Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra matematiky

BAKALÁŘSKÁ PRÁCE

Numerické metody pro advekční rovnici
Numerical methods for advection equation

PLZEŇ, 2012 Zdeňka Baxová

Prohlášení

Předkládám tímto k posouzení a obhajobě bakalářskou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem bakalářskou práci vypracovala samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

PLZEŇ, 2012

Zdeňka Baxová

Poděkování

Ráda bych poděkovala Doc. Ing. Marku Brandnerovi, Ph.D. za odborné vedení mojí práce a rady během jejího zpracovávání.

Abstrakt

Práce se zabývá metodami pro řešení advekční rovnice. Obsahuje teoretický popis těchto metod a popis jejich implementace v programovém vybavení MATLAB. V závěru prezentujeme získané výsledky z provedených experimetů a stanovujeme nejlepší metodu.

Klíčová slova : advekce, numerické metody, upwind, Laxova-Wendroffova metoda, Hartenova-Zwasova metoda, semilagrangeovská metoda, BFECC

Abstract

This thesis deals with numerical methods for solving advection equation. The paper contents theoretic description of the methods and their implementation in MATLAB software. In conclusion we present experimental outcomes and provides the best method.

Keywords : advection, numerical methods, upwind, Lax-Wendroff method, Harten-Zwas method, semilagrangian method, BFECC

Obsah

Seznam obrázků

Úvod

Lidé se odedávna snažili namodelovat dynamiku fyzikálních jevů jako tok řeky, oheň, proudění větru, apod. Na rozdíl od minulosti dnes máme k dispozici velký výpočetní výkon, a proto je možné tuto kapacitu využít a vytvářet numerické modely, které dostatečně přesně odpovídají realitě. Takovéto modely využívají advekční rovnici jako základ pro simulaci dynamiky tekutin či jiného pohybu fyzikálně odpovídajícímu proudění. Lze tedy například provést simulaci chování vody v říčním korytě, pohyby mraků nebo dokonce proudění kouře.

Zmíněné modely neslouží jen vědcům a jejich laboratořím. Velmi často jsou používané v počítačové grafice a při tvorbě počítačových her, ve kterých se vývojáři snaží vytvořit co nejreálnější prostředí (kouř, voda, oheň). Nutno podotknout, že s rostoucím výkonem dnešních počítačů se jim to daří více než dobře.

Pro modelování těchto jevů ovšem nestačí jen advekční rovnice, ale je třeba aplikovat některou z nepřeberného množství metod.

Cílem této práce je poskytnout čtenáři přehled o základních metodách pro řešení advekční rovnice. Pokud by se měl text zaobírat kompletní problematikou numerického modelování problémů s advekční rovnicí či problémů dynamiky tekutin, musel by být o mnoho rozsáhlejší a ve výsledku by mohl působit značně chaoticky. Proto se práce zaměřuje jen na určitý omezený okruh problematiky zahrnující vybranou pětici základních metod. Tyto metody byly vybrány na základě jejich vlastností a typů. Je však nasnadě říci, že je text zaměřen spíše matematicky.

Součástí práce je kromě teorie také rozbor implementace. Čtenář nalezne potřebné informace včetně konkrétních úryvků kódu. Z praktických důvodů je však tento kód napsán v pseudojazyku a nikoliv přímo v nějakém konkrétním jazyku. Hlavním důvodem pro tento krok je zejména omezená šířka stránky a přehlednost uvedených algoritmů.

První kapitola se zabývá teorií potřebnou k pochopení pojmů a uvedených metod. Konkrétně se jedná o metodu typu upwind, Laxovu-Wendroffovu metodu, Hartenovu-Zwasovu metodu , algoritmus BFECC a semilagrangeovskou metodu. Informace o prvních třech zmíněných metodách byly čerpány z [5], k algoritmu BFECC pak z [4] a k semilagrangeovské metodě z [7]. Součástí teoretické části bude samozřejmě vysvětlení základních pojmů.

V druhé kapitole je provedena analýza problematiky z pohledu implementace. Jinými slovy je zde provedena analýza a diskuze jednotlivých možností implementace

metod a použití určitých technik k jejich implementaci z teoretického hlediska.

Kapitola číslo tři se pak zaobírá samotnou implementací metod pro advekční rovnici. Jsou zde rozebrány algoritmy jednotlivých metod a popsány způsoby jakými byly dosaženy výsledky uvedené ve čtvrté a poslední kapitole této práce.

Kapitola 1

Teorie

1.1 Advekce

Výraz advekce je používán pro přenos hmoty prouděním. Například se může jednat o proudění vody v řece, proudění vzduchu, dynamiku kouře a ohně. Všechny tyto děje se dají částečně popsat pomocí advekční rovnice:

$$q_t + uq_x = 0 \tag{1.1}$$

kde *q(x,t)* je funkce reprezentující neznámou veličinu v bodě x a v čase t. Funkce $u = u(x)$ reprezentuje rychlost. V případě 1D bude proměnná x vektor a rychlost bude dána zvolenou konstantou.

Pro advekční rovnici je nutné definovat počáteční a okrajové podmínky, jelikož většina úloh není řešena na celém oboru reálných čísel, ale jen na konečném intervalu. Například na úseku délky L, kde L je předem zvolené. Obvykle se počáteční podmínky definují následovně:

$$q(x, 0) = q_0(x) \tag{1.2}$$

Okrajové podmínky se definují na základě typu řešené úlohy. Tyto podmínky jsou obvykle specifikovány fyzikálním jevem, který chceme modelovat. Některé z nich mohou být definované na pravém okraji, jiné na levém, vždy využijeme k jejich definici rychlost a její směr.

$$u > 0 \quad \Rightarrow \quad q_L(t) \quad \vee \quad u < 0 \quad \Rightarrow \quad q_R(t), \tag{1.3}$$

kde $q_L(t)$, resp. $q_R(t)$ je hodnota, která popisuje chování na levém, resp. pravém okraji zvoleného intervalu.

Advekční rovnici lze řešit analyticky, není to však tak jednoduché, jak by se na první pohled mohlo zdát. Advekční rovnice je parciální diferenciální rovnicí a její řešení pro počáteční podmínku zadanou pomocí hodnot není jednoduché. Právě z tohoto důvodu se řeší spíše numericky. Pro kompletnost však bude dále uvedeno i

řešení analytické.
Mějme advekční rovnici v základním tvaru a počáteční podmínku:

$$q_t + uq_x = 0, \qquad q(x,0) = q_0(x) \tag{1.4}$$

Definujme řešení po přímkách:

$$x = ut + c \tag{1.5}$$

Vezměme obecný bod $\widehat{x}$ v čase $\widehat{t}$ a definujme bod $\tilde{x}$, který určuje polohu zvoleného bodu $\widehat{x}$ v čase 0:

$$q(\ \widehat{x}, \widehat{t}\ ) = q(\tilde{x}, 0) = q_0(\tilde{x}), \tag{1.6}$$

Potom $\widehat{x} = u\widehat{t} + c$. Z této rovnice vyjádříme c a dosadíme jej do obecné rovnice přímky $x = ut + \widehat{x} - u\widehat{t}$. Nyní dosadíme do vztahu pro q:

$$q(\ \widehat{x}, \widehat{t}\ ) = q(\ \widehat{x} - u\ \widehat{t}, 0) = q_0(\widehat{x} - u\ \widehat{t}) \tag{1.7}$$

Z této rovnice vyplyne obecné řešení advekce:

$$q(x,t) = q_0(x - ut) \tag{1.8}$$

V praxi se však často setkáváme s vícerozměrnými případy. Proto jsou všechny simulace v této práci založeny na dvourozměrných datech i dvourozměrném rychlostním poli. Obecně by se tato rovnice dala zapsat stejným způsobem jako rovnice pro 1D případ uvedená výše, ale pro přehlednost bude lépe uvést jí v rozepsaném tvaru.

$$q_t + \mathbf{u} \cdot \nabla q = 0 \tag{1.9}$$

I pro tento tvar advekce platí, že $q = q(x,y,t)$ jsou data v bodě (x,y) a čase t, dále $\mathbf{u} = (u(x,y), v(x,y))$ je vektor, jehož složky jsou rychlostní pole ve směru x (složka u), resp. y (složka v).

Pro advekční rovnici je i zde nutné definovat počáteční a okrajové podmínky, jelikož většina úloh není řešena na celém prostoru, ale jen na konkrétním intervalu. Například $(x,y) \in < 0, K > \times < 0, L >$, kde K a L jsou předem zvolené. Obvykle se počáteční podmínky definují následovně:

$$q(x,y,0) = q_0(x,y) \tag{1.10}$$

Okrajové podmínky se definují na základě typu řešené úlohy. Tyto podmínky jsou specifikovány na základě modelovaného fyzikálního jevu. V tomto případě okrajové podmínky definují pomocí součinu vnější normály plochy či oblasti s vektorem rychlosti. Zde záleží na znaménku, které nám udává směr toku.

Analytické řešení advekční rovnice pro dvě neznámé (x a y) je obdobné jako pro jednorozměrný případ. Pro kompletnost zde bude uvedeno. Mějme advekční rovnici v základním tvaru a počáteční podmínku:

$$q_t + uq_x + vq_y = 0, \qquad q(x,y,0) = q_0(x,y) \tag{1.11}$$

Definujme řešení po přímkách:

$$x = ut + c, \quad y = vt + d \tag{1.12}$$

Vezměme obecný bod $(\widehat{x}, \widehat{y})$ v čase $\widehat{t}$ a definujme bod $(\tilde{x}, \tilde{y})$, který určuje polohu zvoleného bodu $(\widehat{x}, \widehat{y})$ v čase 0:

$$q(\ \widehat{x}, \widehat{y}, \widehat{t}\ ) = q(\tilde{x}, \tilde{y}, 0) = q_0(\tilde{x}, \tilde{y}), \tag{1.13}$$

Pak $\widehat{x} = u\widehat{t} + c$, $\widehat{y} = v\widehat{t} + d$. Z těchto rovnic vyjádříme c a d a dosadíme je do rovnic pro přímku $x = ut + \widehat{x} - u\widehat{t}$ a $y = vt + \widehat{y} - v\widehat{t}$. Nyní dosadíme do vztahu pro q:

$$q(\ \widehat{x}, \widehat{y}, \widehat{t}\ ) = q(\ \widehat{x} - u\ \widehat{t}, \widehat{y} - v\ \widehat{t}, 0) = q_0(\widehat{x} - u\ \widehat{t}, \widehat{y} - v\ \widehat{t}) \tag{1.14}$$

Z této rovnice vyplyne obecné řešení advekce:

$$q(x, y, t) = q_0(x - ut, y - vt) \tag{1.15}$$

Nyní je ovšem nutné stanovit, zda je výše uvedené řešení jednoznačné. Vyslovme tedy následující tvrzení:

Je-li počáteční podmínka q_0 spojitě diferencovatelná, funkce **u** je také spojitě diferencovatelná $\forall x$. Každým bodem množiny $M = ((x,t), x \in < 0, L >, t \in < 0, T >$ prochází jediná charakteristika a tato charakteristika protíná interval $< 0, L >$ v jediném bodě. Pak existuje jediné řešení úlohy. Toto řešení se nazývá klasické.

