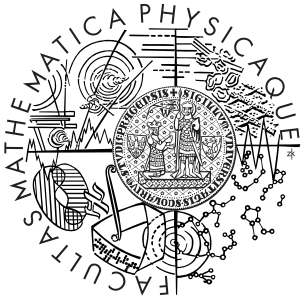
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Zuzana Záhorová

Numerické řešení stlačitelného proudění

Katedra numerické matematiky

Vedoucí bakalářské práce: Doc. RNDr. Jiří Felcman, CSc.
Studijní program: Matematika, obor Obecná matematika

2007

Na tomto místě bych ráda poděkovala svému vedoucímu bakalářské práce doc. RNDr. Jiřímu Felcmanovi, CSc. za odborné vedení a konzultace při psaní práce a panu Mgr. Petru Kuberovi za ochotnou pomoc s programem pro metodu konečných objemů a anizotropní adaptaci sítě.

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

V Praze dne 2. července 2007 Zuzana Záhorová

Obsah

Název práce: Numerické řešení stlačitelného proudění
Autor: Zuzana Záhorová
Katedra (ústav): Katedra numerické matematiky
Vedoucí bakalářské práce: Doc. RNDr. Jiří Felcman, CSc.
e-mail vedoucího: felcman@karlin.mff.cuni.cz

Abstrakt: Práce se zabývá numerickým řešením nevazkého stlačitelného transonického proudění, které je popsáno Eulerovými rovnicemi. Tato soustava parciálních diferenciálních rovnic je řešena pomocí metody konečných objemů na předem dané síti a je doprovázena adaptivním zjemněním sítě. Adaptivní síť se konstruuje pomocí anizotropní adaptace, následuje přepočítání přibližného řešení metodou konečných objemů na novou síť. Hlavním požadavkem metody je splnění geometrického zákona zachování v každém výpočetním kroku, což nám dovoluje řešit i nestacionární problémy.

Klíčová slova: Eulerovy rovnice, metoda konečných objemů, adaptace sítě, nestacionární problém

Title: Numerical solution of compressible flow
Author: Zuzana Záhorová
Department: Department of Numerical Mathematics
Supervisor: Doc. RNDr. Jiří Felcman, CSc.
Supervisor's e-mail address: felcman@karlin.mff.cuni.cz

Abstract: This work deals with the numerical simulation of the multidimensional inviscid compressible transonic gas flow. The adaptive strategy is applied to the numerical solution of problem governed by hyperbolic partial differential equations. An adaptive mesh is constructed in the framework of the cell-centred finite volume scheme. The nonstationary problem is studied. The main attention is paid to an adaptive part of a time marching procedure. The main feature of the proposed method is to keep the mass conservation of the numerical solution at each adaptation step. We apply an anisotropic mesh adaptation. This is followed by a recovery of the approximate solution on the new mesh satisfying the geometric conservation law.

Keywords: Euler equations, finite volume method, mesh adaptation, nonstationary problems

Kapitola 1

Metoda konečných objemů a adaptace sítě

Téma, kterým se tato bakalářská práce zabývá, je numerické řešení a simulování nevazkého stlačitelného transonického proudění plynů. Tato proudění se dají popsat Eulerovými rovnicemi. To je soustava parciálních diferenciálních rovnic, kterou budeme řešit pomocí metody konečných objemů na předem dané síti. Tuto metodu ještě doplníme adaptivním zjemňováním sítě pokrývající výpočetní oblast.

Adaptivní síť je konstruována pomocí anizotroponí adaptace, která je doprovázena přepočítáváním přibližného řešení metodou konečných objemů na novou síť. Hlavním požadavkem této metody je splnění geometrického zákona zachování hmoty v každém výpočetním kroku. To nám dovoluje řešit i nestacionární problémy. V práci je prezentováno numerické řešení v případě nespojité počáteční podmínky.

Obecně je stlačitelné proudění popsáno Navier-Stokesovými rovnicemi. V případech, kdy jsou vazkost a koeficient tepelné vodivosti malé (jako třeba v případě většiny plynů, kdy jsou efekty vazkosti omezeny na malou ohraničenou vrstvu přilehlou k povrchu obtékaných těles), lze Navier-Stokesovy rovnice považovat pouze za jistou perturbaci Eulerových rovnic. V případech, kdy tepelnou výměnu zanedbat nelze, mezní vrstva reaguje na změny, musíme hledat řešení pro vazká, stlačitelná proudění na nestrukturované síti. To lze pomocí Navier-Stokesových rovnic.

Závěr tedy je, že dobrá metoda pro řešení vazkého proudění by měla být založena na dostatečně robustním schématu pro simulaci proudění nevazkého.

Eulerova simulace je speciálně vhodná pro proudění, která obsahují nespojitosti v řešení. Její využití můžeme najít v mnoha vědních oborech, jako je například aeronautika, automobilový průmysl, navrhování turbín, kompresorů nebo pump, v medicíně, meteorologii nebo oceánografii. Tato proudění často uvažujeme v situacích, kdy je velice těžké nebo nemožné provádět spolehlivá měření. V těchto případech mohou výpočetní metody přispět k lepšímu pochopení problému a tudíž i zkrátit navrhovací proces.

1.1 Fyzikální model

Uvažujme proudění nevazkého dokonalého plynu v omezené polygonální, příp. polyhedrální oblasti (ve 2D, příp. ve 3D) $\Omega \subset I\!R^N$ a v časovém intervalu $(0,\, T)$, kde $T > 0$. Předpokládejme $N = 2$, resp. 3 pro proudění ve 2D, resp. 3D. Dále předpokládáme, že Ω je mnohoúhelník ve 2D (mnohostěn ve 3D). Pro zjednodušení budeme zanedbávat vnější síly a předpokládat adiabatičnost proudění. Naším cílem je numericky řešit Eulerovy rovnice

$$\frac{\partial \boldsymbol{w}}{\partial t} + \sum_{s=1}^{N} \frac{\partial \boldsymbol{f}_s(\boldsymbol{w})}{\partial x_s} = 0, \qquad Q_T = \Omega \times (0,T) \tag{1.1}$$

s počáteční podmínkou,

$$\boldsymbol{w}(x,0) = \boldsymbol{w}^0(x), \qquad x \in \Omega \tag{1.2}$$

danou vektorovou funkcí $\boldsymbol{w}^0(x)$ a okrajovými podmínkami

$$B(\boldsymbol{w}(x,t)) = 0, \qquad (x,t) \in \partial\Omega \times (0,T), \tag{1.3}$$

kde B je jistý operátor na hranici. Vektor $\boldsymbol{w} = (\rho, \rho v_1, \ldots, \rho v_N, E)^T$ je z $I\!R^m$, $m = N + 2$ (tzn. $m = 4$ nebo 5 pro 2D, případně 3D proudění), kde ρ značí hustotu, $v_1, \ldots, v_N$ složky rychlosti a E totální energii. Funkce toku $\boldsymbol{f}_s, s = 1, \ldots, N$ jsou m-dimenzionální zobrazení.

1.2 Základní rysy výpočetní metody

Systém popsaný rovnicemi (1.1) - (1.3) můžeme aproximovat pomocí diskrétního explicitního schématu založeného na metodě konečných objemů, které vypadá takto:

$$\boldsymbol{w}_i^{k+1} = \boldsymbol{w}_i^k - \frac{\tau_k}{|D_i|} \sum_{j \in S(i)} \boldsymbol{H}(\boldsymbol{w}_i^k, \boldsymbol{w}_j^k, \boldsymbol{n}_{ij}) |\Gamma_{ij}|, \tag{1.4}$$

$$D_i \in \mathcal{D}_h^k, t_k \in [0,T)$$

Toto schéma je založeno na konstrukci sítě $\mathcal{D}_h^k = \{D_i\}_{i \in J}$ metodou konečných objemů, kde $J \subset Z^+ = \{0, 1, \ldots\}$ je indexová množina, $h > 0$ a $D_i, i \in J$ jsou uzavřené mnohoúhelníky nebo mnohostěny - pro $N = 2$ nebo 3 - s navzájem disjunktními vnitřky a $\boldsymbol{n}_{ij}$ označuje jednotkovou vnější normálu

k ∂D_i na Γ_{ij}. $S(i)$ je indexová množina obsahující informaci o sousedních konečných objemech, případně typu hranice (vstup, výstup, pevná stěna), je-li D_i konečný objem přilehlý k hranici.
Poznámka *V programu (jeho popis následuje ve druhé kapitole) je ošetřeno, jak se pracuje s konečnými objemy ležícími na hranici, kde se musí používat okrajové podmínky. Z hodnot v této množině se tedy dá vyčíst lokalizace daného konečného objemu. Můžeme určit, zda se daný objem nachází na vstupu, na výstupu nebo uvnitř výpočetní oblasti.*
Pro dva sousední konečné objemy $D_i, D_j \in \mathcal{D}_h^k$ označíme symbolem Γ_{ij} jejich společnou hranu, tj. $\Gamma_{ij} = \partial D_i \cap \partial D_j = \Gamma_{ji}$. Označíme symboly $|D_i|$ N-dimenzionální míru D_i a $|\Gamma_{ij}|$ $(N-1)$-dimenzionální míru Γ_{ij}.

Zkonstruujeme dělení $0 = t_0 < t_1 \ldots$ časového intervalu $[0, T]$ a označíme symbolem $\tau_k = t_{k+1} - t_k$ časový krok mezi t_k a t_{k+1}. Schéma (1.4) reprezentuje algoritmus pro hledání aproximace integrálního průměru

$$\frac{1}{|D_i|} \int_{D_i} \boldsymbol{w}(x, t_k) dx \approx \boldsymbol{w}_i^k$$

veličiny $\boldsymbol{w}$ na objemu D_i v čase t_k. Dále aproximujeme tok veličiny $\boldsymbol{w}$ přes Γ_{ij} ve směru $\boldsymbol{n}_{ij}$

$$\sum_{s=1}^{N} \boldsymbol{f}_s(\boldsymbol{w})(\boldsymbol{n}_{ij})_s$$

pomocí numerického toku $\boldsymbol{H}(\boldsymbol{w}_i^k, \boldsymbol{w}_j^k, \boldsymbol{n}_{ij})$, závislého na hodnotě přibližného řešení $\boldsymbol{w}_i^k$ na objemu D_i, na hodnotě $\boldsymbol{w}_j^k$ na D_j a na normále $\boldsymbol{n}_{ij}$:

$$\int_{t_k}^{t_{k+1}} \int_{\Gamma_{ij}} \sum_{s=1}^{N} (\boldsymbol{n}_{ij})_s \boldsymbol{f}_s(\boldsymbol{w}) dS dt \approx \tau_k \boldsymbol{H}(\boldsymbol{w}_i^k, \boldsymbol{w}_j^k, \boldsymbol{n}_{ij}) |\Gamma_{ij}| \tag{1.5}$$

Metoda (1.4) je doplněna počáteční podmínkou $\boldsymbol{w}_i^0, i \in J$, definovanou takto:

$$\boldsymbol{w}_i^0 = \frac{1}{|D_i|} \int_{D_i} \boldsymbol{w}^0(x) dx, \tag{1.6}$$

za předpokladu lokální integrovatelnosti funkce $\boldsymbol{w}^0$ z (1.2) ($\boldsymbol{w}^0 \in L^1_{loc}(\Omega)^m$).

Schéma (1.4) je formálně prvního řádu přesnosti. To je příčinou toho, že nespojitosti v řešení jsou často nepřesné a nějaké detaily nejsou vyřešeny dostatečně přesně. Abychom tomuto jevu předešli, můžeme použít simultánně dvě techniky: metodu vyššího řádu a adaptivní zjemňování sítě.

Protože s tímto termínem budeme dále pracovat, na tomto místě si definujeme pojem přibližného řešení.

