij}} \frac{|\gamma_{ij}^e|}{|\omega_i|}\max(0, \mathbf{n}_{ij}^e \cdot \mathbf{V}_{ij}^e)\right)^{-1}. \tag{3.11}$$

Hodnoty neznámej funkcie ϕ, ako aj hodnoty rýchlosti $\mathbf{V}$ vnútri konečných objemov budeme počítať pomocou lineárnej interpolácie hodnôt vo vrcholoch elementu:

$$\hat{\phi}(\mathbf{x}, t^n) = \sum_i \phi_i^n N_i(\mathbf{x}) \quad \mathbf{x} \in T^e \tag{3.12}$$

kde N_k sú po častiach lineárne funkcie také, že platí $N_i(\mathbf{x}_j) = \delta_{ij}$. Index i prebieha cez indexy všetkých vrcholov elementu T^e.

Numerický gradient $\nabla\phi_i^n$ vypočítame ako priemer gradientov na elementoch obsahujúcich vrchol $\mathbf{x}_i$:

$$\nabla\phi_i^n = \frac{1}{|\omega_i|}\sum_e |T^e \cap \omega_i|\nabla^e\phi_i^n. \tag{3.13}$$

Index e prebieha cez všetky elementy T^e, ktoré obsahujú uzol $\mathbf{x}_i$. ∇^e je aproximácia gradientu na T^e.

3.2 Riešenie krivostnej časti rovnice

V ďalšom budeme pre jednoduchosť, uvažovať rovnicu (2.10) s $V = 1, \delta = 1$ a len s difúznym členom v tvare:

$$\partial_t \phi - |\nabla \phi| \nabla \cdot \frac{\nabla \phi}{|\nabla \phi|} = 0. \tag{3.14}$$

Rovnicu budeme riešiť pomocou numerickej metódy konečných objemov pričom použijeme semi-implicitnú schému v čase [2].
Označenie oblasti, elementov, uzlov siete a hodnôt funckie v uzle siete zostávajú rovnaké ako v predošlej časti. Na vyjadrenie hodnôt funkcie $\phi(\mathbf{x}, t)$ mimo uzlov siete použijeme opäť vzťah (3.12) a gradient $\nabla \hat{\phi}^n(\mathbf{x})$ môžeme potom vyjadriť v tvare:

$$\nabla \hat{\phi}^n(\mathbf{x}) = \sum_{k \in \Lambda_e} \phi_k^n \nabla N_k(\mathbf{x}), \tag{3.15}$$

kde Λ_e je množina indexov vrcholov prislúchajúcich danému elementu.

Opäť skonštruujeme duálnu sieť k primárnej tak, že uzly $\mathbf{x}_i$ primárnej siete budú ťažiská elementov ω_i duálnej siete. Po integrácii v priestore pre $\mathbf{x} \in \omega_i$ a v čase pre $t \in (t^n, t^{n+1})$ dostávame dostávame z (3.14)

$$\phi_i^{n+1} |\omega_i| = \phi_i^n |\omega_i| - |\nabla \phi_i^n| \int_{t^n}^{t^{n+1}} \int_{\partial \omega_i} \frac{\mathbf{n} \cdot \nabla \phi}{|\nabla \phi|} d\gamma dt. \tag{3.16}$$

Integráciu pozdĺž hranice $\partial \omega_i$ nahradíme sumáciou cez segmenty $\gamma_{ij} = T^e \cap \omega_i \cap \omega_j$. Stred segmentu γ_{ij} je bod $\mathbf{x}_{ij}$. Pre gradienty v bodoch $\mathbf{x}_{ij}$ zavedieme notáciu

$$\nabla^e \phi_{ij}^n = \nabla \hat{\phi}^n(\mathbf{x}_{ij}^e). \tag{3.17}$$

Veľkosť gradientu vo vrchole $\mathbf{x}_i$ vyjadríme s použitím vzťahu (3.13) ako

$$|\nabla \phi_i^n| = \frac{1}{|\omega_i|} \sum_{j \in \Lambda_i} \sum_{e \in \Lambda^{ij}} \frac{|T^e \cap \omega_i|}{2} |\nabla^e \phi_{ij}^n|. \tag{3.18}$$

S použitím gradientov (3.17) (3.18) dostávame výslednú schému pre rovnicu

(3.14)

$$\phi_i^{n+1} + \frac{\triangle t^n}{|\omega_i|}|\nabla\phi_i^n| \sum_{j\in\Lambda_i}\sum_{e\in\Lambda^{ij}} |\gamma_{ij}^e| \frac{\mathbf{n}_{ij}^e \cdot \nabla^e \phi_{ij}^{n+1}}{|\nabla^e \phi_{ij}^n|} = \phi_i^n. \tag{3.19}$$

3.3 Výsledný systém rovníc

Schéma (3.19) predstavuje systém rovníc

$$(\mathbb{I} + \mathbb{A})\mathbf{x} = \mathbf{b}. \tag{3.20}$$

Prvky matice $\mathbb{A}$ majú tvar:

$$a_{ik} = \frac{\triangle t^n}{|\omega_i|}|\nabla\phi_i^n| \sum_{j\in\Lambda_i}\sum_{e\in\Lambda^{ij}} |\gamma_{ij}| \frac{\mathbf{n}_{ij} \cdot \nabla N_k(\mathbf{x}_{ij})}{|\nabla \phi_{ij}^n|}. \tag{3.21}$$

Prvky vektora pravej strany $\mathbf{b}$ sa rovnajú pravej strane v schéme 3.10.

Kapitola 4

DUNE

V nasledujúcej kapitole sa venujeme opisu základných charakteristík softvéru Dune. *Uvádzame aj ilustračný príklad vytvorený s použitím modulu dune-grid, ktorý poskytuje všetky potrebné nástroje pri práci s numerickými technikami založenými na výpočtovej sieti.*

4.1 O DUNE

Dune (*Distributed and Unified Numerics Environment*) je softvérová knižnica napísaná v programovacom jazyku C++ na numerické riešenie parciálnych diferenciálnych rovníc. Podporuje implementáciu metód ako Metóda konečných prvkov (MKP), Metóda konečných objemov (MKO), alebo Metóda konečných diferencií (MKD). Dune je voľne dostupná pod licenciou GPL. Je možné aj použitie s iným komerčným softvérom.

Knižnica Dune je rozdelená do samostatných modulov, pre ktoré sa dokumentácia nachádza na oficiálnych stránkach. Pre aktuálnu verziu 2.0 (z 19. mája 2011) sú dostupné moduly stiahnuteľné ako samostatné balíčky:

- `dune-common` obsahuje základné triedy používané všetkými ostatnými Dune modulmi.
- `dune-grid` je najrozvinutejší modul, ktorý je použitý v tejto práci. Definuje nekonformné, hierarchicky vnorené, paralelné mriežky a mriežky

s rôznym typom elementov v priestore s ľubovoľným rozmerom. Podporuje grafický výstup napríklad vo formáte VTK. Možný je aj import z iných generátorov mriežok v rôznych formátoch (Triangle, Tetgen, Gmsh, Amira).

- `dune-istl` - *Iterative Solver Template Library* poskytuje základné triedy riedkych matíc a vektorov a metódy na ich riešenie. V práci sme použili BiCGStab Solver na riešenie systému rovníc, ku ktorému sa dopracujeme pri riešení krivostnej časti rovnice.

- `dune-fem` - modul, ktorý definuje rozhranie pre implementáciu metód ako Metóda konečných prvkov a metóda konečných objemov a nespojité galerkinovské metódy.

- `dune-grid-howto` - manuál [5] s ukážkovou implementáciou metódy konečných objemov na riešenie rovnice advekcie, Poissonovej rovnice a paralelnej verzie metódy konečných objemov

DUNE umožňuje prácu s viacerými typmi mriežok. V práci sme použili konformnú, trojuholníkovú mriežku ALUGrid [1].

Podrobnosti o potrebnom softvérovom vybavení a inštalácii modulov DUNE nájde záujemca na oficiálnej stránke [2]. Je možné vytvorenie aj vlastných modulov, alebo projektov [3]. Popis konceptov implementovaných v module `dune-grid`, ako aj základných tried sa nachádza v [5] a v kapitole 2 v [3].

Pre názornosť uvedieme nami vytvorenú krátku ukážku, ktorá zahŕňa vytvorenie siete a iterovanie cez elementy siete a hrany každého elementu. Program vypíše súradnice ťažiska elementu, veľkosť elementu a následne v cykle cez hrany elementu vypíše súradnice koncových bodov hrany. Na ukážku práce s indexmi program vypočíta aj veľkosti elementov duálnej siete. Kód bude fungovať s rôznymi typmi sietí. Typ siete si zvolí užívateľ pred skompilovaním programu a zadaním napr. `make GRIDTYPE=ALUGRID_SIMPLEX`. Nastavením hodnoty `GRIDDIM` si užívateľ vyberie dimenziu siete. Pre nasledujúci príklad sme vybrali dvojdimenzionálnu, štvorčekovú sieť `YASPGRID`.

