FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

Lineární řešiče v OpenFOAM

Semestrální práce

KMA/PVM

Jan Půlpán

5. května 2021

1 Teoretický úvod

OpenFOAM je sada výpočetních nástrojů pro numerické simulace CFD (computational fluid dynamics) problémů, tedy problémů zabývajících se prouděním tekutin, vedením tepla a podobných procesů. OpenFOAM využívá k hledání řešení metody konečných objemů (FVM).

FVM je numerická metoda na řešení parciálních diferenciálních rovnic na dané 3D geometrii převedené na síť nepřekrývajích se elementů (konečných objemů). Diskretizací úlohy získáme soustavu lineárních algebraických rovnic v neznámé $\boldsymbol{x}$ pro každou ze sledovaných proměnných. Soustava je tvaru

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \tag{1.1}$$

kde $\boldsymbol{A}$ je regulární matice popisující diskretizovaný model, x vektor závislé proměnné a b vektor pravé strany obsahující všechny zdroje, okrajové podmínky a všechny komponenty, které nelze linearizovat.

Zabývat se budeme jen posledním článkem řetězce, tedy lineárními řešiči lineárních algebraických rovnic. Na testovací úloze si ukážeme vhodnost jednotlivých řešičů a budeme sledovat vliv parametrů na konvergenci řešení. Dále budeme používat lineárních řešičích mluvit jen jako o „řešičích“, pokud nebude hrozit záměna kontextu.

Obecně lze soustavu lineárních rovnic řešit buď přímými nebo iteračními řešiči. Úlohy CFD jsou většinou silně nelineární a proto není jejich řešení pomocí přímých řešičů snadné a vyžaduje větší výpočetní výkon. Přesto si ukážeme i tuto cestu.

Přímé řešiče hledají přesné řešení pomocí inverzní matice $\boldsymbol{A}^{-1}$ ve formě

$$\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}.$$

Většina přímých metod je založna na Gaussově eliminici a případné transformaci úlohy na úlohu se stejným řešením ale snadněji řešitelnou například pomocí LU nebo DILU rozkladu.

Většinou ale nejde inverzi snadno nalézt a v tom případě se používají přibližné řešiče iterační. Výhodou těchto metod je mnohem menší výpočetní náročnost i náročnost na paměť. Pomocí iterací počítáme posloupnost řešení $\boldsymbol{x}^n$, která za daných podmínek konverguje k řešení $\boldsymbol{x}$. Využívané iterační řešiče jsou Gaussova-Seidelova metoda, metody Krylovových prostorů (kam patří například metoda sdružených gradientů) a víceúrovňové metody

Iterační metody obecně měří chybu aproximace řešení v každém kroku pomocí reziduí, tedy dosazením aproximovaného řešení do původních rovnic a vypočtením rozdílu oproti pravé straně $\boldsymbol{b}$. Reziduum $\boldsymbol{r}^{(n)}$ definujeme jako

$$\boldsymbol{r}^{(n)} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(n)}. \tag{1.2}$$

Zastavení řešiče je pak definováno obecně jako pokles residua pod nastavenou hodnotu

$$\|\boldsymbol{r}^{(n)}\| < \epsilon.$$

OpenFOAM obsahuje kromě přímých řešičů hlavně iterační řešiče pomocí gradientních metod, víceúrovňových metod a řešiče pomocí smootherů (vyhlazovačů).

Gradientní metody jsou založeny na tom, že úlohu 1.1 lze přeformulovat na úlohu minimalizace kvadratické funkce

$$\boldsymbol{Q}(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x} + \boldsymbol{c}. \tag{1.3}$$