Řešení se dá zobecnit i na nespojitá data, ale tímto se v této práci nebudeme zabývat. Trzení lze formulovat i pro vícerozměrný případ počáteční podmínky.
Jelikož většinou nemáme zadáno analyticky q_0 a zároveň potřebujeme efektivně vyhodnotit neznámou q, tak se pro řešení používají numerické metody, jejichž typy a vlastnosti budou uvedeny dále v tomto textu.

1.2 Numerické metody

Obecně se numerické metody pro řešení advekční rovnice dělí na dva druhy a to metody konečných diferencí neboli eulerovské a metody využívající chrakteristik neboli lagrangeovské. Oba dva druhy metod mají specifické vlastnosti. V dalším textu budou popsány zejména metody eulerovského typu. Jedinou vyjímku bude tvořit semilagrangeovská metoda, která spojuje oba přístupy.

Metody konečných diferencí využívají pro řešení problému poměrné diference, tedy rozdíl hodnot mezi dvěma body mřížky, kterými nahrazují derivace. Všechny tyto metody jsou snadné na pochopení a implementaci ale mají velkou nevýhodu, že nejsou vhodné pro nehladká data. Některé z těchto metod jsou v bodech nespojitosti náchylné k oscilacím. Jak bude dále uvedeno mezi metody konečných diferencí

patří například metoda upwind, Laxova-Wendroffova metoda a Hartenova-Zwasova metoda. Tyto metody budou porovnány, jak mezi sebou, tak s metodou semilagrangeovskou.

Druhým typem metod jsou metody lagrangeovské. Tyto metody využívají k řešení charakteristiky dat neboli v tomto případě jejich rychlostní pole. Tyto metody narozdíl od eulerovských nemají pevně stanovenou mřížku pro data, respektive má tato mřížka proměnlivý krok. Jako příklad uveďme siločáry okolo tyčového magnetu. Ty jsou v blízkosti magnetu více nahuštěné než ve větších vzdálenostech od objektu. Body mřížky by pro tento příklad byly rozmístěny stejným způsobem. Na rozdíl od pevně dané mřížky v případě diferenčních metod můžou lagrangeovské metody lépe vystihnout rozmístění bodů v prostoru včetně míst nespojitosti dat. lagrangeovské metody jsou proto obvykle stabilnější než metody konečných diferencí.

Jednotlivé numerické metody uvedené v této části budou řazeny podle svého řádu a složitosti. Jinými slovy bude nejprve uvedena metoda upwind zastupující metody prvního řádu. Následně budou zmíněny metody druhého řádu (např. Laxova-Wendroffova metoda) a nakonec se text zaměří na metodu BFECC, která ke svému běhu využívá jednodušší metody upwind a semilagrangeovskou metodu. V případě metody BFECC se nejedná o samostatnou metodu řešení advekční rovnice, ale o algoritmus zpřesňující řešení.

Pro potřeby této práce bylo nutné zvolit mřížku, na kterou budou aplikovány jednotlivé metody a která bude rozměrově odpovídat zvoleným datům. Byla zvolena čtvercová, pravidelná mřížka s konstantním krokem. Tato mřížka byla zavedena následovně:

$$\begin{aligned} \Delta x > 0, \qquad & \Delta t > 0 \\ t_n = n\Delta t \qquad & n \in \mathbb{N} \\ x_j = j\Delta x \qquad & j \in \mathbb{N} \end{aligned}$$

Tyto vztahy popisují jednoduchou mřížku aplikovanou na daná data. Δt a Δx jsou zvolené kroky času a délky, j je celé číslo udávající velikost dat a n je celé číslo udávající počet časových kroků.
V této části textu bude použito označení q pro přesnou hodnotu dat a Q pro aproximaci dat.

1.2.1 Metoda typu upwind

Tato metoda je nejjednodušší metodou z metod uvedených v této práci. Upwind lze odvodit přímo z advekční rovnice rozvinutím do Taylorova polynomu, kde užijeme všechny členy až do členů s derivacemi prvního řádu. Tyto derivace nahradíme poměrnými diferencemi prvního řádu. Jedná se tedy o metodu prvního řádu.

Metoda ke své činnosti využívá dopředných a zpětných diferencí. Diferencí rozumíme rozdíl hodnot dvou bodů nacházejících se vedle sebe ve zvoledné mřížce dat dělený krokem mřížky :

$$\frac{Q_j - Q_{j-1}}{\Delta x}$$

V závislosti na znaménku hodnoty rychlostního pole bereme buď bod před nebo bod za aktuálním bodem vzhledem ke směru vektoru rychlosti. V uvedené diferenci je aktuální bod x_j s hodnotou Q_j bodem, ve kterém stanovujeme novou hodnotu.
Při implementaci metody je možné brát v úvahu jen levé nebo jen pravé diference. Přesnější je ovšem kompaktní verze metody, která bere v úvahu oba druhy diferencí. Kompaktní metoda upwind má následující tvar :

$$Q_j^{n+1} = Q_j^n - \frac{\Delta t}{\Delta x}(u_j^+(Q_j^n - Q_{j-1}^n) + u_j^-(Q_{j+1}^n - Q_j)), \tag{1.16}$$

kde u_j^+, resp. u_j^-, jsou definované jako maximum, respektive minimum, z příslušné hodnoty rychlostního pole a hodnoty 0, tedy $u_j^+ = max\{u_j, 0\}$, resp. $u_j^- = min\{u_j, 0\}$. Vzorec metody ve zmíněném tvaru odpovídá metodě upwind pro jednorozměrná data. Jelikož budou v této práci používána výhradně dvojrozměrná data, bude nutné používat také metodu upwind ve tvaru odpovídajícím vstupním datům :

$$\begin{aligned} Q_{i,j}^{n+1} = Q_{i,j}^n - \frac{\Delta t}{\Delta x}(u_{i,j}^+(Q_{i,j}^n - Q_{i,j-1}^n) \ + \ u_{i,j}^-(Q_{i,j+1}^n - Q_{i,j}^n)) - \\ -\frac{\Delta t}{\Delta x}(v_{i,j}^+(Q_{i,j}^n - Q_{i-1,j}^n) \ + \ v_{i,j}^-(Q_{i+1,j}^n - Q_{i,j}^n)), \end{aligned} \tag{1.17}$$

kde $v_{i,j}^+$, resp. $v_{i,j}^-$, jsou definované jako maximum, respektive minimum, z příslušné hodnoty rychlostního pole a hodnoty 0 ve směru kolmém na u, tedy $v_{i,j}^+ = max\{v_{i,j}, 0\}$,resp. $v_{i,j}^- = min\{v_{i,j}, 0\}$.

Výhodou metody je její implementační jednoduchost, rychlost výpočtu a paměťová nenáročnost.
Hlavní nevýhodou metody typu upwind je fakt, že zkresluje data, čímž se stává prakticky nepoužitelnou. V případě již zkreslených dat metoda nepřesnosti umocňuje a výsledná chyba naprosto znehodnotí získané výsledky.

1.2.2 Laxova-Wendroffova metoda

Stejně jako předchozí metoda je i tato odvoditelná z Taylorova rozvoje advekční rovnice s jedním významným rozdílem. Je nutné brát i člen reprezentující druhý řád Taylorova polynomu. Jedná se tedy o metodu druhého řádu.
Dalším možným způsobem pro její získání je přidání korekčního členu do rovnice pro metodu upwind. Následně pak dostaneme tvar Laxovy-Vendroffovy metody pro jednorozměrná data:

$$Q_j^{n+1} = Q_j^n - \frac{\Delta t}{\Delta x}(u_j^+(Q_j^n - Q_{j-1}^n) + u_j^-(Q_{j+1}^n - Q_j^n)) + \frac{1}{2}|u_j|(1 - \frac{\Delta t}{\Delta x}|u_j|)(Q_{j+1}^n - 2Q_j^n + Q_{j-1}^n) \tag{1.18}$$

Tato metoda funguje velice dobře na již zkreslená data zejména v porovnání s metodou upwind. Na druhou stranu může nastat problém u dat, která mají příliš strmý gradient nebo pro data s nespojitostmi. V takových případech se v procesu této metody vytvářejí oscilace a tím se snižuje přesnost metody.
Ovšem v tomto tvaru je metoda téměř nepoužitelná kvůli velkým oscilacím a proto v této práci bude použita metoda v následujícím tvaru převzatá z článku [5]. Tento tvar metody taktéž produkuje oscilace při nespojitosti dat, ovšem tyto oscilace jsou menší než u výše uvedeného tvaru této metody.

$$Q_j^{n+1} = Q_j^n - \frac{\Delta t}{2\Delta x} u_j (Q_{j+1}^n - Q_{j-1}^n) + \frac{1}{2}\left(\frac{\Delta t}{\Delta x}\right)^2 u_j^2 \, (Q_{j+1}^n - 2Q_j^n + Q_{j-1}^n) \quad (1.19)$$

Ten je ovšem pouze pro jednorozměrná data a proto zde uvedu i metodu ve tvaru použitém pro experimenty:

$$Q_{i,j}^{n+1} = Q_{i,j}^n - \frac{\Delta t}{2\Delta x} \, u_{i,j} \, (Q_{i,j+1}^n - Q_{i,j-1}^n) + \frac{1}{2}\left(\frac{\Delta t}{\Delta x}\right)^2 \, u_{i,j}^2 \, (Q_{i,j-1}^n - 2Q_{i,j}^n + Q_{i,j+1}^n)$$

$$-\frac{\Delta t}{2\Delta x} \, v_{i,j} \, (Q_{i+1,j}^n - Q_{i-1,j}^n) + \frac{1}{2}\left(\frac{\Delta t}{\Delta x}\right)^2 \, v_{i,j}^2 \, (Q_{i-1,j}^n - 2Q_{i,j}^n + Q_{i+1,j}^n) \quad (1.20)$$