[1] `http://www.mathematik.uni-freiburg.de/IAM/Research/alugrid/`
[2] `http://www.dune-project.org/doc/installation-notes.html`
[3] `http://www.dune-project.org/doc/installation-notes.html`

```
1  #ifdef HAVE_CONFIG_H
2  #include "config.h"
3  #endif
4  #include <iostream>
5  #include "dune/common/exceptions.hh"
6  #include <dune/grid/io/file/dgfparser/dgfgridtype.hh>
7
8  template< class G >
9  void gridTraverse( G& grid )
10 {
11     // dimenzia referenceho elementu
12     const int dim = G::dimension;
13
14     // dimenzia siete
15     const int dimworld = G::dimensionworld;
16
17     // typ suradnic siete (double)
18     typedef typename G::ctype ct;
19     typedef Dune::FieldVector< ct, dimworld > GlobalCoordinateType;
20
21     // typ pohladu na siet
22     typedef typename G::LeafGridView GridView;
23
24     // typ iteratora cez elementy
25     typedef typename GridView::template Codim< 0 >::Iterator
26         LeafIterator;
27
28     // typ geometrie elementu
29     typedef typename LeafIterator::Entity::Geometry LeafGeometry;
30
31     // typ iteratora cez hrany elementu
32     typedef typename GridView::IntersectionIterator
33         IntersectionIterator;
34
35     // typ geometrie hrany elementu
36     typedef typename IntersectionIterator::Intersection::Geometry
37         IntersectionGeometry;
38
39     // typ mnoziny indexov
40     typedef typename GridView::IndexSet LeafIndexSet;
41
42     // ziskanie pohladu na siet
43     GridView gridView = grid.leafView();
44
45     // ziskanie referencie na mnozinu indexov
46     const LeafIndexSet& set = gridView.indexSet();
47
48     std::vector< double > area( gridView.size( dim ) );
49     for ( int i = 0; i < area.size(); i++ ) area[i] = 0.0;
```

```
50
51      std::cout << "␣Siet␣sa␣sklada␣z␣"
52          << gridView.size( 0 ) << "␣elementov,␣"
53          << gridView.size( dim-1 ) << "␣hran␣a␣"
54          << gridView.size( dim ) << "␣vrcholov.␣"
55          << std::endl;
56
57      // cyklus cez elementy siete
58      LeafIterator endit = gridView.template end< 0 >();
59      for ( LeafIterator it = gridView.template begin< 0 >();
60              it!=endit; ++it )
61      {
62          // ziskanie referencie na geometriu elementu
63          const LeafGeometry &geo = it->geometry();
64
65          // velkost elementu
66          double volume = geo.volume();
67
68          // pocet vrcholov elementu
69          int vertexSize = geo.corners();
70
71          // tazisko elementu
72          Dune::FieldVector< ct, dimworld > ec = geo.center();
73
74          std::cout << std::endl << "Element␣s␣taziskom␣" << ec
75              << "␣ma␣velkost␣" << volume
76              << "␣a␣" << vertexSize
77              << "␣vrcholy␣" << std::endl;
78
79          // ziskanie geometrie referencneho elementu
80          Dune::GeometryType gt = it->type();
81
82          // ziskanie referencie na referencny element
83          const Dune::GenericReferenceElement< ct, dim > &ref =
84              Dune::GenericReferenceElements< ct, dim >::general(gt);
85
86          // cyklus cez hrany elementu
87          IntersectionIterator isend = gridView.iend( *it );
88          for ( IntersectionIterator is = gridView.ibegin( *it );
89                  is!=isend; ++is )
90          {
91              // ziskanie referencie na geometriu hrany
92              const IntersectionGeometry &geof = is->geometry();
93
94              // suradnice koncovych vrcholov hrany
95              GlobalCoordinateType svg = geof.corner( 0 );
96              GlobalCoordinateType evg = geof.corner( 1 );
97
98              // suradnice stredu hrany
```

```
 99              GlobalCoordinateType edc = geof.center();
100
101              std::cout << "␣Prechadzam␣hranu␣so␣stredom␣" << edc <<
102                  "␣a␣koncovymi␣bodmi␣" << svg << "␣a␣" << evg
103                  << std::endl;
104
105              // ziskanie pociatocneho a koncoveho bodu hrany
106              // v ref. elemente
107              int startVertex =
108                  ref.subEntity( is->indexInInside(), dim-1, 0, dim );
109
110              int endVertex =
111                  ref.subEntity( is->indexInInside(), dim-1, 1, dim );
112
113              // indexy koncovych vrcholov hrany
114              int indexi = set.subIndex( *it, startVertex, dim );
115              int indexj = set.subIndex( *it, endVertex, dim );
116
117              area[indexi] += 0.5 * volume / vertexSize;
118              area[indexj] += 0.5 * volume / vertexSize;
119          }
120      }
121
122      std::cout << std::endl;
123      for ( int i = 0; i < area.size(); i++ )
124          std::cout << "␣Konecny␣objem␣" << i
125              << "␣ma␣velkost␣" << area[i] << std::endl;
126  }
127
128  int main( int argc, char** argv )
129  {
130      try
131      {
132          // typ siete
133          typedef Dune::GridSelector::GridType GridType;
134
135          // vytvorenie siete z .dgf suboru
136          Dune::GridPtr< GridType > gridPtr( "unitcube2.dgf" );
137
138          // ziskanie referencie na siet
139          GridType& grid = *gridPtr;
140
141          // jedenkrat globalne zjemnena siet
142          grid.globalRefine( 1 );
143
144          // funkcia na prechod siete
145          gridTraverse( grid );
146
147          return 0;
```

```
148       }
149
150       catch ( Dune::Exception &e )
151       {
152           std::cerr << "Dune reported error: " << e << std::endl;
153       }
154
155       catch (...)
156       {
157           std::cerr << "Unknown exception thrown!" << std::endl;
158       }
159   }
```

Hlavný program sa začína na riadku 127. V riadku 135 je vytvorená sieť zo súboru `unitcube2.dgf`, ktorý obsahuje definíciu makrosiete (v tomto prípade sa jedná a jednotkový štvorec). V riadku 141 je sieť jedenkrát globálne zjemnená, teda výsledok je 2×2 sieť. Vo funkcii `gridTraverse()`, po definovaní typov, je v riadku 43 získaný pohľad na koncovú (leaf) sieť. Iný typ pohľadu (`levelView()`) využijeme pri lokálnom zjemňovaní siete. Viď kapitola 5. V riadku 58 začína cyklus cez elementy siete a v riadku 87 cyklus cez hrany elementu. Výstup z programu vyzerá takto:

```
 1  Siet sa sklada z 4 elementov, 12 hran a 9 vrcholov.
 2
 3 Element s taziskom 0.25 0.25 ma velkost 0.25 a 4 vrcholy
 4  Prechadzam hranu so stredom 0 0.25 a koncovymi bodmi 0 0 a 0 0.5
 5  Prechadzam hranu so stredom 0.5 0.25 a koncovymi bodmi 0.5 0 a 0.5 0.5
 6  Prechadzam hranu so stredom 0.25 0 a koncovymi bodmi 0 0 a 0.5 0
 7  Prechadzam hranu so stredom 0.25 0.5 a koncovymi bodmi 0 0.5 a 0.5 0.5
 8
 9 Element s taziskom 0.75 0.25 ma velkost 0.25 a 4 vrcholy
10  Prechadzam hranu so stredom 0.5 0.25 a koncovymi bodmi 0.5 0 a 0.5 0.5
11  Prechadzam hranu so stredom 1 0.25 a koncovymi bodmi 1 0 a 1 0.5
12  Prechadzam hranu so stredom 0.75 0 a koncovymi bodmi 0.5 0 a 1 0
13  Prechadzam hranu so stredom 0.75 0.5 a koncovymi bodmi 0.5 0.5 a 1 0.5
14
15 Element s taziskom 0.25 0.75 ma velkost 0.25 a 4 vrcholy
16  Prechadzam hranu so stredom 0 0.75 a koncovymi bodmi 0 0.5 a 0 1
17  Prechadzam hranu so stredom 0.5 0.75 a koncovymi bodmi 0.5 0.5 a 0.5 1
18  Prechadzam hranu so stredom 0.25 0.5 a koncovymi bodmi 0 0.5 a 0.5 0.5
19  Prechadzam hranu so stredom 0.25 1 a koncovymi bodmi 0 1 a 0.5 1
20
21 Element s taziskom 0.75 0.75 ma velkost 0.25 a 4 vrcholy
22  Prechadzam hranu so stredom 0.5 0.75 a koncovymi bodmi 0.5 0.5 a 0.5 1
23  Prechadzam hranu so stredom 1 0.75 a koncovymi bodmi 1 0.5 a 1 1
```

```
24  Prechadzam hranu so stredom 0.75 0.5 a koncovymi bodmi 0.5 0.5 a 1 0.5
25  Prechadzam hranu so stredom 0.75 1 a koncovymi bodmi 0.5 1 a 1 1
26
27  Konecny objem 0 ma velkost 0.0625
28  Konecny objem 1 ma velkost 0.125
29  Konecny objem 2 ma velkost 0.0625
30  Konecny objem 3 ma velkost 0.125
31  Konecny objem 4 ma velkost 0.25
32  Konecny objem 5 ma velkost 0.125
33  Konecny objem 6 ma velkost 0.0625
34  Konecny objem 7 ma velkost 0.125
35  Konecny objem 8 ma velkost 0.0625
```