V případě symetrické a positivně definitní matice $\boldsymbol{A}$ je iterační metoda řešení úlohy 1.1 definována vztahem

$$\boldsymbol{x}^{(n+1)} = \boldsymbol{x}^{(n)} + \alpha^{(n)}\left(\boldsymbol{\delta x}^{(n)}\right), \tag{1.4}$$

kde $\alpha^{(n)}$ je relaxační faktor a $\boldsymbol{\delta x}^{(n)}$ je člen minimalizující 1.3, který lze získat například metodou největšího spádu, kdy použijeme jako minimalizující prvek vektor ve směru největšího spádu. To ale často vede k oscilacím okolo minima a proto se většinou používá metoda sdružených gradientů. Ta pracuje s posloupností N směrů $\boldsymbol{d}^{(0)}, \boldsymbol{d}^{(1)}, \boldsymbol{d}^{(2)} \cdots \boldsymbol{d}^{(N-1)}$ takových, že splňují

$$\left(\boldsymbol{d}^{(n)}\right)^T \boldsymbol{A}\boldsymbol{d}^{(m)} = 0$$

a jsou tedy $\boldsymbol{A}$-ortogonální. Iterační metoda je popsaná následujícím vztahem

$$\boldsymbol{x}^{(n+1)} = \boldsymbol{x}^{(n)} + \alpha^{(n)}\boldsymbol{d}^{(n)}.$$

V případě, že není matice $\boldsymbol{A}$ symetrická, lze použít metodu bi-sdružených gradientů, která konverguje úlohu na úlohu se symetrickou maticí následovně.

$$\begin{bmatrix} 0 & \boldsymbol{A} \\ \boldsymbol{A}^T & 0 \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}} \\ \boldsymbol{x} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b} \\ 0 \end{bmatrix} \tag{1.5}$$

V současnosti asi nejpoužívanější jsou **víceúrovňové (multigrid) metody**. Ty odstraňují základní problém Krylovovských metod, tedy vysokou náročnost na hardware už pro středně velké úlohy. Základní myšlenkou je využití několika různě velkých sítí. Ty jemnější pro detailnější řešení a odtsranění vysokofrekvenčních chyb, hrubší pak pro rychlé odstranění nízkofrekvnčních chyb, které nejsou na jemnějších rozpoznatelné a proto často prodlužují konvergenci.

Víceúrovňová metoda postupuje v následujících krocích:

1. několik kroků iterační metody (např. Gauss-Seidel) na jemné síti - presmoothing
2. projekce na hrubší síť - restriction
3. řešení na hrubé síti (na velmi hrubé síti možné použít i přímý řešič), odstranění nízkofrekvenčních chyb
4. projekce zpět na jemnou síť - prolongation
5. několik kroků iterační metody na vyhlazení vysokofrekvenčních chyb - postsmoothing

Úrovní může být i více než 2 a podle způsobu přechodů (restriction a prolongation) mezi nimy se mluví o V-cyklech, W-cyklech, případně F-cyklech. Víceúrovňové metody se využívají ve 2 variantách:

- Geometrické - na úrovni sítě, využívá informaci o sousedních elementech
- Algebraické - na úrovni matice, bez znalosti gometrie, je univerzální pro jakékoliv problémy

Iterační metody řešení lineárních algebrických rovnic často využívají **předpodmínění** (preconditiong). To převede úlohu do tvaru s ekvivalentním řešením ale lepšímy spektrálnímy vlastnostmi matice soustavy. Při levém předpodmínění (existuje i pravé a centrální) předpokládáme soustavu ve tvaru

$$\boldsymbol{P}^{-1}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{P}^{-1}\boldsymbol{b}$$

kde $\boldsymbol{P}$ je předpodmiňovač. Ten zajistí, že konvergence předpodmíněného systému je rychlejší, než bez něj. Při hledání předpodmiňovače $\boldsymbol{P}$ je často uvažována nějaká snadno invertovatelná aproximace matice $\boldsymbol{A}$. Nejjednodušší volbou je tak $\boldsymbol{P} = \boldsymbol{I}$, což ovšem vede k nepodmíněnému problému. Ideální je naopak $\boldsymbol{P} = \boldsymbol{A}$, to ovšem znamená, že inverze předpodmiňovače je stejně náročná jako původní matice. Často se tak volí různé rozklady matice $\boldsymbol{A}$, jako jsou LU, ILU nebo DILU rozklady.