Definice *Přibližné řešení systému (1.1) vypočítané pomocí metody konečných objemů definujeme jako po částech konstantní vektorovou funkci* $\mathbf{w}^k_{\mathcal{D}^k}$, $k = 0, 1, \ldots$ *na polygonální (polyhedrální) oblasti* Ω *předpisem* $\mathbf{w}^k_{\mathcal{D}^k}|_{D_i{}^\circ} = \boldsymbol{w}^k_i$ *pro* $i \in J$, *kde* $D_i{}^\circ$ *je vnitřek* D_i *(tj.* $D_i{}^\circ = D_i - \partial D_i$*) a* $\boldsymbol{w}^k_i$ *je získáno pomocí předpisu (1.4). Symbol* $\mathcal{D}^k$ *značí síť sestávající z uzavřených objemů* D_i. *Funkce* $\mathbf{w}^k_{\mathcal{D}^k}$ *značí přibližné řešení v čase* $t = t_k$ *a vektor* $\boldsymbol{w}^k_i$ *je hodnota přibližného řešení na konečném objemu* D_i *v čase* t_k.

1.3 Adaptace sítě

Vycházíme z výsledků prezentovaných v [2]. Zde byla prezentována a testována metoda konečných objemů pro nestacionární problém v 1D. V této práci předložíme výsledky experimentů ve 2D. Algoritmus lze ale obecně formulovat i pro třírozměrné úlohy.

Algoritmus se sestává z těchto částí:

- Vycházíme ze sítě v k-té iteraci a odpovídajícího řešení.
- Toto řešení na staré síti přepočítáme metodou konečných objemů a získáme nové řešení na původní síti ($\mathbf{w}^{k+1}_{\mathcal{D}^k}$). Tedy 'předpovídáme' vývoj řešení v (k+1)-ním kroku. Toto řešení dále využijeme při adaptaci sítě.
- V druhém kroku adaptujeme síť (na $\mathcal{D}^{k+1}$) pomocí původní sítě a nového, 'předpovězeného' řešení. Pro další výpočet můžeme řešení z minulého kroku zapomenout, dále se již nevyužívá.
- Nyní přepočítáme řešení ze staré sítě na síť novou, adaptovanou (dostaneme $\tilde{\mathbf{w}}^k_{\mathcal{D}^{k+1}}$) pomocí geometrického zákona zachování. K tomu použijeme původní řešení a původní i novou síť.
- Nakonec pomocí nové sítě a řešení přepočítaného na tuto síť dostáváme nové přibližné řešení dané na nové síti ($\mathbf{w}^{k+1}_{\mathcal{D}^{k+1}}$). Tedy jsme zase u výchozího bodu a celý postup můžeme opakovat.

Nyní si algoritmus popíšeme o něco přesněji:

1. Predikce: $\mathbf{w}_{\mathcal{D}^k}^{k+1} :=$ FVsol $(\mathbf{w}_{\mathcal{D}^k}^k, \mathcal{D}^k)$ *(Finite volume solution)*
   Zde symbol $\mathcal{D}^k$ je síť v k-té iteraci schématu (1.4) a $\mathbf{w}_{\mathcal{D}^k}^k$ přibližné řešení na síti $\mathcal{D}^k$ v časovém kroku t_k. Numerické řešení $\mathbf{w}_{\mathcal{D}^k}^{k+1}$ v čase t_{k+1} na síti $\mathcal{D}^k$ počítáme na základě známé sítě $\mathcal{D}^k$ a odpovídajícího řešení $\mathbf{w}_{\mathcal{D}^k}^k$ pomocí schématu (1.4).

2. Adaptace: $\mathcal{D}^{k+1} :=$ MeshAdapt $(\mathcal{D}^k, \mathbf{w}_{\mathcal{D}^k}^{k+1})$ *(Mesh adaptation)*
   Nová síť $\mathcal{D}^{k+1}$ v (k+1)-ním kroku je konstruována pomocí anisotropní adaptace na základě vypočítané predikce $\mathbf{w}_{\mathcal{D}^k}^{k+1}$ a sítě na původní hladině t_k, tj. $\mathcal{D}^k$.

3. Aproximace řešení na nové síti pomocí zákona zachování:
   $\tilde{\mathbf{w}}_{\mathcal{D}^{k+1}}^k :=$ SolRecovery $(\mathbf{w}_{\mathcal{D}^k}^k, \mathcal{D}^k, \mathcal{D}^{k+1})$ *(Solution recovery)*
   Nyní staré řešení $\mathbf{w}_{\mathcal{D}^k}^k$ dané na staré síti $\mathcal{D}^k$ přepočítáme pomocí zákona zachování (viz. dále kapitola 1.5) na novou síť $\mathcal{D}^{k+1}$. Řešení označíme $\tilde{\mathbf{w}}_{\mathcal{D}^{k+1}}^k$ a budeme ho používat dále k výpočtům.

4. Přepočítání řešení: $\mathbf{w}_{\mathcal{D}^{k+1}}^{k+1} :=$ FVsol $(\tilde{\mathbf{w}}_{\mathcal{D}^{k+1}}^k, \mathcal{D}^{k+1})$
   V posledním kroku výpočtu získáme (k+1)-ní přiblížení řešení dané již na nové síti $\mathcal{D}^{k+1}$. Využijeme k tomu síť v (k+1)-ní iteraci $\mathcal{D}^{k+1}$ a odpovídající aproximaci řešení splňující geometrický zákon zachování $\tilde{\mathbf{w}}_{\mathcal{D}^{k+1}}^k$. Přibližné řešení $\mathbf{w}_{\mathcal{D}^{k+1}}^{k+1}$ na časové hladině t_{k+1} je počítáno opět pomocí explicitní metody konečných objemů (1.4).

1.4 Anisotropní adaptace sítě

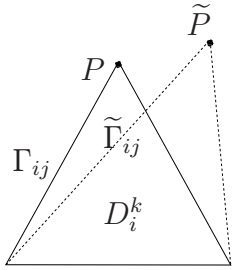
Pro danou síť $\mathcal{D}^k$ a příslušné řešení $\mathbf{w}_{\mathcal{D}^k}^{k+1}$ definujeme parametr kvality sítě takto:

$$Q_{\mathcal{D}^k} := \frac{1}{\#\mathcal{D}^k} \sum_{D_i \in \mathcal{D}^k} \sum_{e \,=\, hrana\ D_i} (\|e\|_{\mathbf{w}_{\mathcal{D}^k}^{k+1}} - c_N)^2, \tag{1.7}$$

kde $\#\mathcal{D}^k$ je počet konečných objemů v $\mathcal{D}^k$, $\|\cdot\|_{\mathbf{w}_{\mathcal{D}^k}^{k+1}}$ označuje energetickou normu vektoru tvořícího hranu e konečného objemu D_i, která je daná maticí závislou na aproximaci Hessovy matice numerického řešení, a c_N je konstanta závislá na toleranci chyby interpolace a dimenzi problému. Detaily lze nalézt v [1]. Odpovídající ekvilaterální N-dimenzionální síť (ve smyslu nejmenších čtverců) vzhledem k Riemannově normě $\|\cdot\|_{\mathbf{w}_{\mathcal{D}^k}^{k+1}}$, je konstruována minimalizováním parametru kvality sítě $Q_{\mathcal{D}^k}$.

V algoritmu tedy adaptujeme síť $\mathcal{D}^k$ pomocí minimalizace parametru $Q_{\mathcal{D}^k}$ a chceme najít novou síť $\mathcal{D}^{k+1}$ tak, aby kvalitativní parametr $Q_{\mathcal{D}^{k+1}}$ sítě $\mathcal{D}^{k+1}$ byl nižší než původní parametr $Q_{\mathcal{D}^k}$.

1.5 Geometrický zákon zachování hmoty



Obrázek 1.1: Adaptace sítě, přesunutí: $P \to \widetilde{P}$, $\Gamma_{ij} \to \widetilde{\Gamma}_{ij}$

Po adaptaci sítě je nutné přepočítat řešení $\mathbf{w}_{\mathcal{D}^k}^k$ dané na staré síti $\mathcal{D}^k$ na řešení $\tilde{\mathbf{w}}_{\mathcal{D}^{k+1}}^k$ dané na nové síti $\mathcal{D}^{k+1}$. V každém kroku musí být splněn geometrický zákon zachování hmoty. To znamená

$$\sum_{i \in J^k} |D_i^k| \boldsymbol{w}_i^k = \sum_{i \in J^{k+1}} |D_i^{k+1}| \tilde{\boldsymbol{w}}_i^k, \tag{1.8}$$

kde $\tilde{\boldsymbol{w}}_i^k = \tilde{\mathbf{w}}_{D^{k+1}}^k|_{D_i^{k+1^\circ}}$ ($D_i^{k+1^\circ}$ označuje vnitřek D_i^{k+1}).

V následujícím se soustředíme na přepočítávání řešení po přesunutí vrcholů (tj. po adaptaci sítě) - viz. ilustrační obrázek (1.1). V tomto případě počet konečných objemů $\#\mathcal{D}^k$ sítě $\mathcal{D}^k$ je stejný jako parametr $\#\mathcal{D}^{k+1}$ sítě $\mathcal{D}^{k+1}$.

Pomocí posunu vrcholu P konečného objemu D_i^k do nové pozice $\widetilde{P}$ konečného objemu D_i^{k+1} je definováno lineární zobrazení $\boldsymbol{c}$. Bod $x \in D_i^k$ je transformován do bodu $\tilde{x} \in D_i^{k+1}$ vztahem

$$\tilde{x} = x - \boldsymbol{c}(x) \tag{1.9}$$

Za předpokladu, že posunutí $\boldsymbol{c}$ je malé, můžeme použít aproximaci

$$\int_{D_i^{k+1}} \boldsymbol{w}(\tilde{x})d\tilde{x} \approx \int_{D_i^k} \boldsymbol{w}dx - \int_{\partial D_i^k} \boldsymbol{w}c_n dS, \tag{1.10}$$

kde zanedbáváme členy vyšších řádů. V (1.10) klademe $c_n = \boldsymbol{c} \cdot \boldsymbol{n}$, kde $\boldsymbol{n}$ je jednotková vnější normála k ∂D_i^k. Přechod k integrálním středním hodnotám a aproximace plošného integrálu v (1.10) vede k následující formuli pro výpočet $\tilde{\boldsymbol{w}}_i^k$ splňující geometrický zákon zachování hmoty (1.8)

$$|D_i^{k+1}|\tilde{\boldsymbol{w}}_i^k = |D_i^k|\boldsymbol{w}_i^k - \sum_{j \in S(i)} |\Gamma_{ij}|(c_{n_{ij}}^+ \boldsymbol{w}_i^k + c_{n_{ij}}^- \boldsymbol{w}_j^k). \tag{1.11}$$

Většina symbolů je definována již dříve (ve vzorci (1.4)), $\pm$ označuje pozitivní ($\geq$) a negativní ($\leq$) část skalární veličiny. Konstanty $c_{n_{ij}}$ v (1.11)jsou vypočítávány v těžištích hran Γ_{ij}. V případě, že $c_{n_{ij}}$ vypočítané v těžištích jsou nezáporné (jako na našem ilustračním obrázku), lze aproximovat

$$\int_{\Gamma_{ij}} \boldsymbol{w}c_n dS \approx |\Gamma_{ij}|c_{n_{ij}}\boldsymbol{w}_i^k. \tag{1.12}$$

V tomto případě težiště hrany $\widetilde{\Gamma}_{ij} = \partial D_i^{k+1} \cap \partial D_j^{k+1}$ leží v D_i^k. Ekvivalentně, aproximace použitá v (1.12) může být vyjádřena

$$\int_{\Gamma_{ij}} \boldsymbol{w}c_n dS \approx \begin{cases} |\Gamma_{ij}|c_{n_{ij}}\boldsymbol{w}_i^k & \text{je-li } \tilde{e_{ij}} \in D_i^k, \\ |\Gamma_{ij}|c_{n_{ij}}\boldsymbol{w}_j^k & \text{je-li } \tilde{e_{ij}} \in D_j^k, \end{cases} \tag{1.13}$$

kde $\tilde{e_{ij}}$ je těžiště $\widetilde{\Gamma}_{ij}$.

