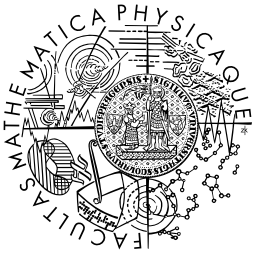


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Jakub Rychlík

Numerické řešení lineárních parciálních diferenciálních rovnic prvního řádu metodou charakteristik

Katedra numerické matematiky

Vedoucí bakalářské práce: doc. Mgr. Petr Knobloch, Dr., DSc.

Studijní program: Obecná matematika

Praha 2024

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ....

Podpis autora

Rád bych poděkoval svému vedoucímu bakalářské práce panu doc. Mgr. Petr Knobloch, Dr., DSc. za vstřícnost, rady a čas, který mi věnoval. Dále bych rád poděkoval studentce MFF Bc. Lucii Vomelové za pomoc a rady ohledně syntaxe a úpravy kódu v programovacím jazyku Matlab.

Název práce: Numerické řešení lineárních parciálních diferenciálních rovnic prvního řádu metodou charakteristik

Autor: Jakub Rychlík

Katedra: Katedra numerické matematiky

Vedoucí bakalářské práce: doc. Mgr. Petr Knobloch, Dr., DSc., Katedra numerické matematiky

Abstrakt: Hlavním tématem práce je navržení numerického algoritmu, který bude řešit lineární parciální diferenciální rovnice 1. řádu pomocí metody charakteristik. Shrneme princip teoretických výpočtů využívajících metodu charakteristik, zkonstruujeme numerický algoritmus a aplikujeme ho s pomocí programovacího jazyku Matlab. Budeme využívat numerické metody jako Rungeovu-Kuttovu metodu pro řešení obyčejných diferenciálních rovnic, složené lichoběžníkové pravidlo pro aproximaci integrálů nebo také barycentrickou interpolaci pro aproximaci hodnoty funkce. Nakonec použijeme algoritmus pro konkrétní příklady, zanalyzujeme je a vykreslíme aproximované řešení v grafickém softwaru Paraview.

Klíčová slova: parciální diferenciální rovnice prvního řádu, metoda charakteristik, Rungeova-Kuttova metoda

Title: Numerical solution of linear first order partial differential equations using the method of characteristics

Author: Jakub Rychlík

Department: Department of numerical mathematics

Supervisor: doc. Mgr. Petr Knobloch, Dr., DSc., Department of numerical mathematics

Abstract: The main topic of the thesis is the design of a numerical algorithm which can solve 1st order linear partial differential equations using the method of characteristics. We will summarize the principle of theoretical calculations using the method of characteristics, construct a numerical algorithm and apply it with the help of the Matlab programming language. We will use numerical methods such as the Runge-Kutta method for solving ordinary differential equations, composite trapezoidal rule for approximation of integrals, or barycentric interpolation for an approximation of function values. Finally, we will use the algorithm on specific examples, analyze them and plot the approximate solutions in Paraview graphics software.

Keywords: first order partial differential equations, method of characteristics, Runge-Kutta method

Obsah

Seznam použitého značení

$\triangle[\mathbf{a}, \mathbf{b}, \mathbf{c}]$ — trojúhelník s vrcholy a, b, c

$\mathbb{N}_\mathbf{0}$ — $\mathbb{N} \cup \{0\}$

$\mathbf{u_x}$ — $\frac{\partial u}{\partial x}$

MoC — Metoda charakteristik (Method of characteristics)

ODR — Obyčejná diferenciální rovnice

PDR — Parciální diferenciální rovnice

Úvod

Pomocí parciálních diferenciálních rovnic jsme schopni popsat mnoho fyzikálních, chemických či biologických jevů. Existují metody, pomocí kterých jsme schopni přesně vyřešit určité typy těchto rovnic. Nicméně v některých případech musíme řešení hledat numerickými metodami. V této práci se zaměříme na numerické řešení *lineárních parciálních diferenciálních rovnic 1. řádu* pomocí metody charakteristik na otevřených množinách $\Omega \subset \mathbb{R}^2$. *Metoda charakteristik* funguje na bázi redukce parciální diferenciální rovnice (PDR) na systém obyčejných diferenciálních rovnic (ODR), jehož vyřešením získáme tzv. *charakteristické křivky* nebo *charakteristiky*. V 1. kapitole se dozvíme, že řešení původní *zjednodušené PDR* jsme schopni vyjářit díky těmto křivkám pomocí *okrajové podmínky*, což je funkce definovaná na *vstupní hranici* $\partial\Omega^- \subset \partial\Omega$.

Cílem práce je navrhnout a naprogramovat vhodný numerický algoritmus, který bude schopen aproximovat řešení zadané PDR co nejpřesněji a na co nejobecnější otevřené množině $\Omega \subset \mathbb{R}^2$ se zadanou okrajovou podmínkou na vstupní hranici. Ukážeme si:

- Jak řešit úlohu na množinách definovaných jako vnitřky uzavřených křivek.
- Jak aproximovat vstupní hranici $\partial\Omega^-$.
- Jak nalézt aproximaci charakteristik pomocí Rungeovy-Kuttovy metody 4. řádu pro numerické řešení ODR.
- Jak nalézt vhodné jemné pokrytí množiny Ω trojúhleníkovou sítí s vrcholy ležícími na charakteristikách.
- Jak nalézt aproximaci řešení PDR s využitím barycentrické interpolace.

V první kapitole zavedeme potřebné pojmy a vysvětlíme podrobněji, jak funguje metoda charakteristik. V následující kapitole popíšeme, jak funguje řešící algoritmus. V poslední kapitole předvedeme, jak program funguje na různých příkladech a ilustrujeme si výsledky na obrázcích.

1 Základní definice a vlastnosti metody charakteristik

1.1 Zavedení pojmů

Než se pustíme do numerického řešení parciálních diferenciálních rovnic, musíme nejdříve zadefinovat, na jaké typy rovnic budeme numerické výpočty aplikovat. Teorie níže je inspirovaná z [1] (kapitola 2.1 a část kapitoly 2.2) a z [2] (kapitola 1.3).

Definice 1 (Lineární parciální diferenciální rovnice 1. řádu v $\mathbb{R}^2$)**.** *Nechť $\Omega \subseteq \mathbb{R}^2$ je neprázdná otevřená množina.* ***Lineární parciální diferenciální rovnicí 1. řádu*** *pro neznámou funkci $u : \Omega \to \mathbb{R}$ nazveme výraz tvaru:*

$$b_1(w)u_x(w) + b_2(w)u_y(w) + c(w)u(w) = f(w), \qquad \text{(LPDR)}$$

kde b_1, b_2, c, f jsou reálné funkce definované na Ω a $w = (x, y) \in \Omega$.

Nyní bychom rádi řešili rovnici (LPDR). Říkáme, že úloha s danou rovnicí je korektně zadaná, jestliže má řešení, které je jednoznačné a spojitě závislé na datech z Ω. Poslední vlastnost říká, že je třeba, aby se při malých změnách v datech vstupujících do LPDR změnilo málo i řešení. Pro splnění vlastnosti jednoznačnosti zavedeme tzv. okrajovou podmínku.

Definice 2 (Okrajová podmínka pro LPDR)**.** *Nechť $H \subseteq \partial\Omega$ je část hranice $\Omega \subseteq \mathbb{R}^2$. Funkci $g : H \to \mathbb{R}$ nazveme* ***okrajovou podmínkou*** (LPDR) *na H, jestliže $g \in C^1(H, \mathbb{R})$ a g řeší LPDR na množině H.*

Poznámka.

- V případě, že $\Omega = \mathbb{R}^2$, bereme H jako nějakou nadrovinu v $\mathbb{R}^2$.
- Rovnici (LPDR) nazveme Zjednodušenou LPDR, jestliže $f \equiv c \equiv 0$.
- Úloha složená z PDR a její okrajové podmínky se obvykle zadává pomocí složených závorek (viz Příklady níže).

Příklad (LPDR s okrajovou podmínkou).

1. $\begin{cases} u_x + 2xu_y + 2xu = 0, & \text{v } \Omega = \mathbb{R}^2 \\ u(x, y) = x^2, & \text{pro } (x, y) \in H = \mathbb{R} \times \{0\}. \end{cases}$

2. $\begin{cases} u_x + 4u_y = x, & \text{v } \Omega = \mathbb{R}^2 \\ u(x, y) = \cos(x), & \text{pro } (x, y) \in H = \mathbb{R} \times \{0\}. \end{cases}$

3. $\begin{cases} yu_x - xu_y = 0, & \text{v } \Omega = \{(x,y) \in \mathbb{R}^2 : x^2 + y^2 < 4;\ y > 0\}, \\ u(x,y) = \sin(x), & \text{pro } (x,y) \in [-2,0] \times \{0\}. \end{cases}$

   V této úloze máme zadanou množinu Ω omezenou. Jedná se o vnitřek půlkružnice s poloměrem 2. S úlohou se ještě setkáme v kapitole 3 a vyřešíme ji numericky.

1.2 Metoda charakteristik pro zjednodušenou LPDR

V následující podkapitole popíšeme, jak funguje metoda charakteristik (MoC). Jedná se o metodu, která může sloužit nejen k řešení PDR 1. řádu v $\mathbb{R}^n$, ale po vhodné modifikaci i k řešení jiných typů PDR. Budeme zde vycházet převážně z [3] (kapitola 10.1).
Uvažme zadanou úlohu se zjednodušenou LPDR pro otevřenou množinu $\Omega \subseteq \mathbb{R}^2$:

$$\begin{cases} b_1 u_x + b_2 u_y = 0, & \text{v } \Omega \subseteq \mathbb{R}^2, \\ u = g, & \text{na } H \subseteq \partial\Omega \end{cases} \tag{1}$$

Předpokládejme navíc, že b_1, b_2 jsou spojité na $\bar{\Omega}$.

Ať $u \in C^1(\Omega)$ je libovolná funkce, která zatím nemusí nutně řešit úlohu (1). Pokusíme se najít vhodné křivky s hodnotami v Ω, pomocí nichž parametrizujeme funkci u. Zaveďme proto charakteristické křivky:

Definice 3 (Charakteristické křivky)**.** *Nechť $I = [\alpha, \beta] \subset \mathbb{R}$. Regulární křivku $\phi : I \to \bar{\Omega}$, $\phi = (\phi_1, \phi_2)^T$, nazveme **charakteristickou křivkou** (zkráceně **charakteristikou**) úlohy (1), pokud řeší následující systém ODR:*

$$\frac{d}{ds}\begin{pmatrix}\phi_1(s)\\ \phi_2(s)\end{pmatrix} = \begin{pmatrix} b_1(\phi(s)) \\ b_2(\phi(s))\end{pmatrix},\ s \in I,$$

*s počáteční podmínkou $\phi(\alpha) = a$ (resp. $\phi(\beta) = a$) pro nějaké $a \in H$. Tomuto systému ODR se říká **charakteristický systém**.*

Definice je korektní, protože vektor pravých stran je spojitý a dle Peanovy věty existuje lokálně řešení pro každou počáteční podmínku $a \in H$. Je užitečné si uvědomit, že volbou počáteční podmínky ukotvíme křivky na části hranice Ω, tedy H, a jsme schopni regulovat, jaké je vybíráme (dle volby $a \in H$) a kolik jich vybíráme. Počátečních podmínek může být nekonečně mnoho, protože H je zpravidla nějaká nadrovina v $\mathbb{R}^2$.