Pro zrychlení konvergence se u iteračních řešičů využívá často i tzv. **vyhlazovač** (smoother), pro který lze používat např. základní Jacobiho, nebo Gauss-Seidelova metodu. Po každé iteraci řešiče metody se pustí několik kroků vybraného vyhlazovače, který „vyhladí“ špičky v reziduu a tím i sníží jeho normovanou hodnotu. Na rozdíl od předpodmínění dokáže vyhlazovač snížit počet kroků nezávisle na velikosti sítě. Pro příklad uveďme iterační vztah pro Gaussův-Seidelův vyhlazovač, kde $\boldsymbol{L}$ je dolní trojúhelníková část matice $\boldsymbol{A}$.

$$\boldsymbol{x}^{(\boldsymbol{k}+\mathbf{1})} = \boldsymbol{x}^{(\boldsymbol{k})} + \boldsymbol{L}^{-1}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(\boldsymbol{k})}), \tag{1.6}$$

Další vyhlazovače jsou pak založeny na principu LU, DILU, nebo Choleského rozkladu.

2 Lineární řešiče v OpenFOAM

OpenFOAM obsahuje kromě základního přímého řešiče `diagonalSolver` tři základní typy řešičů:

1. řešiče metodou sdružených gradientů
2. víceúrovňové řešiče
3. řešiče založené na vyhlazovači

Konkrétní řešiče jsou popsány v tabulce 2.2.

Tabulka 2.1: řešiče v rámci OpenFOAM

Řešič	**Označení**	**Matice $\boldsymbol{A}$**
Předpodmíněné sdružené gradienty	PCG	sym
Předpodmíněné bi-conjugate gradient	PBiCG	asym
Stabilizované předpodmíněné (bi-)conjugate gradient	PBiCGStab	sym/asym
Řešič používající vyhlazovač	smoothSolver	sym/asym
Víceúrovňový řešič	GAMG	sym/asym
Diagonální přímý řešič	diagonal	

To který řešič je možné použít je závislé na matici $\boldsymbol{A}$ a tom, jestli je symetrická nebo nesymetrická. Symetrie $\boldsymbol{A}$ samozřejmě závisí na rovnicích, které chceme řešit. V případě, že zvolíme nesprávný řešič vzhledem k symetričnosti matice $\boldsymbol{A}$ OpenFOAM sám volbu opraví.

Doporučeným řešičem v OpenFOAM je, jak jsme již uvedli v teoretické části obecně, víceúrovňový řešič GAMG (geometric-algebraic multi-grid). Dál je obsažen i řešič založený na vyhlazovači, kdy je možné vybrat použitý smoother. Většinou je Gauss-Seidel nejlepší volbou.

V průběhu řešení je reziduum vyhodnocováno během jednotlivých iterací pomocí vztahu (1.2). Každý řešič má k reziduu trochu jiný přístup, obecně ale dochází k normalizaci a škálování rezidua vztahy

$$n = \sum \left(|\boldsymbol{A}x - \boldsymbol{A}\hat{x}| + |\boldsymbol{b} - \boldsymbol{A}\hat{x}| \right),$$
$$r = \frac{1}{n} \sum |\boldsymbol{b} - \boldsymbol{A}x|.$$

Pro každý typ řešice je třeba nastavit parametr pro velikost rezidua (`tolerance`) a také relativní toleranci (`relTol`) pro poměr aktuálního rezidua v rámci jedné iterace a počátečního rezidua před začátkem iterace. Lineární řešič se následně zastaví pokud je splněna alespoň jedna z těchto podmínek:

1. reziduum je menší než nastavená `tolerance`,
2. relativní tolerance je menší než nastavená `relTol`,
3. je dosaženo nastaveného maximálního počtu iterací `maxIter`.

Předpodmičovač je u většiny řešičů volitelný i když velmi doporučovaný. OpenFOAM implementuje následující předpodmiňovače:

Tabulka 2.2: přdpodmiňovače obsažené v OpenFOAM

Předpodmiňovač	**Označení**
Diagonal incomplete-Cholesky (symetrický)	DIC
Faster diagonal incomplete-Cholesky (DIC with caching)	FDIC
Diagonální neúplný LU rozklad	DILU
Diagonální přímý	diagonal
Víceúrovňový předpodmiňovač	GAMG