Kapitola 2

Softwarové produkty ZK 2.1

Na KNM MFF UK je v současné době vyvíjen software s názvem ZK 2.1. Program řeší nestacionární Eulerovy rovnice pro nevazké stlačitelné proudění pomocí metody konečných objemů a anizotropní adaptace sítě. Program je implementován pod operačním systémem Linux. V této kapitole uvedeme popis tohoto software.

Program je napsán v jazyce C++, část doplňkových programů v jazyce fortran, pro překlad je tedy potřeba mít nainstalovány příslušné překladače. Pro vizualizaci výsledků se využívá program gnuplot, který je volně dostupný.

Adresář, ve kterém budeme pracovat, můžeme pojmenovat libovolně. V dalším budeme používat název `ZK 2.1`. Do tohoto adresáře se rozbalí 3 podadresáře potřebné k běhu programu. Jsou to adresáře `res`, `src` a `triang`. V adresáři `res` probíhá samotný výpočet, v adresáři `src` jsou uloženy zdrojové kódy používaných programů. Adresář `res` obsahuje podadresář `etc`, ve kterém jsou uloženy konfigurační soubory, nastavují se zde parametry výpočtu. Adresář `triang` je pouze pomocný, obsahuje dvě sítě pro testování.

Potřebný je ještě pomocný program `shock-generator`, který generuje počáteční podmínky pro funkce testované v této práci. Tento program uložíme do samostatného adresáře.

Vstupní i výstupní data jsou ve formě textových souborů. Používání tohoto typu sice zpomaluje načítání, vypisování dat nebo průběh výpočtu, ovšem na druhou stranu je tento formát uživatelsky přívětivější, přehlednější a tudíž vhodnější pro širší použití.

2.1 Vstupní data

Vstupní data jsou uložena v adresáři `res`, případně v podadresáři `etc`.

- adresář `res` obsahuje soubory:
  - potřebné programy, jsou to `mama`, `add`, `compare`, `ader2`, `megen` a `kresli`, k programu `kresli` je potřeba mít v adresáři i spouštěcí skript `kreslic` (použití programů bude popsáno dále)
  - `file.dt2` - obsahuje okrajové podmínky úlohy, nutné pro spuštění programu `ader2`
  - `file.dt3` - tento soubor obsahuje konfigurace a nastavení výpočtu pro program `ader2`, důležitý je hlavně třetí řádek `time`, kde se udává, po jakých časových krocích má výpočet postupovat - tj. po jakých krocích se má adaptovat řešení a odpovídajícím způsobem také síť
  - `result_start.dat` - původní řešení, nutné pro zákon zachování (tj. $\mathbf{w}^k_{\mathcal{D}^k}$)
  - `triang` - soubor s použitou triangulací (V rámci bakalářské práce byla použita síť v předem daném formátu. V tomto formátu může uživatel vytvořit vlastní triangulaci, popis viz http://www.karlin.mff.cuni.cz/∼ dolejsi)
  - `start-moving` - startovací skript, který spustí program `mama` a předá mu příslušné parametry uložené v pevně daných souborech
  - `xy-slice-noadapt.txt` - pomocný soubor pro ukládání výsledků
  - `adapt_solve.sh` - spouštěcí skript, pomocí kterého se provádí výpočet (popis bude dále)
- adresář `etc` obsahuje soubory
  - `paramet` - parametry pro anizotropní zjemňování
  - `optim` - optimalizační parametry pro různé optimalizační metody obsažené ve výpočtu

Během programu se do adresáře uloží několik pomocných souborů, se kterými program dále pracuje. Stojí za to zmínit hlavně tyto:

- `vol_sol.dat` - řešení pomocí metody konečných objemů, používá se pro zjemnění sítě, ($\mathbf{w}_{\mathcal{D}^k}^{k+1}$)
- `result_mod.dat` - nové řešení na adaptované triangulaci získané metodou konečných objemů ($\mathbf{w}_{\mathcal{D}^{k+1}}^{k+1}$)
- `triang.new` - datový soubor obsahující adaptovanou triangulaci ($\mathcal{D}^{k+1}$)

2.2 Překlad programů, spuštění a výpočet

Před prvním spuštění programu je nutné všechny potřebné programy přeložit a nakopírovat z příslušných podadresářů adresáře `src` do pracovního adresáře `res`. Totéž je potřeba udělat při každé změně zdrojových kódů. Jednotlivé části programu, které je nutné v pracovním adresáři mít, jsou následující:

- `mama` (*Mesh Adaptation*) - program provádějící anizotropní adaptaci sítě
- `add` - pomocný program na sčítání dvou reálných čísel, zjišťuje aktuální čas výpočtu
- `compare` - pomocný program na porovnávání dvou reálných čísel, zjišťuje, zda program doběhl již do konečného času nebo ne
- `ader2` - program provádějící aproximaci řešení na základě metody konečných objemů
- `megen` - pomocný program pro `ader2`, zpracovává soubor `triang` do podoby vhodné pro řešič
- `kresli` - pomocný program, který upraví datové soubory adaptovaných řešení do formátu vhodného pro vykreslení v programu gnuplot (tj. do textových souborů)

Zdrojové kódy jsou uloženy v adresáři `src` v odpovídajících podadresářích. Nyní si popíšeme postup překladu jednotlivých částí programu.

V adresáři `src` jsou uloženy následující podadresáře:

- `adaptace` - zde se překládají programy `mama`, `add` a `compare`, a sice následujícími posloupnostmi příkazů:
  - `mama`: *make clean* (adresář se uvolní pro nový překlad programu)
    make (vytvoří se program `mama`, popis překladu je uveden v souboru `makefile`)
  - `add`: *gcc -o add add.c* (vytvoří se program `add`)
    chmod u+x add (přidělí se práva programu `add`)
  - `compare`: analogicky jako program `add`
    gcc -o compare compare.c
    chmod u+x compare
- `kreslic` - zde se překládá program `kresli`, a to následujícím postupem:
  make clean (vyčistí se adresář před novým přeložením programu)
  make (vytvoří se program `kresli`)
- `resic` - zde se překládají programy `ader2` a `megen` následujícími příkazy:
  - `ader2`: *make -f ader2.make*
  - `megen`: *make -f megen.make*

Nyní v těchto souborech máme přeložené všechny potřebné programy. Překopírujeme je do pracovního adresáře `res` a program je připravený na spuštění.

Program se spustí příkazem *./adapt_solve.sh time*. Do proměnné *time* se uloží čas, do kterého chceme výpočet provádět - například *0.2* nebo *1.05*.

Spouštěcí skript `adapt_solve.sh` je napsán tak, že se provádí výpočet po časových krocích, jejichž velikost je dána nastavením v souboru `file.dt3` na řádku `time`, až do požadovaného času. Všechny tyto mezivýsledky se ukládají a skript obsahuje while cyklus, který hlídá, kdy se dosáhne předepsaného času zadaného v příkazu *./adapt_solve.sh time*. Poté se výpočet ukončí.

2.3 Výstupy

Výsledkem programu jsou výstupy ve formě textových souborů, které se dají dále použít například v programu gnuplot. Tyto soubory se v průběhu výpočtu ukládají do adresáře `SOL`, který je uložen v adresáři `res`. Zde se při každém výpočtu automaticky vygeneruje adresář, jehož název má tvar `time_date-adapt`, kde `time_date` je čas a datum spuštění výpočtu. Sem se ukládají textové soubory ze všech provedených kroků (jak je to popsáno v minulé podkapitole), které se dají použít k vykreslení sítě a odpovídajícího řešení v daném čase. Tyto soubory mají název `time-mesh.txt` pro síť a `time-slice-adapt.txt` pro řešení.

Soubor `time-slice-adapt.txt` slouží k vykreslení řezu funkcí. Řez je brán po diagonále výpočetní oblasti (popsána dále) a vykresluje se první složka přibližného řešení $\mathbf{w}^k_{\mathcal{D}^k}$, což je hustota.

Oba dva typy těchto souborů se dají vykreslit v programu gnuplot, lze je uložit i do formátu postscript. Zde je postup:
set terminal postscript
set size square - příkaz pro čtvercový formát obrázku
set output 'file', kde *file* je jméno výstupního souboru,
např. *'/home/Bc/vystup/sit1.ps'*
plot 'name.txt' w l (pokud vynecháme příkaz *w l* na konci, vykreslí se pouze body, ale nespojí se hranami)

Na konci každého výpočetního kroku se na obrazovce objeví základní údaje o provedeném výpočtu týkající se aproximace sítě. Lze si zde ověřit, zda se vlastnosti sítě zlepšují (`Quality`), kolik se v jednom kroku provádí iterací sítě (`iteration 0, iteration 1, ...`) a další údaje, které mohou napovědět, zda výpočet probíhá podle očekávání.

Příklad výpisu programu na obrazovku:

```
MASS before: 103.14191766 -0.00000007 0.00000001 128.92739765
Initial Quality: 13.671137
iteration 0
iteration 1
iteration 2
iteration 3
iteration 4
Total time = 2.720000
0     Quality: 11.344560
1     Quality: 9.356190
2     Quality: 7.834059
3     Quality: 6.749697
4     Quality: 5.847678
max 0.677373     min 0.062998     epsilon (max/min) 10.752341
MASS after: 103.14191766 -0.00000007 0.00000001 128.92739765
```

Popis některých parametrů:
`MASS before/after:` hustota, první a druhá složka hybnosti, energie před a po adaptaci sítě
`Initial Quality:` počáteční parametr kvality sítě ($Q_{\mathcal{D}^k}$)
`k     Quality:` parametr kvality sítě po k-té iteraci v daném kroku
`max, min, epsilon (max/min):` délka nejdelší, nejkratší hrany, jejich podíl

2.4 Generátor počátečních podmínek

Před spuštěním programu `ZK 2.1` je potřeba do vstupního souboru `result_start.dat` uložit počáteční podmínku ve tvaru, s kterým umí program pracovat. To zajistí pomocný program `shock-generator`, který tuto podmínku vygeneruje.

V rámci bakalářské práce byly testovány dva příklady - *Vortex evolution problem* a *Explosion*. V adresáři `shock-generator` jsou dva podadresáře, každý pro jednu počáteční podmínku. Jejich názvy jsou `init-explosion` a `init-ve`. Program je napsán v jazyce fortran. Pro danou počáteční podmínku se vypočítá řešení na původní síti (soubor `triang`), tj. řešení $\mathbf{w}^k_{\mathcal{D}^k}$.

Podadresáře obsahují soubory potřebné pro generování počátečních podmínek. Popíšeme si význam dvou z nich - `soldata.f` a `ader2.f`. Soubor `soldata.f` obsahuje počáteční podmínku k daným problémům, soubor `ader2.f` je program řešení metodou konečných objemů upravený tak, že nedělá žádnou iteraci a pouze vygeneruje soubor s řešením na původní síti (síť uložena v souboru `triang`). Při změně souboru `triang` (použití jiné sítě) je před vlastním generováním nutné tuto síť převést do tvaru vhodného pro řešič. To zajistí program `megen`, použije se příkaz *./megen*.

Samotný program se jmenuje `ve`, případně `explosion`. Dále si popíšeme přesný postup, jak počáteční podmínku vygenerovat do tvaru potřebného pro program. Postup bude popsán pro adresář `init-ve`, v adresáři `init-expl` je postup obdobný.