Kapitola 5

Adaptivita

V tejto kapitole popíšeme najprv význam adaptivity pre numerické riešenie. V ďalšej časti načrtneme koncept lokálneho zjemňovania v DUNE. *Nakoniec vysvetlíme, ako sme myšlienku adaptivity integrovali do našej aplikácie a samotný algoritmus.*

5.1 Význam adaptivity

Adaptívna sieť je sieť zjemnená, resp. zhrubená len na niektorých miestach - teda lokálne, na základe nejakého kritéria.

Poznámka 3. *Zavedieme označenie S_g^l, $l \geq g$, ktoré bude označovať sieť, ktorá je zjemnená g-krát globálne a $(l-g)$-krát lokálne. Ak bude sieť zjemnená len g-krát globálne, označíme ju S_g.*

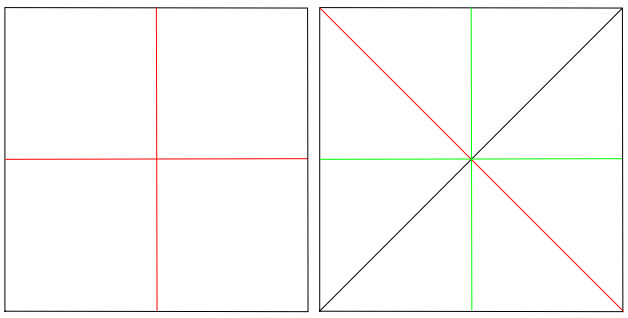
Hlavný faktor, ktorý vplýva na presnosť numerického riešenia je stupeň zjemnenia (jemnosť) výpočtovej siete. So zvyšujúcou sa jemnosťou siete a teda vyšším počtom elementov narastá aj výpočtový čas. (viď napr. tabuľka 6.1, alebo obrázok 6.10). Sú však prípady, keď je pre nás dôležitá len určitá oblasť/oblasti na sieti, kde chceme, aby bol výpočet presnejší ako na iných miestach. Typickým príkladom sú miesta, kde je gradient neznámej funkcie príliš vysoký a chceme ho dostatočne presne zachytiť, kým na iných miestach je gradient napr. veľmi „plochý“. Podobne sa lokálne zjemnenie môže použiť pre presnejšiu geometrickú aproximáciu v miestach s komplikovaným tvarom

oblasti, ako sú napríklad ostré rohy, alebo naopak zaoblené hrany. Opakom zjemňovania siete je jej zhrubenie, ktoré má význam najmä pri časových úlohách ako je tá naša.

5.2 Adaptivita v DUNE

Skôr ako sa dostaneme k realizácii adaptivity v DUNE, vysvetlíme niekoľko pojmov súvisiacich s konceptom zjemňovania ako takého.

Nech S je uniformná, dvojdimenzionálna, štvorčeková sieť pozostávajúca z elementov T a ∂S je jej okraj o dĺžke L. Ak je okraj ∂S tvorený N hranami príslušných elementov, potom veľkosť jednej hrany elementu pozdĺž ∂S je $h = L/N$. Pod pojmom *jedenkrát zjemniť sieť*, či už globálne (t.j. všetky elementy siete), alebo lokálne, budeme rozumieť zjemniť sieť tak, aby pre zjemňované elementy platilo $h \to \frac{h}{2}$. Pre štvorčekovú sieť je to triviálne, avšak pre iné typy sietí, ako napríklad trojuholníková ALUGrid, ktorú sme použili na naše výpočty, nastáva malý rozdiel. Vysvetlenie je na obrázku 5.1 s popisom.

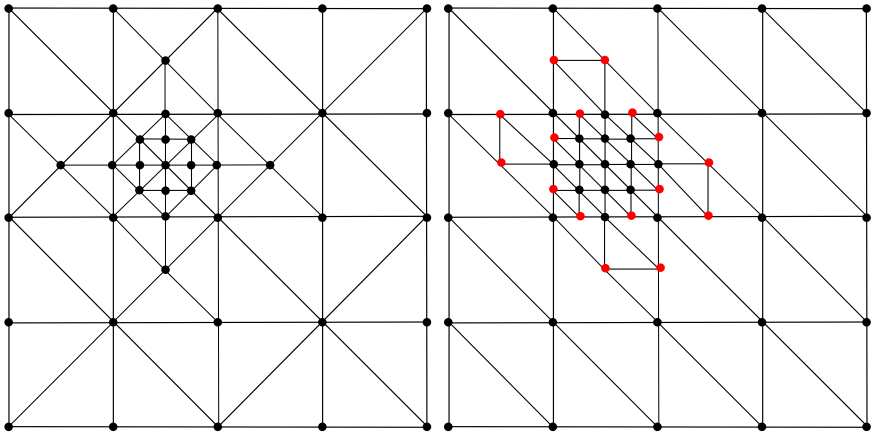


Obr. 5.1: Porovnanie zjemnenia štvorčekovej siete a trojuholníkovej siete ALUGrid. Počiatočná sieť je v tvare štvorca, v prípade ALUGrid tvorená dvoma čiernymi trojuholníkovým elementami. Pri štvorčekovej sieti jedenkrát globálne zjemniť znamená pridať červené hrany. Pri trojuholníkovej však globálne zjemnenie podľa definície znamená pridanie červenej a aj dvoch zelených hrán.

V DUNE je funkcia `refineStepsForHalf()`, ktorá pre každý typ siete vracia správnu hodnotu tak, aby výsledok zjemnenia zodpovedal definícii vyššie.

Ak teda chceme sieť zjemniť n-krát, DUNE v skutočnosti zjemní sieť $n *$ `refineStepsForHalf()`-*krát*. Takéto globálne zjemenie má význam hlavne pre prípady, keď pri numerických experimentoch sledujeme vývoj (chyby) riešenia vzhľadom na jemnosť siete. Pre nás je podstatné hlavne to, že pri vertex-centered metóde, tak pri jednom globálnom zjemnení ALUGrid, naozaj dostaneme dvojnásobný počet uzlov siete.

Užívateľ má na výber dva typy siete: ALUGRID_SIMPLEX a ALUGRID_CONFORM. V aplikácii sme zvolili druhú verziu, pretože pri nej nevznikajú lokálnym zjemňovaním tzv. *hanging nodes*, pre ktoré nie je možné zostrojiť konečný objem tak, ako to bolo popísané v kapitole 3, viď. obrázky 3.1 a 5.2 pre ilustráciu.

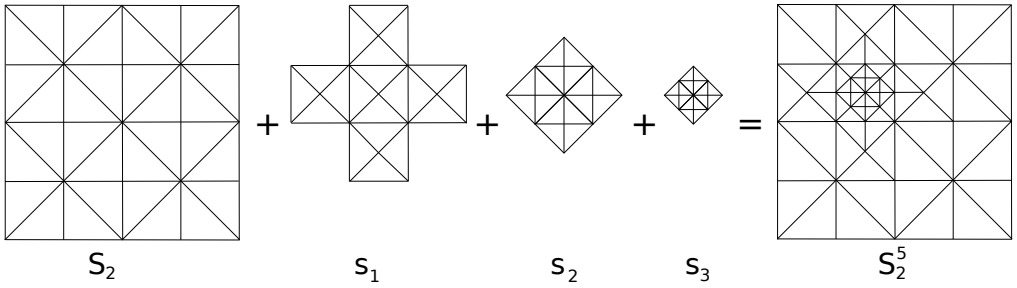


Obr. 5.2: Porovnanie stratégie zjemňovania pre ALUGrid_Conform (vľavo) a ALUGrid_Simplex. Červenou farbou sú znázornené tzv. hanging nodes.