Také můžeme říci, že H je část hranice Ω, kterou charakteristiky vstupují dovnitř do Ω. H by měla být zadána tak, aby opravdu reprezentovala celou podmnožinu $\partial\Omega$, kterou křivky vstupují do Ω (vysvětleno na příkladě 1.2). To vede k následující definici:

Definice 4. *Nechť $n : \partial\Omega \to \mathbb{R}^2$ je vnější normálový vektor k $\partial\Omega$. Označme si $n = (n_1, n_2)$ a $b = (b_1, b_2)$, kde b_1, b_2 jsou funkce zadané v úloze (1). **Vstupní hranicí** rozumíme množinu*

$$\{(x,y) \in \partial\Omega : b(x,y) \cdot n(x,y) < 0\},$$

kde operace „·“ mezi b a n je standardní skalární součin. Tuto množinu značíme $\partial\Omega^-$.

Nadále už budeme místo H používat značení $\partial\Omega^-$. Pro intuitivnější představu dále ukážeme, jak vypadají charakteristiky na konkrétním příkladě.

Příklad (Transportní rovnice 1).

$$\begin{cases} u_x + u_y = 0, & \text{na } \{(x,y) \in \mathbb{R}^2 : (x,y) \in (0,\pi)^2;\ x > y\}, \\ u(x,y) = \sin(x), & \text{na } [0,\pi] \times \{0\}. \end{cases}$$

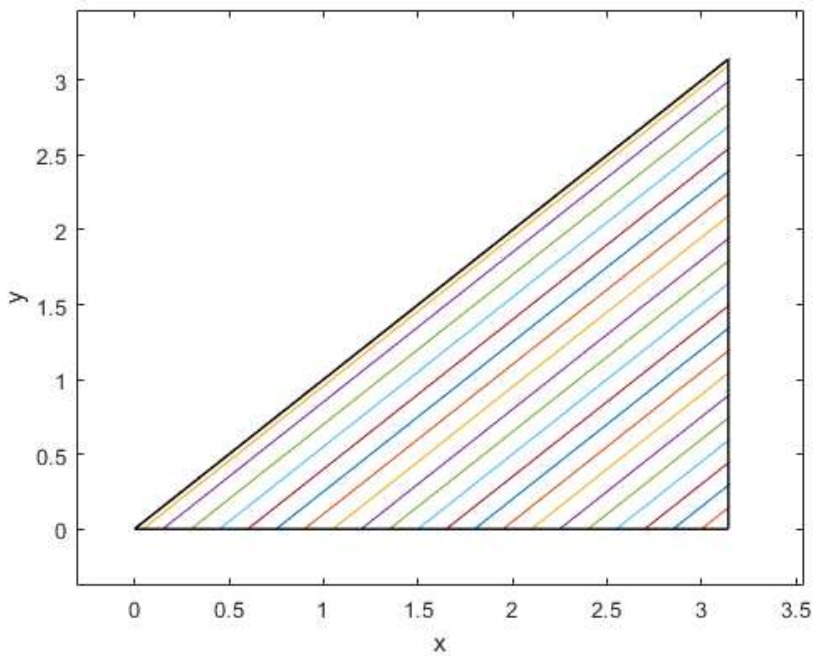
Zjednodušená LPDR v této rovnici se nazývá transportní rovnice.
Platí, že $\Omega = \{(x,y) \in \mathbb{R}^2 : (x,y) \in (0,\pi)^2;\ x > y\}$, $\partial\Omega^- = [0,\pi] \times \{0\}$ a $g(x,y) = \sin(x)$. Ω je vnitřek pravoúhlého trojúhelníku s odvěsnami dlouhými π a přeponou ležící na přímce $y = x$. Vstupní hranicí je podstava trojúhelníku. Charakteristiky spočteme z charakteristického systému

$$\frac{d}{ds}\begin{pmatrix}\phi_1(s)\\ \phi_2(s)\end{pmatrix} = \begin{pmatrix}1\\1\end{pmatrix}, s \in I.$$

Dostáváme

$$\begin{pmatrix}\phi_1(s)\\ \phi_2(s)\end{pmatrix} = \begin{pmatrix}s+a\\ s+b\end{pmatrix}, s \in I, a, b \in \mathbb{R}.$$

Na obrázku vidíme několik charakteristik (barevně) a hranici množiny Ω (černě). Ve skutečnosti jich je ale nekonečně mnoho a pokrývají celý trojúhelník.



Obrázek 1.1 Vybrané charakteristiky uvnitř Ω

Proč je důležité správně zadat vstupní hranici? Pokud bychom v příkladě zadali množinu Ω jako čtverec $(0,\pi)^2$, ale nezměnili bychom vstupní hranici, pak nám charakteristiky pokryjí jen půlku čtverce. Jak uvidíme níže, pro body z nepokryté části bychom nebyli schopni spočítat řešení. Spravně by se tedy vstupní hranice měla změnit na $([0,\pi] \times \{0\}) \cup (\{0\} \times [0,\pi])$

Pojďme se v následujícím lemmatu podívat na to, jak se chová derivace u podél charakteristik. Inspirováno je lemmatem z neoficiálních skript k bakalářskému předmětu Úvod do parciálních diferenciálních rovnic ([4], podkapitola 2.2 - Cauchyova úloha pro kvazilineární PDR 1. řádu, Lemma 2.3; převzato ke dni 14.7.2024)

Lemma 1 (O konstantnosti řešení podél charakteristik)**.**
Nechť $u \in C^1(\bar{\Omega})$ je reálná funkce.

1) *Nechť $\phi : I \to \bar{\Omega}$ je charakteristická křivka úlohy* (1)*. Je-li u konstantní podél křivky ϕ a splňuje $u = g$ na H, pak u řeší úlohu (1) v bodech $\phi(s), s \in I$.*

2) *Řeší-li u úlohu (1), pak je konstantní podél každé charakteristické křivky úlohy (1).*

Důkaz.

1) Mějme charakteristiku ϕ příslušející úloze (1) a $u \in C^1(\bar{\Omega})$ reálnou funkci, pro kterou platí
$$0 = \frac{d}{ds} u(\phi(s)).$$
S použitím řetízkového pravidla a definice charakteristiky dostáváme
$$\begin{aligned} 0 = \frac{d}{ds} u(\phi(s)) &= u_x(\phi(s)) \frac{d}{ds}\phi_1(s) + u_y(\phi(s)) \frac{d}{ds}\phi_2(s) \\ &\overset{Def\,3}{=} u_x(\phi(s)) b_1(\phi(s)) + u_y(\phi(s)) b_2(\phi(s)). \end{aligned}$$
Z definice charakteristických křivek také platí $Im(\phi) \subset \bar{\Omega}$, z čehož vyplývá, že u řeší zjednodušenou LPDR v každém bodě $\phi(s) \in \Omega$, $s \in I$. Navíc $\exists \alpha \in I : \phi(\alpha) \in H$, protože tak byla zadána počáteční podmínka při výpočtu ϕ. Tedy $u(\phi(\alpha)) = g$. Tzn. u řeší úlohu (1) v každém bodě $\phi(s), s \in I$.

2) Volíme opačný postupu jako v důkazu *1)*, ale ϕ volíme jako libovolnou charakteristiku. Z tohoto důvodu taky tvrzení *1)* a *2)* nejsou ekvivalentní.

□

Z odstavce pod obrázkem 1.1 a bodu 1) v lemmatu plyne jednoduchý důsledek.

Důsledek. Pokrývají-li charakteristické křivky celou množinu Ω a $u \in C^1(\bar{\Omega})$ je konstantní podél každé z křivek, pak u řeší úlohu (1) na celé množině Ω.

Toto je silný nástroj k počítání zjednodušených LPDR. Jsme totiž schopni převést problém hledání řešení nějaké PDR na problém hledání řešení systému ODR 1. řádu, což může být často jednodušší. Postup výpočtu si opět ukážeme na transportní rovnici.

Příklad (Transportní rovnice 2)*.*
Z příkladu 1.2 už máme nalezeny charakteristické křivky ve tvaru:
$$\phi(s) = \begin{pmatrix} \phi_1(s) \\ \phi_2(s) \end{pmatrix} = \begin{pmatrix} s + a \\ s + b \end{pmatrix}, s \in I, a, b \in \mathbb{R}.$$

Dle lemmatu hledáme funkci u takovou, která bude podél přímek konstantní. Na množině $\partial\Omega^-$ však máme zadanou známou funkci g a charakteristické křivky

protínají vstupní hranici. Jako řešení úlohy se tedy nabízí vhodně posunutá funkce g.

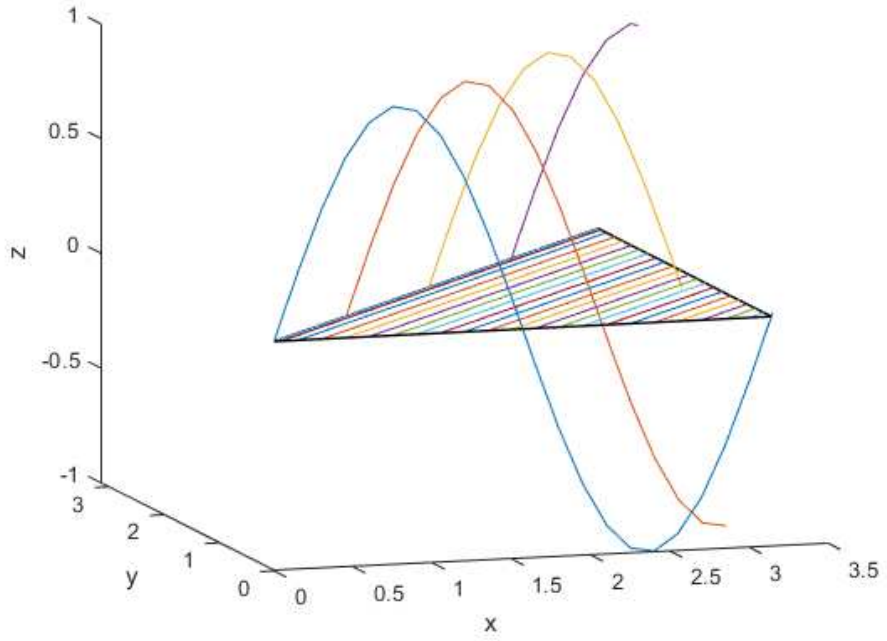
Zafixujme libovolný bod $(x, y) \in \Omega$. Není těžké si rozmyslet, že křivka, pro níž zvolíme konstanty $a := x$ a $b := y$, prochází bodem (x, y) v $s = 0$. Dostáváme křivku

$$\begin{pmatrix} \phi_1(s) \\ \phi_2(s) \end{pmatrix} = \begin{pmatrix} s + x \\ s + y \end{pmatrix}, s \in I.$$

ϕ protne osu x pro $s = -y$. V následujícím výpočtu využíváme toho, že u musí být konstantní podél ϕ, jinak by nemohlo být řešením dle bodu *2)* v lemmatu 1.

$$u(x, y) = u(\phi(0)) = u(\phi(-y)) = u(x - y, 0) = \sin(x - y)$$

Poslední „=“ je vynucené a vyplývá z faktu, že kdyby u nebylo rovno okrajové podmínce na $\partial\Omega^-$, tak nemůže být řešením úlohy (1). Jelikož bod (x, y) byl libovolný, dostáváme řešení $u(x, y) = \sin(x - y)$, $(x, y) \in \Omega$.
Na obrázku vidíme vybrané řezy řešením této úlohy. Ve skutečnosti řešením bude plocha složená z nekonečně mnoha takovýchto posunutých sinusoid. Zároveň můžeme vidět, proč se této rovnici říká transportní.



Obrázek 1.2 Řezy řešením transportní rovnice s okrajovou podmínkou sin

1.3 Metoda charakteristik pro LPDR

Teď už můžeme postup MoC zobecnit pro (LPDR). Mějme úlohu pro otevřenou množinu $\Omega \subseteq \mathbb{R}^2$:

$$\begin{cases} b_1 u_x + b_2 u_y + cu = f, & \text{v } \Omega \subseteq \mathbb{R}^2, \\ u = g, & \text{na } \partial\Omega^-. \end{cases} \tag{2}$$

Předpokládejme navíc, že b_1, b_2, c, f jsou spojité na $\bar{\Omega}$. Pojmy charakteristických křivek a vstupní hranice zůstavají stejné, ale mění se znění lemmatu 1.