V adresáři `init-ve` zadáme příkaz *make*, což vede k přeložení programu a vytvoří se nový program `ve`. Nyní lze samotný program spustit pomocí příkazu *./ve*. Vznikne soubor `vol_sol.dat`, kde je uloženo řešení na počáteční síti. Tento soubor tedy zkopírujeme do adresáře `/ZK 2.1/res` a přejmenujeme ho na `result_start.dat`. Nyní máme vše připraveno a program lze spustit postupem popsaným výše (viz. kapitola 2.2, Překlad programů, spuštění a výpočet).

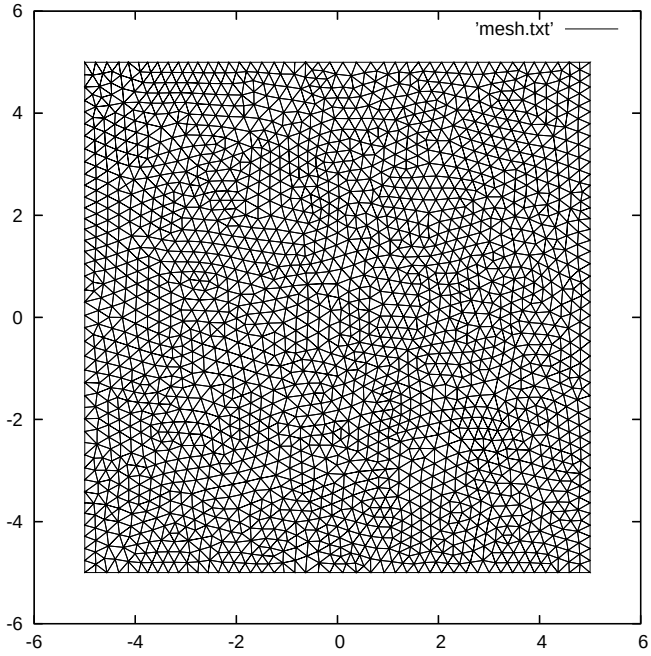
2.5 Popis testovaných příkladů

Výpočet se provádí na oblasti $[-5,5]\times[-5,5]$. Na této oblasti konstruujeme síť (viz obr. 2.1), na které později upravujeme řešení. V souboru `/ZK 2.1/triang` jsou k dispozici dvě sítě, které se liší počtem elementů. Menší síť s 4096 elementy a větší síť s 16384 elementy. Obecně je vhodnější větší síť, řešení je přesnější, ale výpočet zase bývá o hodně pomalejší (při výpočtech na PC se 4 procesory typu Intel(R) Xeon(TM) CPU 3.20GHz a 2GB ram byl rozdíl v délce výpočtu v řádu desítek minut).

Na zvolené síti tedy dále řešíme Eulerovy rovnice pomocí explicitního schématu založeného na metodě konečných prvků (1.4). Výpočet probíhá podle adaptace sítě popsané v kapitole 1.3. K výpočtu je samozřejmě nutná i počáteční a okrajová podmínka daného problému (ty si pro jednotlivé testované příklady blíže popíšeme v následujících kapitolách).

Vykreslovat budeme vždy adaptovanou síť a odpovídající řešení získané pomocí schématu (1.4). Řez řešením se vykresluje po diagonále, tzn. po přímce $\boldsymbol{z}$ z bodu $(-5,-5)$ do bodu $(5,5)$. Vykresluje se první složka vektoru přibližného řešení $\mathbf{w}^k_{\mathcal{D}^k}$, což je hustota.

Pomocí metody konečných objemů dostaneme výsledné řešení $\mathbf{w}^k_{\mathcal{D}^k}$ dané na síti $\mathcal{D}^k$. Toto řešení je po částech konstantní, na každém konečném objemu $D_i \in \mathcal{D}^k$ je jeho hodnota $\boldsymbol{w}^k_i$. Nyní chceme zjistit a vykreslit hodnotu v každém bodě, kde se přímka $\boldsymbol{z}$ protne s hranou Γ_{ij} společnou konečným objemům D_i a D_j, $D_i, D_j \in \mathcal{D}^k$. U této hrany zjistíme pomocí váženého průměru z hodnot na všech okolních konečných objemech hodnotu funkce ve vrcholech přislušných hraně Γ_{ij} a tyto hodnoty interpolujeme do průsečíku. Nyní máme spočítanou požadovanou hodnotu, která se následně vykreslí do grafu.



Obrázek 2.1: Počáteční síť (4096 elementů) $\mathcal{D}^0$ na oblasti $(-5,5) \times (-5,5)$

2.6 Vlastnosti sítě

V této části popíši, jak lze měnit do určité míry vlastnosti výpočetní sítě. Bude ukázáno na příkladu s nespojitou počáteční podmínkou, který je popsán v další kapitole. K obrázkům sítě je přidáno i vykreslení řezu výslednou funkcí.

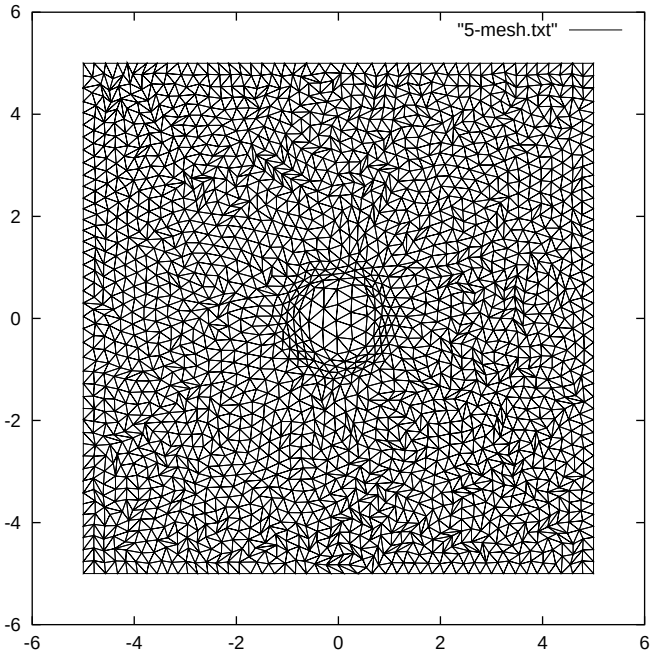
Budu ukazovat vliv parametru, který určuje poměr nejkratší a nejdelší strany sítě, tudíž její 'hustotu' v místě nespojitosti. Protože algoritmus pracuje pouze s posunem vrcholů a ne jejich přidáváním, tento parametr vpodstatě určuje, jak moc se v místech, kde není funkce spojitá, může síť zjemnit a tím se i zpřesnit výpočet.

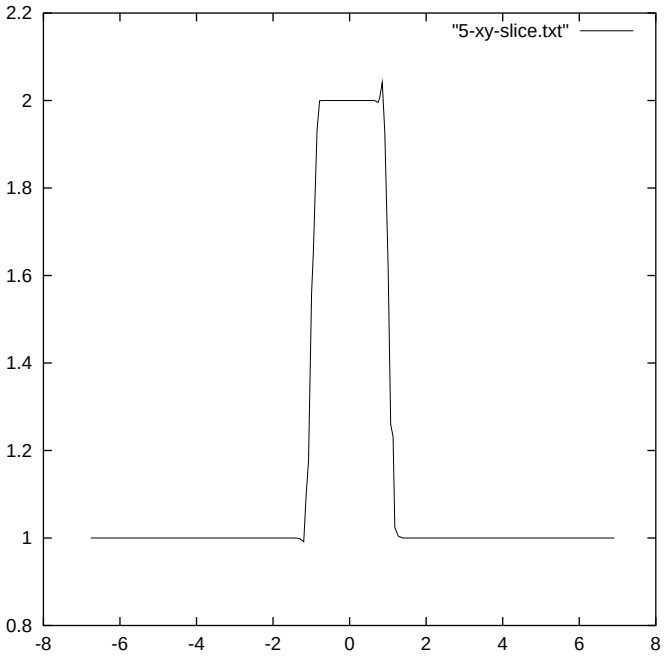
Parametr se nastavuje v souboru `/ZK 2.1/res/etc/paramet`, kde je jako šestý údaj uložena jeho hodnota pod názvem *epsilon1*. Tuto hodnotu může uživatel měnit a poté sledovat její vliv na výpočty.

V programu je standardně přednastavena tato hodnota na 50, což považuji za přiměřené (v dalších kapitolách tedy bude toto nastavení zachováno). Na obrázcích je vidět, že pro malá čísla (například 5) se síť moc neadaptuje, není úplně poznat, kde přesně je řešení nespojité. Pro větší hodnoty (cca od 150) je zas problém s vykreslením řezu funkcí. Můžeme zde pozorovat tzv. "Gibbsův jev", což jsou nepřesnosti v řešení (řešení ve vrcholech 'střílí') způsobené numerickým výpočtem.

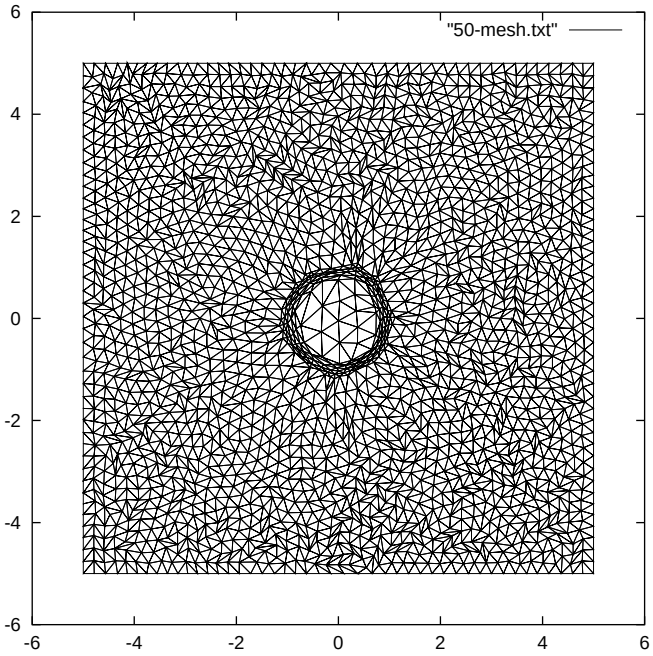
Nyní si již představíme výsledky pro různé hodnoty parametru, který dále budu značit ϵ.