Elementy, ktoré chceme zjemniť sa v DUNE musia označiť pomocou funkcie `mark()`. V DUNE nie je dovolené sieť nijako modifikovať. Teda nemôžeme ručne pridávať alebo odoberať elementy, vrcholy a pod. Preto pred zjemnením siete musíme zavolať funkciu `preAdapt()`, ktorá „prepne“ sieť do módu, kedy je dovolené ju modifikovať. Následne môžeme zavolať funkciu `adapt()`, ktorá prevedie zjemnenie a nakoniec funkciu `postAdapt()`, ktorá vymaže označenia elementov.

Ďalej vysvetlíme niektoré dôležité pojmy, ktoré súvisia s lokálnym zjemňovaním siete v DUNE.

G-krát globálne a $(l-g)$-krát lokálne zjemnenú sieť S_g^l nazývame *koncová sieť* (leaf grid) a ju môžeme interpretovať aj ako zjednotenie siete S_g s vnorenými „podsieťami“ $s_i, i = 1, 2, \ldots, (l-g)$, ktoré vzniknú lokálnym zjemnením. Tieto vnorené siete sa v DUNE nazývajú *úrovne* (levely) siete S_g^l, viď obrázok 5.3. Každý element siete s_i nazývame *synom* elementu siete s_{i-1}, resp. siete $S_g \,\mathrm{ak}\, i = 1$, ktorého zjemnením vznikol. A naopak, každý element siete s_i, resp. S_g, nazývame *otcom* elementu siete s_{i+1}, ktorý vznikol zjemnením otca. Po jednotlivých úrovniach je možné iterovať pomocou iterátora `LevelIterator`.



Obr. 5.3: Úrovne lokálne zjemnenej siete. Výsledná sieť S_2^5 sa skladá z troch vnorených sietí. Z obrázku je vidieť, že zjemňovať ALUGrid v podstate znamená „sekať“ prepony elementov. Teda nový uzol sieti sa môže zjemňovaním vytvoriť len na prepone elementu. Naopak - zhrubiť sieť znamená „odobrať“ preponu elementu.

Pri lokálnom zjemňovaní siete nie je vhodné používať číslovanie entít siete pomocou postupnosti indexov, pretože modifikáciou siete sa táto postupnosť naruší a musí byť vygenerovaná nanovo. Ako uvidíme neskôr, po modifikácii siete potrebujeme nejako zistiť, ktoré uzly siete sú pôvodné a ktoré sú nové. Preto sú v DUNE zavedené *ID čísla* entít. ID číslo nie je nutne číslo v pravom slova zmysle, môže to byť aj skupina cifier jedinečná pre entitu siete pred, aj po modifikácii siete. ID potom vystupujú ako kľúče pre pole typu `std::map`. Pre takéto pole ukladáme hodnoty a aj kľúč, podľa ktorého potom k hodnote pristupujeme, buď pomocou iterátora a funkcie `find( ID )`.

5.3 Použitie adaptivity pri modelovaní šíriaceho sa požiaru.

Nech hranica požiaru je opäť daná krivkou Γ, ktorá je implicitne popísaná nulovou izočiarou level set funkcie. Sieť budeme zjemňovať v miestach, kde sa nachádza hranica požiaru, v okolí o šírke σ. Počas vývoja našej aplikácie sa ukázalo, že zjemňovať sieť len v okolí nulovej izočiary nie je dostatočné pre prípad s palivom rozloženým nerovnomerne po oblasti. Preto sme sa rozhodli, že sieť budeme zjemňovať aj v miestach, kde je náhla zmena v hodnote popisujúcej kvalitu paliva.

Algoritmus pozostáva z dvoch základných krokov: i) výber a označenie elementov, ktoré majú byť zjemnené resp. odstránené a ii) prenos riešenia z pôvodnej siete na novú sieť:

1. Cyklus cez všetky elementy siete

   - Ak element pretína nulová izočiara, alebo je rozdiel hodnôt paliva vo vrcholoch elementu vysoký, označ element na zjemnenie

   - Koniec cyklu cez elementy siete

2. Uloženie dvojice ID uzla siete a aktuálnej hodnoty v uzle siete do poľa `hodnoty` pre všetky uzly siete Typ poľa `hodnoty` je `std::map`

3. Uloženie ID prepony každého elementu a hodnoty v jej strede do poľa `noveHodnoty`. Uvažujeme lineárnu interpoláciu, teda hodnota v strede prepony bude priemer hodnôt v jej vrcholoch

4. Volanie `preAdapt()` a `adapt()`

5. Cyklus pre úrovne siete od 0 po `maxLevel()`

- Cyklus cez všetky elementy úrovne
  - Ak pre vrchol elementu existuje ID v poli `hodnoty`, nastav príslušnú hodnotu vrcholu
  - Ak pre vrchol elementu neexistuje ID v poli `hodnoty`, najdi hodnotu pre ID hrany otca, na ktorej leží vrchol, v poli `noveHodnoty` a priraď ju vrcholu
- Koniec cyklu cez všetky úrovne

6. Volanie `postAdapt()`

7. Koniec

Vplyv adaptivity na presnosť riešenia a výpočtový čas pre jednoduchý príklad uvádzame v kapitole 6.

Kapitola 6

Numerické experimenty

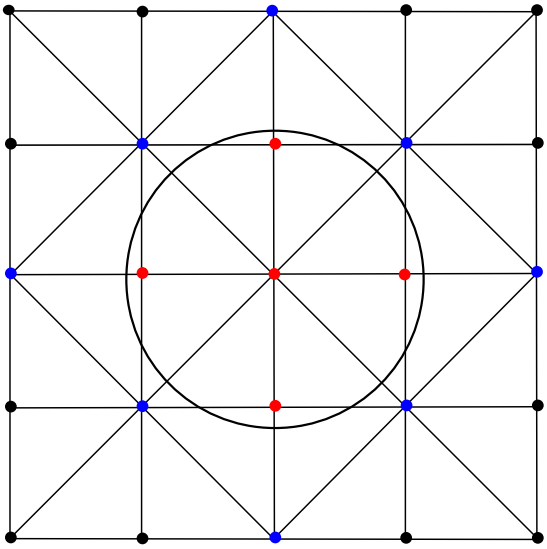
V tejto kapitole prezentujeme výsledky dosiahnuté numerickým riešením advekčno-difúznych rovníc. V prvej podkapitole ukážeme konvergenciu a rád presnosti numerickej metódy pre advekčnú časť rovnice. Ďalej uvádzame výsledky riešenia kompletnej rovnice modelu požiaru a ich porovnania pre rôzne voľby vstupných parametrov.

6.1 Výpočet znamienkovej funkcie vzdialenosti

V prvom príklade ukážeme konvergenciu a rád numerickej metódy na riešenie rovnice advekcie v tvare:

$$\frac{\partial \phi(\mathbf{x},t)}{\partial t} + \mathbf{N} \cdot \nabla \phi(\mathbf{x},t) = 1. \tag{6.1}$$

Úlohu sme riešili na jednotkovom štvorci, pre uzly kde $\phi_0 > 0$, s počiatočnou podmienkou $\phi(\mathbf{x},0) = \phi^0(\mathbf{x}) = |\mathbf{x} - 0.5| - 0.1$, čo je vlastne znamienková funkcia vzdialenosti a aj presné riešenie našej úlohy. $\mathbf{N} = \frac{\nabla\phi}{|\nabla\phi|}$ je vektor jednotkovej vonkajšej normály k príslušnej izočiare level set funkcie ϕ. Okrajové podmienky sme zvolili tak, že sme na počiatočnej hodnote fixovali hodnoty na a v okolí krivky (teda hodnoty pre modré uzly na obrázku 6.1).



Obr. 6.1: Prípad keď nepoznáme presnú pozíciu krivky pretože krivka neprechádza žiadnym uzlom siete.

Našou motiváciou je nájsť stacionárne riešenie úlohy 6.1, ktoré je znamienkovou funkciou vzdialenosti k počiatočnej polohe krivky, pozri aj 1.9.

Počítali sme L1 chybu

$$E = \frac{1}{|\omega_i|} \sum_i (\phi_i^N - \phi_i^0), \tag{6.2}$$

kde N je index posledného časového kroku, kedy sa riešenie ustálilo.

Rád konvergencie EOC definujeme následovne:

$$EOC = \frac{\log(e_2) - \log(e_1)}{\log(0.5)}, \tag{6.3}$$

kde e_1 je chyba riešenia na sieti s veľkosťou elementu h a e_2 je chyba riešenia na sieti s veľkosťou elementu $\frac{h}{2}$.

Výsledky sú v tabuľke 6.1.