1.2.3 Hartenova-Zwasova metoda

V tomto případě se jedná o metodu, která je odvozená z výše uvedené metody. Jedná se o zpřesnění a zobecnění Laxovy-Wendroffovy metody. Tvar metody bude velice podobný rovnici uvedené výše. Pouze korekční člen bude přenásoben tzv. limiterem Φ. Základní myšlenka je taková, že pokud bude limiter vhodně zvolen, budou výsledky metody přesnější a omezí se rozkmitání, které by způsobovalo v případě klasické Laxovy-Wendroffovy metody větší chybu. Předpis metody pak bude vypadat následovně [5]:

$$Q_j^{n+1} = Q_j^n - \frac{\Delta t}{\Delta x}(u_j^+(Q_j^n - Q_{j-1}^n) + u_j^-(Q_{j+1}^n - Q_j^n)) + \frac{1}{2}|u_j|(1 - \frac{\Delta t}{\Delta x}|u_j|)(Q_j^n - Q_{j-1}^n)\Phi_j^n. \quad (1.21)$$

Vhodnou definicí limiteru může být například funkce ve tvaru:

$$\Phi_j^n = \phi(\theta_j^n), \qquad \theta_j^n = \frac{Q_J^n - Q_{J-1}^n}{Q_j^n - Q_{j-1}^n},$$

kde J je definované na základě znaménka u příslušné hodnoty rychlostního pole (podobně jako prvky u^+ a u^- u metody upwind), tedy $u > 0 \rightarrow J = j - 1$ a pro $u \leq 0 \rightarrow J = j + 1$. Z výše uvedeného vztahu pro θ_j^n je vidět, že se jedná o poměr diferencí. Diference ve jmenovateli je dána jako diference ve směru rychlostního pole a diference v čitateli je zpětnou diferencí pro aktuální bod.
Limiterů existuje celá řada. Například [8]:

van Albada $\phi(\theta) = \frac{\theta^2 + \theta}{\theta^2 + 1}$

koren $\phi(\theta) = max(0, min(2\theta, \frac{(1+2\theta)}{3}, 2))$

minmod $\phi(\theta) = max(0, min(1, \theta))$

van Leer $\phi(\theta) = \frac{\theta + |\theta|}{1 + |\theta|}$

Limiter van Leer bude použit v této práci při veškerých experimentech s touto metodou. Z předpisu pro tuto metodu je patrné, že limiter Φ_j může do značné míry měnit tvar metody. Pro $\Phi_j = 0 \quad \forall j$ se vynuluje celý korekční člen a výsledkem je rovnice základní metody typu upwind. Při dosazení $\Phi_j = 1 \quad \forall j$ je pro změnu výsledkem Laxova-Wendroffova metoda. Tento fakt může velice usnadnit implementaci obou výše zmíněných metod. V praxi postačí vytvořit metodu s limiterem a pomocí parametru pouze měnit tvar limiteru.

1.2.4 Semilagrangeovská metoda

V následujícím odstavci bude popsána semilagrangeovská metoda. Tato metoda se od výše popsaných metod liší, jelikož spojuje vlastnosti lagrangeovských metod a metod konečných diferencí, k nimž se řadí všechny metody popsané výše. Metoda snoubí přesnost řešení metod konečných diferencí se stabilitou metod založených na lagrangeovském přístupu.

Celá tato metoda je založena na základním vztahu pro rychlost:

$$\frac{dx}{dt} = u(x)$$

Každá hodnota bodu x se dá určit pomocí vztahu:

$$q(x,t) = q(x - ut, 0),$$

což znamená, že každý bod lze určit pomocí hodnoty rychlostního pole a daného času. Z výše uvedeného vztahu lze odvodit platnost následující rovnosti:

$$q(x,t) = q(x - u\Delta t, t - \Delta t).$$

Vztah pro metodu lze odvodit na základě koeficientu posunu α a následujícího vztahu:

$$\alpha = x_{k+1} - x_k$$
$$x_{k+1} = x_k + u(x_{k+1} - \frac{\alpha}{2}) \tag{1.22}$$

Po úpravách a dosazení za α dostaneme metodu ve tvaru:

$$x_{k+1} = x_k + u \left(\frac{x_k + x_{k+1}}{2} \right) \tag{1.23}$$

Z výše uvedených vztahů vyplývá, že výpočtem nového bodu x se dostaneme mimo body mřížky a tedy hodnotu v tomto novém bodě je třeba interpolovat z hodnot v okolních bodech. Tedy hodnota v bodě x_{k+1} je interpolována z bodů ležících na jeho okolí, do kterého se posuneme vlivem rychlosti.

Tento tvar metody je aplikovatelný na jednorozměrná data. Pro úplnost bude uveden i tvar metody pro dvojrozměrná data. V tomto případě máme bod *(x,y)* a stejné jako je uvedeno výše můžeme uvažovat platnost následujících vztahů:

$$q(x, y, t) = q(x - ut, y - vt, 0),$$

$$q(x, y, t) = q(x - u\Delta t, y - v\Delta t, t - \Delta t).$$

Tvar metody se poté lze rozepsat do následujících vztahů:

$$\alpha = x_{k+1} - x_k$$

$$\beta = y_{k+1} - y_k$$

$$(x_{k+1}, y_{k+1}) = (x_k + u(x_{k+1} - \frac{\alpha}{2}, y_k + v(y_{k+1} - \frac{\beta}{2}))$$

Po dosazení za koeficienty a úpravách dostaneme metodu v konečném tvaru:

$$(x_{k+1}, y_{k+1}) = (x_k + u \left(\frac{x_k + x_{k+1}}{2} \right), y_k + v \left(\frac{y_k + y_{k+1}}{2} \right)) \tag{1.24}$$

Dále stejně jako v 1D případě následuje interpolace hodnot v bodech určených výše uvedeným vztahem.

1.2.5 Metoda BFECC

V tomto případě se opět nejedná o jednoduchou metodu, nýbrž (stejně jako u metody Hartenovy-Zwasovy) se jedná o algoritmus zpřesnění metody použité v jeho základu. Algoritmus BFECC má iterativní charakter.

Nechť funkce $L(.,.)$ přestavuje implementaci metody upwind nebo semilagrangovské metody. Je možné zapsat rovnici pro funkci L:

$$Q^{n+1} = L(\mathbf{u}, Q^n).$$

Rovnice představuje jedno provedení funkce, respektive algoritmu definovaném v L. Metodu BFECC je pak možné zapsat ve tvaru:

$$Q^{n+1} = L(\mathbf{u}, Q^n + \frac{1}{2}(Q^n - \overline{Q})), \tag{1.25}$$

kde

$$\overline{Q} = L(-\mathbf{u}, L(\mathbf{u}, Q^n))$$

Jedná se tedy o algoritmus zpřesnění výsledků metody definované v L pomocí vícenásobného volání zmíněné metody s různými vstupními daty.

Kapitola 2

Algoritmizace

V této části práce budou popsány cíle a možná úskalí při implementaci jednotlivých metod z teoretického hlediska. Postupně budou popsány všechny potřebné části, které budou pak v další kapitole implementovány.

2.1 Cíl implementační části práce

Hlavním cílem je implementovat jednotlivé metody popsané v kapitole 2 a prověřit jejich vlastnosti. Z toho vyplývá, že bude potřeba použít různá vstupní data a různá rychlostní pole tak, aby byly vlastnosti metod potvrzeny nebo vyvráceny.

2.2 Matlab

Jednotlivé metody budou implementovány v prostředí Matlab firmy Mathworks. Mezi přednosti tohoto systému patří jednoduchost zápisu programu a obrovské množství předvytvořených funkcí. Díky tomu je možné v Matlabu velice rychle vytvořit funkční prototyp algoritmu a teprve poté se věnovat programu z pohledu estetiky, přehlednosti a rychlosti. Poslední zmíněná vlastnost tohoto programového vybavení je jeho největší nevýhodou. Rychlost výpočtů je oproti jiným programovacím jazykům velice malá. To je také důvod proč musí být použit výkonný počítač i pro relativně malé množství dat.

2.3 Data

V práci budou použita různá data, jejichž přesná podoba bude zmíněna v kapitole 4. Obecně však algoritmy budou testovány na programově vytvořených datech aproximujících nespojité i spojité objekty, černobílém a barevném obrázku. Ač se dle názvu jedná o 4 druhy naprosto rozdílných dat, mají společnou reprezentaci v počítači. Jinými slovy se ve všech případech jedná o matici řádu MxN. Dá se k nim

tedy po vytvoření/načtení přistupovat stejným způsobem.

Obrázek 2.1: Příklad dat

To je zároveň důvod, proč je nasnadě vytvořit matlabovský skript, který v reakci na určitý parametr nabídne ta data, která jsou potřeba při experimentu. V zásadě bude potřeba jen jeden textový parametr a tím je typ zvolených dat. Návrh podoby tohoto parametru ukazuje následující tabulka :

Typ dat	**Parametr**
Válec	'valec'
Kvádr	'ctverec'
Gaussova funkce	'gauss'
Černobílý obrázek	'cb_obr'
Barevný obrázek	'bar_obr'

2.4 Rychlostní pole

Je velice důležitou součástí všech experimentů, neboť metody pracují s daty právě v závislosti na podobě rychlostního pole. Základním rychlostním polem může být například konstantní pole ve směru osy x i y. Dalším polem, které bude v experimentech použito bude pole otáčející se kolem určeného středu. Takovéto pole má velice zajímavou funkci z pohledu obrázků. Vše bude ukázáno v kapitole 4.