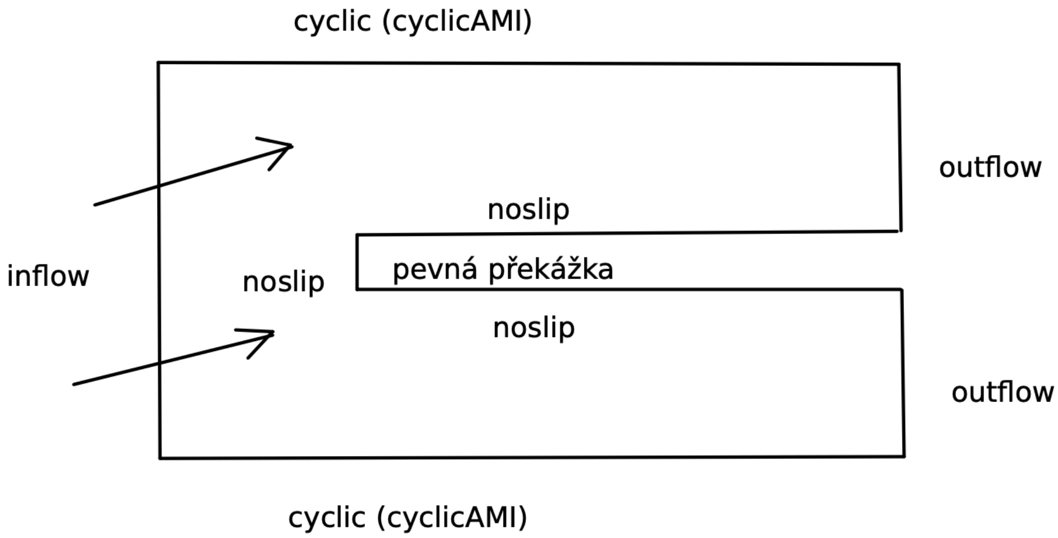
NĚCO O RELAXACI

under-relaxation, a technique used for improving stability of a computation, particularly in solving steady-state problems. Under-relaxation works by limiting the amount which a variable changes from one iteration to the next, either by modifying the solution matrix and source prior to solving for a field or by modifying the field directly. An under-relaxation factor $\alpha, 0 < \alpha \leq 1$ specifies the amount of under-relaxation, ranging from none at all for $\alpha = 1$ and increasing in strength as $\alpha- > 0$ The limiting case where $\alpha = 0$ represents a solution which does not change at all with successive iterations. An optimum choice of α is one that is small enough to ensure stable computation but large enough to move the iterative process forward quickly; values of α as high as 0.9 can ensure stability in some cases and anything much below, say, 0.2 are prohibitively restrictive in slowing the iterative process

1. jaké jsou možnosti, jaké jsou parametry
   (a) diagonalSolver - pokud je matic A diagonální, pak vypočítám $A^{-1} = A/diag(A)$ a jsem hotov (https://www.openfoam.com/documentation/guides/latest/solvers.html)