Lemma 2 (Řešení LPDR)**.** *Nechť charakteristické křivky úlohy* (2) *pokrývají celou* Ω *a nechť* $u \in C^1(\bar{\Omega})$ *je reálná funkce splňující* $u = g$ *na* $\partial\Omega^-$*. Pak následující tvrzení jsou ekvivalentní:*

1) *Pro každou charakteristiku* $\phi : I \to \bar{\Omega}$ *úlohy (2) platí* $\frac{d}{ds}u(\phi(s)) = f(\phi(s)) - c(\phi(s))u(\phi(s))$, $s \in I$.

2) *u řeší úlohu (2) na* Ω.

Pokud je jedno z tvrzeních pravdivé, pak navíc platí, že řešení je ve tvaru

$$u(\phi(s)) = g(\phi(\alpha)) + \int_\alpha^s f(\phi(\xi))\,d\xi - \int_\alpha^s c(\phi(\xi))u(\phi(\xi))\,d\xi \tag{3}$$

pro každou charakteristiku ϕ *spočítanou s příslušnou počáteční podmínkou* $\phi(\alpha) \in \partial\Omega^-$.

Důkaz. Důkaz ekvivalence vyplývá z důkazu lemmatu 1.

1) $\Rightarrow$ *2)*

Z řetízkového pravidla opět dostaneme, že u řeší (LPDR) podél všech charakteristik. Jelikož křivky pokrývají Ω, tak u řeší (LPDR) na Ω. Navíc $u = g$ na $\partial\Omega^-$. Tedy u řeší úlohu (2) na Ω.

2) $\Rightarrow$ *1)*

Vezmeme-li libovolnou charakteristiku, pak z řetízkového pravidla a faktu, že u je řešením, plyne rovnost pro derivaci u v *1)*.

Nyní dokážeme, že řešení je skutečně ve tvaru 3. Ať ϕ je charakteristika spočítaná s počáteční podmínkou $\phi(\alpha) \in \partial\Omega^-$. Zkusme nyní zintegrovat vztah pro derivaci u podél ϕ v bodě *1)* (integrujeme od α do s):

$$\int_\alpha^s \frac{d}{d\xi}u(\phi(\xi))\,d\xi = \int_\alpha^s f(\phi(\xi))\,d\xi - \int_\alpha^s c(\phi(\xi))u(\phi(\xi))\,d\xi.$$

Dle Newton-Leibnizovy formule ($u \in C^1(\bar{\Omega})$) můžeme psát

$$u(\phi(s)) - u(\phi(\alpha)) = \int_\alpha^s f(\phi(\xi))\,d\xi - \int_\alpha^s c(\phi(\xi))u(\phi(\xi))\,d\xi.$$

A platí $u(\phi(\alpha)) = g(\phi(\alpha))$, takže po přičtení tohoto členu k rovnici dostáváme onu formuli. □

2 Numerický algoritmus

V této kapitole zkonstruujeme s využitím numerických metod algoritmus, jenž bude umět řešit úlohu 2 pomocí MoC. Napišme ji zde znova, abychom s ní mohli lépe pracovat.

$$\begin{cases} b_1 u_x + b_2 u_y + cu = f, & \text{v } \Omega \subset \mathbb{R}^2, \\ u = g, & \text{na } \partial\Omega^-. \end{cases} \tag{3}$$

Zaveďme si dodatečné předpoklady: Nechť b_1, b_2, c, f, g jsou lipschitzovsky spojité reálné funkce, množina Ω je omezená otevřená. Poslední předpoklad nemusí být nutně splněn, pokud nepotřebujeme dostat řešení na celé množině Ω.

Algoritmus jsem naprogramoval v Matlabu. Kód a dokumentace k jeho použití jsou odevzdány společně s bakalářskou prací.

Stručná idea algoritmu:

1. Aproximace vstupní hranice $\partial\Omega^-$ pomocí nějakého dělení.
2. Nalezení aproximace charakteristik pomocí Rungeovy-Kuttovy metody.
3. Regulace vzájemné vzdálenosti sousedních bodů na jedné charakteristice vytvořením nových bodů nebo odebraním zbytečných (postupně provedeme pro všechny křivky). Stejně tak regulace relativní vzdálenosti sousedních charakteristických křivek. Požadujeme, aby tyto vzdálenosti byly menší než námi zvolená *tolerance*.
4. Vytvoření trojúhelníkové sítě pokrývající Ω, jejíž vrcholy budou ležet na charakteristikách.
5. Následné spočtení aproximované hodnoty řešení ve vrcholech trojúhelníků. Vyskytuje-li se v úloze zjednodušená LPDR, lze to udělat triviálně s pomocí okrajové podmínky (viz podkapitola 1.2). Pokud ne, pak se pomocí numerické kvadratury spočítají určité integrály (viz podkapitola 1.3)

Algoritmus jsem rozdělil na několik částí. Každé z nich je věnována jedna podkapitola. Po vysvětlení numerického postupu v podkapitolách bude následovat rámeček MATLAB, v němž stručně vysvětlím, jak jsem k algoritmu přistupoval při programování a o jakou část kódu se jedná. Hlavní program je `Main.m`, přes který se spouští všechny ostatní funkce. Celý program a jeho dokumentace jsou přiloženy k této práci.

Nakonec bude užitečné, z důvodu přehlednosti a sjednocení značení s Matlabem, psát místo desetinné čárky desetinnou tečku, např.: 3,6 → 3.6.

2.1 Inicializace algoritmu

V první fázi algoritmu probíhá inicializace. Zadáme tedy funkce b_1, b_2, c, f, g. Následně musíme nějak algoritmu popsat předpis otevřené množiny Ω a její vstupní

hranice, což není jednoduchý úkol.

S důrazem na co největší obecnost jsem vymyslel algoritmus řešící úlohu (3) na množině zadané jako vnitřek uzavřené křivky. Bereme v úvahu jen takové křivky, které rozdělují $\mathbb{R}^2$ na dvě komponenty souvislosti, z nichž jedna má nekonečný diametr. Druhá komponenta bude naše Ω. Nakonec by tato křivka neměla být příliš oscilující. Až se budou později provádět různé aproximace, mohla by se podoba vstupní hranice množiny Ω razantně změnit a nedostali bychom relevantní výsledky.

Mějme tedy uzavřenou křivku $\omega : J \to \mathbb{R}^2$. Pak $\Omega = \mathrm{Int}(M)$, kde M je množina s hranicí $\omega(J)$ a $\mathrm{diam}(M) < \infty$, a $\omega(U) = \partial\Omega^-$, kde $U \subset J$. Je důležité podotknout, že musíme znát jak parametrické vyjádření (i s intervaly J a U), tak obecnou rovnici křivky ω (viz Matlab 1).

Jako poslední, což je volitelná záležitost, můžeme zadat tzv. *toleranci* (zkráceně *tol*). Ta omezuje obsah jednotlivých trojúhelníků v později vytvořené trojúhelníkové síti (viz podkapitola 2.5). Čím menší se zvolí *tol*, tím přesnější aproximaci řešení úlohy dostaneme, ale algoritmus bude pomalejší. V základu je nastaveno $tol = 0.15$.

Matlab 1

- Pomocí sady nerovností definuji vnitřek uzavřené křivky jakožto množinu Ω. Proto je třeba obecná rovnice ω. Lze pak kontrolovat, zda se nějaký bod při výpočtu nevychýlil z množiny Ω a v tom případě nás nezajímá.
- Vstupní hranici $\partial\Omega^-$ definuji parametricky s pomocí intervalu U. Zde budeme vybírat body, z nichž povedou charakteristiky.
- Kvůli double-precision aritmetice Matlabu přičítám k *tol* ještě 10^{-10}. Stručně řečeno, Matlab nepočítá vždy přesně a může dojít k miniaturním chybám ve výpočtech (např.zaokrouhlování, viz [5], podkapitola 2 - Floating-Point Numerical Representation). Při porovnávání čísel proto počítám i s nějakou odchylkou.
- V kódu:
  `B.m`,
  `Ivc.m`,
  `Entry_bdd.m`,
  `Set_bdd.m`,
  `Set_def.m`

Tímto bychom měli inicializaci úlohy hotovou.

2.2 Aproximace vstupní hranice

Dále chceme aproximovat vstupní hranici tak, aby vzdálenost sousedních bodů, které na ní zvolíme, byla v rozmezí [0,*tol*]. Nejprve mějme fixní ekvidistantní dělení intervalu U s krokem rovným *tol*, přičemž poslední dva členy budou k

sobě pravděpodobně blíže. Označme $U := [a, b]$ a dělení $D = \{x_j\}_{j=0}^{n}$ takové, že $a = x_0 < x_1 < \cdots < x_{n-1} < x_n = b$ a $\|x_j - x_{j-1}\| \leqslant tol$, $j = 1, \ldots, n$.

Nyní však nemusí platit, že $\|\omega(x_j) - \omega(x_{j-1})\| \leqslant tol$ pro nějaká $j \in \{0, \ldots, n\}$. Budeme proto adaptivně zjemňovat dělení $\omega(D)$, dělící vstupní hranici $\partial\Omega^-$. Chceme získat dělení $\omega(D')$, pro které bude platit, že sousední body $\omega(D')$ jsou od sebe vzdáleny nejvýše o hodnotu *tol*.

Ať $k = 1$ a $\omega(D') = \{\omega(x_0)\}$. Níže je zapsaná konstrukce $\omega(D')$ pomocí pseudokódu. Algoritmus vytváří nové hodnoty na vstupní hranici mezi dvěma sousedními body, které jsou od sebe moc daleko. Postupně body prochází a upravuje celé původní dělení $\omega(D)$ (viz Kroky 1 a 18 v Algoritmu 1).

Algoritmus 1 Nalezení vhodně jemného dělení vstupní hranice $\partial\Omega^-$

1: $p_1 \leftarrow \omega(x_k)$
2: $p_2 \leftarrow$ poslední prvek přidaný do $\omega(D')$
3: $j \leftarrow 250$
4: **if** $\|p_2 - p_1\| \leqslant tol$
5: $\quad \omega(D') \leftarrow$ Přidám na konec $\omega(D')$ bod p_1
6: **else**
7: $\quad$ **while** $\|p_2 - p_1\| > tol$
8: $\quad\quad i \leftarrow 1$
9: $\quad\quad$ **while** $\|p_2 - p_1\| > tol$
10: $\quad\quad\quad p_1 \leftarrow \omega(x_k - \frac{i}{250}\, tol)$ ▷ Trochu se přiblížíme k p_2
11: $\quad\quad\quad i \leftarrow i + 1$
12: $\quad\quad\quad$ **if** $i = j$ **then** break
13: $\quad\quad j \leftarrow i - 1$ ▷ Nová horní hranice pro i
14: $\quad\quad$ **if** $i = 2$ **then** break
15: $\quad\quad \omega(D') \leftarrow$ Přidám na konec $\omega(D')$ bod p_1
16: $\quad\quad p_2 \leftarrow p_1$
17: $\quad\quad p_1 \leftarrow \omega(x_k)$
18: $k \leftarrow k + 1$ a vracím se ke Kroku 1, dokud $k < n + 1$

Proč při zjemňování D volíme krok 1/250? Po otestování se to ukázalo jako jedna z nejlepších voleb zaručující vcelku přesné výsledky a zároveň nezabírající tolik času. Kdyby se stalo, že ω bude rychle rostoucí nebo klesající funkce, pak i v algoritmu může dosáhnout hodnoty až 250 (resp. hodnoty z předchozí iterace, pro kterou už byl bod vytvořen - Kroky 12 a 14). Tzn. že by nám hodnoty p_1 a p_2 splynuly. Takže nepokračujeme dál, ale i tak přidáme ten nejbližší možný p_1, co jsme našli a provedeme reset pro hodnot p_1, p_2. Algoritmus tak vrátí výsledek, ale nebude už splňovat striktní odhady pro vzdálenost sousedních bodů, a tedy nebude splňovat příslušné odhady pro rozměry trojúhelníkové sítě.