Tvorba jednotlivých polí je založena na stejném postupu. Pro posuvné pole je třeba zvolit pouze konstanty v obou směrech, abychom určili, kam chceme posouvat. Tedy dvojrozměrné posuvné pole $\mathbf{u} = (u, v)$ má dvě složky u a v, které jsou konstantami a určují směr posunu. V případě rotačního pole $\mathbf{u} = (u, v)$ není tvorba takto jednoduchá. Zde je třeba zvolit střed rotace (v našem případě střed mřížky) a poté pomocí rovnic kružnice napočítávat body a jednotlivé hodnoty pole. Tedy souřadnice bodů (x, y) jsou stanoveny:

$$x = stred + r \sin \varphi$$

$$y = stred + r \cos \varphi,$$

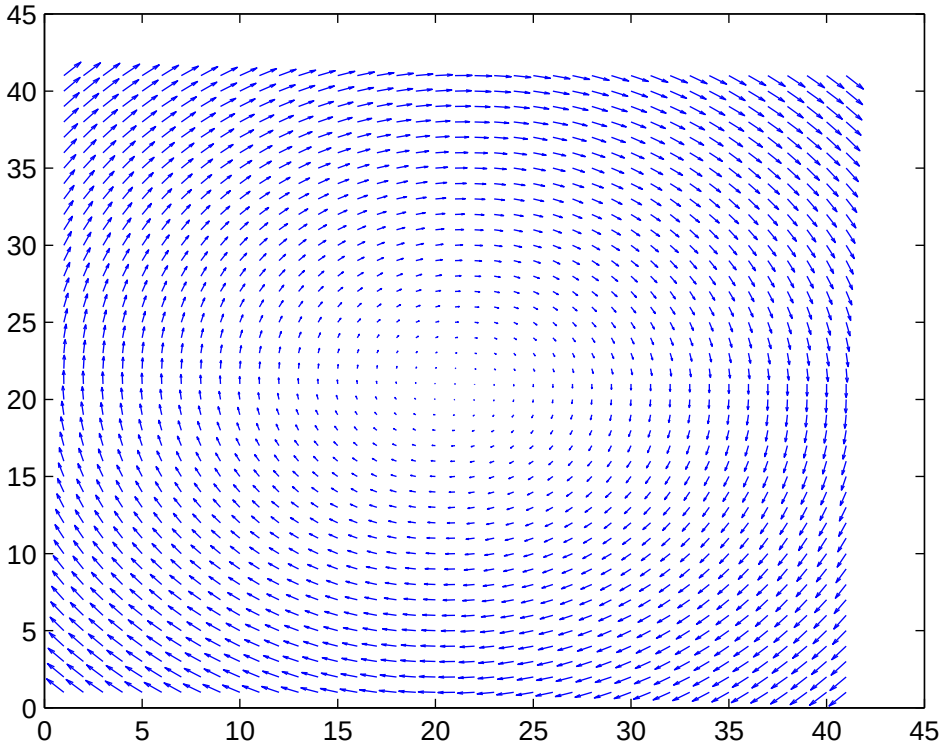
kde r je poloměr kružnice, na které se právě nacházíme (ve skriptu dáno pomocí hodnoty cyklu, kde počítáme x a y). Hodnoty u a v v bodech (x,y) jsou dány:

$$u(x, y) = 2\pi r \Delta x \sin \varphi$$

$$v(x, y) = -2\pi r \Delta x \cos \varphi,$$

kde r je poloměr kružnice, na které leží bod (x,y) a úhel φ je stejný jako u výpočtu bodu. Vzhledem k tomu, že struktura všech polí je stejná, bude dobré opět vytvořit jeden skript, který bude na základě parametrů vracet potřebné rychlostní pole. Narozdíl od skriptu pro vytváření dat zde bude potřeba více parametrů. Konkrétně se jako ideální jeví kombinace těchto tří :

- Typ rychlostního pole
- Velikost dat podle kterých se nastaví i velikost rychlostního pole
- Krok mřížky (vzdálenost dvou sousedních bodů pole)



Obrázek 2.2: Příklad rychlostního pole

2.5 Metody

V této sekci bude uveden pouze obecný rozbor pro metody. Podrobněji budou metody rozebrány v následující kapitole. Každá z metod použitých při experimentech bude potřebovat vlastní skript, který na základě zadaných parametrů - to jest vstupních dat, rychlostního pole, konstant vrátí data upravená. Následuje výpis předpokládaných parametrů pro jednotlivé metody :

- Matice dat
- Rychlostní pole
- Časový krok
- Velikost mřížky
- Krok mřížky (prostorový krok)

Cílem této kapitoly bylo teoreticky popsat postup při implementaci metod. Vše bude podrobněji rozebráno v další kapitole.

Kapitola 3

Implementace

Jak už bylo řečeno v předešlé kapitole, bude implementace jednotlivých metod provedena v programovém vybavení Matlab společnosti Mathworks. V dalších odstavcích bude popsána implementace jednotlivých částí skriptu pro experimenty a také implementace jednotlivých metod.

3.1 Tvorba dat pro experimenty

V této části bude podrobně popsáno vytváření dat, nad kterými bude algoritmus jednotlivých metod provádět experimenty. Všechny typy dat budou z pohledu algoritmu dvou-dimenzionální.

3.1.1 Válec a kvádr

Tato data byla použita pro nejjednodušší experimenty. V případě válce jejich vytvoření spočívá v nastavení matice dat tak, aby byla do určitého poloměru (kolem středu matice) vyplněna jedničkami a mimo tuto oblast nulami. Pro kvádr je vytvoření jednoduší neboť jde pouze o naplnění matice jedničkami a nulami tak, aby jedničky tvořily čtverec. Jedná se tedy o programově vytvořená nespojitá data. Na následujících obrázcích je graf těchto dat vykreslený v programu Matlab. Výpočet dat ve tvaru válce je jednoduchý. Pro válec se nejprve počítají v cyklu body na kružnicích jak již bylo uvedeno u rychlostního pole. Tedy vytvoříme 2 vnořené cykly vnější přes úhel a vnitřní přes poloměr. Tímto cyklem počítáme jednotlivé body.

$$x = stred + r \sin \varphi$$

$$y = stred + r \cos \varphi$$

Dále se v tomto cyklu nachází podmínka, která bodům, ležícím uvnitř námi zvoleného poloměru válce, přiřadí hodnotu 1. Pro kvádr se pouze dosadí matice jedniček do větší matice o rozměru dat naplněné nulami tak, aby se jejich středy kryly.

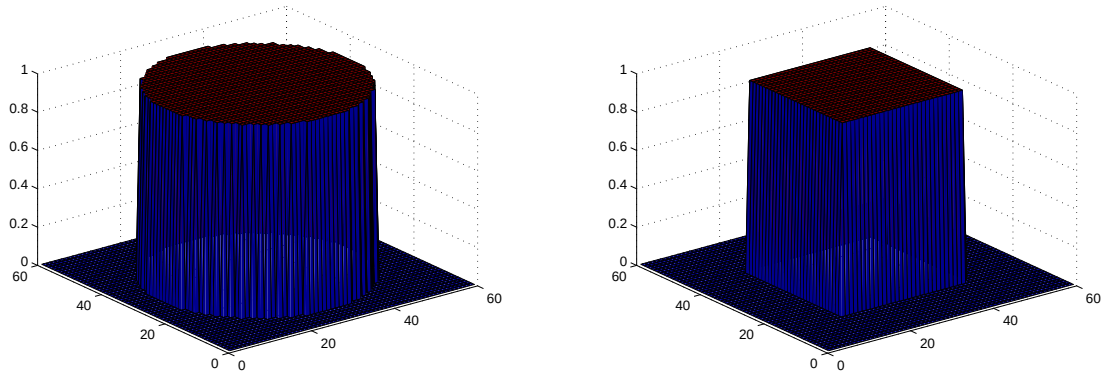
Tedy pro válec dostaneme:

$$q_0(x, y) = 1 \forall x \in (stred - r, stred + r) \forall y \in (stred - r, stred + r),$$

kde r je poloměr válce. Pro kvádr dostaneme:

$$q_0(x, y) = 1 \forall (x, y) \in (gridSize/4, 3/4 gridSize) x (gridSize/4, 3/4 gridSize)$$

Je potřeba říci, že pro dokonalý válec by bylo potřeba, aby se válec skládal z nekonečného množství bodů a byl tedy v proměnných x a y spojitý. To ovšem při numerických pokusech není možné. Proto je použita mřížka určité velikosti a vzniklá data jsou pouze aproximací skutečného válce.



Obrázek 3.1: Data ve tvaru válce a kvádru

3.1.2 Gaussova funkce

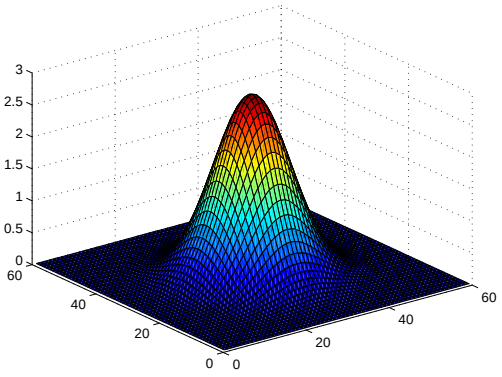
Pro potřeby pokusů bylo potřeba vytvořit také hladká data. To splňuje právě Gaussova funkce. Její vytvoření spočívá ve využití funkce Gaussova pravděpodobnostního normálního rozdělení, které lze v Matlabu snadno rozšířit do dvou rozměrů a následně vykreslit pomocí funkce surf(). Pro vytvoření je využita funkce hustoty Gaussova normálního rozdělení

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}},$$

kde μ je střední hodnota a σ^2 je rozptyl. Na následujícím obrázku je graf Gaussovy funkce. I zde se jedná o určitou aproximaci požadované funkce.

Algoritmus 1: Algoritmus tvorby dat ve tvaru Gaussovy funkce

1. vytvoření Gaussova rozdělení s požadovanými parametry pomocí matlabovské funkce **normpdf**(), $g = normpdf(body\ pro\ tvorbu, průměr, směrodatná\ odchylka)$ v této práci g = normpdf(1:gridSize, $\frac{1}{2}$gridSize, $\frac{1}{8}$gridSize)
2. rozmnožení dat na dvojrozměrný tvar (funkce **repmat**() - nazveme maticí X), tedy $X = repmat(g, gridSize, 1)$
3. znásobení matice X s transpozicí sama sebe $s = X * X'$
4. přenásobit číslem c tak, abychom dostali hodnoty blízké k 1, $q = cs$



Obrázek 3.2: Data ve tvaru dvojrozměrné gaussovy funkce

3.1.3 Obrázek

Posledním typem dat je černobílý a barevný obrázek. Ty byly použity společně s rychlostním polem otáčejícím se kolem určeného středu. Použití černobílého obrázku je prakticky stejné jako v případě vytvořených dat. Má to velice jednoduchý důvod. Obrázek je z pohledu počítače stejná matice jako v případě výše zmíněných dat.