N	t	L1 chyba	EOC	CPU
64	0.85	4.7040e-05	-	6s
128	0.67	1.7750e-05	1.41	30s
256	0.65	4.5572e-06	1.96	3min 52s

Tabuľka 6.1: Chyby a rád konvergencie (EOC) pre príklad so znamienkovou funkciou vzdialenosti na štvorcovej oblasti 1×1. Počiatočná uzavretá krivka so stredom v $[0.5, 0.5]$ má polomer 0.1. N je počet delení siete pozdĺž okraja siete, $h = 1/N$ a t je čas dosiahnutia stacionárneho riešenia. CPU je výpočtový čas.

6.2 Ukážky výsledkov pre šírenie sa požiarov v rôznych podmienkach.

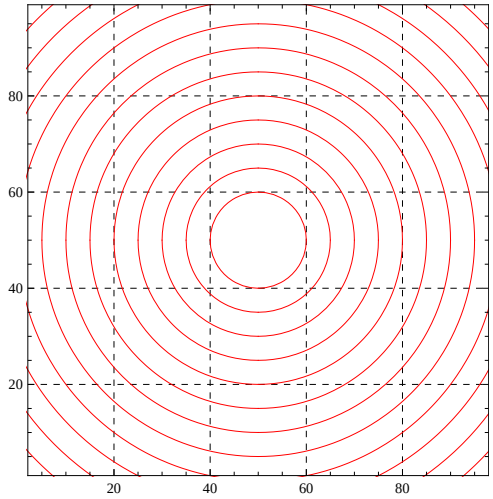
V tejto časti uvádzame ukážky príkladov pre rôzne zjednodušené prípady šíriacich sa požiarov. Len pre pripomenutie uvedieme tvar rýchlostného poľa $\mathbf{V}$:

$$\begin{aligned} \mathbf{V} &= \beta \mathbf{N} \\ \beta &= V(1 - \delta\kappa) \\ V &= e^{\lambda \mathbf{N} \cdot \mathbf{q}} f(\mathbf{x}), \end{aligned}$$

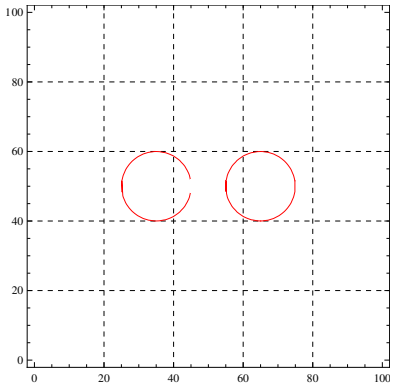
kde $\mathbf{N}$ je jednotková vonkajšia normála k hranici požiaru, $\mathbf{q}$ je konštantná rýchlosť vetra, κ je krivosť a $f(\mathbf{x})$ je kvalita paliva (v tejto časti sa zatiaľ obmedzíme na $f(\mathbf{x}) = 1$). Parametre δ a λ si môžeme zvoliť.

6.2.1 Príklad 1

Uvažujme teraz dva požiare, ktoré sú od seba vzdialené, pozri obrázok 6.3.



Obr. 6.2: Vypočítaná znamienková funkcia vzdialenosti so znázornenými izočiarami, ktoré popisujú polohu expandujúcej krivky v časoch $t = 0, 5, 10 \ldots$

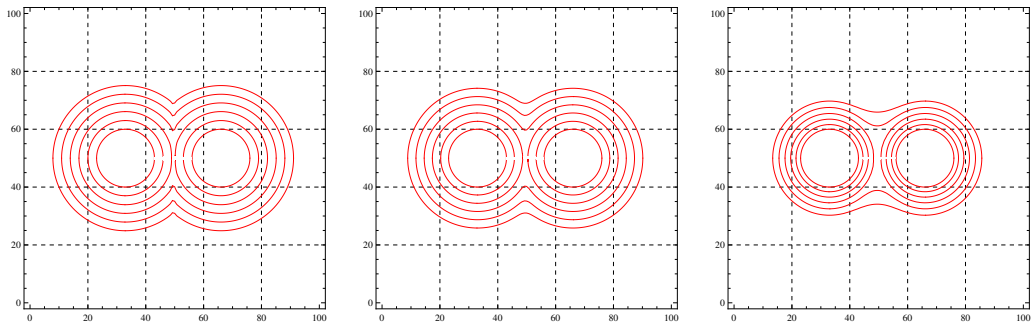


Obr. 6.3: Dva vzdialené požiare.

Predpokladajme nulový účinok vetra $\mathbf{q} = [0, 0]$, teda požiare sa šíria jednotkovou rýchlosťou v smere normály a v určitom čase sa stretnú a spoja do jedného. Vývoj hranice požiarov v čase je zobrazený na obrázku 6.4. Výhodou level set metódy je, že topologickú zmenu v hranici požiaru nemusíme nijako zvlášť ošetrovať, jediné čo musíme urobiť je zobraziť nulovú izočiaru level set funkcie.

Ďalej si môžeme na obrázku všimnúť vplyv koeficientu δ v definícii rýchlosti $\mathbf{V}$ na riešenie. Vidíme, že krivosť pôsobí „proti“ pohybu expandujúceho

požiaru. Túto vlastnosť budeme v ďalších príkladoch kompenzovať práve druhým parametrom λ.



$\delta = 0$ $\delta = 1.0$ $\delta = 5.0$

Obr. 6.4: Vývoj dvoch požiarov s rôznou voľbou vplyvu krivosti. Iný pohľad na túto situáciu môžeme vidieť aj v kapitole 1, na obrázku 1.3, kde je znázornený tvar level set funkcie. Izočiary vo všetkých troch obrázkoch sú zobrazené v tých istých časoch.

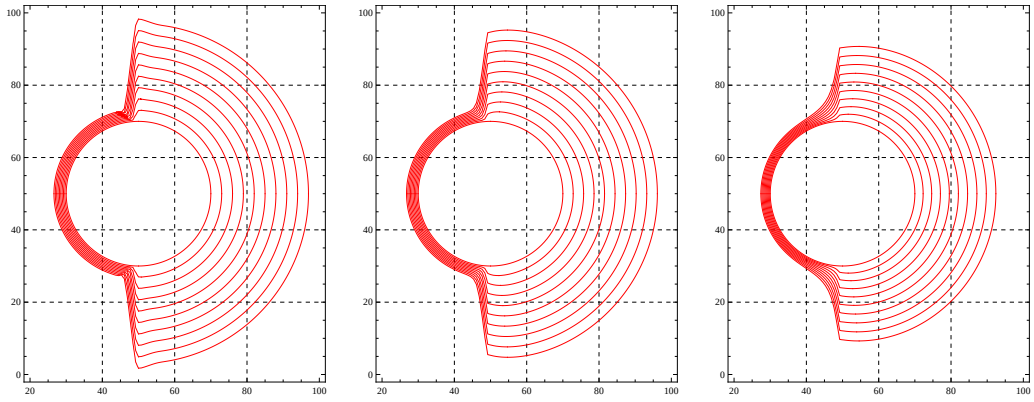
6.2.2 Príklad 2

V tomto príklade zostaneme stále pri prípade $\mathbf{q} = [0, 0]$ bez vetra a predvedieme vplyv δ krivosti a paliva $f(\mathbf{x})$ na vývoj požiaru. Počiatočný požiar je v tvaru kruhu s polomerom 10m so stredom v bode $[50, 50]$. Palivo môže nadobúdať hodnoty od 0 do 1, pričom nulové palivo je v miestach, ktoré nehoria vôbec a naopak, palivo s hodnotou 1 horí najľahšie. Výsledky sú zobrazené na obrázku 6.5.

Okrem vplyvu paliva si tu môžeme všimnúť zhladzujúci efekt krivosti na riešenie. Kým na obrázku pre prípad $\delta = 0$ vznikne pomerne ostrý prechod v miestach, kde sa hodnoty paliva menia, na obrázku pre prípad $\delta = 5.0$ je tento efekt odstránený.

6.2.3 Príklad 3

V tomto príklade uvádzame výsledky zahŕňajúce vplyv krivosti aj vetra pre rôzne voľby hodnôt parametrov δ a λ. Postupujeme podľa [8], kde δ a λ boli



$\delta = 0$ $\delta = 1.0$ $\delta = 5.0$

Obr. 6.5: Vplyv krivosti a paliva na riešenie. Palivo bolo zvolené $f = 0.125$ v časti naľavo od stredu počiatočného požiaru a $f = 1$ napravo.