MATLAB 2

- Mimo jiné si jako vedlejší produkt ukládá program i pomocnou trojici indexů při nalezení nového bodu: [aktuální index k; zjemňovací krok $250 - i$; kolikatý bod jsme přidali do $\omega(D')$]. Tyto informace se nám budou hodit později v podkapitole 2.4
- V kódu:
  `Disc_entry_bdd.m`

Tímto máme aproximovanou vstupní hranici $\partial\Omega^- \approx \omega(D')$, se kterou umí Matlab pracovat. Označme $Q := \omega(D')$

2.3 Aproximace charakteristik

V dalším kroku aproximujeme charakteristické křivky vedoucí z bodů v Q, které slouží jako počáteční podmínky. Na řešení charakteristického systému ODR použijeme známou Rungeovu-Kuttovu metodu 4. řádu. Proč se vyplácí nejvíce použít metodu 4. řádu, lze nalézt v [6] (chapter 3, theorems 324A, 324B). Víceřádové metody už by se nevyplatily z hlediska výpočetní složitosti, protože by bylo třeba spočíst více koeficientů než je řád metody. Jedná se o explicitní jednokrokovou metodu, která generuje body, jež mají aproximovat charakteristické křivky. Při postupném počítání nových bodů používám čtveřici koeficientů používaných pro tzv. „klasickou Rungeovu-Kuttovu metodu“ ([6] chapter 3, page 180).

Dále budeme označením ϕ, ψ rozumět charakteristické křivky a Φ, Ψ jejich aproximace pomocí Rungeovy-Kuttovy metody.

Nechť $q_0 \in Q$ je zvolená počáteční podmínka a Φ bude seznam bodů aproximující charakteristiku ϕ spočtenou s počáteční podmínkou q_0 z charakteristického systému. Metoda je čtvrtého řádu, proto volím za iterační krok $h := 0.05$, což je více než dostačující. Vektor pravých stran z charakteristického systému si označme jako $b(x, y) = (b_1(x, y), b_2(x, y))^T$. Jakmile máme $\Phi = [q_0, q_1, \ldots, q_i]$, tak spočtení dalšího aproximujícího bodu q_{i+1} pro $i \in \mathbb{N}$ můžeme zapsat následovně:

Algoritmus 2 Runge-Kutta 4. řádu

1: $k_1 \leftarrow b(\Phi(i))$ ▷ $\Phi(i) = q_i$
2: $k_2 \leftarrow b(\Phi(i) + \frac{1}{2}hk_1)$
3: $k_3 \leftarrow b(\Phi(i) + \frac{1}{2}hk_2)$
4: $k_4 \leftarrow b(\Phi(i) + hk_3)$
5: $step \leftarrow \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4)$
6: $\Phi(i + 1) \leftarrow \Phi(i) + step$

Musíme však ohlídat jednu věc. Chceme opět dostat body takové, že všechny sousední budou mít od sebe navzájem vzdálenost v rozmezí $[0, tol]$. Tento problém se řeší úplně stejně jako v podkapitole 2.2. Kód zkouší hledat nové body mezi dvěma vzdálenými tak, že zmenšuje h. Tentokrát pro přiblížení se neodečítá $\frac{1}{250}h$, ale $\frac{1}{150}h$.

Víme, že Φ aproximuje nějakou charakteristickou křivku $\phi : I \to \bar{\Omega}$, tedy $q_i \approx \phi(s_i), s_i \in I, i \in \mathbb{N}_0$ a navíc $q_0 = \phi(s_0) = \phi(\alpha) \in \partial\Omega^-$. Co dělá Rungeova-Kuttova metode je, že kromě vytváření aproximovaných hodnot vytváří i body v nichž hodnotu aproximujeme a to následovně: $s_i = s_{i-1} + h_i, i \in \mathbb{N}$, kde h_i je iterační krok, pomocí kterého jsme spočítali hodnotu q_i. Proto si během tohoto algoritmu budeme ukládat i kroky h_i, které budeme potřebovat v podkapitole 2.6 při numerické integraci.

Matlab 3

- Po přidání každého bodu zkontroluji, zda neležel už mimo množinu Ω. Pokud ano, nechám ho stejně přidaný (kvůli co nejlepšímu pokrytí trojúhelníkovou sítí v podkapitole 2.5) a poté přestanu tvořit aktuální charakteristickou křivku.
- V kódu:
  `Create_characteristics.m`

2.4 Přidávání více křivek

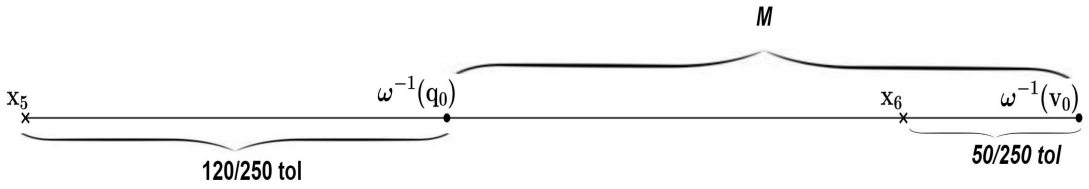
Ačkoliv se zdá, že jsme s charakteristikami hotovi, je třeba ohlídat ještě nějakou jejich relativní vzdálenost. Jelikož jsme předpokládali lipschitzovskou spojitost pro b_1, b_2, pak z Picardovy-Lindelöfovy věty ([7], chapter 1, theorem 1.1) plyne lokální jednoznačnost řešení charakteristického systému. To znamená, že jednotlivé charakteristiky se nemohou protínat. Uvažujeme-li, že „začínají“ na vstupní hranici, pak se může stát, že se k sobě postupně **přibližují**, nebo naopak se od sebe **vzdalují**, nebo začínají blízko sebe na vstupní hranici, pak se **někde v Ω vzdálí** a než vystoupí ven tak se zase přiblíží.

V prvním případě nic dělat nemusíme. Body vstupní hranice už jsou k sobě dostatečně blízko a když z nich křivky vyrazí, tak se jejich vzdálenost jen snižuje. V druhém případě je třeba přidat křivek více. Chci, aby body, které leží na dvou sousedních křivkách, ale už neleží v Ω (ty body, které byly naposledy přidány k aproximaci charakteristických křivek), od sebe byly vzdáleny maximálně v rozmezí $[0, tol]$. Největší vzdálenost takových dvou sousedních křivek v Ω nastane právě v těchto dvou koncových bodech, což zvládnu vyřešit (viz níže v této podkapitole). Poslední případ jsem neošetřoval, jelikož je složitý a nenastává moc často. Program vrátí nějaký výsledek stejně, jen nemusí splňovat příslušné odhady pro rozměry trojúhelníkové sítě v podkapitole 2.5.

Mějme tedy dvě sousední aproximované křivky $\Phi_1 = [q_0, q_1, \ldots, q_{m1}]$ a $\Phi_2 = [v_0, v_1, \ldots, v_{m2}]$. Nechť nastane druhý případ, tedy $d := \|q_{m1} - v_{m2}\| > tol$. Je potřeba přidat nové body do aproximace vstupní hranice Q a z nich vést nové charakteristické křivky. Kontrolovat chování křivek na části hranice Ω, ze které křivky vystupují ven, neumíme, proto upravujeme jen aproximaci vstupní hranice. Nyní nám pomůžou dříve uložené indexy (viz podkapitola 2.2 rámeček Matlab 2). Nechť $T_1 = [t_1^1, t_1^2, t_1^3], T_2 = [t_2^1, t_2^2, t_2^3]$ jsou trojice indexů příslušejících q_0 a v_0. První a druhé členy z trojice indexů nám řeknou, kolik nových bodů vstupní hranice jsme ještě schopni přidat s pomocí našeho zjemňovacího procesu pro dělení

$\omega(D')$ (resp. kolik kroků s velikostí $\frac{1}{250}$ jsme ještě schopni udělat). Označme toto číslo M.

Příklad.
Pro $T_1 = [5, 120, 9]$, $T_2 = [6, 50, 10]$ tam máme prostor pro $M = 50 + (250 - 120) = 180$ nových bodů.



Dále můžeme heuristicky říct, jakými body přibližně vést nové křivky. Pokud předpokládáme, že rozložení koncových bodů nových křivek mezi Φ_1 a Φ_2 bude relativně rovnoměrné, pak jich stačít přidat přibližně $\lceil \frac{d}{tol} \rceil$. Já toto číslo násobím ještě 2, abych se pokusil přidat jich 2x víc, pokud by předpoklad výše nebyl pravdivý. Jistě bychom dokázali najít nějaké charakteristické křivky reprezentované třeba exponenciálami, pro které by tento postup nestačil. Nicméně už tak pracujeme vcelku s malými vzdálenostmi a v praxi to ve většině případů funguje. Zároveň pro aplikaci algoritmu využíváme počítač, který počítá jen s konečnými čísly, tudíž vždy by šla najít funkce, která by se vymezila těmto předpokladům.

Následně M rozdělíme v poměru $1 : 2\lceil \frac{d}{tol} \rceil$, což nám vytvoří nové zjemňovací koeficienty. V případě, že M nelze rozdělit tímto poměrem, bereme větší poměr, kterým to lze. Dále tedy algoritmus přidává nové body do Q a z nich pak vede nové křivky (popsáno v 2.3). Pokud by koncové body křivek byly stále moc vzdáleny, pokusí se zmenšit krok mezi body v D' a přiblížit body aproximované vstupní hranice Q.

Příklad.
Mějme stejné T_1, T_2, M jako v příkladu výše a $tol = 0.1, d = 2$, pak rozdělíme M v poměru $1 : 40$. Číslo 180 však nelze rozdělit na 40 částí, tak vezmeme 45. Dostaneme dílky po 4. Víme, že z $\omega(x_5 + \frac{120}{250}tol) = q_0$ vede Φ_1. Vezměme tedy potenciální nový bod $\omega(x_5 + \frac{124}{250}tol)$ a veďme z něj křivku. Pokud jsou koncové body této křivky a té sousední (aktuálně Φ_1) daleko, zkusím krok $\frac{123}{250}tol$, atd... Pokud dojdu ke $\frac{121}{250}tol$, přidám alespoň tu nejbližší křivku, co jsem našel. Pro další novou křivku zkouším krok $\frac{128}{250}tol$. Pak postupně dojdu k $\omega(x_5 + \frac{252}{250}tol) = \omega(x_6 + \frac{2}{250}tol)$ a zkouším konstruovat křivky až do $\omega(x_6 + \frac{50}{250}tol) = v_0$, odkud už vede Φ_2.

Matlab 4

- V kódu:
  `Add_characteristics.m`

2.5 Filtrace všech bodů a tvorba trojúhelníkové sítě

V předchozích částech jsme aproximovali vstupní hranici a charakteristiky. Přidali jsme dostatečně bodů i křivek, aby platily určité odhady. Pravdou je, že spousta přidání byla zbytečná, takže nyní držíme v paměti hodně přebytečných bodů a křivek, které nejsou potřeba držet. Cílem však právě bylo vytvořit je i za cenu toho, že jich bude více. Je to jednodušší cesta, protože odstraňování přebytečných křivek a bodů z Q není těžké. Těžké je kontrolované přidávání. Pro další práci s křivkami si zaveďme značení:

Nechť $\Phi = [q_0, q_1, \ldots, q_n]$ je aproximace charakteristické křivky, pak v této kapitole budeme značit i-tou hodnotu jako $\Phi(i) := q_i$ a poslední hodnotu jako $\Phi(end) := q_n$.
Dále pro body $a, b, c \in \mathbb{R}^2$ rozumíme značením $\triangle[a, b, c]$ trojúhelník s vrcholy a, b, c.