Algoritmus 2: Algoritmus tvorby dat ve tvaru černobílého obrázku

1. načtení obrázku pomocí matlabovské funkce **imread**(), tedy $obr1 = imread(name)$
2. pokud je obrázek barevný, ale je požadován černobílý použijeme funkci pro převod **rgb2gray**() jinak pokračujeme bodem 3, $obr = rgb2gray(obr1)$.
3. převedení typu dat z **uint8** na **double**, $obr\_new = double(obr)$

Obrázek 3.3: Data ve tvaru černobílého obrázku

V případě barevných obrázků je v první řadě potřeba vědět jak jsou v Matlabu (respektive v technologii, ve které se budou experimenty provádět) uchovávány. V této práci byly použity pouze obrázky v barevné paletě RGB. Tyto obrázky mají data uložená v trojrozměrné matici o rozměru obrázku a třetí rozměr je 3 (počet barevných složek). Pro aplikaci je třeba je rozložit na jednotlivé matice pro barevné složky. Matice R - červený kanál, matice G - zelený kanál, matice B - modrý kanál.

Obrázek 3.4: Data ve tvaru barevného obrázku

Algoritmus 3: Algoritmus tvorby dat ve tvaru barevného obrázku

1. načtení obrázku pomocí matlabovské funkce **image** = **imread**()
2. rozložení obrázku na jednotlivé matice reprezentující každou ze 3 základních barev pomocí jednoduchého příkazu: např pro červený kanál $R = (double)(image(:,:,1))$, zároveň převedení na typ **double**

3.2 Rychlostní pole

V této části popíši tvorbu jednotlivých rychlostních použitých při experimentech z implementačního hlediska. Celkem byly použity tyto typy rychlostních polí:

- rotační pole
- posuvné pole

Na následujících řádcích bude popsán algoritmus tvorby polí:

Algoritmus 4: Algoritmus tvorby rychlostních polí

1. Vytvoření mřížky velikostí odpovídající datům $q = zeros(vel.\ mřížky,\ vel.\ mřížky)$
2. počítání jednotlivých bodů na kruhu pomocí rovnic kružnice $x = stred + r\sin\varphi, \quad y = stred + r\cos\varphi$
3. Každému bodu se podle typu pole přiřadí hodnota (konstanta nebo hodnota pro rotaci)
4. Pole se vykreslí pomocí funkce **quiver**

3.3 Metoda typu upwind a Laxova-Wendroffova metoda

Implementace těchto dvou metod je jednoduchá. Jedná se pouze o přepsání rovnic metod do matlabovské syntaxe a aplikace na data s využitím rychlostního pole. V následujícím algoritmu bude ukázáno schéma, které bude použité u všech metod. Každá metoda bude implementována jako funkce jak již bylo popsáno v předchozí kapitole.

3.4 Hartenova-Zwasova metoda

Opět se jedná o obdobnou implementaci jako výše pouze je zde třeba ošetřit správné nastavení hodnot u^+ a u^- popřípadě v^+ a v^- v rámci cyklu, který počítá jednotlivé

Algoritmus 5: Schéma pro tvorbu metod v pseudokódu

1.načtení parametrů - data, rychlostní pole, konstanty
2.v cyklu určení potřebných hodnot v každém bodě(např. pro upwind hodnoty $u^+ = max(u_j, 0), u^- = min(u_j, 0)$,
3.vlastní metoda - přepsání metody uvedené v teorii do matlab. syntaxe, zde konec cyklu
4.vrácení výstupu funkce - vždy data

iterace. Proto je v této práci k implementaci využit výsledek metody upwind, jelikož toto stačí pro správné výsledky. Ve skriptu implementující tuto metodu se nachází i cyklus počítající metodu upwind. Algoritmus pro tento skript je stejný jako u metody upwind, ale navíc se počítají hodnoty limiteru.

3.5 Semilagrangeovská metoda

Implementace téhle metody se dá rozdělit do dvou kroků. Prvním krokem je výpočet koeficientu posunu α pomocí metody prosté iterace, která je realizována *while* cyklem, který končí po splnění zvolené podmínky. Druhým krokem je interpolace. Zde bylo využito funkce *interp*, resp. *interp2*, která je součástí základní sady matlabovských funkcí. Tato funkce má parametry - maticemi x, resp x a y, reprezentující body mřížky ($[x, y] = meshgrid(1 : \Delta x : gridSize, 1 : \Delta x : gridSize)$), dále data a poté body, ve kterých je prováděna interpolace. Tyto body jsou dány pomocí koeficientu α, resp. α a β, a tvoří matici *xi*, resp. *xi, yi*, $xi(i, j) = x(i, j) - \alpha$. Funkce *interp* umožňuje počítat interpolaci pro všechny body najednou nebo interpolovat polohy jednotlivých bodů postupně.
Pro dvojrozměrnou metodu je potřeba algoritmus rozšířit o druhý koeficient tak, aby každý koeficient byl pro jeden směr, tedy α pro směr osy x a β pro směr osy y. Oba koeficienty je třeba počítat ve stejném cyklu. Pro dvojrozměrná data je použita funkce *interp2*, která má následující zápis:

$$Q\_new = interp2(x, y, Q, xi, yi),$$

kde x a y jsou matice určující mřížku, Q jsou počáteční data a xi a yi jsou matice bodů, ve kterých provádíme interpolaci.

Algoritmus 6: Algoritmus semilagrangeovské metody v pseudokódu

1.výpočet α pomocí prosté iterace (cyklus **while**),
$[\alpha, \beta] = \Delta tpole(x(i) - \frac{\alpha}{2}, y(j) - \frac{\beta}{2})$ v cyklu přes rozměry dat (koeficienty i a j) a výpočet matic xi,yi. (zde x(i),y(j) body vektorů x = 1: Δx:gridSize, y = 1: Δx:gridSize)
2.interpolace na daných bodech (xi,yi) pomocí funkce **interp2**,
$q = interp2(x, y, q, xi, yi, typ\ interopolace)$

3.6 Metoda BFECC

Tato metoda se od výše uvedených liší, jelikož je implementována pomocí vlastního algoritmu uvedeného níže a pomocí metody upwind nebo semilagrangeovské metody. Příkaz **First-Order-Step** může v algoritmu představovat implementaci

Algoritmus 7: Algoritmus BFECC v pseudokódu

1. $First - Order - Step(u, v, q^n) \longrightarrow q^{\#}$
2. $First - Order - Step(-u, -v, q^{\#}) \longrightarrow \overline{q}$
3. $q^* := q^n + \frac{1}{2}(q^n - \overline{q})$
4. $First - Order - Step(u, v, q^*) \longrightarrow q^{n+1}$

metody upwind nebo metody semilagrangeovské. Tento příkaz odpovídá funkci L popsané v teorii. Vstupem do příkazu **First-Order-Step** je rychlostní (parametry u a v) a datové (všechny parametry q) pole. Do tohoto skriptu je opět vstupem rychlostní pole, data a konstanty.

3.7 Vykreslování

Pro vykreslování jednotlivých výsledků je v Matlabu k dispozici několik funkcí, jejichž použití značně ulehčí práci. Na druhou stranu neexistuje metoda, která by na základě typu dat rozhodla, jak daná data vykreslit. Proto bylo potřeba takovýto skript vytvořit. V zásadě skript obaluje a sjednocuje použití dvou matlabovských funkcí :

- Funkci *imshow()*, která vykreslí obrazová data tak jako klasický prohlížeč obrázků - tj. hodnoty reprezentuje jako jednotlivé barvy černobílého nebo RGB spektra.
- Funkci *surf()*, která trojrozměrně vykreslí zadaná dvojrozměrná data.

Obě zmíněné funkce mají jen jeden parametr, který reprezentuje data pro vykreslení. Obalující skript má proto dva parametry. Prvním jsou data stejně jako u vestavěných

funkcí a pak typ dat, aby funkce věděla, jak data vykreslit. Skript vytvořený pro vykreslení má v této práci parametrů více, jelikož prvních n parametrů určuje data od n metod určená k vykreslení.

3.8 Testovací skript

V této sekci bude popsána struktura skriptu vytvořeného pro testování. Základem skriptu je soubor **index.m**. V tomto souboru jsou volány jednotlivé skripty pro načtení dat, tvorbu rychlostního pole, skripty pro metody, vykreslení výsledku a nakonec jsou tu i vypočítávány normy pro porovnání metod. Následující algoritmus znázorňuje postup jakým je prováděn test.

Algoritmus 8: Postup testování metod

1.určení koeficientů (časový a prostorový krok, velikost mřížky) - pevně zadané
2.načtení nebo vytvoření dat pomocí skriptu **data**()
3.vytvoření rychlostního pole pomocí skriptu **pole**()
4.časový cyklus obsahující jednotlivé metody
5.vykreslení dat pomocí skriptu **vykresleni**()
6.výpočet jednotlivých norem

Mimo vizuálního porovnání metod pomocí obrázků je třeba vytvořit i porovnání, které lze vyjádřit přesněji (např. číselně). Proto pro jednotlivé metody testovací skript vypočítá normy. Tyto normy určují, jak moc se liší původní data od dat, na které byla alplikována daná metoda. Pro tento výpočet je použita funkce *norm()*, která je implementována přímo Matlabem.

$$norma = norm(Q - Q_{upraven}),$$

kde $Q_upraven$ jesou data po aplikaci některé z metod. Pro každou metodu bude vypočítána tato norma a poté bude provedeno porovnání. Platí, že čím menší je norma, tím je metoda přesnější. Podrobné výsledky budou znázorněny v další kapitole.

Kapitola 4

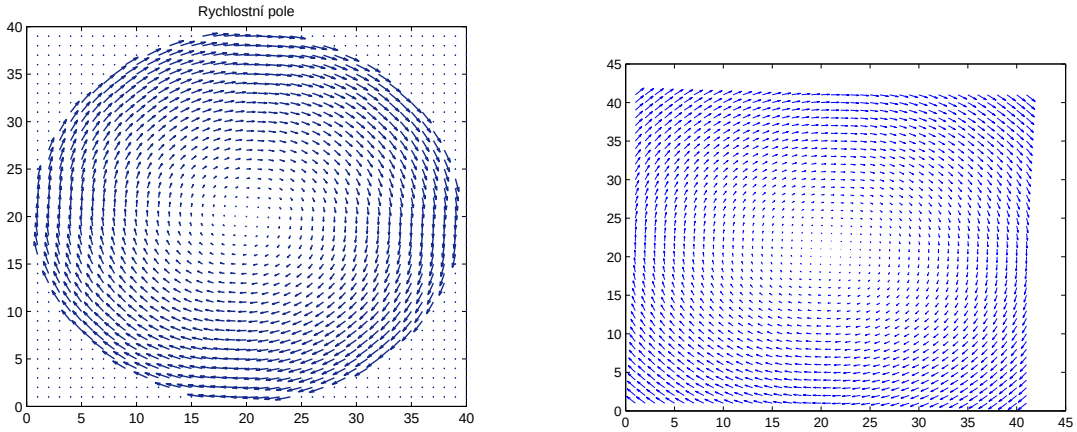
Zhodnocení výsledků

V této kapitole budou ukázány výsledky metod na několika typech dat a poté bude provedeno porovnání přesnosti metod pomocí norem, jak již bylo zmíněno v minulé kapitole. Celá kapitola bude členěna podle typů dat, na kterých je prováděn konkrétní test. Pro každou podkapitolu bude vždy uvedena sada obrázků, kde budou znázorněny výsledky původních metod. Na konci každé podkapitoly bude uvedena tabulka s číselnými hodnotami norem pro jednotlivé metody. Následně provedu porovnání metod a slovně popíši, co normy vyjadřují.