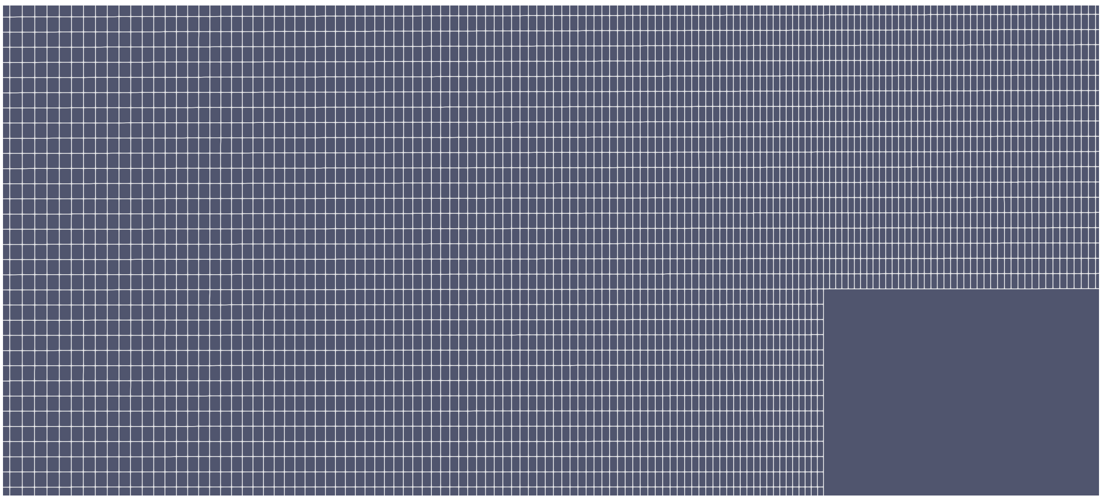
3 Porovnání řešičů

Lineární řešiče porovnáme na konkrétní úloze stacionárního proudění v lopatkové mříži. Zachováme přitom jednotné parametry nastavení úlohy až na nastavení jednotlivých lineárních řešičů. Zajímat nás bude konvergence řešení a výpočetní náročnost. Zadání úlohy je na obrázku 3.1, který definuje geometrii úlohy i okrajové podmínky.

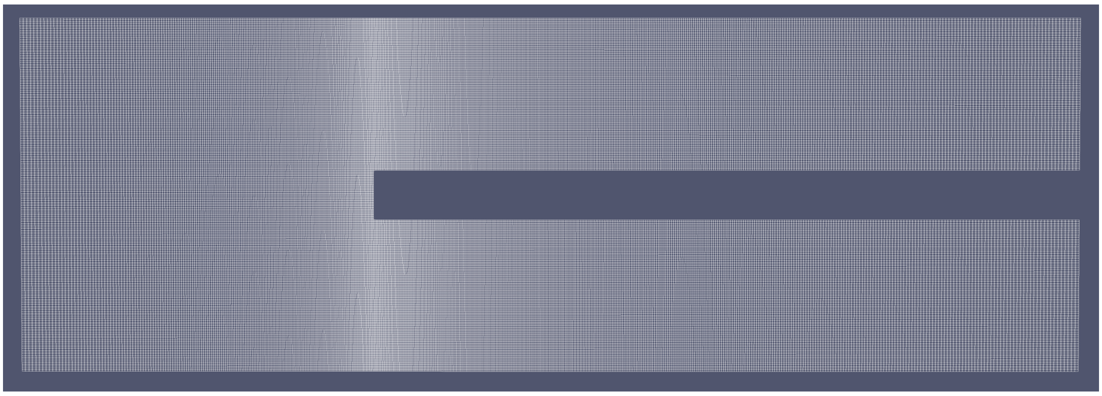


Obrázek 3.1: Popis řešeného modelu

Nejprve musíme definovat síť na které budeme naši úlohu řešit. Úloha je v 2D, ale OpenFOAM řeší vše ve 3D. Třetí souřadnici tedy nastavíme minimální, jen jako jednu buňku, a budeme ji ve výsledcích vlastně ignorovat. Nejvíce se toho bude dít kolem „vykousknuté" části a proto v těchto místech síť uděláme hustší jak je vidět na obrázku 3.2. Celá síť je pak na obrázku 3.3 a obsahuje celkem 108800 jednotlivých buněk, na kterých budeme úlohu numericky integrovat. Konkrétní definice je v souboru `blockMeshDict` v příloze A.