2.5.1 Filtrace

Nejdříve budeme procházet všechny charakteristické křivky a jestliže najdeme tři po sobě jdoucí Φ_1, Φ_2, Φ_3 splňující: $\|\Phi_1(1) - \Phi_3(1)\| < tol$ a $\|\Phi_1(end) - \Phi_3(end)\| < tol$, pak můžeme Φ_2 zapomenout (společně i s bodem na vstupní hranici, ze kterého vycházela!). Takto procházím křivky do té chvíle, kdy je profiltruji kompletně.

Následně si vezmu pevně jednu z křivek $\Phi = [q_0, q_1, \ldots, q_n]$ a koukám na trojice po sobě jdoucích bodů q_{i-1}, q_i, q_{i+1}. Opět mohu zapomenout q_i, jestliže $\|q_{i-1} - q_{i+1}\| < tol$. Takto profiltruji body na všech křivkách.

2.5.2 Trojúhelníková síť

Nyní můžeme vytvořit trojúhelníkovou síť, která bude pokrývat množinu Ω, a jejíž vrcholy budou ležet na charakteristikách. Síť slouží k tomu, abychom mohli najít aproximovanou hodnotu řešení úlohy (3) kdekoliv v Ω. Ve vrcholech je to jednoduché a hodnotu uvnitř trojúhelníku získáme pomocí interpolace hodnot ve vrcholech (viz podkapitola 2.7).

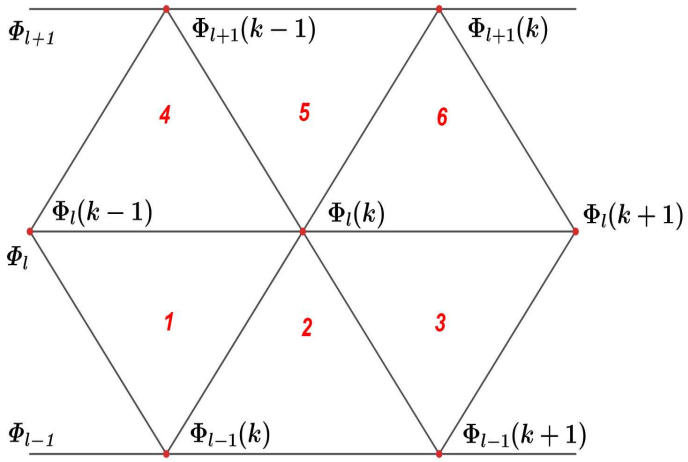
Proč trojúhelníky? Trojúhelník (nebo také *element sítě*) je nejjednoduší n-úhelník, jakým mohu množinu pokrýt. Pokud bych se rozhodl pro elementy s více vrcholy, pak by pokrytí množiny Ω tak, aby vrcholy ležely na charakteristikách, bylo složitější.

Zbývá teď nalézt způsob jak vhodně pospojovat body na charakteristikách. Pokud se nacházíme v k-tém bodě nějaké vnitřní charakteristiky Φ_l (nevychází z prvního ani posledního bodu vstupní hranice), vytvoříme z něj 6 následujících trojúhelníků ($\triangle$):

1. $\triangle[\Phi_l(k-1), \Phi_l(k), \Phi_{l-1}(k)]$
2. $\triangle[(\Phi_{l-1}(k), \Phi_l(k), \Phi_{l-1}(k+1)]$

3. $\triangle[(\Phi_l(k+1), \Phi_l(k), \Phi_{l-1}(k+1)]$
4. $\triangle[(\Phi_l(k-1), \Phi_l(k), \Phi_{l+1}(k-1)]$
5. $\triangle[(\Phi_{l+1}(k-1), \Phi_l(k), \Phi_{l+1}(k)]$
6. $\triangle[(\Phi_l(k+1), \Phi_l(k), \Phi_{l+1}(k)]$

Pro lepší představu je tvorba těchto elementů znázorněna na obrázku níže (pro zjednodušení uvažujeme charakteristiky jako přímky)



Obrázek 2.1 Vytvoření trojúhelníků kolem jednoho bodu

Díky tomu, že jsem provedl filtraci, tak je velmi malá pravděpodobnost, že by vznikaly degenerované trojúhelníky.

Pokud se nacházím v bodě vstupní hranice, na okrajové charakteristické křivce nebo v jiném bodě množiny $\partial\Omega$, pak definuji trojúhelníky pomocí stejných pravidel ale jen pro body, které existují.

Speciální případ: Pokud jsem u části hranice množiny Ω, ze které křivky vystupují ven, a Φ_l má N bodů a Φ_{l-1} má M bodů, $N > M$, pak dotvořím trojúhelníky mezi těmito křivkami jen s pomocí M-tého bodu Φ_{l-1} a zbylých $M, \ldots, N$ bodů z Φ_l. Tedy tvořím trojúhelníky:

$\triangle[\Phi_l(M), \Phi_l(M+1), \Phi_{l-1}(M)]$,
$\triangle\,[\Phi_l(M+1), \Phi_l(M+2), \Phi_{l-1}(M)]$,
$\ldots$,
$\triangle\,[\Phi_l(N-1), \Phi_l(N), \Phi_{l-1}(M)]$.

A stejně postupuji pro další podobné případy.

2.5.3 Odhad velikosti obsahů trojúhelníků

Když jsme si dali práci s tvorbou všech bodů se zohledněním konstanty *tolerance*, podívejme se na horní odhad obsahů trojúhelníků. Mějme dvě sousední

charakteristické křivky $\Phi = [q_0, \ldots, q_n], \Psi = [v_0, \ldots, v_m]$. Nechť Φ, Ψ jsou přímky a jejich vzdálenost je rovna *toleranci* v celé množině Ω, což je nejhorší možný případ. Pokud mají body na Φ mezi sebou maximální vzdálenost *tol* a stejně tak ji mají mezi sebou i body na Ψ. Pak obsah libovolného trojúhelníku mezi těmito křivkami můžeme odhadnout obsahem čtverce, jenž ho zahrnuje v sobě:

$$S_{\triangle} \leqslant (\text{vzdálenost křivek}) \times (\text{vzdálenost bodů}) = tol^2$$

To ale není nejpesimističtější odhad. Jestliže by tyto křivky nebyly přímky a různě se vlnily, pak horní odhad obsahu n-tého vytvořeného trojúhelníku mezi křivkami Φ, Ψ bude roven $c_n tol^2$. Přičemž c_n je nějaká kladná reálná konstanta a platí, že posloupnost $\{c_n\}_{n=1}^{N}$ je neklesající. Tedy čím dál od vstupní hranice tvořím trojúhelníky tím větší mohou mít obsah. Toto je však velice hrubý odhad a ve většině případů jsou obsahy trojúhelníků menší.

Matlab 5

- Během práce na této části jsem změnil reprezentaci charakteristických křivek a místo seznamu bodů jsem vytvořil seznam struktur. Každá struktura obsahuje stále souřadnice bodu a potom maximálně 6 vrcholů, které s ním tvoří trojúhelníky.
- Souběžně s tvorbou sítě vytvářím i soubor typu `.vtk`, který pak slouží k vykreslení sítě v grafickém softwaru Paraview.
- V kódu `Create_net.m`

2.6 Aproximace řešení ve vrcholech trojúhelníků

V této chvíli už můžeme spočítat řešení. Začneme ve vrcholech elementů sítě, kde je to jednodušší. Z podkapitol 1.2 a 1.3 víme, v jakém tvaru bude řešení. Proto jsme také položili vrcholy na charakteristické křivky. Označme aproximované řešení úlohy (3) jako U, tzn. $U \approx u$. Mějme libovolný element $\triangle[a, b, c]$ z trojúhelníkové sítě. Řekněme, že chceme zjistit $U(a)$. Nalezneme aproximovanou charakteristickou křivku $\Phi = [q_0, \ldots, q_n]$ takovou, že existuje $i \in \{0, \ldots, n\} : q_i = a$. Jestliže je úloha (3) zadaná pouze se zjednodušenou LPDR, pak $U(a) = U(q_i) = U(q_0) = g(q_0)$, protože řešení je konstantní podél charakteristik. Tedy platí rovnost $u(a) = g(q_0)$, protože $g(q_0)$ je definováno jako $g(\phi(s_0))$.

Nechť f, c zadané v úloze (3) jsou nenulové funkce, Φ, a jsou jako v předchozím odstavci a $a = q_i = \Phi(s_i) \approx \phi(s_i)$. Dle lemmatu 2 platí

$$u(\phi(s_i)) = g(\phi(\alpha)) + \int_{\alpha}^{s_i} f(\phi(\xi))\, d\xi - \int_{\alpha}^{s_i} c(\phi(\xi)) u(\phi(\xi))\, d\xi.$$

První sčítanec už máme spočtený jako $g(\phi(\alpha)) = g(\phi(s_0)) = g(q_0)$. Pro aproximaci integrálů jsem se rozhodl použít metodu, které se říká složené lichoběžníkové pravidlo ([8], chapter 2 - Trapezoidal rule). Jen je potřeba myslet na to, že nepracujeme s konstantním dělením intervalu $[\alpha, s_i] = [s_0, s_i]$. V podkapitole 2.3

jsme si zaznamenávali kroky h_j mezi s_{j-1} a s_j, které mohou být různé. Proveďme aproximaci charakteristik a prvního sčítance ve vzorci výše. Dostaneme:

$$g(q_0) + \int_{s_0}^{s_i} f(\Phi(\xi))\,d\xi - \int_{s_0}^{s_i} c(\Phi(\xi))U(\Phi(\xi))\,d\xi,$$

což ještě nemůžeme prohlásit za $U(a)$. Nejdříve musíme nahradit integrály jejich aproximacemi. První integrál můžeme jednoduše odhadnout jako

$$\int_{s_0}^{s_i} f(\Phi(\xi))\,d\xi = \sum_{j=1}^{i} \int_{s_{j-1}}^{s_j} f(\Phi(\xi))\,d\xi \approx \sum_{j=1}^{i} h_j \frac{f(\Phi(s_j)) + f(\Phi(s_{j-1}))}{2} =: I_1$$

Druhý integrál spočítáme stejnou metodou, ale musíme si pohlídat to, že na pravé straně se v sumě objeví člen $U(a)$. Provedeme proto úpravu vzorce:

$$\int_{s_0}^{s_i} c(\Phi(\xi))U(\Phi(\xi))\,d\xi \approx \sum_{j=1}^{i} h_j \frac{c(\Phi(s_j))U(\Phi(s_j)) + c(\Phi(s_{j-1}))U(\Phi(s_{j-1}))}{2} =$$

$$= \sum_{j=1}^{i-1} h_j \frac{c(\Phi(s_j))U(\Phi(s_j)) + c(\Phi(s_{j-1}))U(\Phi(s_{j-1}))}{2} + h_i \frac{c(\Phi(s_{i-1}))U(\Phi(s_{i-1}))}{2}$$
$$+ h_i \frac{c(a)U(a)}{2}$$

Označme si

$$I_2 := \sum_{j=1}^{i-1} h_j \frac{c(\Phi(s_j))U(\Phi(s_j)) + c(\Phi(s_{j-1}))U(\Phi(s_{j-1}))}{2} + h_i \frac{c(\Phi(s_{i-1}))U(\Phi(s_{i-1}))}{2}.$$

Jestliže by nastal případ $h_i \frac{c(a)}{2} = -1$, pak touto metodou nemůžeme aproximovat druhý integrál. Nicméně se zaokrouhlovacími chybami, které Matlab dělá, je to velice málo pravděpodobné.

Aproximované řešení úlohy (3) ve vrcholu a pak můžeme zapsat jako

$$U(a) = (g(q_0) + I_1 + I_2)\frac{1}{1 + h_i \frac{c(a)}{2}},$$

což je aproximace přesného řešení $u(a) \approx U(a)$.