Pro všechny typy dat bylo použito jednotné nastavení a to bylo následující:

- časový krok $\Delta t = 0.0005$ (ve skriptu označeno písmenem k)
- krok mřížky $\Delta x = 1$ (ve skriptu označeno písmenem h)
- rotační pole (typ 1 nebo 2, viz. obr.4)

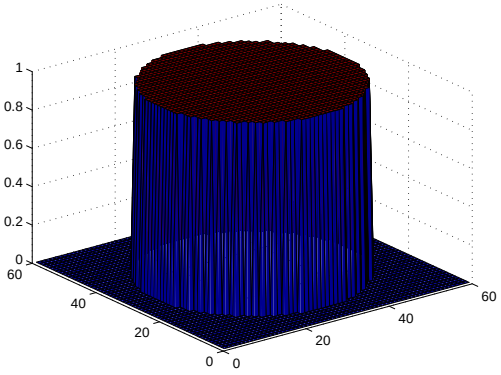
Normy uvedené v tabulkách a výsledky jsou pro rotační pole typ 1 a časový údaj je pro jeden časový krok dané metody. Počet časových kroků pro jednotlivé typy dat byl volen tak, abychom dostali stejný tvar kvůli posouzení přesnosti pomocí norem. Pro další typy pole se normy liší jen minimálně a je zachován poměr mezi normami pro jednotlivé metody.



Obrázek 4.1: Rychlostní pole použitá k testování metod (vlevo typ 1, vpravo typ 2)

4.1 Válec

Původní data byla ve tvaru znázorněném na následujícím obrázku. Je to válec o poloměru 3/4 poloviny velikosti mřížky. Mřížka má rozměry 60x60. Počet iterací byl



Obrázek 4.2: Data ve tvaru válce

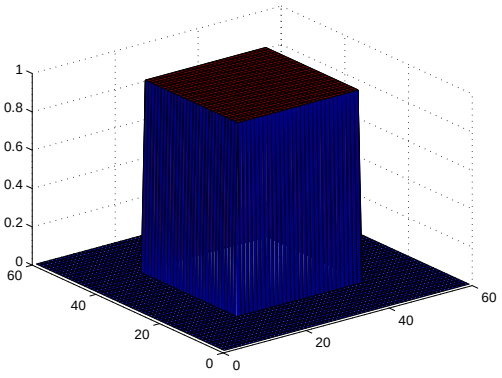
volen $T = 250$, tedy otočení o osminu (není třeba volit více díky souměrnosti válce). Výsledky metod jsou uvedeny v příloze (5). Následuje tabulka norem pro jednotlivé metody.

typ metody	norma	doba výpočtu jednoho časového kroku [s]
metoda upwind	4.0891	0.103485
Laxova-Wendroffova metoda	2.8415	0.006319
Hartenova-Zwasova metoda	2.5457	0.110195
semilagrangeovská metoda	4.1147	0.502500
metoda BFECC se zákl. upwind	2.7848	0.274589
metoda BFECC se zákl. semilagrange	2.9210	1.703359

Tabulka 4.1: Tabulka norem pro válec

4.2 Kvádr

Původní data byla ve tvaru znázorněném na následujícím obrázku. Je to kvádr o rozměrech polovičních než rozměry mřížku. Mřížka má rozměry 60x60. Počet iterací byl



Obrázek 4.3: Data ve tvaru kvádru

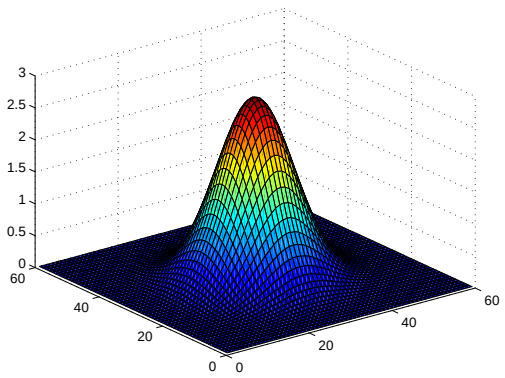
volen $T = 500$, tedy otočení o čtvrtinu (tímto dostaneme stejný čtverec). Výsledky metod jsou uvedeny v příloze (5). Následuje tabulka norem pro jednotlivé metody.

typ metody	norma	doba výpočtu jednoho časového kroku [s]
metoda upwind	8.5384	0.138074
Laxova-Wendroffova metoda	6.3651	0.007146
Hartenova-Zwasova metoda	6.2704	0.128437
semilagrangeovská metoda	8.9406	0.527506
metoda BFECC se zákl. upwind	6.3834	0.286875
metoda BFECC se zákl. semilagrange	6.1027	1.661483

Tabulka 4.2: Tabulka norem pro kvádr

4.3 Gaussova funkce

Původní data byla ve tvaru znázorněném na následujícím obrázku. Počet iterací



Obrázek 4.4: Data ve tvaru dvojrozměrné Gaussovy funkce

byl volen $T = 250$, tedy otočení o osminu (není třeba volit více díky souměrnosti funkce). Výsledky metod jsou uvedeny v příloze (5). Následuje tabulka norem pro jednotlivé metody.

typ metody	norma	doba výpočtu jednoho časového kroku [s]
metoda upwind	1.2108	0.122016
Laxova-Wendroffova metoda	1.6359	0.026933
Hartenova-Zwasova metoda	1.6441	0.161784
semilagrangeovská metoda	1.2927	0.896581
metoda BFECC se zákl. upwind	1.6591	0.291043
metoda BFECC se zákl. semilagrange	0.0363	1.618665

Tabulka 4.3: Tabulka norem pro Gaussovu funkci

4.4 Černobílý obrázek

Původní data byla ve tvaru znázorněném na následujícím obrázku. Počet iterací byl

Obrázek 4.5: Data ve tvaru černobílého obrázku

volen $T = 2000$, tedy otočení o celý kruh. Výsledky metod jsou uvedeny v příloze (5). Následuje tabulka norem pro jednotlivé metody.

typ metody	norma	doba výpočtu jednoho časového kroku [s]
metoda upwind	3.66785e03	0.287784
Laxova-Wendroffova metoda	1.4808e03	0.035861
Hartenova-Zwasova metoda	1.4180e03	0.342171
semilagrangeovská metoda	4.9376e03	1.691374
metoda BFECC se zákl. upwind	1.5379e03	0.812007
metoda BFECC se zákl. semilagrange	1.0388e03	4.753137

Tabulka 4.4: Tabulka norem pro černobílý obrázek

4.5 Barevný obrázek

Původní data byla ve tvaru znázorněném na následujícím obrázku. Počet iterací byl

Obrázek 4.6: Data ve tvaru barevného obrázku

volen $T = 2000$, tedy otočení o celý kruh. Výsledky metod jsou uvedeny v příloze (5). V případě těchto dat jsou normy vytvořeny jako průměr norem pro matice jednotlivých barevných kanálů obrázku. Následuje tabulka norem pro jednotlivé metody.

typ metody	norma	doba výpočtu jednoho časového kroku [s]
metoda upwind	3.5309e03	0.306232
Laxova-Wendroffova metoda	1.6070e03	0.034638
Hartenova-Zwasova metoda	1.4883e03	0.376165
semilagrangeovská metoda	4.5953e03	1.856436
metoda BFECC se zákl. upwind	1.6292e03	0.901415
metoda BFECC se zákl. semilagrange	1.1035e03	4.368316

Tabulka 4.5: Tabulka norem pro barevný obrázek

4.6 Zhodnocení

Z hodnot uvedených v jednotlivých tabulkách vyplývá, že ve většině případů je nejlepší metodou Hartenova-Zwasova metoda. Další metodou, která dává velmi dobré výsledky je metoda BFECC se základem semilagrangeovské metody. Tato metoda je velmi dobrá na hladká data (Gaussova funkce, obrázek), ale výsledky na data s nespojitostmi jsou horší. Další metody dávají špatné výsledky a to z několika důvodů.

Pro metodu upwind vychází norma velká, jelikož upwind je metoda prvního řádu. Pro Laxovu-Wendroffovu metodu dostáváme lepší výsledek, který není díky oscilacím nejlepší, ale je lepší než u metody upwind díky vyššímu řádu metody. Dále ovšem výsledek Laxovy-Wendroffovy metody zhoršuje fázový posun, což je dobře patrné na datech ve tvaru obrázku (5,5) a proto je samotná Laxova-Wendroffova metoda nepoužitelná.

Semilagrangeovská metoda dává špatný výsledek kvůli typu použité interpolace. Oba dva typy metody BFECC dávají přiměřeně dobrý výsledek ale stále nejsou nejlepší, jelikož produkují oscilace. Pro BFECC se základem semilagrangeovské metody dostáváme relativně lepší výsledky. Oscilace v tomto případě jsou menší, než u druhého typu BFECC a tato metoda je velmi dobrá na hladká data (viz. 4.3,).

Pro lepší výsledky všech metod konečných diferencí by bylo třeba ošetřit vznik oscilací a obdobně jako je to u Hartenovy-Zwasovy metody tomuto jevu zabránit. Pro semilagrangeovskou metodu je třeba pro lepší výsledek použít jiný typ interpolace než lineární, která je použita v tomto případě. Pro metodu BFECC je třeba taktéž ošetřit vznik oscilací. Tímto bychom docílili lepších výsledků než dostáváme nyní.