Obrázek 3.2: Detail sítě řešeného modelu



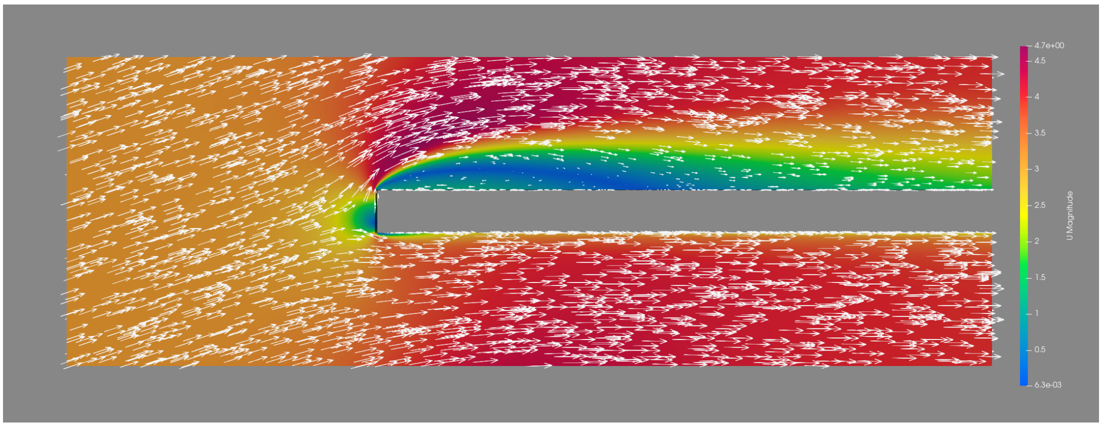
Obrázek 3.3: Síť řešeného modelu

Okrajové podmínky pro jednotlivé veličiny, pro nás primárně pro rychlost U a tlak p jsou definovány v souborech ve složce `0`.

Ostatní parametry řešiče úlohy, netýkající se přímo lineárních řešičů jsou pro všechny simulace totožné a jsou v souborech `transportProperties`, `turbulenceProperties`, `fvSchemes` a `controlDict`. Zde uvedeme jen vybrané parametry popisující naší úlohu.

- application: simpleFoam
- simulationType: RAS
- RASmodel: kEpsilon
- turbulence: on
- transportModel: Newtonian
- nu: 1e-05
- residualControl p: 1e-02
- residualControl ostatní: 1e-04

Na ukázku je zde obrázek 3.4 výsledného klidového stavu pořízený v ParaView. Výsledky úlohy řešené pomocí různých lineárních řešičů vypadají prakticky identicky.



Obrázek 3.4: Klidový stav s vektory rychlosti U v ParaView

Na lineárních řešičích budeme zkoumat vliv následujících parametrů na konvergenci simulace:

- typ použitého řešiče
- typ použitého předpodmiňovače
- typ použitého smootheru
- hodnotu pod-relaxace

Nastavení řešičů se v OpenFOAM dělá v souboru `system/fvSolution`. Ukázkový soubor je v příloze B.

Nejprve nás zajímá typ použitého řešiče, případně přidruženého předpodmiňovače nebo smootheru. V tabulkách 3.1, 3.2 a 3.3 jsou uvedená nastavení a výsledné počty iterací řešení úlohy a čas běhu simulace testovací úlohy.

Konvergenci metody měříme pomocí počtu využitých iterací metody. Maximální počet iterací byl nastaven na 500. Tolerance lineárního řešiče je shodně nastavena na 10^{-6}, relativní tolerance na 0. Velikost reziduí v rámci SIMPLE algoritmu pomocí `residualControl` parametru pro tlak p nastaveno na 10^{-3} a pro rychlost U na 10^{-4}.