Matlab 6

- V kódu `Approximate_solution_vertices.m`

2.7 Aproximace řešení uvnitř trojúhelníků

Nechť $p \in \Omega$ a chceme zjistit $U(p)$. Nejdříve najdeme charakteristickou křivku $\Phi = [q_0, \ldots, q_n]$ takovou, že $\exists i \in \{0, \ldots, n\}$, pro které platí:

$$\forall \text{ char. křivky } \Psi = [v_0, \ldots, v_m], \Psi \neq \Phi, \forall j \in \{0, \ldots, m\} : \|p - q_i\| < \|p - v_j\|.$$

Vezměme i takové, že $\forall j \in \{0, \ldots, n\}, j \neq i : \|p - q_i\| < \|p - q_j\|$. Máme tedy bod q_i, který je k p nejblíže a q_i je vrcholem až šesti trojúhelníků. Zkusíme jestli v jednom z nich neleží bod p.

Máme-li element ze sítě $\triangle_{abc} = \triangle[a, b, c]$, pak p leží uvnitř $\triangle_{abc}$ právě tehdy, když

$$S_{abc} = S_{apb} + S_{apc} + S_{bpc},$$

kde

$$S_{abc} = \left| \frac{\det([b - a, c - b])}{2} \right|$$

je obsah trojúhelníku s vrcholy a, b, c. Jelikož body na křivkách nejsou rozloženy rovnoměrně, může se stát, že p nebude ležet uvnitř žádného z 6 trojúhelníků. Pak zkoušíme najít druhý nejbližší bod po q_i a opakujeme postup.

Označme si nyní $\triangle_p = \triangle[p_1, p_2, p_3]$ element, ve kterém leží p. Hodnotu $U(p)$ vyjádříme pomocí barycentrické interpolace hodnot ve vrcholech ([9], chapter 1 - Introduction):

$$u(p) \approx U(p) = \frac{S_{p_2p_3p}}{S_{p_1p_2p_3}} U(p_1) + \frac{S_{p_1p_3p}}{S_{p_1p_2p_3}} U(p_2) + \frac{S_{p_2p_1p}}{S_{p_1p_2p_3}} U(p_3),$$

kde $U(p_i)$, $i = 1, 2, 3$, dokážeme spočíst z 2.6.

Matlab 7

- Barycentrickou interpolaci nevyužíváme explicitně v kapitole 3, ale je součástí kódu. Je využívaná k vykreslení vybraných řezů řešením.
- Při kontrole zda leží bod uvnitř trojúhelníku povoluji odchylku součtu obsahů malých trojúhelníků od obsahu velkého trojúhelníku o 5 desetinných míst a více. To je opět kvůli double-precision aritmetice.
- V kódu:
  `Approximate_solution_inside.m`,
  `Find_closest_vertex.m`,
  `Is_in_triangle.m`,
  `Barycentric_interp.m`

3 Experimenty

V této kapitole otestuji a okomentuji funkčnost algoritmu na různorodých příkladech lineárních PDR 1. řádu. Dovolil bych si připomenout, že algoritmus byl implementován v Matlabu. Grafy, které zde uvidíme, jsou vykresleny pomocí Matlabu a grafického softwaru Paraview. Kód i návod k jeho použití jsou odevzdány společně s bakalářskou prací.

Každý příklad si rozebereme jak z teoretické stránky, tak z praktické stránky. Mimo jiné se u každého z příkladů bude nacházet čas, který program potřeboval k vyřešení úlohy, *tolerance*, se kterou zrovna počítáme a 3 grafy. Na prvním grafu bude vykreslena hranice množiny Ω společně s nalezenými body, které aproximují charakteristiky. Zde budeme moci vidět hustotu bodů na křivkách. Na druhém obrázku budou pomocí nalezených bodů spojitě vykresleny křivky, abychom měli lepší představu o tom, jak vypadají a kolik jich je. Poslední obrázek bude vyobrazené aproximované řešení úlohy v $\mathbb{R}^3$ v Paraview. Softwaru předám vytvořenou trojúhelníkovou síť a aproximované hodnoty řešení ve vrcholech trojúhelníků. Paraview pak sám dodefinuje hodnoty uvnitř trojúhelníků a vykreslí spojitou plochu.

3.1 Příklad s charakteristikami jako kružnicemi

Na začátek si ukážeme úlohu se zjednodušenou LPDR, kde uvidíme vzorové grafické předvedení vlastnosti konstantnosti řešení podél charakteristik. Uvažme úlohu:

$$\begin{cases} yu_x - xu_y = 0, & \text{v } \Omega = \{(x,y) \in \mathbb{R}^2 : x^2 + y^2 < 4;\ y > 0\}, \\ u(x,y) = \cos(x + \pi/2), & \text{pro } (x,y) \in [-2,0] \times \{0\}. \end{cases}$$

Řešení hledáme na kladné polokružnici s poloměrem 2. Jako vstupní hranici bereme úsečku vedoucí z bodu $[-2,0]$ do bodu $[0,0]$.

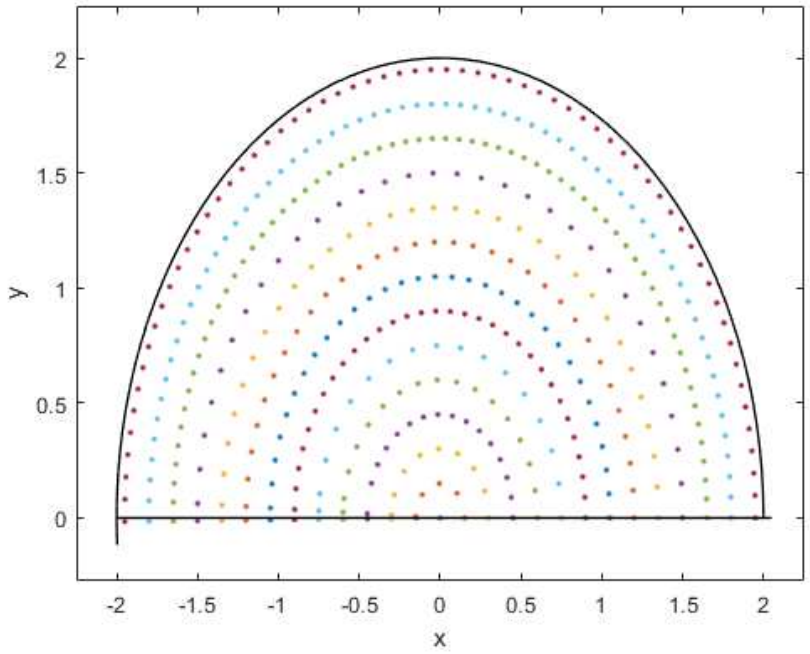
Charakteristický systém bude ve tvaru:

$$\frac{d}{ds}\begin{pmatrix}\phi_1(s)\\ \phi_2(s)\end{pmatrix} = \begin{pmatrix}\phi_2(s)\\ -\phi_1(s)\end{pmatrix}.$$

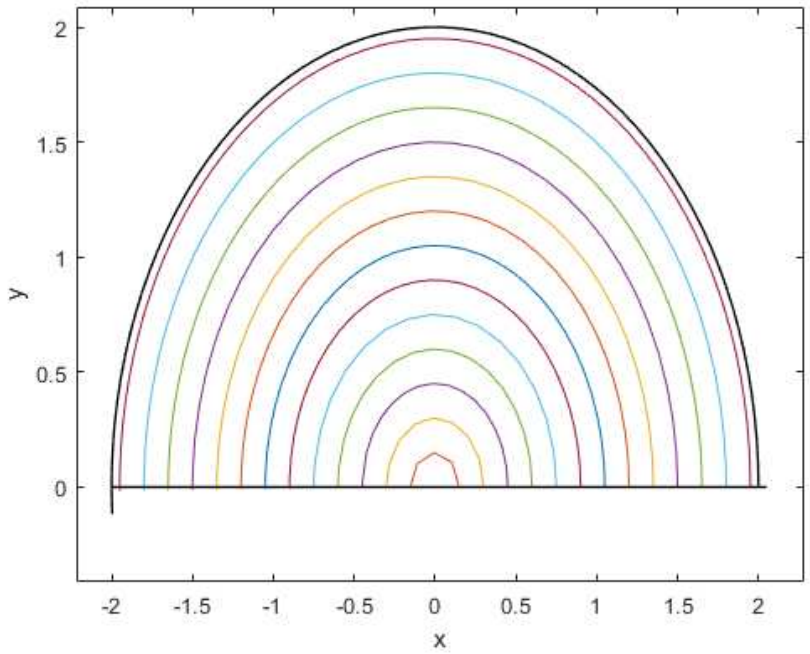
Jednoduše lze spočítat nebo uhádnout řešení:

$$\begin{pmatrix}\phi_1(s)\\ \phi_2(s)\end{pmatrix} = \begin{pmatrix}\cos(s) + a\\ \sin(s) + b\end{pmatrix}, a, b \in \mathbb{R}.$$

Charakteristickými křivkami budou tedy kružnice, resp. půlkružnice. Algoritmus pro tuto úlohu spustíme s přednastavenou *tolerancí* rovnou 0.15. Na obrázku 3.1 můžeme vidět barevně vykreslené body ležící na charakteristikách a černě vykreslenou hranici $\partial\Omega$. Na obrázku 3.2 vidíme barevné spojitě vykreslené charakteristiky.

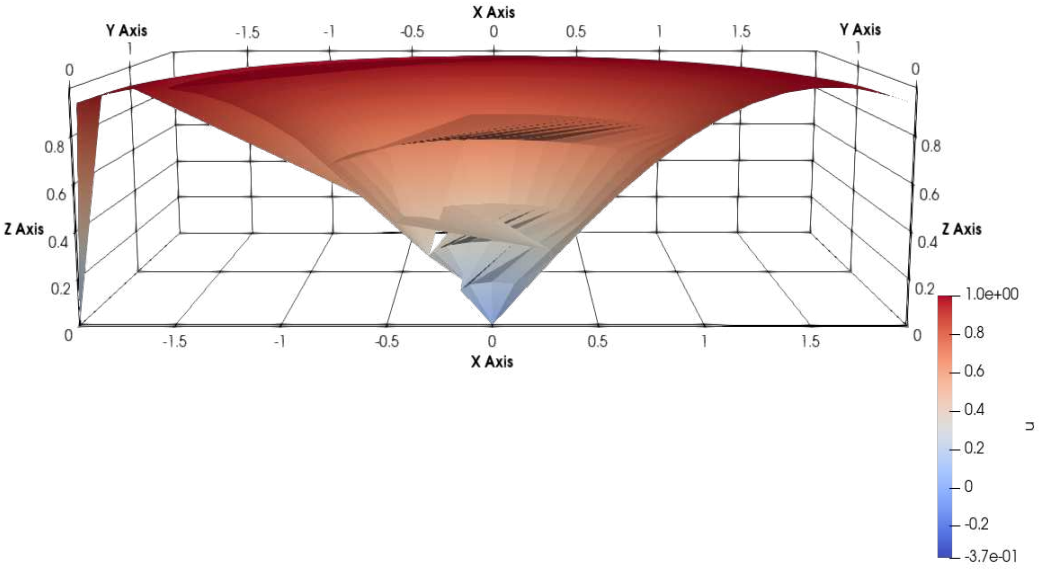


Obrázek 3.1 Body aproximující charakteristiky, $tol = 0.15$



Obrázek 3.2 Spojitě vykreslené charakteristiky, $tol = 0.15$

Jelikož okrajová podmínka je cos, tak na základě teorie bychom měli jako řešení dostat část posunuté sinusoidy zrotované kolem osy z. Intuitivně se bude aproximované řešení podobat trychtýři, o čemž se můžeme hned přesvědčit.