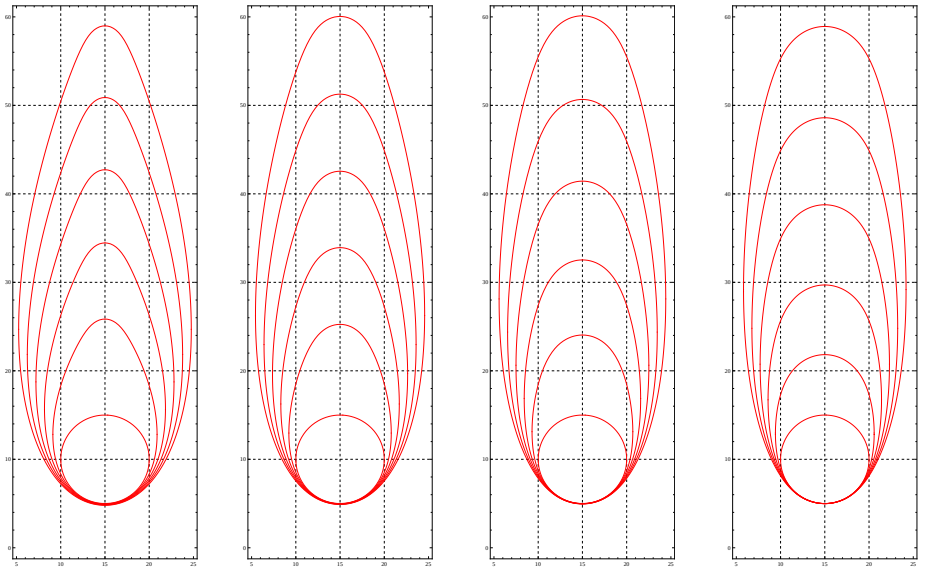
zvolené tak, aby sa požiar dostal za určitý čas do rovnakej vzdialenosti od počiatočnej pozície. Vietor bol vždy zvolený $\mathbf{q} = [0, 3]$, palivo bolo zvolené $f(\mathbf{x}) = 1$. Hodnoty parametrov δ a λ sú uvedené v popise obrázku 6.6.

Na obrázku 6.8 zobrazujeme výsledky realizované Lagrangeovskou metódou [8].

Výsledky dosiahnuté level set metódou a Lagrangeovskou sú porovnané na obrázku 6.9. Vidíme, že výsledný tvar požiaru je pre obe metódy totožný. Výsledky boli vypočítané na sieti S_5^6 a počiatočná sieť S_0 je zobrazená na obrázku 6.7.

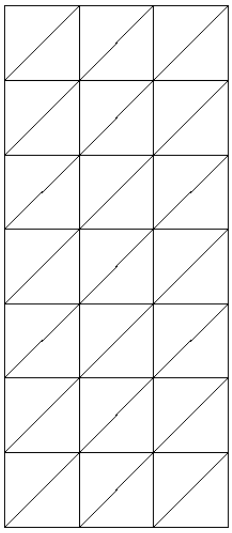
6.2.4 Príklad 4

V poslednom príklade (obrázok 6.10) predvedieme vplyv lokálne zjemnenej siete na presnosť riešenia. Príklad je identický s príkladom 3, pričom zvolené parametre boli $\lambda = 1, \delta = 1, \mathbf{q} = [0, 3]$ a palivo $f = 1$ na celej oblasti.

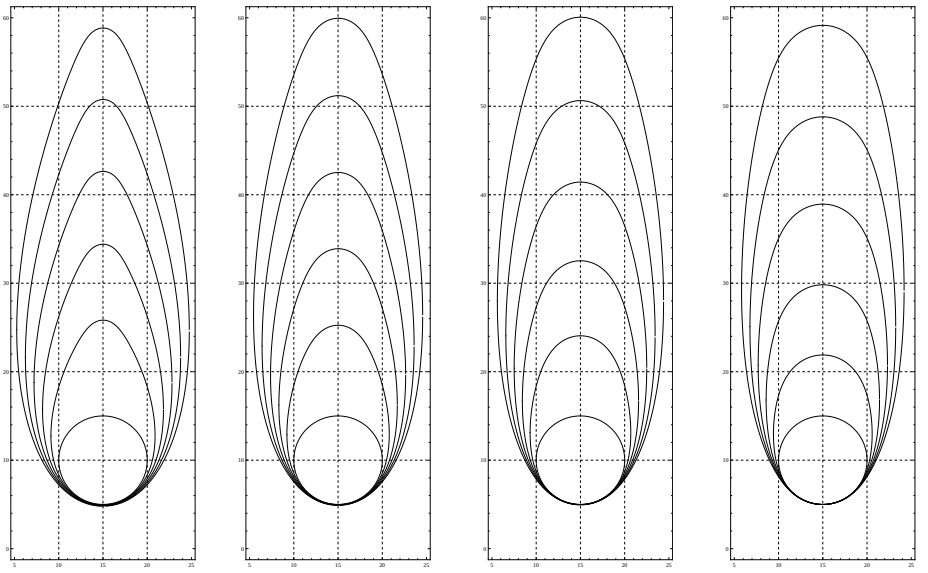


$\lambda = 1.0, \delta = 1.0$ $\lambda = 1.13, \delta = 2.0$ $\lambda = 1.27, \delta = 3.0$ $\lambda = 1.45, \delta = 4.0$

Obr. 6.6: Porovnanie výsledkov pre rôzne voľby parametrov λ, δ na rovnakej sieti. Počiatočný požiar má tvar kruhu so stredom v bode $[10, 15]$ a s polomerom 5 v obdĺžnikovej oblasti 30×70. Parametre boli zvolené tak, aby sa požiar dostal približne do bodu $[60, 15]$ v čase $t = 5$. Nulové izočiary level set funkcie sú zobrazené v časoch $t = 0, 1, 2, 3, 4, 5$.

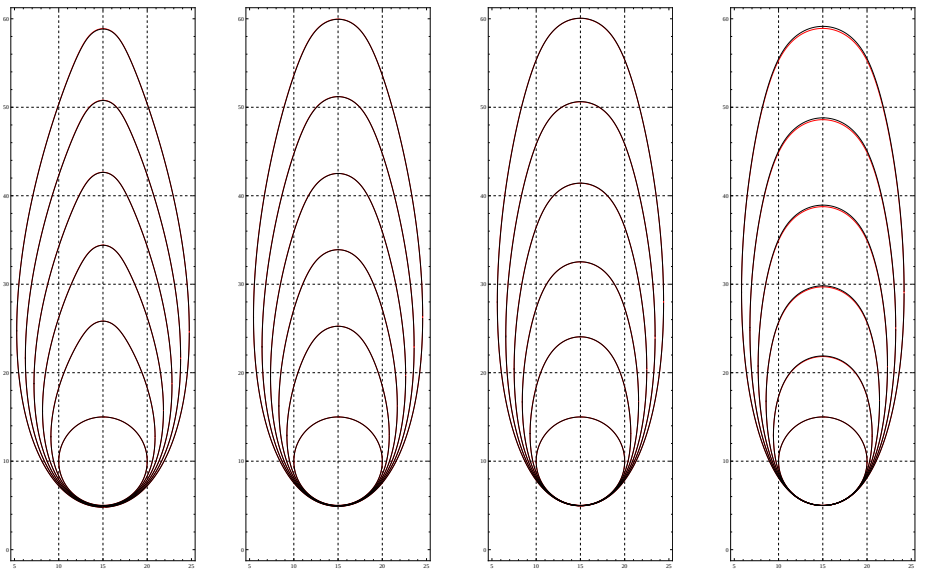


Obr. 6.7: Sieť S_0 pre príklad 3.



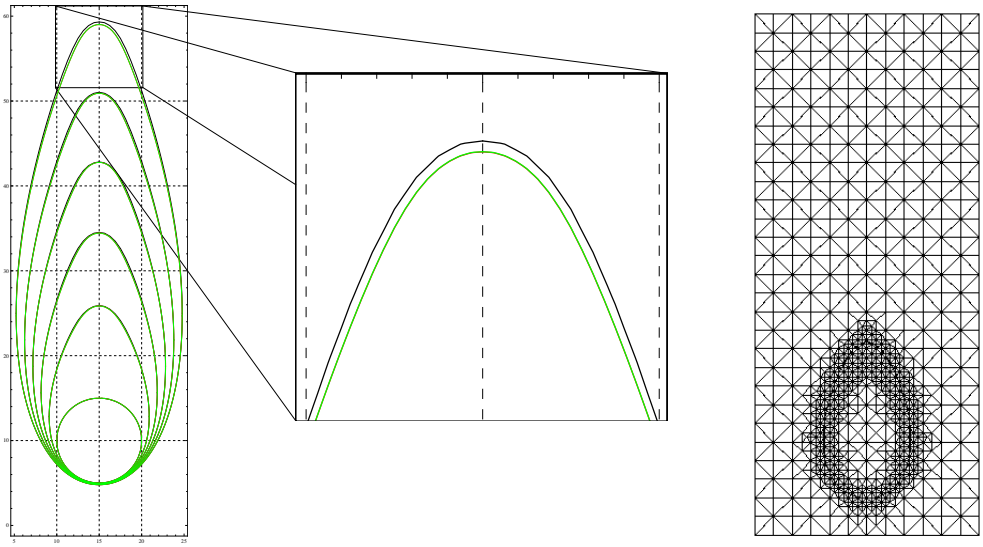
$\lambda = 1.0, \delta = 1.0$ $\lambda = 1.13, \delta = 2.0$ $\lambda = 1.27, \delta = 3.0$ $\lambda = 1.45, \delta = 4.0$

Obr. 6.8: Výsledky dosiahnuté Lagrangeovskou metódou.