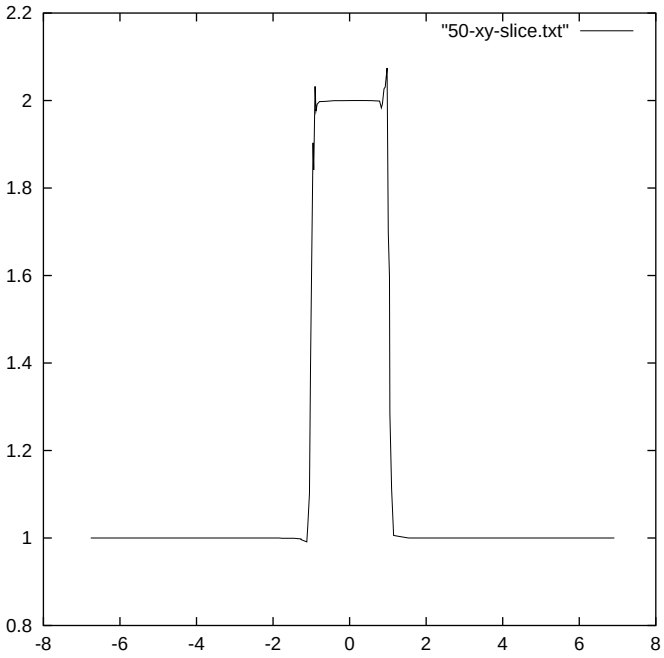
Výsledná síť a řez funkcí na oblasti $(-5, 5) \times (-5, 5)$ pro $\epsilon = 5$:



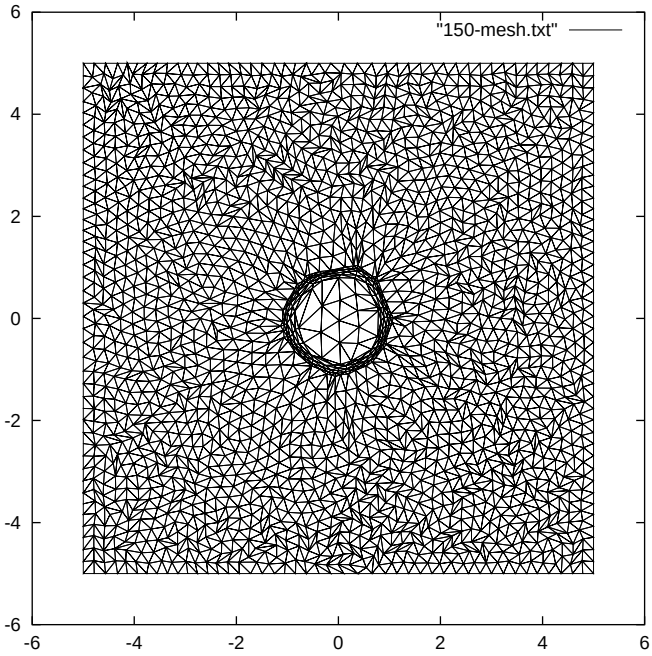


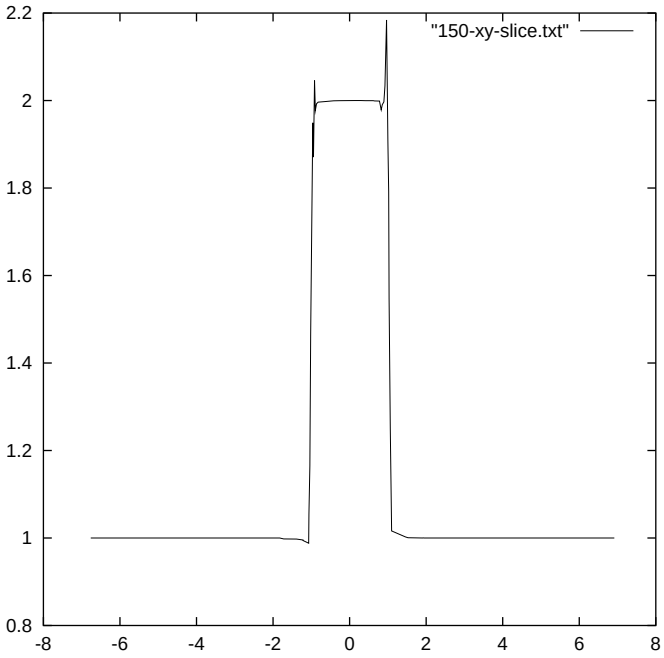
Výsledná síť a řez funkcí na oblasti $(-5, 5) \times (-5, 5)$ pro $\epsilon = 50$:



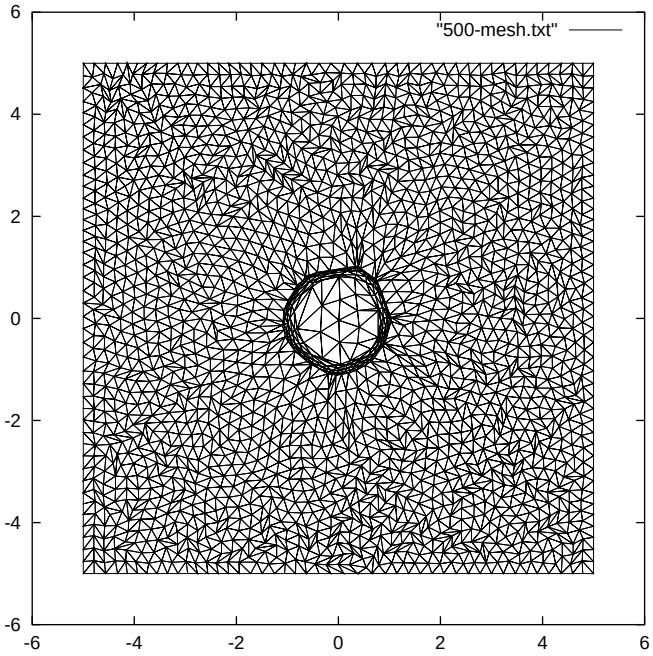


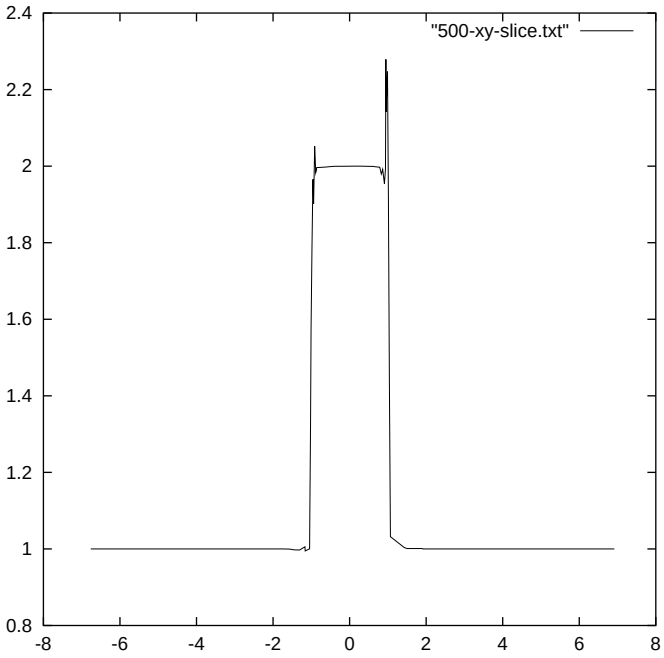
Výsledná síť a řez funkcí na oblasti $(-5, 5) \times (-5, 5)$ pro $\epsilon = 150$:





Výsledná síť a řez funkcí na oblasti $(-5, 5) \times (-5, 5)$ pro $\epsilon = 500$:





Kapitola 3

Úloha s nespojitou počáteční podmínkou

První testovací příklad se týká nespojité počáteční podmínky. Je to simulování dvourozměrného proudění. Počáteční podmínka se generuje v adreáři `/shock-generator/init-explosion` pomocí programu `explosion`, viz. kapitola 2.4, Generátor počátečních podmínek.

Cílem je řešit Eulerovy rovnice (1.1) - (1.3) ve 2D, tzn. pro $N = 2$. Výpočet bude probíhat v oblasti $Q_T = [-5, 5] \times [-5, 5] \times (0, T), T > 0$.

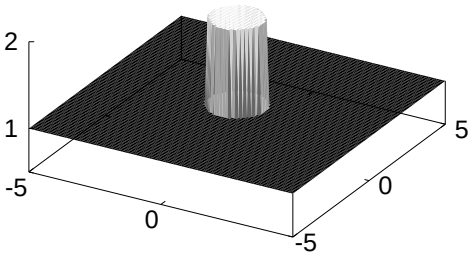
3.1 Počáteční a okrajová podmínka

Okrajová podmínka je daná prouděním zleva doprava.

Počáteční podmínka (viz obr. 3.1) je dána předpisem (1.2) s následující funkcí $\boldsymbol{w}^0$:

$$\boldsymbol{w}^0(x) = \begin{cases} (2,0,0,1)^T & \text{je-li } x_1^2 + x_2^2 \leq 1, \\ (1,0,0,0.5)^T & \text{je-li } x_1^2 + x_2^2 < 1. \end{cases} \tag{3.1}$$

Pracujeme ve dvou-dimenzionálním prostoru, takže vektor $\boldsymbol{w}$ má 4 složky: $\boldsymbol{w} = (\rho, \rho v_1, \rho v_2, E)^T$, kde ρ je hustota, v_1, v_2 složky rychlosti (tedy $\rho v_1, \rho v_1$ jsou složky hybnosti) a E značí energii.



Obrázek 3.1: Počáteční podmínka $\boldsymbol{w}^0(x)$ in $(-5,5) \times (-5,5)$

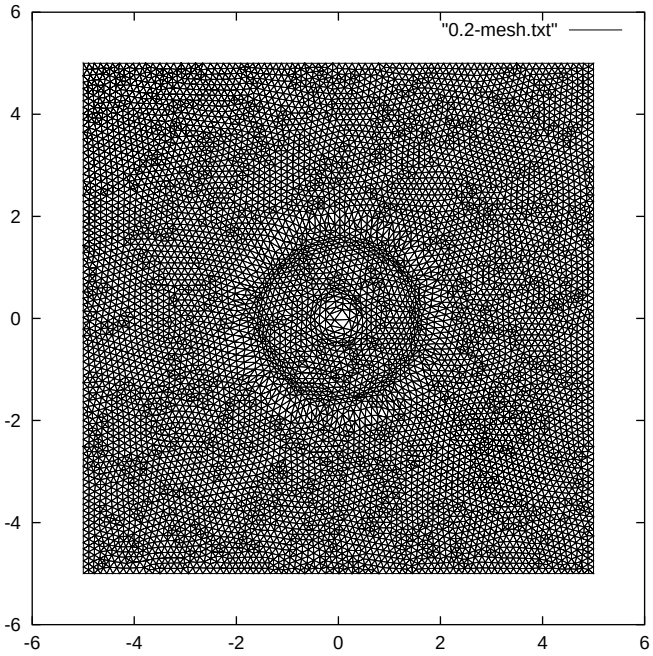
3.2 Výsledky prvního testovaného příkladu pomocí metody konečných objemů

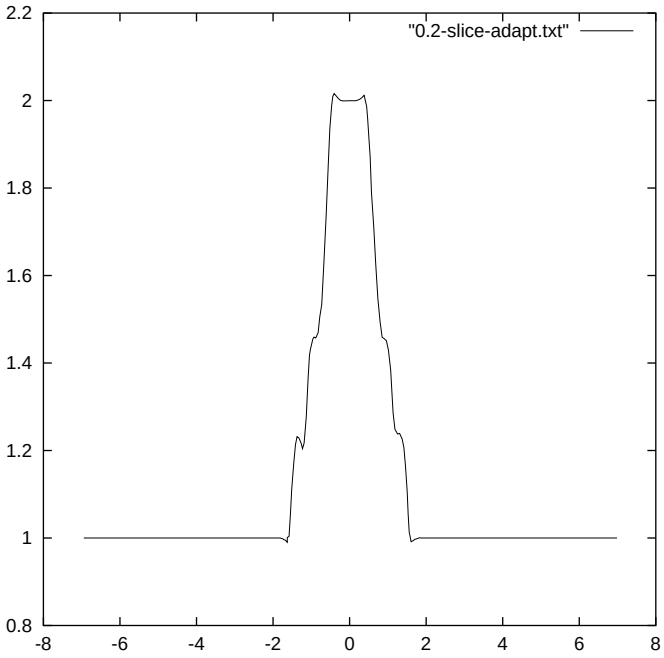
Výsledky testování tohoto příkladu předložím na větší síti s 16384 elementy. Experimentálně jsem zjistila, že menší síť s 4096 elementy není schopna nespojitosti v řešení dostatečně přesně rozlišit a síť odpovídajícím způsobem adaptovat. Toto pozorování předložím pomocí obrázku menší sítě a odpovídajícího řešení v čase 0.8, kde je možné se přesvědčit, že zatímco větší síť je jednotlivé nespojitosti schopna detekovat, rozlišit, a síť odpovídajícím způsobem adaptovat, menší síť nespojitosti neodlišuje a nejsme schopni rozeznat, kde se jednotlivé nespojitosti vyskytují.