Kapitola 5

Závěr

Tato práce se zabývala metodami pro řešení advekční rovnice. Celkem byly zmíněny 4 metody eulerovského typu a metoda semilagrangeovská. Tyto metody byly popsány z teoretického hlediska a poté rozebrány z hlediska implementačního. Na základě tohoto rozboru byly následně metody implementovány. Ovšem implementací a testováním bylo prověřeno mnohem více. Zejména je řeč o vlastnostech jednotlivých metod.

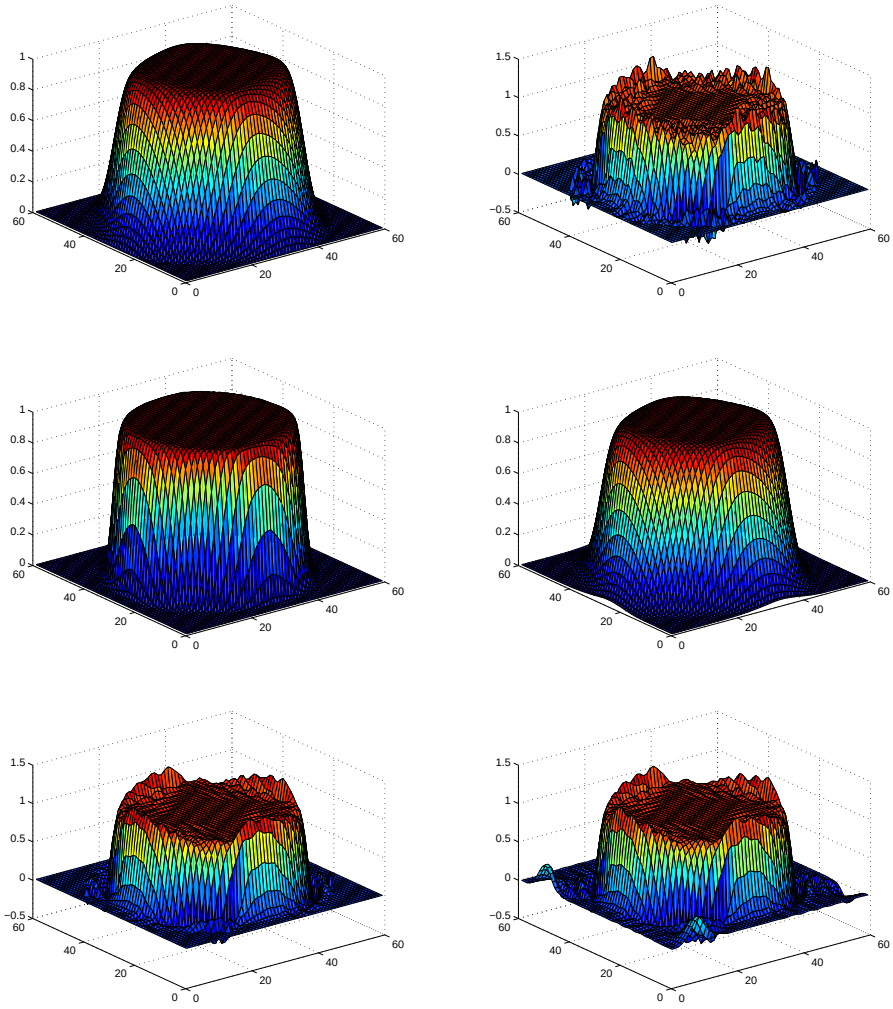
Metody byly testovány na několika různých typech dat a na základě těchto testů bylo zjištěno pořadí jednotlivých metod dle přesnosti výsledku a rychlosti výpočtu. Ve většině případů se jako nejlepší metoda jevila Hartenova-Zwasova metoda. U ostatních metod se ve výsledku projevily vlastnosti, které zhoršily výsledek, např. u Laxovy-Wendroffovy metody se projevily oscilace.

Dalšími úkoly, které je třeba vyřešit je vylepšení jednotlivých metod. U metod konečných diferencí se jedná o ošetření vzniku oscilací, u metody semilagrangeovské se jedná o vytvoření lepší interpolace. Následně je třeba vytvořit komplexnější systém posouzení správnosti a přesnosti metod a dalším úkolem je vytvořit a aplikovat časově proměnné pole.

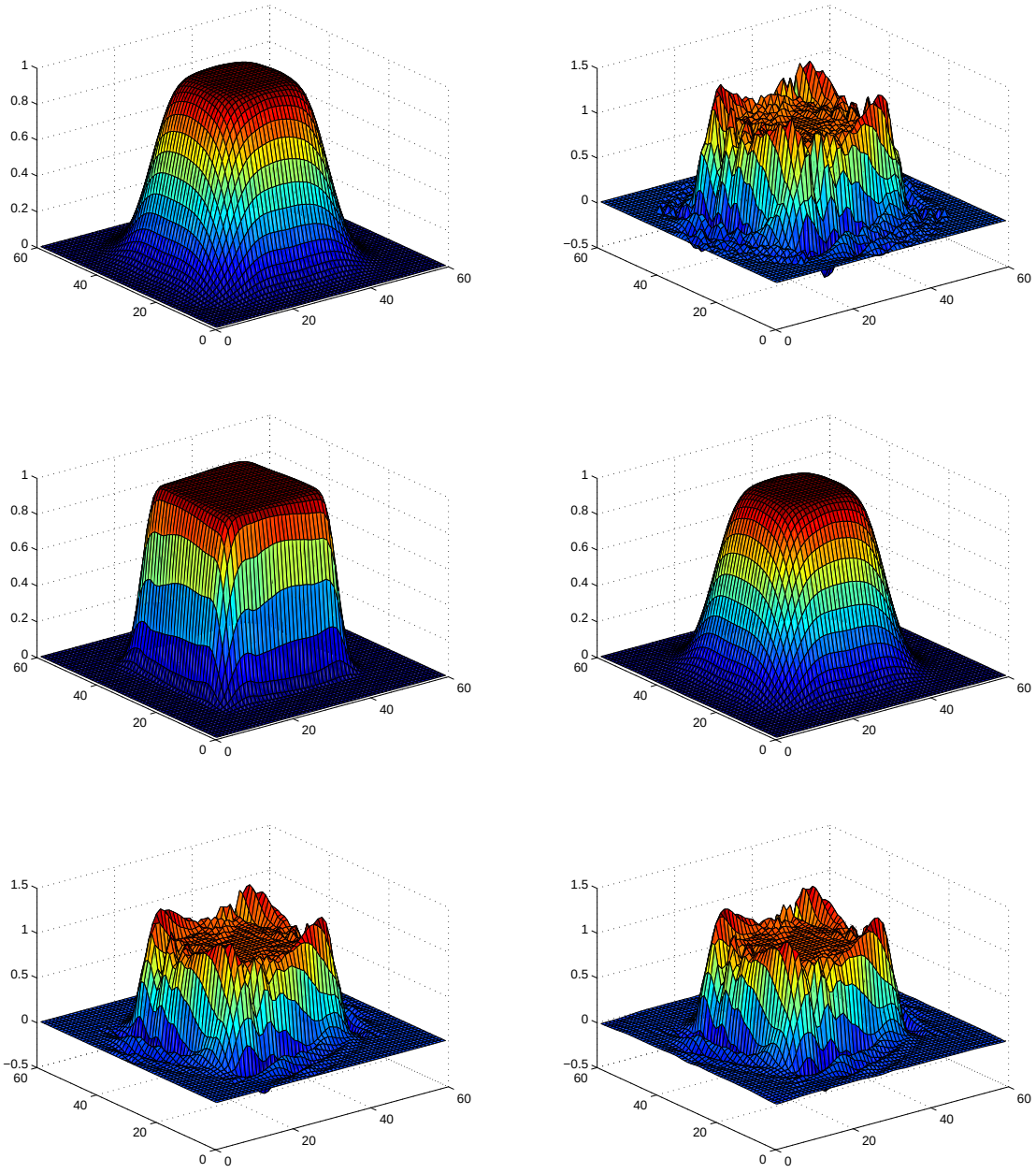
Výsledkem celé této práce tedy bylo jednak implementování metod pracujících s advekční rovnicí dále vyhodnocení výsledků a stanovení průměrně nejlepší metody. Dalším cílem této práce bylo přehledně sepsat a uspořádat metody pro práci s advekční rovnicí a to takovým způsobem, aby byly pochopitelné i pro člověka, který s advekční rovnicí dosud nepracoval.

Výsledky a postupy obsažené v této práci jsou popsány relativně obecně a při případné aplikaci na konkrétní problém z reálného světa bude dozajista potřeba do daných postupů zavést vlastnosti a zákonitosti fyzikální podstaty daného problému.