Tabulka 3.1: Nastavení lineárních řešičů pro jednotlivé simulace

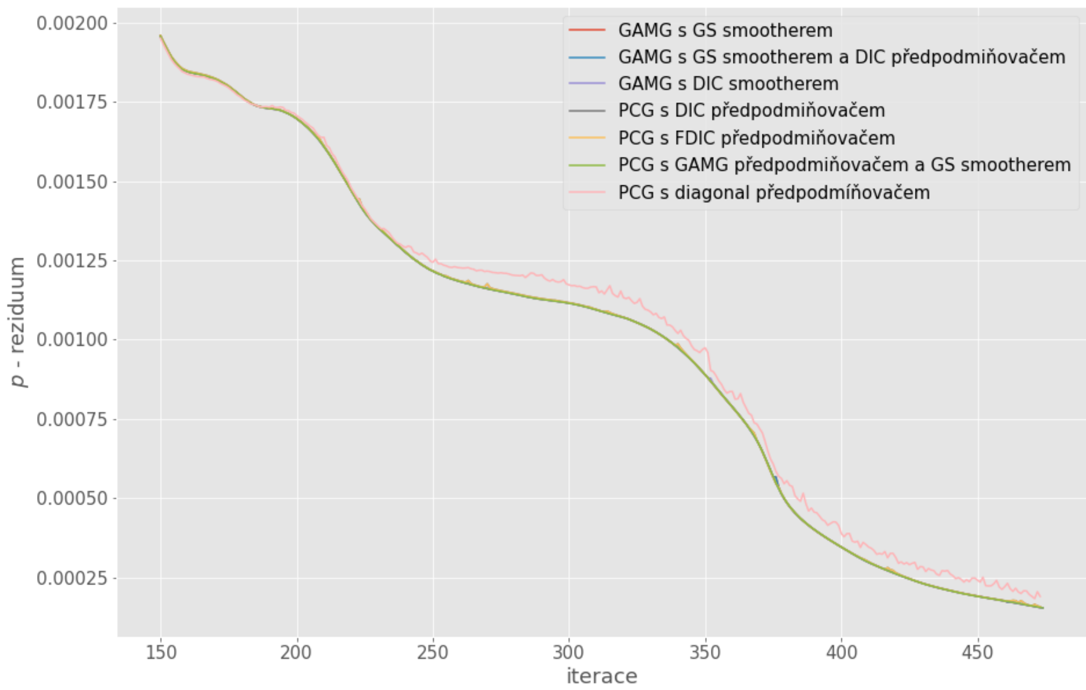
Tlak p: **GAMG**		Rychlost U: **GAMG**			
Předpod.	Smoother	Předpod.	Smoother	# iter	Čas
—	DIC	—	DILU	475	157.94
—	GaussSeidel	—	GaussSeidel	475	176.99

Tabulka 3.2: Nastavení lineárních řešičů pro jednotlivé simulace

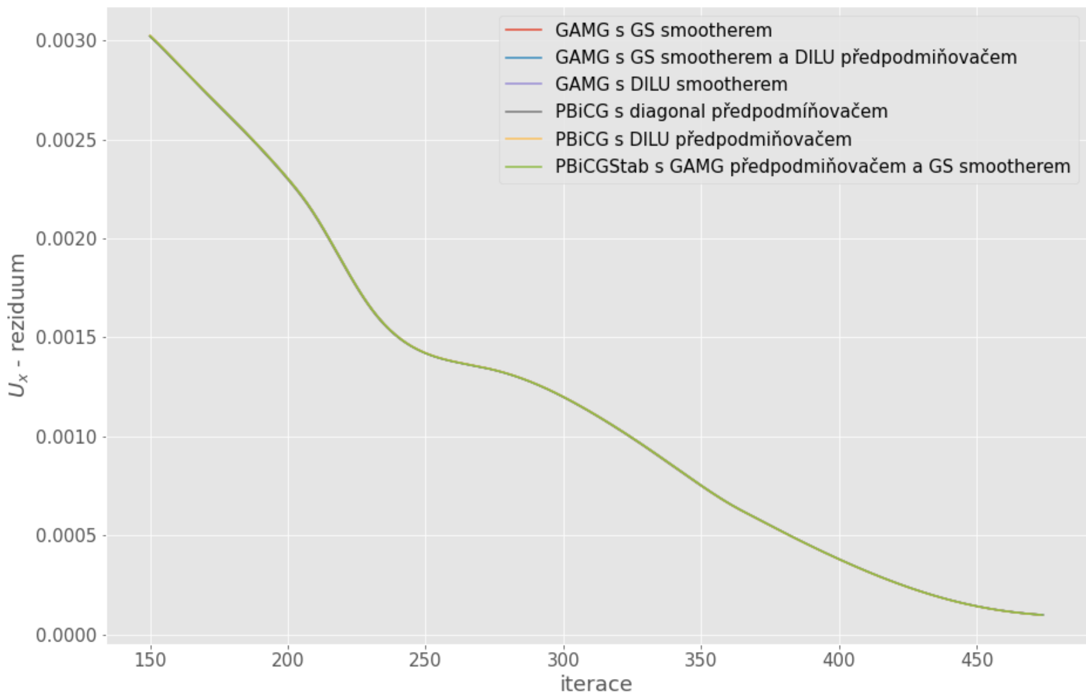
Tlak p: **PCG**		Rychlost U: **PBiCG**			
Předpod.	Smoother	Předpod.	Smoother	# iter	Čas
DIC	—	DILU	—	474	540.48
FDIC	—	DILU	—	474	531.14
GAMG GS smooth.	—	GAMG GS smooth.	—	475	184.85
diagonal	—	diagonal	—	474	679.26