Následující obrázky byly tedy - s jednou výjimkou - získány dříve popsaným programem na síti s 16384 elementy a následujícím nastavením:

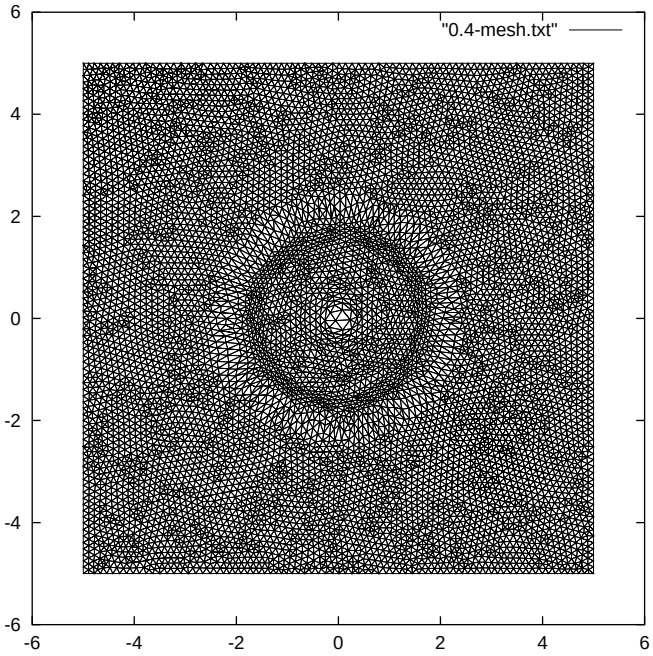
- /ZK 2.1/res/file.dt3
  - `time` - hodnota nastavena na 0.05
- /ZK 2.1/res/etc/paramet
  - `numel` - hodnota nastavena na 10000
  - `epsilon1` - hodnota nastavena na 50
  - `p` - hodnota nastavena na 30

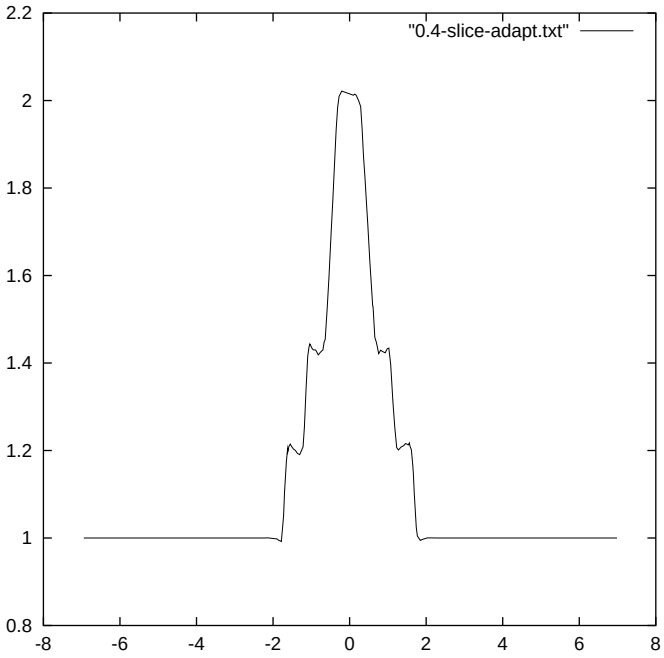
Síť a funkce v čase $t = 0.2$:



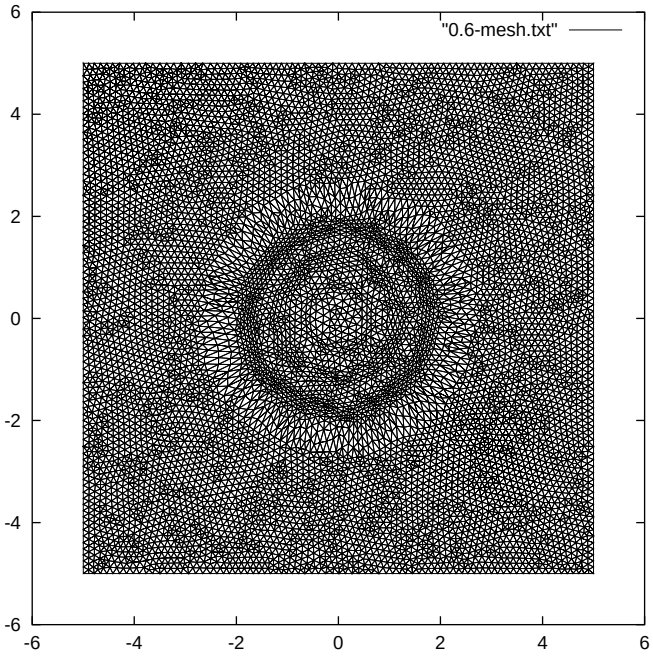


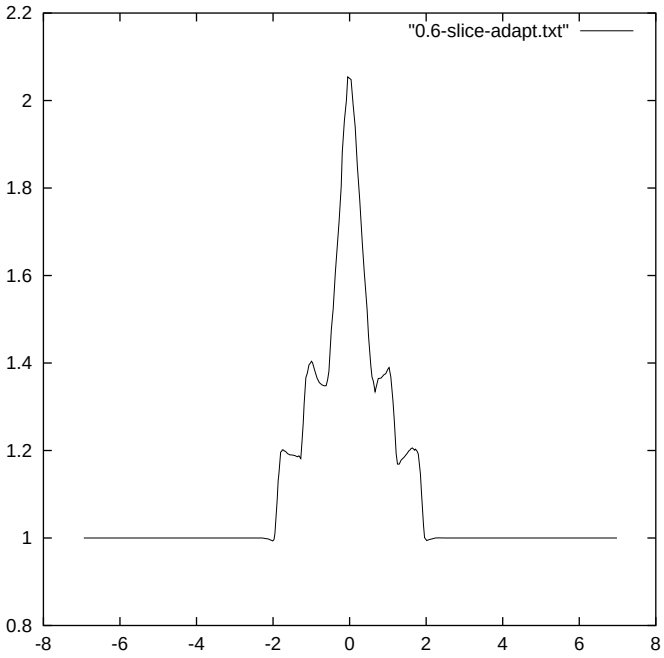
Síť a funkce v čase $t = 0.4$:



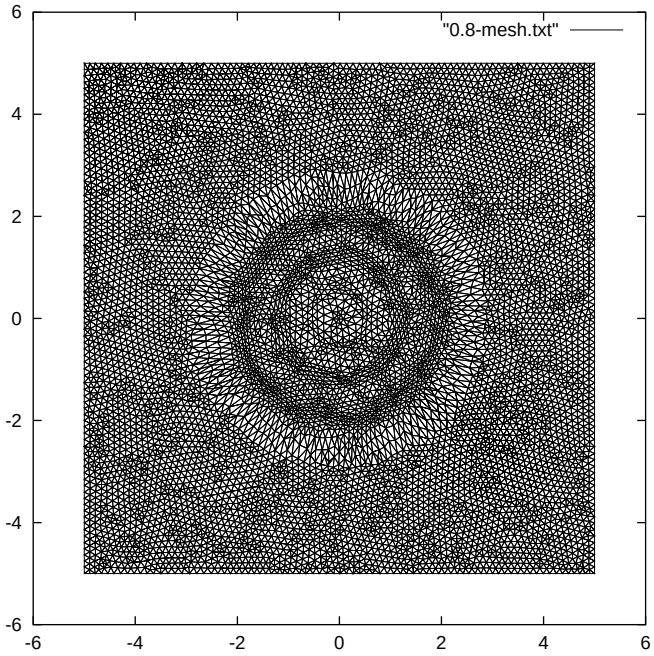


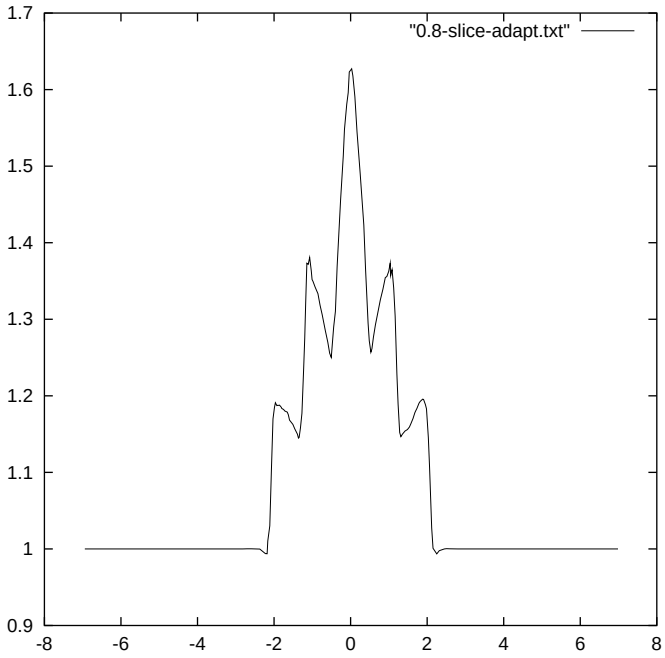
Síť a funkce v čase $t = 0.6$:



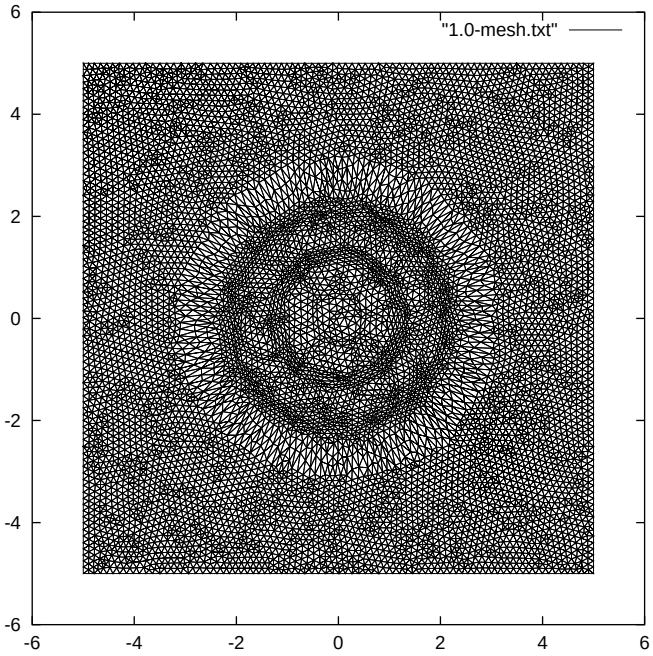


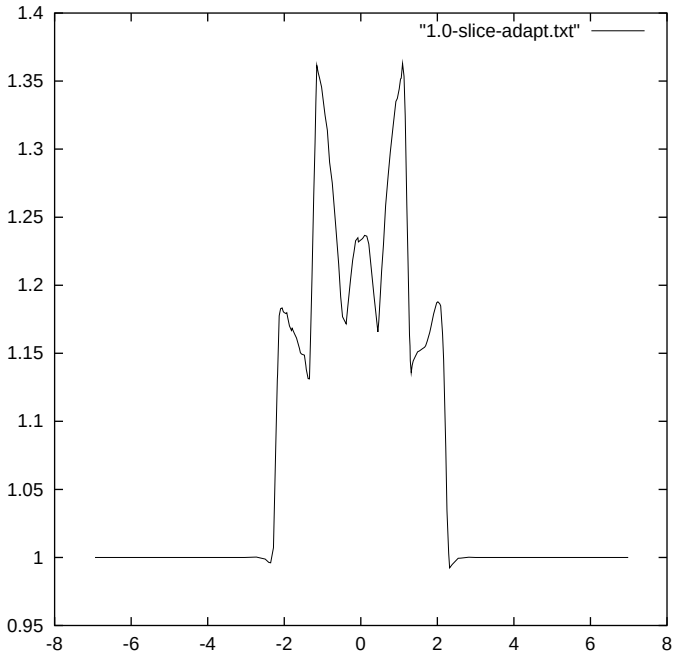
Síť a funkce v čase $t = 0.8$:



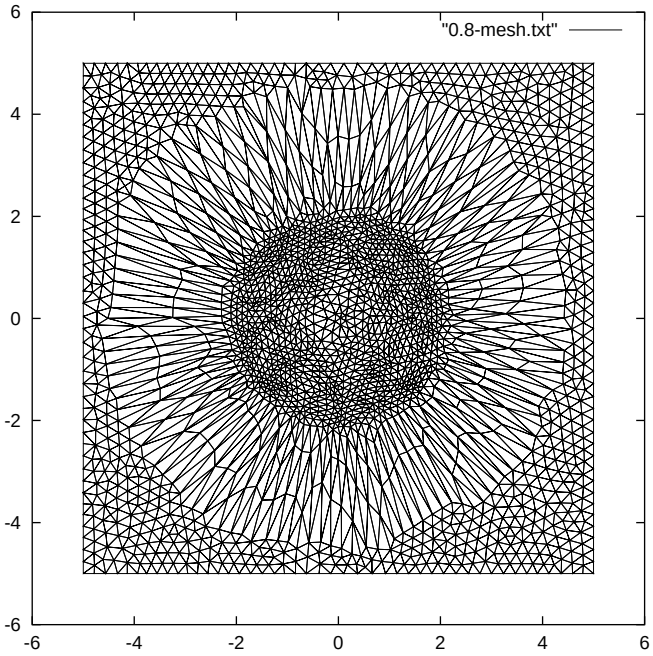


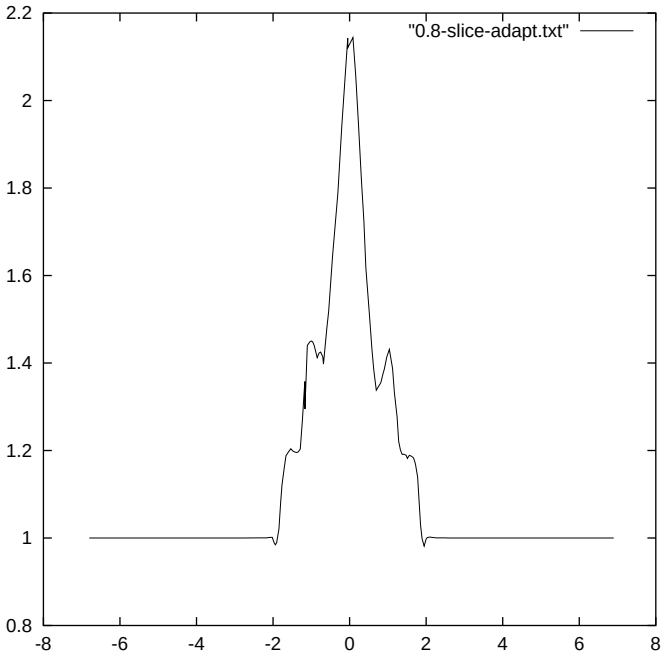
Síť a funkce v čase $t = 1.0$:





Menší síť a odpovídající řešení (pro srovnání) v čase $t = 0.8$:





Kapitola 4

Vortex evolution problem

Druhý příklad simuluje pohybující se vír na výpočetní oblasti. Počáteční podmínka se generuje v adreáři `/shock-generator/init-ve` pomocí programu `ve`. Popis je opět v kapitole 2.4, Generátor počátečních podmínek.

Cílem je řešit Eulerovy rovnice (popsáno v první kapitole) pro veličiny a funkce definované dále. Výpočet budeme provádět na oblasti
$Q_T = [-5, 5] \times [-5, 5] \times (0, T), T > 0$.

4.1 Popis problému

Budeme řešit Eulerovy rovnice

$$\frac{\partial \boldsymbol{w}}{\partial t} + \frac{\partial \boldsymbol{f}_1(\boldsymbol{w})}{\partial x_1} + \frac{\partial \boldsymbol{f}_2(\boldsymbol{w})}{\partial x_2} = 0 \qquad \text{v oblasti } Q_T. \tag{4.1}$$

Opět máme vektor $\boldsymbol{w} = (\rho, \rho v_1, \rho v_2, E)^T \in I\!R^4$.

Dále se definují čtyř-dimenzionální zobrazení $\boldsymbol{f}_1$ a $\boldsymbol{f}_2$ následujícími předpisy:

$$\boldsymbol{f}_1(\boldsymbol{w}) = (\rho v_1, \rho v_1^2 + p, \rho v_1 v_2, v_1(E + p)),$$
$$\boldsymbol{f}_2(\boldsymbol{w}) = (\rho v_2, \rho v_1 v_2, \rho v_2^2 + p, v_2(E + p)),$$

kde p značí tlak.

4.2 Počáteční a okrajové podmínky

Okrajová podmínka je dána periodicky, a to v obou směrech.

Počáteční podmínka je dána následujícími předpisy.

- Hustota: $\rho(x, y, 0) = T^{\left(\frac{1}{\gamma-1}\right)}$
- Rychlost: $(v_1, v_2)(x, y, 0) = (1, 1) + \frac{\epsilon}{2\pi} e^{0.5(1-r^2)}(-\bar{y}, \bar{x})$
- Energie: $E(x, y, 0) = \frac{p}{\gamma-1} + \frac{1}{2}p(v_1^2 + v_2^2)$
- Tlak: $p(x, y, 0) = \rho T$
- Teplota: $T(x, y, 0) = 1 - \frac{(\gamma-1)\epsilon^2}{8\gamma\pi^2} e^{1-r^2}$,

kde $r^2 = \bar{x}^2 + \bar{y}^2$ a $\bar{x} = x - S_x$, $\bar{y} = y - S_y$, označíme-li S_x a S_y počáteční pozici centra víru, ϵ je šířka víru a $\gamma = 1.4$ je koeficient specifického tepla vzduchu.

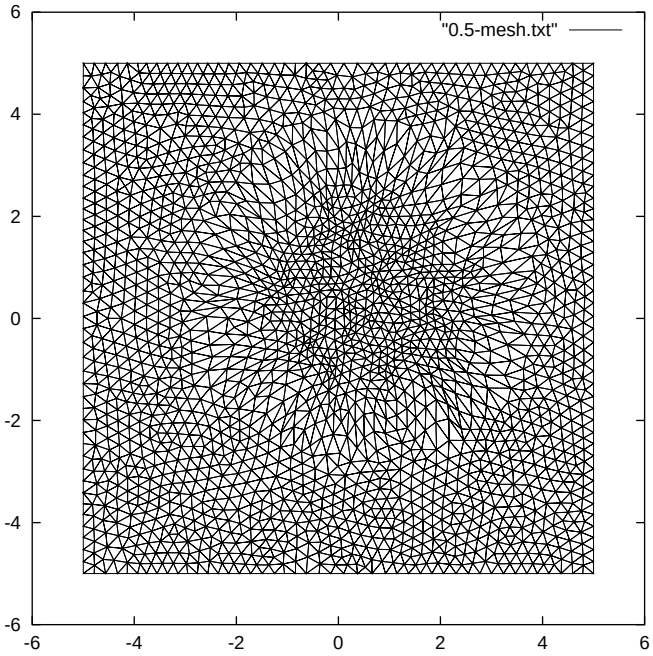
4.3 Výsledky druhého testovaného příkladu pomocí metody konečných objemů

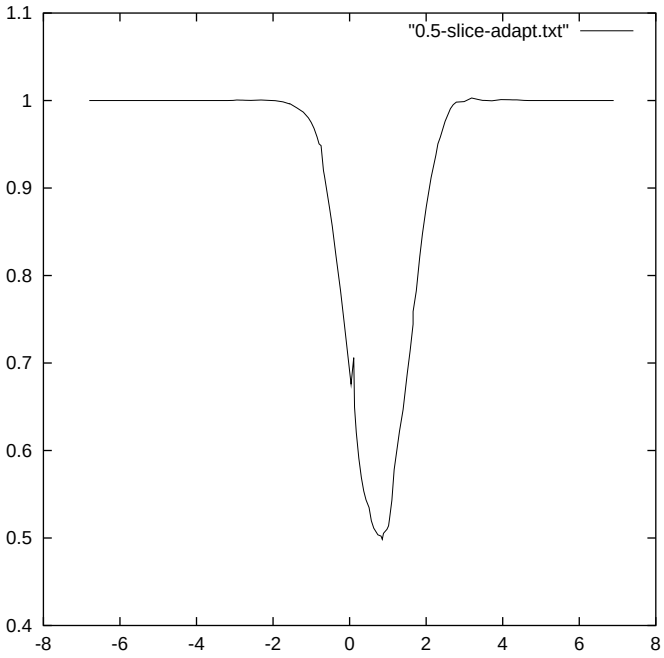
Na rozdíl od minulého příkladu zde předložím výsledky získané při výpočtu na menší síti s 4096 elementy. Hlavním důvodem je to, že na této síti jsou změny sítě a pohyb víru lépe vidět. Je to proto, že tato funkce neobsahuje nespojitosti. Její gradient je relativně malý a větší síť s 16384 elementy je pro tuto funkci 'dostatečně dobrá' a změny sítě se ani v okolí největšího gradientu neprojeví nijak výrazně. Lepší vizualizace na větší síti jsem se snažila dosáhnout pomocí změn počátečních parametrů, povedlo se to ovšem jen částečně a za cenu dlouho trvajících výpočtů. Bylo totiž nutné postupovat po malých časových krocích (0.02) a takovýto výpočet s větší sítí je časově náročný (při výpočtech na PC se 4 procesory typu Intel(R) Xeon(TM) CPU 3.20GHz a 2GB ram byla délka výpočtu v řádu hodin). Výsledek na větší síti přesto pro úplnost uvedu.

Následující výsledky tedy byly - opět s jednou výjimkou - získány na menší síti s 4096 elementy a následujícím nastavením:

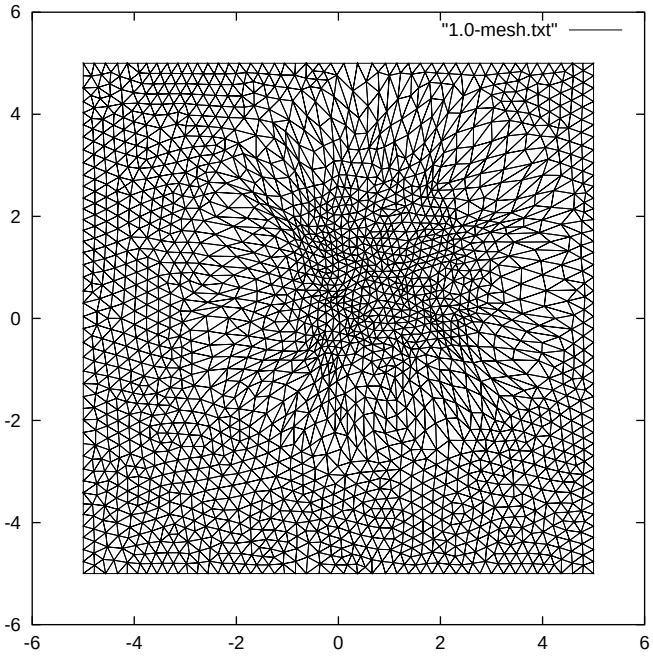
- /ZK 2.1/res/file.dt3
  - `time` - hodnota nastavena na 0.02
- /ZK 2.1/res/etc/paramet
  - `numel` - hodnota nastavena na 1000
  - `epsilon1` - hodnota nastavena na 50
  - `p` - hodnota nastavena na 30