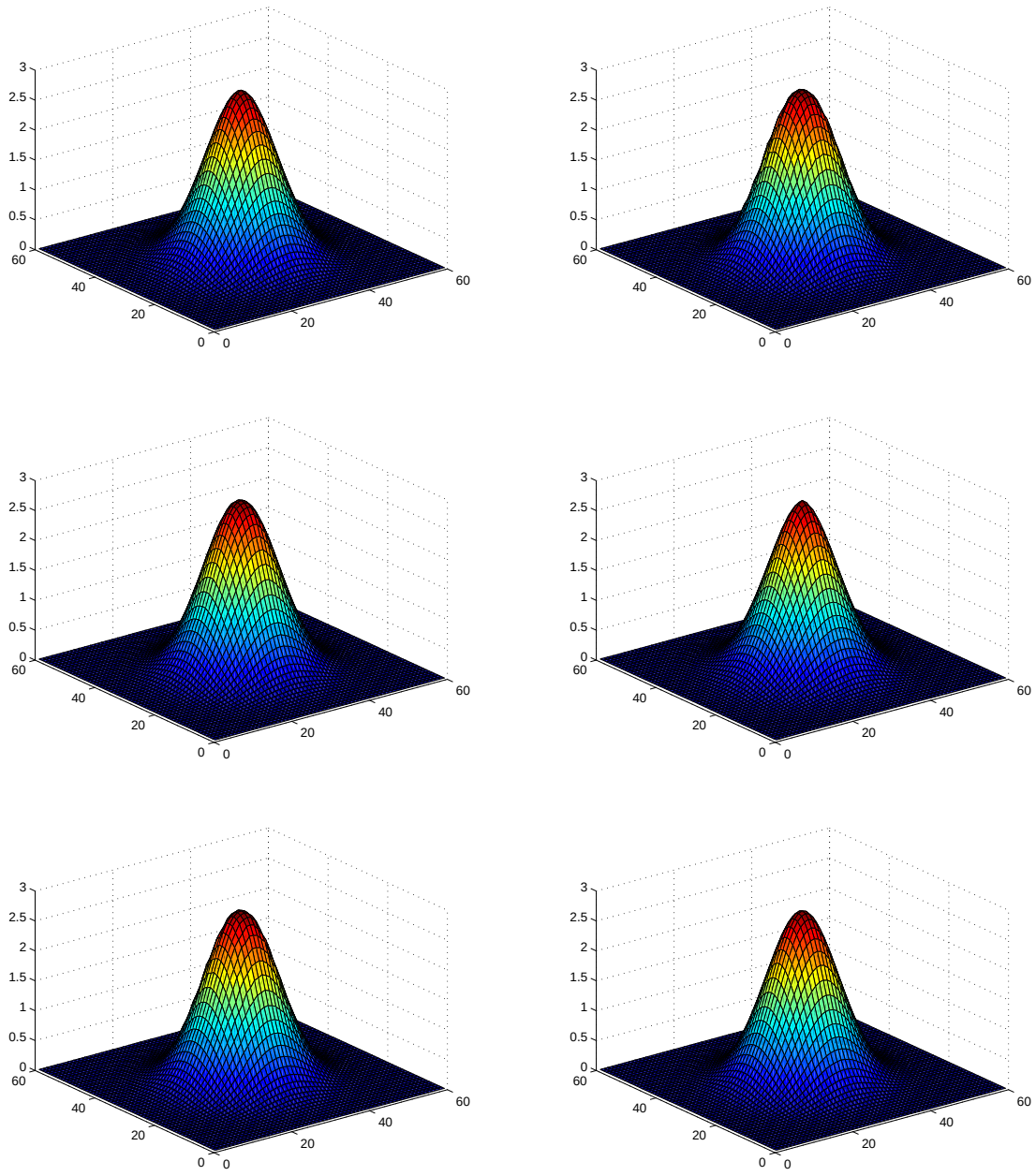
Příloha



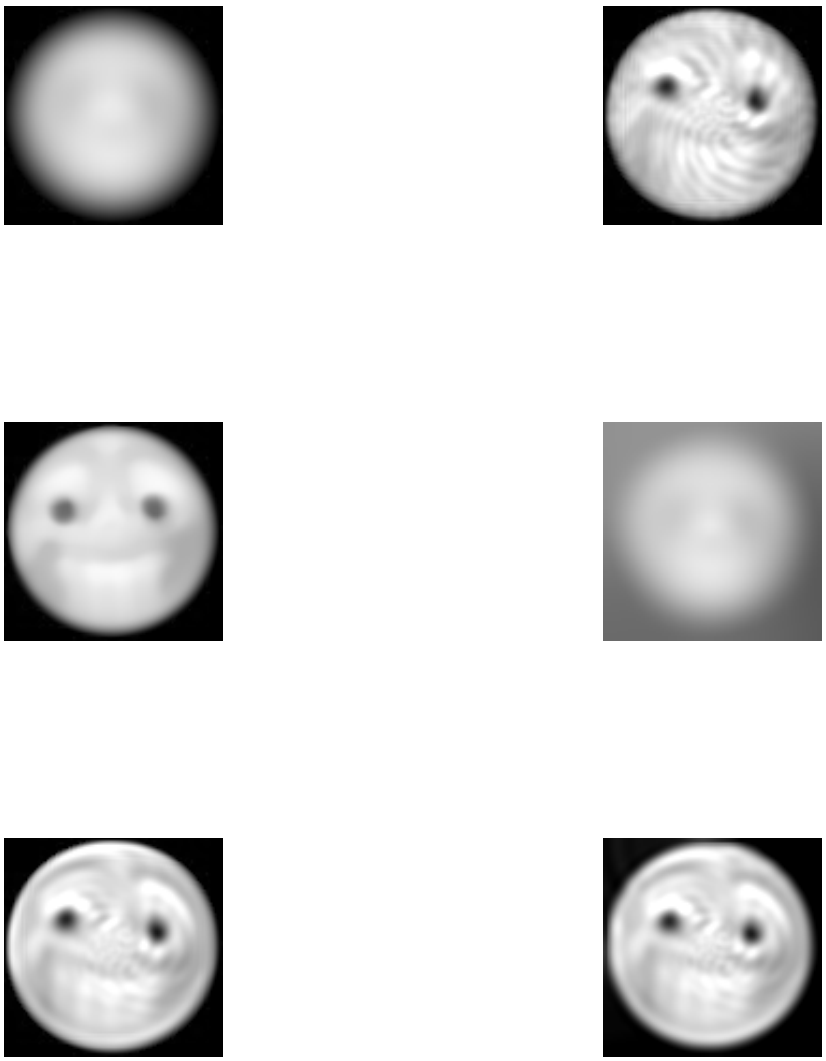
Obrázek 5.1: Výsledky metod - vlevo nahoře metoda upwind, vpravo nahoře Laxova-Wendroffovametoda, uprostřed Hartenova-Zwasova (vlevo) a semilagrangeovská metoda, dole metoda BFECC se základem upwind (vlevo) a semilagrangeovské metody (vpravo)



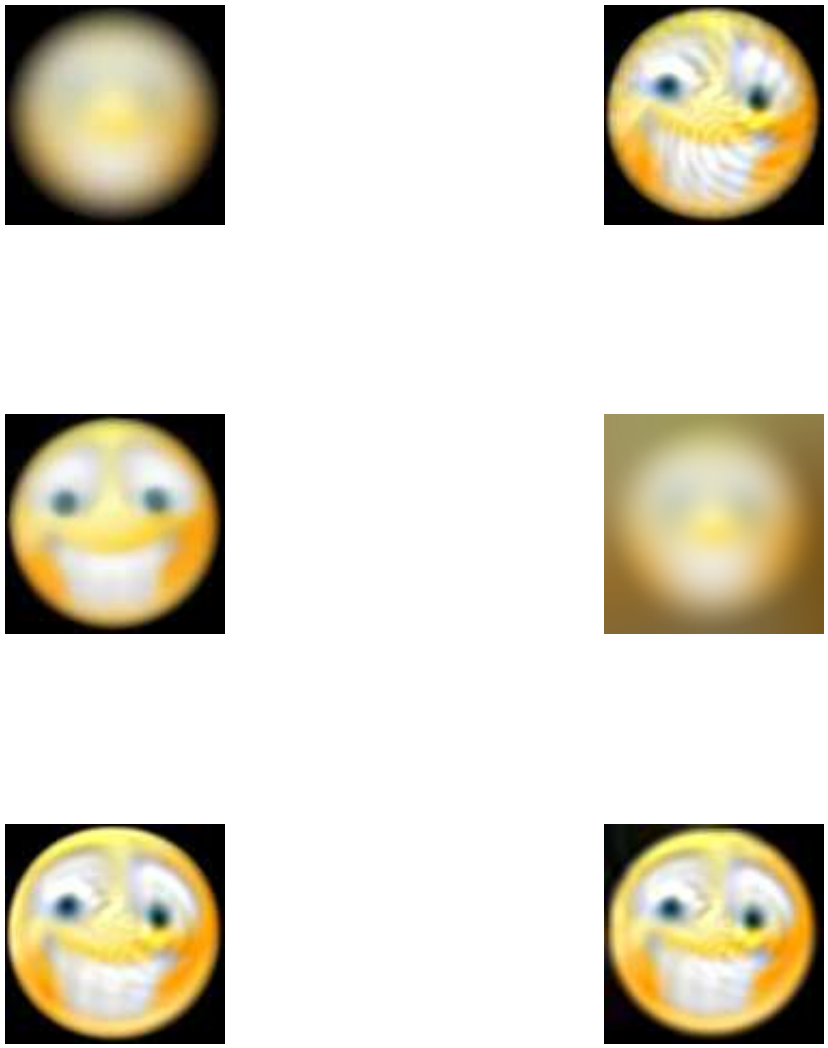
Obrázek 5.2: Výsledky metod - vlevo nahoře metoda upwind, vpravo nahoře Laxova-Wendroffovametoda, uprostřed Hartenova-Zwasova (vlevo) a semilagrangeovská metoda, dole metoda BFECC se základem upwind (vlevo) a semilagrangeovské metody (vpravo)



Obrázek 5.3: Výsledky metod - vlevo nahoře metoda upwind, vpravo nahoře Laxova-Wendroffovametoda, uprostřed Hartenova-Zwasova (vlevo) a semilagrangeovská metoda, dole metoda BFECC se základem upwind (vlevo) a semilagrangeovské metody (vpravo)



Obrázek 5.4: Výsledky metod - vlevo nahoře metoda upwind, vpravo nahoře Laxova-Wendroffovametoda, uprostřed Hartenova-Zwasova (vlevo) a semilagrangeovská metoda, dole metoda BFECC se základem upwind (vlevo) a semilagrangeovské metody (vpravo)



Obrázek 5.5: Výsledky metod - vlevo nahoře metoda upwind, vpravo nahoře Laxova-Wendroffovametoda, uprostřed Hartenova-Zwasova (vlevo) a semilagrangeovská metoda, dole metoda BFECC se základem upwind (vlevo) a semilagrangeovské metody (vpravo)

Literatura

[1] LeVeque R.J.: *Finite-Volume Methods for Hyperbolic Problem* , Cambridge University Press, 2004

[2] LeVeque R.J.: *Finite Difference Methods for Ordinary and Partial Differential Equations* , SIAM, 2007

[3] Kim B.,Liu Y.,Llamas I.,Rossignac J.,Jiao X.: *Simulation of Bubbles in Foam with the Volume Control Method*, [online] *http://www.gvu.gatech.edu/∼jarek/papers/foam.pdf*

[4] Kim B.,Liu Y.,Llamas I.,Rossignac J.: *: Advections with Significantly Reduced Dissipation and Diffusion* , [online] *http://www.gvu.gatech.edu/∼jarek/papers/Advect.pdf*

[5] LeVeque R.J.: *: High-resolution conservative algorithms for advection in incompressible flow* , SIAM, 1996

[6] Brandner M., Egermaier J., Kopincová H.: *: Numerické modelování v hydrologii* , , 2011

[7] Le Roux D.Y., Lin Ch.A., Staniforth A.: *: An accurate interpolating scheme for semi-Lagrangian advection on an unstructured mesh for ocean modeling* , C2GCR Report No. 95-11, July 1995

[8] *Wikipedia.org, otevřená encyklopedie* , [online] *http://en.wikipedia.org/wiki/Flux_limiter*