$\lambda = 1.0, \delta = 1.0$ $\lambda = 1.13, \delta = 2.0$ $\lambda = 1.27, \delta = 3.0$ $\lambda = 1.45, \delta = 4.0$

Obr. 6.9: Porovnanie výsledkov pre Lagrangeovskú a level set metódu.



Obr. 6.10: Na obrázku vidíme hranicu požiaru v časoch $t = 0, 1, \ldots, 5$. Výsledky boli dosiahnuté na sieti S_4 (čierna), S_4^5 (zelená) a S_5 (červená). Vidíme, že riešenie na sieti S_4^5 je totožné s riešením na sieti S_5, preto je ním prekryté. Čas výpočtu bol 7 sekúnd pre sieť S_4, pre sieť S_4^5 3 minúty a 51 sekúnd, a pre sieť S_5 7 minút 38 sekúnd. Vpravo je výpočtová sieť v čase $t = 1$.

Kapitola 7

Aplikácie

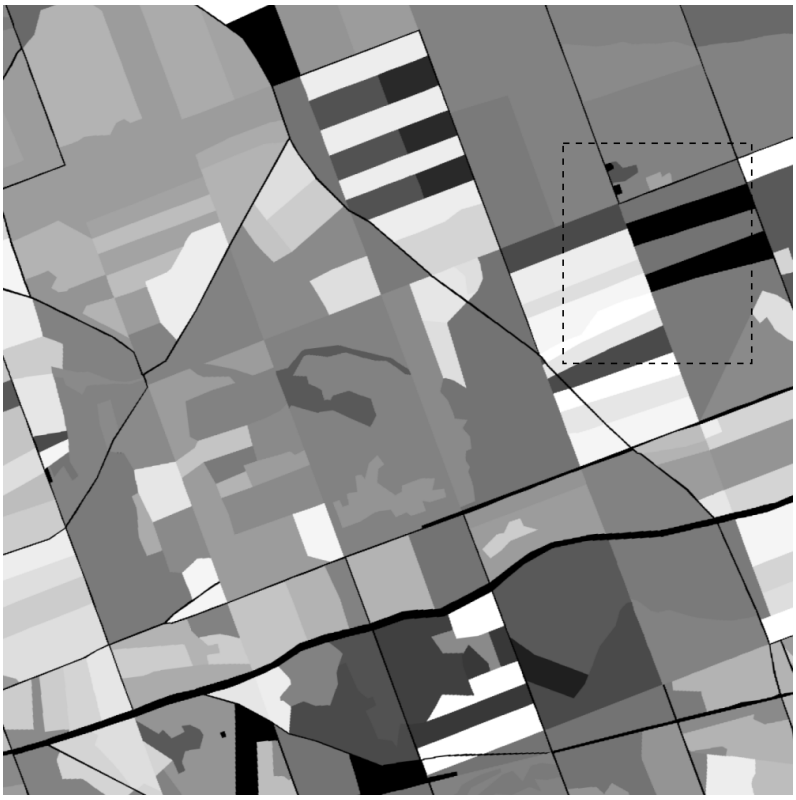
7.1 Popis vstupných dát

Náš model sme testovali aj na reálnych dátach. Ako vstup načítavame výrez z obrázku 7.1, na ktorom je zobrazená mapa Záhoria [7, 8]. Táto mapa zobrazuje odtiene šedi, kde svetlejšie odtiene predstavujú miesta, cez ktoré sa požiar šíri rýchlejšie a naopak tmavé miesta, kde sa šíri pomalšie alebo vôbec (napríklad prázdne pole s ornou pôdou).

Obrázok je vo formáte .pgm a obsahuje všetky odtiene šedej zapísane v hodnotách 0 až 255, ktoré preškálujeme na hodnoty 0 až 1, čo sú hodnoty paliva $f(\mathbf{x})$.

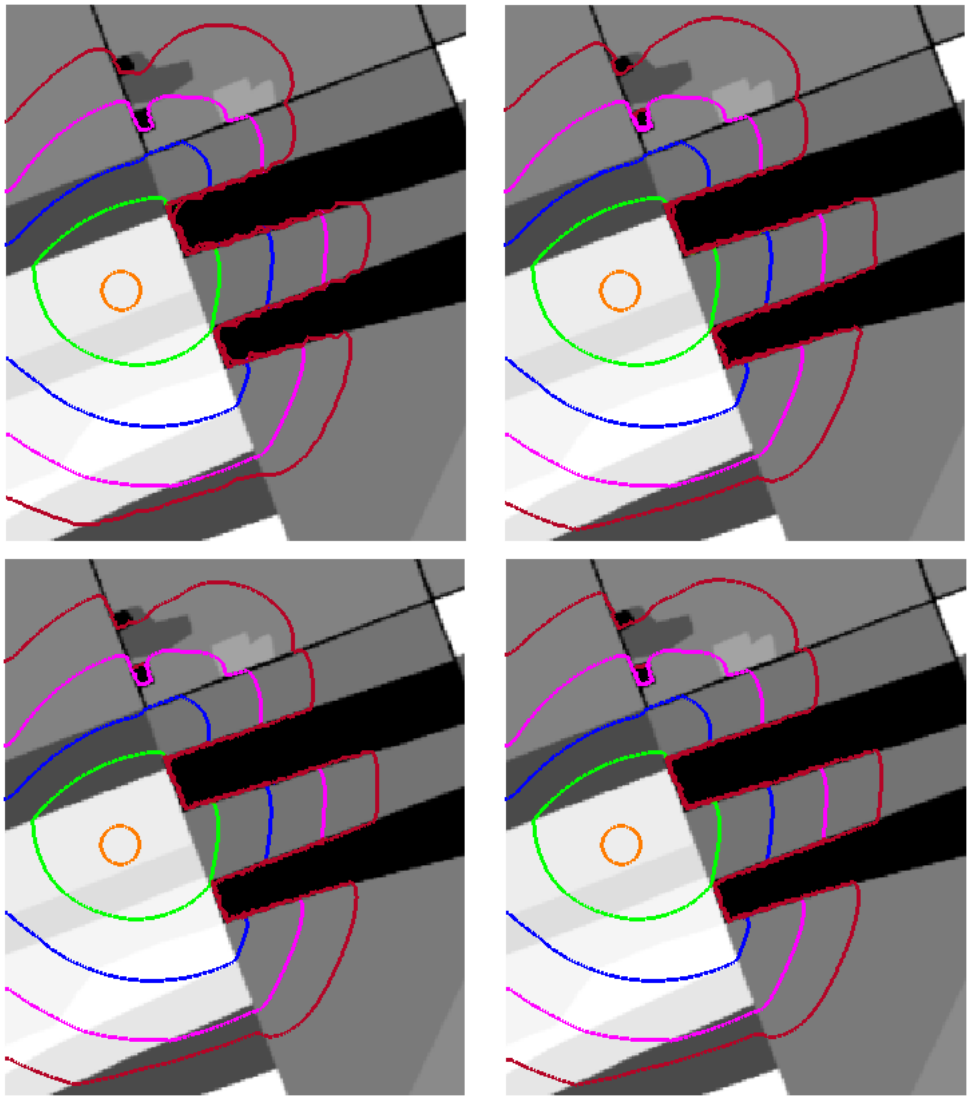
7.2 Porovnanie výsledkov

Na obrázku 7.2 môžeme vidieť výsledky pre rôzne voľby zjemnenia výpočtovej siete na výreze 240×280 z mapy na obrázku 7.1. Pre sieť S_3^6 bol zvolený časový krok $\triangle t = 1.0$ a postupne pre každú jemnejšiu sieť bol zmenšovaný na polovicu. Použitím adaptivity sa výsledky zlepšia najmä v oblastiach s vysokým „gradientom“ v palive. Najdôležitejší prínos integrácie adaptivity do našej aplikácie je výrazné skrátenie výpočtového času, pričom výsledky dosiahnuté na sieti S_5^8 môžeme prehlásiť na identické s výsledkami na sieti S_8. Sieť je zobrazená na obrázku 7.4.