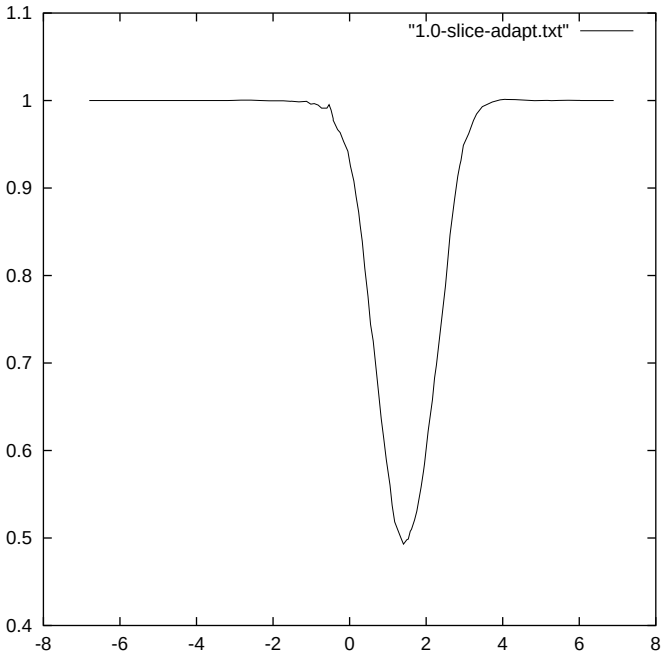
Síť a funkce v čase $t = 0.5$:



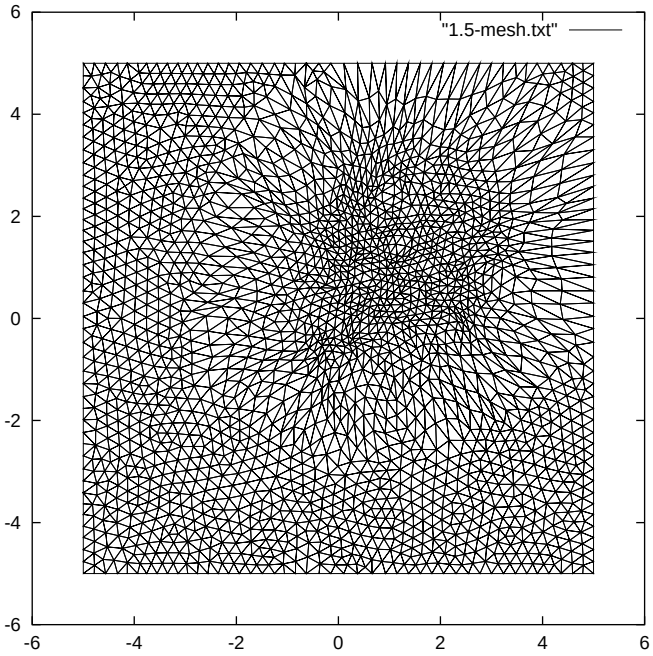


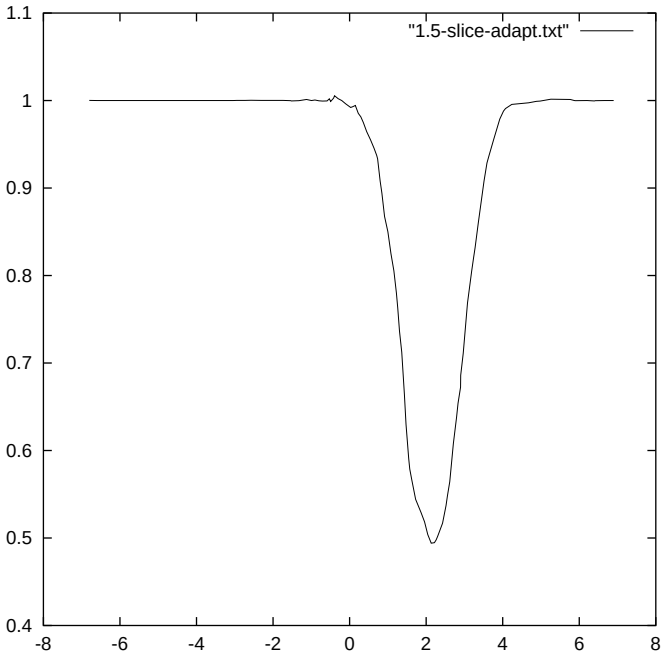
Síť a funkce v čase $t = 1.0$:



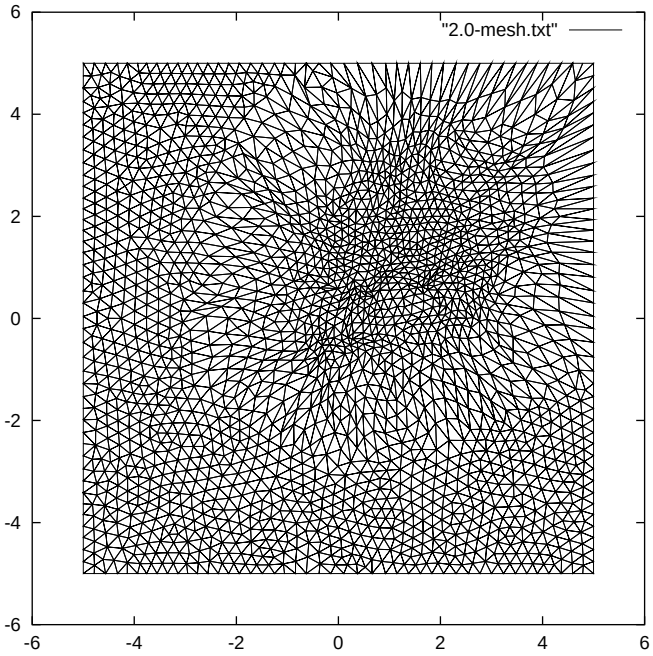


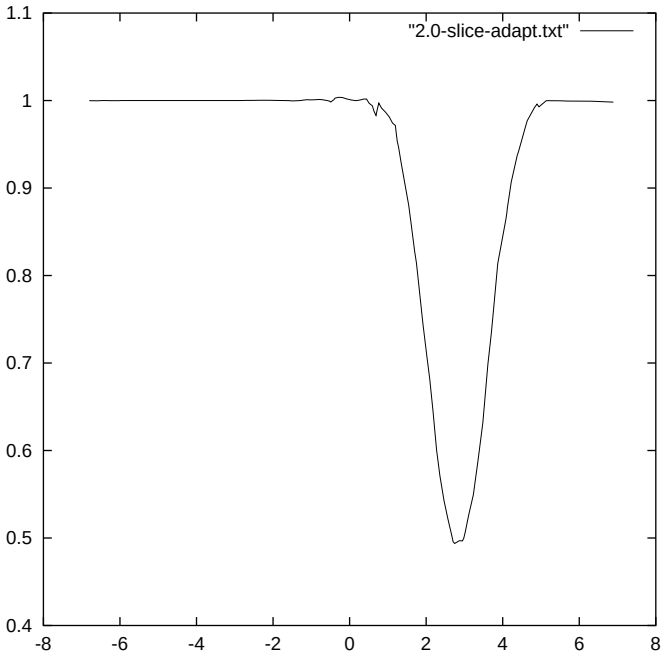
Síť a funkce v čase $t = 1.5$:



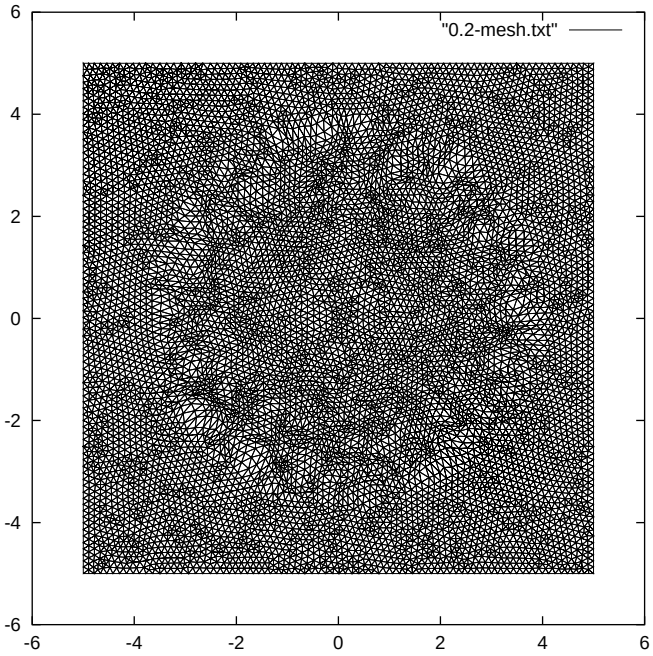


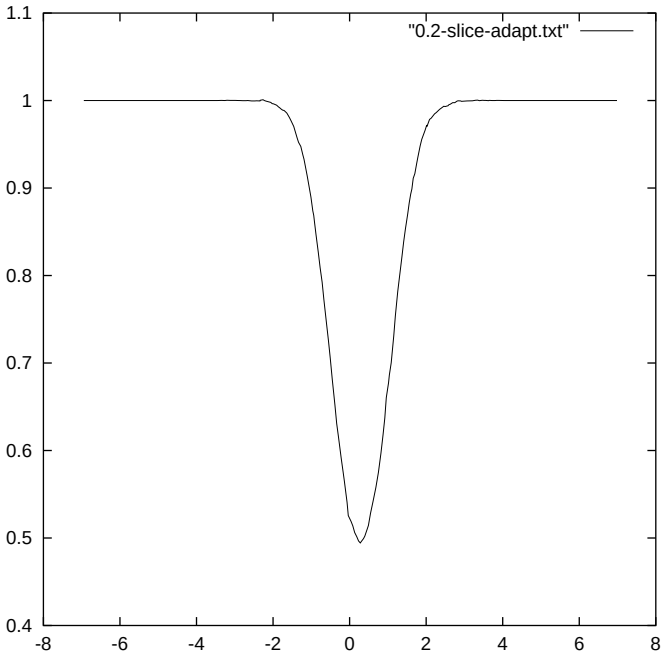
Síť a funkce v čase $t = 2.0$:





Větší síť a odpovídající řešení v čase $t = 0.2$:





Příloha A

Popis CD

Na přiloženém CD je uložen program popsaný v této práci. Program je určen pro operační systém Linux.

Struktura CD:

- `Priklady` - adresář obsahuje podadresáře `explBig`, `explSmall` a `veSmall`. V každém podadresáři je spustitelný program včetně vhodného počátečního nastavení.
  - `explBig` - připravené na výpočet testovacího příkladu *Explosion* na větší síti
  - `explSmall` - připravené na výpočet testovacího příkladu *Explosion* na menší síti
  - `veSmall` - připravené na výpočet testovacího příkladu *Vortex-evolution problem* na větší síti
- `shock-generator` - adresář obsahuje podadresáře `init-expl` a `init-ve` pro generování počátečních podmínek testovaných funkcí, popis viz kapitola 2.4, Generátor počátečních podmínek

- `ZK 2.1` - tento adresář obsahuje potřebné soubory pro spuštění programu. Vstupní parametry musí uživatel sám zadat. Adresář je rozdělen na 3 podadresáře:
  - `res` - adresář s uloženým programem, spouští se postupem popsaným v kapitole 2.2, Překlad programů, spuštění a výpočet
  - `src` - adresář obsahuje podadresáře se zdrojovými soubory k potřebným programům, překlad a použití opět popsáno v kapitole 2.2, Překlad programů, spuštění a výpočet
    Je možné, že po překladu nebudou programy `ader2` a `megen` z podaresáře `resic` fungovat. V tomto případě se obraťte na kubera@sci.ujep.cz.
  - triang - pomocný soubor, obsahuje dvě používané triangulace - menší se 4096 elementy a větší s 16384 elementy

Literatura

[1] Dolejší V., Felcman J.: *Anisotropic mesh adaptation for numerical solution of boundary value problems*, Numerical Methods for Partial Differential Equations (2003), 576–608.

[2] Felcman J., Kubera P.: *Mesh adaptation for a time marching procedure*, in: Bermúdez A. (editor): *Numerical Mathematics and Advanced Applications*, ENUMATH 2005, Springer, Berlin, 2005.