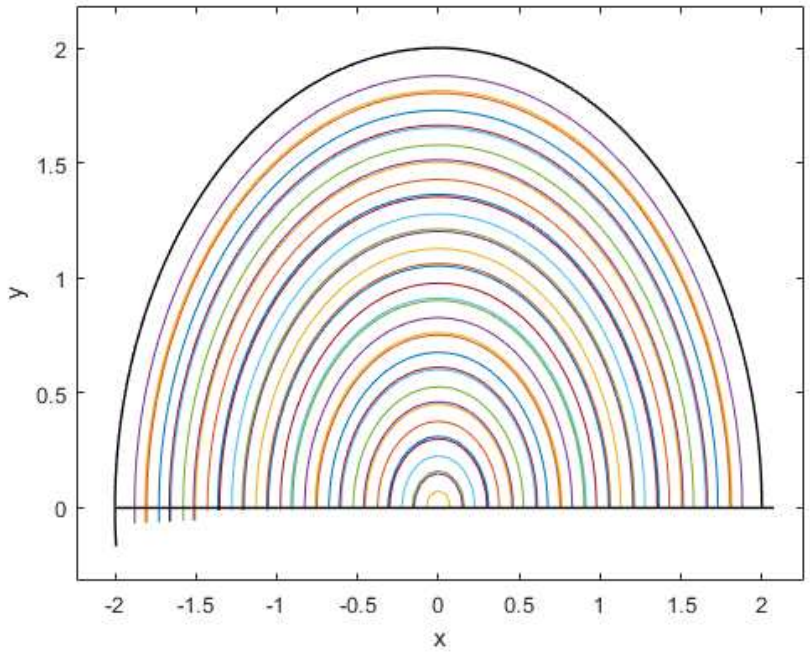


Obrázek 3.3 Aproximované řešení U, $tol = 0.15$

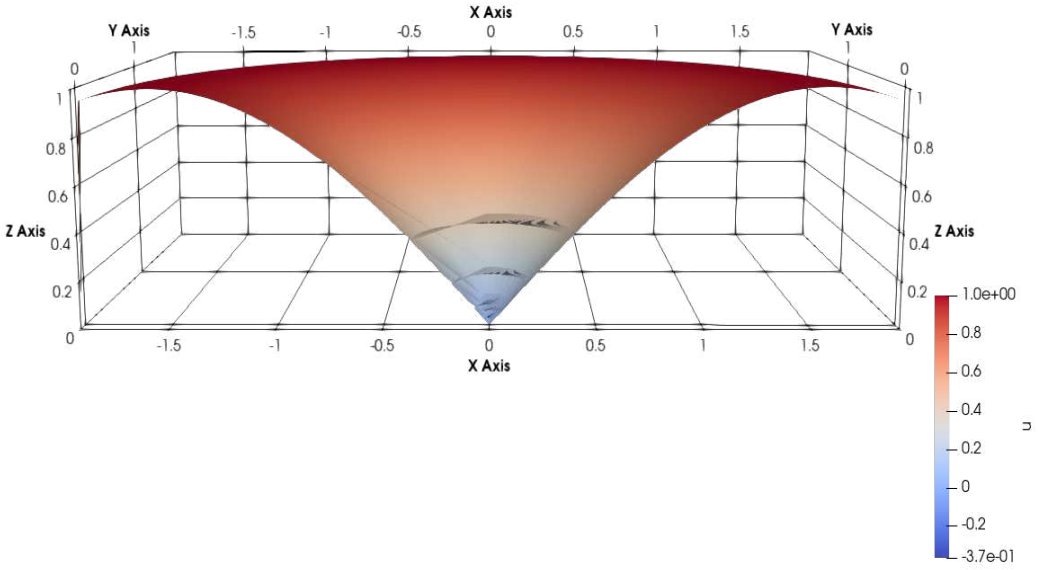
Programu trvalo najít řešení přibližně 1 s. Červená místa na grafu představují nejvyšší hodnoty, kterých aproximované řešení U dosahuje a naopak modrá místa představují ty nejnižší hodnoty. Na první pohled však můžeme vidět, že řešení je v nějakých částech zvláštně definované nebo je místy velice zubaté.

Podívejme se proto na to, jak budou vypadat charakteristiky a aproximace řešení, pokud spustíme algoritmus s $tol = 0.075$, což je 2-krát menší hodnota *tolerance*. V tomto případě algoritmus bežel 37 s. Můžeme si povšimnout, že program našel více jak 2-krát více charakteristik než v prvním případě. Body na nich jsou také hustěji rozložené, ale je jich tolik, že nemá smysl vykreslovat graf bodů.

Nakonec na grafu aproximovaného řešení můžeme vidět téměř dokonalou hladkost. Na dvou hladinách se pak vyskytují lehká vychýlení. Ty mohly být způsobeny nějakými výpočetními chybami. Každopádně z toho vyplývá, že pokud řešení není spočítáno dostatečně přesně, lze snížit hodnotu *tolerance* a dosáhnout přesnějších výsledků na úkor časové složitosti algoritmu.



Obrázek 3.4 Spojitě vykreslené charakteristiky, $tol = 0.075$



Obrázek 3.5 Aproximované řešení U, $tol = 0.075$

3.2 Příklad s nenulovou funkcí f

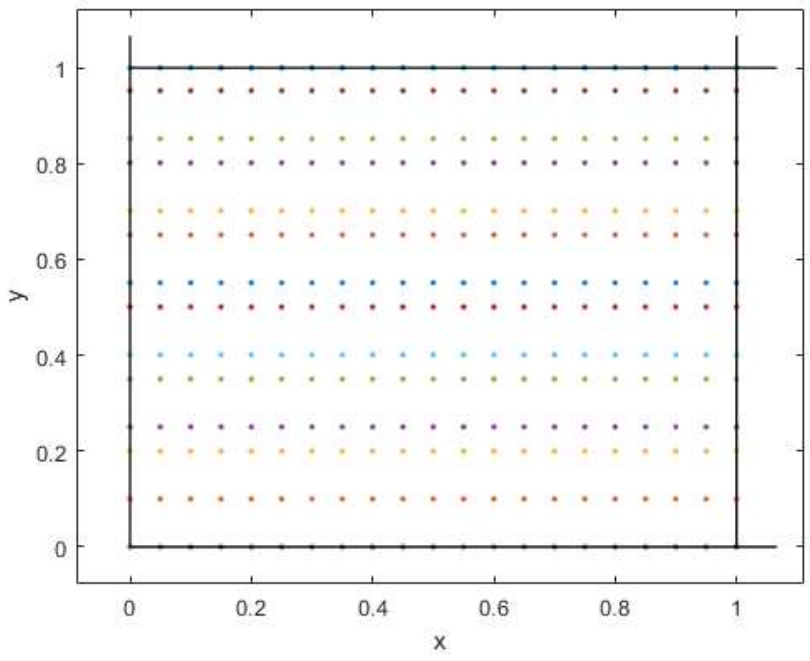
$$\begin{cases} u_x = -16x + 8, & \text{v } \Omega = (0,1)^2, \\ u(x,y) = 0, & \text{pro } (x,y) \in \{0\} \times [0,1]. \end{cases}$$

Úlohu řešíme na jednotkovém čtverci, kde vstupní hranicí je úsečka vedoucí z bodu $[0,0]$ do bodu $[0,1]$. Spočtěme si charakteristické křivky:

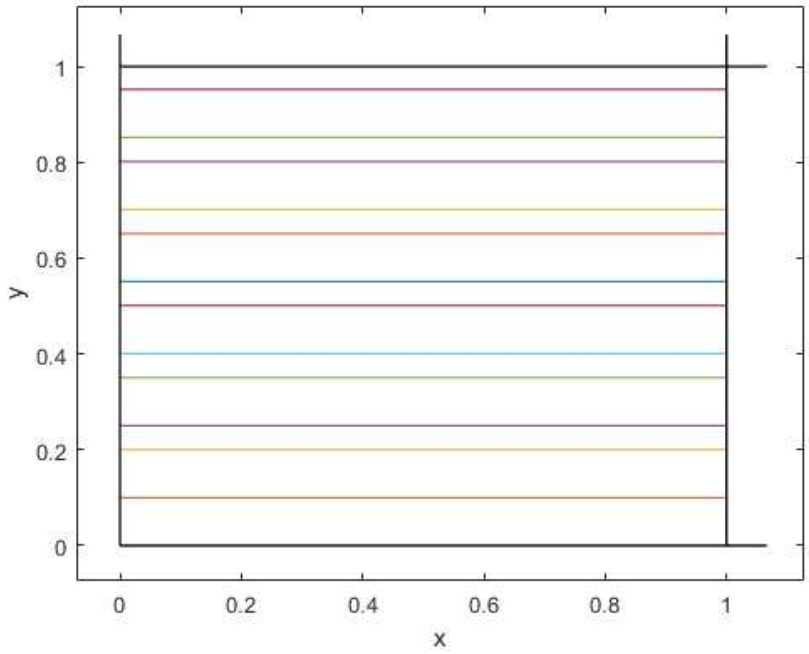
$$\frac{d}{ds}\begin{pmatrix}\phi_1(s)\\ \phi_2(s)\end{pmatrix} = \begin{pmatrix}1\\0\end{pmatrix},$$

$$\begin{pmatrix}\phi_1(s)\\ \phi_2(s)\end{pmatrix} = \begin{pmatrix}s+a\\ b\end{pmatrix}, a, b \in \mathbb{R}.$$

Vidíme, že půjde o přímky rovnoběžné s osou x. Program jsem spustil s $tol = 0.10$ a trvalu mu doběhnout přibližně 1 s. Na obrázku můžeme vidět, jakými body algoritmus aproximoval charakteristiky. Proložením těchto bodů křivkami dostáváme ony přímky.

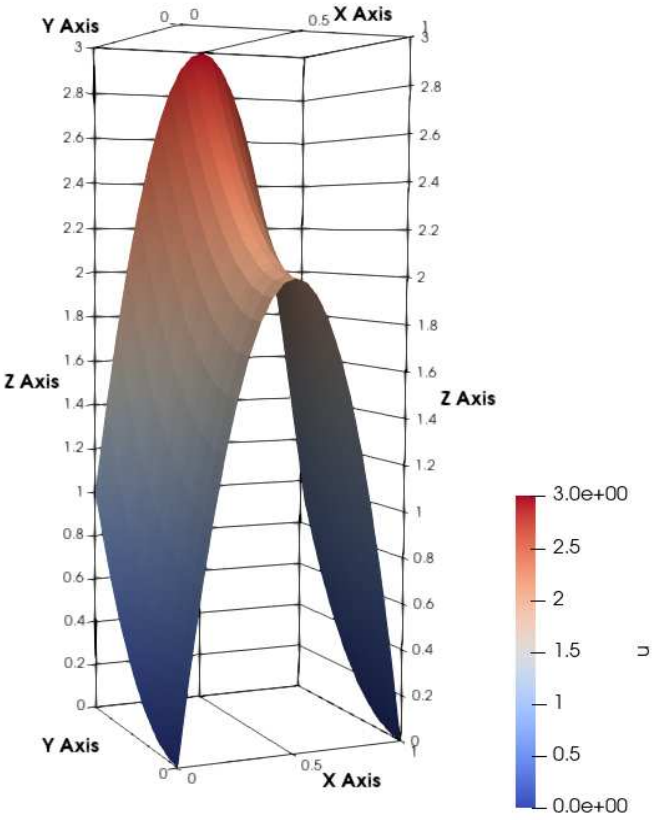


Obrázek 3.6 Body aproximující charakteristiky, $tol = 0.10$



Obrázek 3.7 Spojitě vykreslené charakteristiky, $tol = 0.10$

Mohli jsme si povšimnout, že pravá strana rovnice v úloze je nenulová. Hlavním cílem tohoto příkladu je otestovat funkčnost numerické integrace v algoritmu. Z lemmatu 2 plyne, že se v řešení bude vyskytovat integrál z funkce f. Charakteristiky a funkce f jsou polynomy 1. stupně, tudíž pokud dosadíme křivky do funkce f, dostaneme opět polynom 1. stupně. V řešení této úlohy se pak f integruje v bodech $\phi(s)$. Dostaneme tedy kvadratickou funkci se záporným koeficientem u kvadratického členu. To znamená, že graf funkce f podél křivek $\phi(s)$ bude parabola. Navíc máme ještě zadanou okrajovou podmínku, takže k těmto parabolám budeme přičítat příslušnou hodnotu na vstupní hranici. Tato hodnota se bude s rostoucím y zvyšovat, protože okrajovou podmínkou je funkce y^2. Intuitivně tedy dostaneme graf podobný tunelu jehož výška bude růst. Můžeme se o tom přesvědčit na obrázku níže.