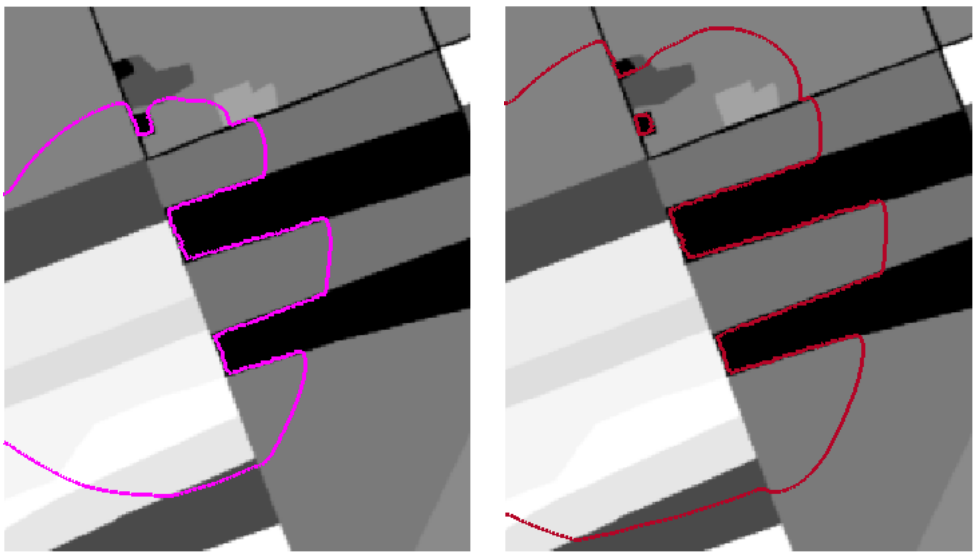


Obr. 7.1: Mapa Záhoria. Skutočná veľkosť obrázku je 1000×1000 pixelov. 1 pixel predstavuje plochu $1m^2$. Pre naše výpočty sme si vybrali výrez o veľkosti 240×280 ohraničený prerušovanou čiarou.

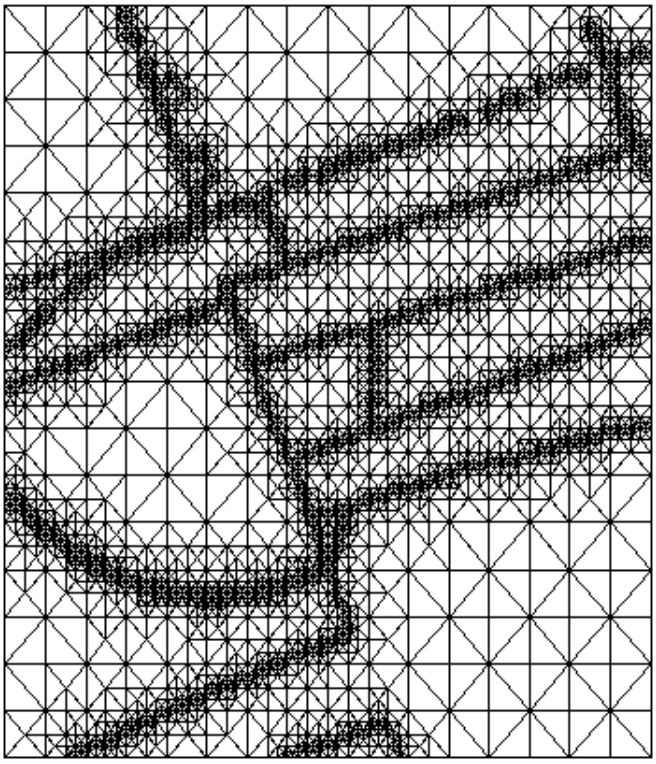
Na obrázku 7.3 môžeme lepšie vidieť topologickú zmenu v hranici požiaru.



Obr. 7.2: Porovnanie výsledkov dosiahnutých na sieťach (zľava zhora po riadkoch) S_3^6, S_4^7, S_5^8 a S_8 pre výrez z obrázku 7.1 o veľkosti 240×280. Sieť S_0 pozostáva z dvoch trojuholníkov, sieť S_8 má 257×257 uzlov. Počiatočný požiar má polomer 10 so stredom v bode $[60, 130]$. Vietor bol zvolený $\mathbf{q} = [0.23, 0.41]$ a parametre $\delta = 1$ a $\lambda = 1$. Oranžová izočiara zodpovedá počiatočnej polohe požiaru v čase $t = 0$ a ďalšie izočiary zodpovedajú časom $t = 50, 100, 150, 200$ v poradí zelená, modrá, ružová, červená. V okolí dvoch nehorľavých prekážok vpravo sú zelená, modrá a ružová izočiara prekryté čevenou. Výpočet na sieti S_3^6 trval 24 sekúnd, na sieti S_4^7 1 minúta 13 sekúnd, na sieti S_5^8 3 minúty a 11 sekúnd a na globálne zjemnenej sieti S_8 16 minút 44 sekúnd.



Obr. 7.3: Topologická zmena v hranici požiaru v ľavej hornej časti výrezu. Na ľavom obrázku je znázornená hranica požiaru v čase $t = 150$, ktorá sa krátko nato kôli nehorľavej prekážke rozdelí na dve časti. Na pravom obrázku je hranica požiaru v čase $t = 200$.



Obr. 7.4: Výpočtová sieť S_3^6 v čase $t = 100$. Sieť je lokálne zjemnená v okolí hranice požiaru a v miestach, kde je skok v palive vyšší ako 0.2.

Súhrn

V práci sme úspešne aplikovali numerickú level set metódu na riešenie matematického modelu šírenia sa pozemného požiaru. Matematický model sme podrobne opísali, implementovali a otestovali. Na numerických experimentoch sme predviedli výhody implicitného vyjadrenia hranice pozemného požiaru a dosiahnuté výsledky sme porovnali s výsledkami dosiahnutými Lagrangeovskou metódou. Ukázalo sa, že výsledky dosiahnute level set metódou sú porovnatelné s Lagrangeovskou metódou, pričom sme nemuseli riešiť niektoré problémy vyskytujúce sa pri Lagrangeovskej metóde. Náročnosť level set metódy na výpočtový čas sa nám podarilo znížiť integráciou lokálneho zjemňovania siete, čo sa významne prejavilo pri testovaní našej aplikácie na reálnych dátach. Celú aplikáciu sme implementovali pomocou softvérovej knižnice Dune. Level set metóda je teda vhodnou alternatívou na aplikáciu v modelovaní šíriaceho sa pozemného požiaru.

Summary

In this work we successfully applied the numerical level set method to solve a mathematical model of the spread of surface fire. We described mathematical model in detail, implemented and tested it succsessfully. We showed the advantages of the implicid expressed fire front and the results were compared with results from Lagrangian methods in numerical experiments. It turned out that the results achieved with level set method are comparable with the Lagrangian methods, and we did not solve some problems one can encounter when working with Lagrangian methods. We succsessfully integrated local refinement and decreased CPU time of the level set method, which is significantly reflected when testing our application on real data. The entire application was implemented using software library DUNE. Level set method is therefore a suitable alternative for application in modeling the propagation of surface fire.

Literatúra

[1] Peter Frolkovič, Christian Wehner: *Flux-based level set method on rectangular grids and computation of first arrival time functions*, Computing and Visualization in Science, Vol. 12, No. 6. s. 297-306, 2009

[2] Peter Frolkovič, Karol Mikula: *Flux-based level set method: A finite volume method for evolving interfaces*, Applied Numerical Mathematics, 4 (57), p. 436-454, 2007

[3] Maroš Bohunčák: *Riešenie rovnice advekcie pomocou softvéru DUNE*, Bakalárska práca, 2009

[4] Maroš Bohunčák: *Numerické modelovanie problému šírenia pozemných požiarov pomocou softvéru DUNE*, práca ŠVK, 2010

[5] Peter Bastian, Markus Blatt, Andreas Dedner, Christian Engwer, Robert Klöfkorn, Martin Nolte, Mario Ohlberger, Oliver Sander: *The Distributed and Unified Numerics Environment (DUNE) Grid Interface HOWTO*, Version 2.2svn, March 18, 2011

[6] J. A. Sethian: *Level set methods and fast marching methods: Evolving interfaces in computational geometry, fluid mechanics, computer vision, and material science* , Cambridge University Press, New York, 1999

[7] Jozef Urbán: *Modelovanie lesných požiarov pomocou evolúcie rovinných kriviek*, práca ŠVK, 2011

[8] M. Balažovjech, M. Bohunčák, P. Frolkovič, L. Halada, K. Mikula, M. Petrášová, J. Urbán: *Numerical modeling of forest fire propagation*, in preparation, 2011

[9] D.X. Viegas, L.P. Pita, L. Matos, P. Palheiro: *Slope and wind effects on fire spread*, Forest Fire Research and Wildland Fire Safety, Millpress, Rotterdam, ISBN 90-77017-72-0, 2002