Obrázek 3.8 Aproximované řešení U, $tol = 0.10$

3.3 Příklad s charakteristikami jako exponenciálami

$$\begin{cases} 2xu_x + u_y + x^2u = 0, & \text{v } \Omega = \{(x,y) \in \mathbb{R}^2 : x^2 + y^2 < 4;\ x > 0;\ y > 0\}, \\ u(x,y) = \sin(x), & \text{pro } (x,y) \in [0,2] \times \{0\}. \end{cases}$$

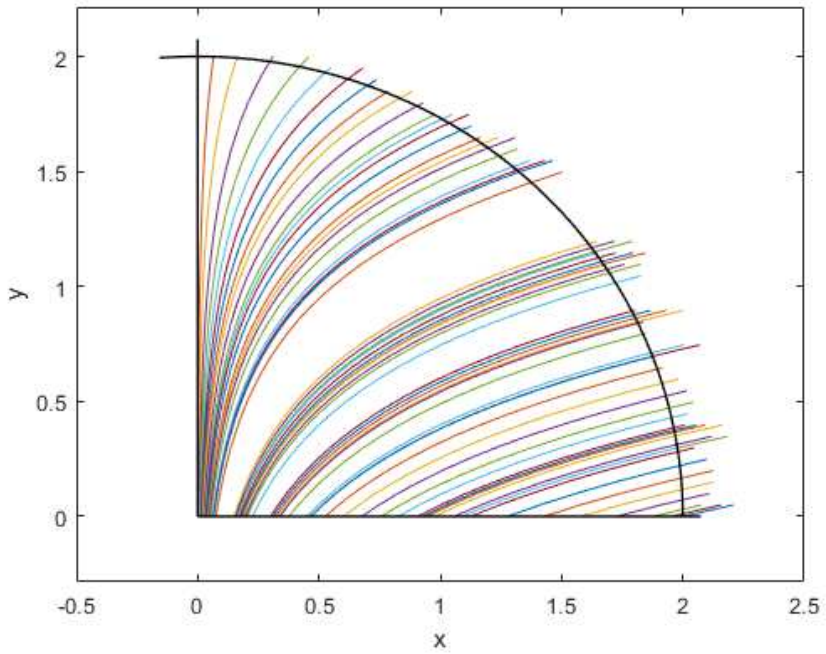
V této úloze si ukážeme, jak algoritmus pracuje s rychle rostoucími exponenciálami jakožto charakteristikami a jak vypadá aproximace řešení, když se v úloze vyskytuje nulová derivace funkce u. Spočtěme opět charakteristiky:

$$\frac{d}{ds}\begin{pmatrix}\phi_1(s)\\ \phi_2(s)\end{pmatrix} = \begin{pmatrix}2\phi_1(s)\\ 1\end{pmatrix},$$

$$\begin{pmatrix}\phi_1(s)\\ \phi_2(s)\end{pmatrix} = \begin{pmatrix}a\exp(2s)\\ s+b\end{pmatrix}, a, b \in \mathbb{R}.$$

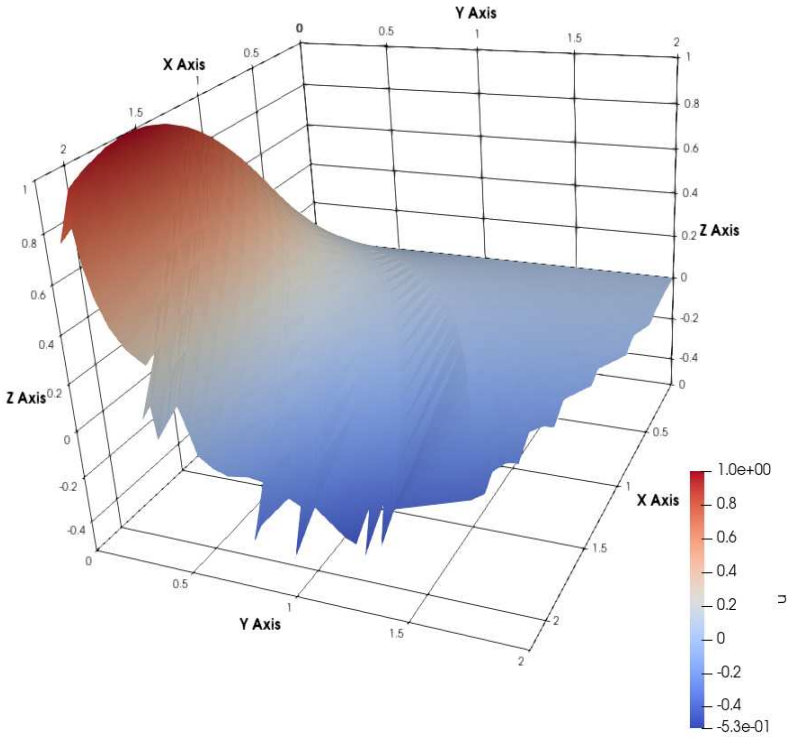
Algoritmus v tomto příkladě spouštím s volbou $tol = 0.075$. K takto malé hodnotě jsem došel postupným zkoušením různých čísel. Přestože pracujeme na čtvrtkružnici s poloměrem 2, tak se stihnou charakteristické křivky od sebe oddálit více,

než bychom chtěli. Programu trvá kolem 34 s až 38 s než dojde k výsledku. Většinu tohoto času stráví algoritmus právě snahou o dostatečně husté pokrytí množiny Ω charakteristickými křivkami. Na obrázku níže můžeme vidět, že existují části množiny Ω, které stále nejsou charakteristikami pokryty dostatečně (obrázek pouze s aproximovanými body vykreslovat nebudeme, protože je jich obrovské množství a obrázek by tedy nebyl užitečný).



Obrázek 3.9 Spojitě vykreslené charakteristiky, $tol = 0.075$

Nyní se můžeme už podívat na to, jak vypadá aproximace řešení této úlohy. Jakmile se v rovnici vyskytuje nenulová funkce $c(x, y) = x^2$, kterou násobíme nultou derivaci u, je obtížné přijít s intuitivní představou popisující, jak bude graf vypadat. Pojďme si proto hned ukázat, jak vypadá aproximované řešení (Pozor: tento graf, na rozdíl od minulých příkladů, není orientovaný vstupní hranicí čelem k nám)



Obrázek 3.10 Aproximované řešení U, $tol = 0.075$

Na první pohled je vidět nepravidelnost grafu na části hranice množiny Ω. Zde je to ta část, ze které charakteristiky vystupují ven a kde jsou od sebe různě daleko. Právě proto dostáváme takto zubaté řešení.

Závěr

V této práci jsme se soustředili na navržení a implementaci numerického algoritmu aproximující řešení lineárních parciálních diferenciálních rovnic 1. řádu pomocí metody charakteristik.

Postup při numerickém řešení úloh jsme rozdělili na sedm částí, které jsme si podrobně popsali v 2. kapitole. Pro pokrytí množiny Ω trojúhelníkovou sítí jsme se snažili hledat vrcholy této sítě tak, aby jejich vzdálenost byla menší než nějaká zvolená konstanta. Díky tomu jsme získali dostatečně jemnou síť a aproximace řešení byla přesnější. V algoritmu jsme použili numerické metody jako Rungeovu-Kuttovu metodu, složené lichoběžníkové pravidlo a barycentrickou interpolaci.

Ve 3. kapitole jsme otestovali naprogramovaný algoritmus na příkladech a vykreslili podobu charakteristik a aproximovaných řešeních. V prvním příkladu bylo vidět, jak vypadá řešení úlohy se zjednodušenou LPDR a jak se mění jeho přesnost v závislosti na hodnotě toleranční konstanty. V druhém příkladu jsme otestovali numerickou integraci v algoritmu a zanalyzovali, jaký vliv má nenulová pravá strana rovnice na řešení. V třetím příkladu jsme zadali LPDR obsahující nulovou derivaci funkce u. Zároveň jsme zpozorovali, že jakmile reprezentujeme charakteristiky pomocí exponenciál, tak je velmi těžké dostatečně pokrýt určité části množiny Ω.

Další zlepšení

Pravděpodobně by šel zlepšit způsob pokrývání množiny Ω charakteristikami tak, abychom ji dokázali dostatečně pokrýt i rychle rostoucími/klesajícími křivkami. Dále by bylo možné lépe vytvářet elementy sítě na části hranice množiny Ω, ze které vystupují charakteristiky ven. V posledním příkladu 3. kapitoly bychom pak mohli dostat hladší hranici plochy reprezentující aproximaci řešení.

Literatura

1. Velazquez, G.L. *Partial Differential Equations Of First Order And Their Applications To Physics (2nd Edition)*. World Scientific Publishing Company, 2012. isbn 9789814397506. Dostupné také z: https://books.google.cz/books?id=Jcw5DwAAQBAJ.
2. Evans, L.C. *Partial Differential Equations*. American Mathematical Society, 2010. Graduate studies in mathematics. isbn 9780821849743. Dostupné také z: https://books.google.cz/books?id=Xnu0o_EJrCQC.
3. Hillen, T.; Leonard, I.E.; Roessel, H. van. *Partial Differential Equations: Theory and Completely Solved Problems*. FriesenPress, 2019. isbn 9781525550249. Dostupné také z: https://books.google.cz/books?id=0aOcDwAAQBAJ.
4. Kaplický, Petr. *Skript k přednášce Úvod od parciálních diferenciálních rovnic*. 2023. Dostupné také z: https://www.karlin.mff.cuni.cz/~kaplicky/pages/pages/2023z/nmma339.php.
5. Paschalakis, S.; Lee, P. Double precision floating-point arithmetic on FPGAs. In: *Proceedings. 2003 IEEE International Conference on Field-Programmable Technology (FPT) (IEEE Cat. No.03EX798)*. 2003, s. 352–358. Dostupné z doi: 10.1109/FPT.2003.1275775.
6. Butcher, J.C. *Numerical Methods for Ordinary Differential Equations*. Wiley, 2008. isbn 9780470753750. Dostupné také z: https://books.google.cz/books?id=opd2NkBmMxsC.
7. Shalchian, Erfan. *A generalization of Picard-Lindelof theorem/ the method of characteristics to systems of PDE*. 2018. Dostupné z arXiv: 1812.08925 [math.AP].
8. Smyth, Gordon K. Numerical integration. *Encyclopedia of biostatistics*. 1998, s. 3088–3095.
9. Floater, Michael; Hormann, Kai; Kós, Géza. A general construction of barycentric coordinates over convex polygons. *Adv. Comput. Math.* 2006, roč. 24, s. 311–331. Dostupné z doi: 10.1007/s10444-004-7611-6.

Seznam obrázků

A Přílohy

K této práci přikládám přílohu. Jedná se o archiv ve formátu .zip obsahující zdrojové kódy a dokumentaci k programu.

A.1 Zdrojové kódy

V příloze se nachází implementace popsaného algoritmu v podobě zdrojových kódů v jazyce Matlab. Tyto zdrojové kódy se nachází ve složce `src`.

A.2 Dokumentace

V příloze se nachází dokumentace k programu ve formátu PDF.