Z výsledků je patrné, že pro naši úlohu ideální GAMG řešič, což odpovídá i obecným doporučením pro řešení úloh pro klidový stav. PCG/PBiCG řešiče dospějí ke stejným výsledkům, v podobném počtu iterací, jen za víc jak dvojnásobný čas. Obecně al GAMG není tak dobře škálovatelný jaklo PCG/PBiCG a je tedy proto možné, že pro jinou úlohu, případě i stejnou úlohu ale řádově větší síť by mohl být PCG/PBiCG řešič vhodnější.

Na obrázku 3.5 jsou zobrazeny průběhy simulací pomocí závislosti velikosti rezidua pro tlak p a různé řešiče. Na obrázku 3.6 je pak totéž, jen pro rychlost v $x$ové souřadnici U_x.



Obrázek 3.5: Konvergence lineárních řešičů v různé konfiguraci pro tlak p



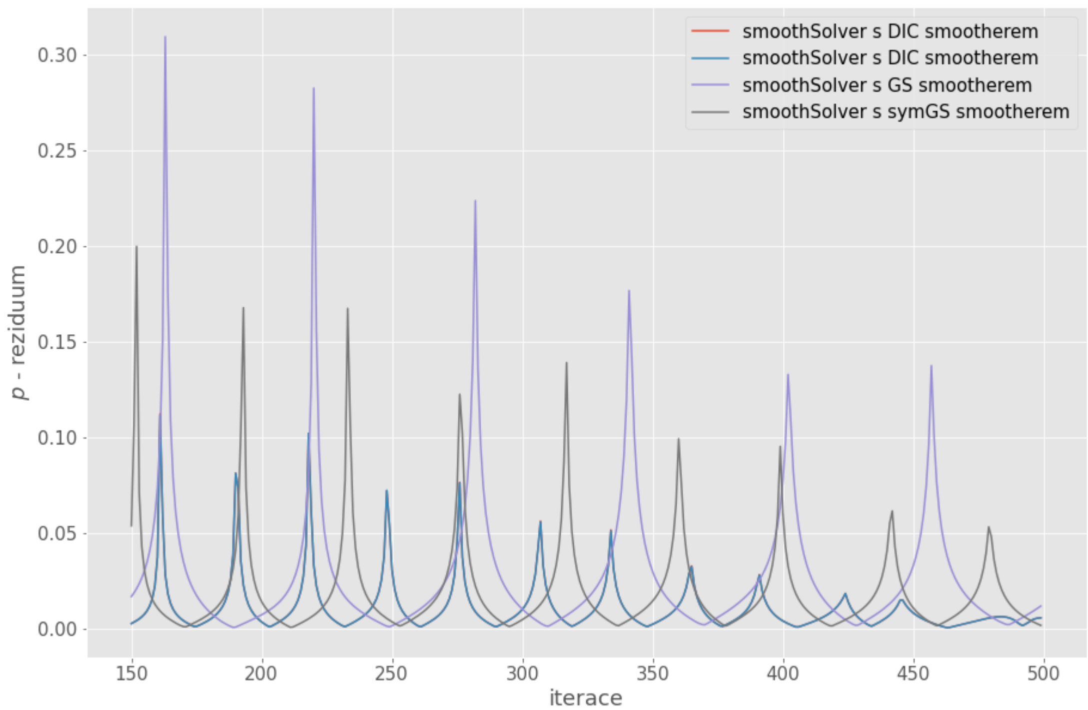
Obrázek 3.6: Konvergence lineárních řešičů v různé konfiguraci pro rychlost U_x

Z výsledků v tabulce 3.3 je vidět, že smoothSolver není pro naši úlohu vhodný. Ani v jednom případě nedokonvergovala úloha ke stacionárnímu stavu v stanovém maximálním počtu 500 iterací. I čas potřebný ke zpracování této úlohy je výrazně vyšší.

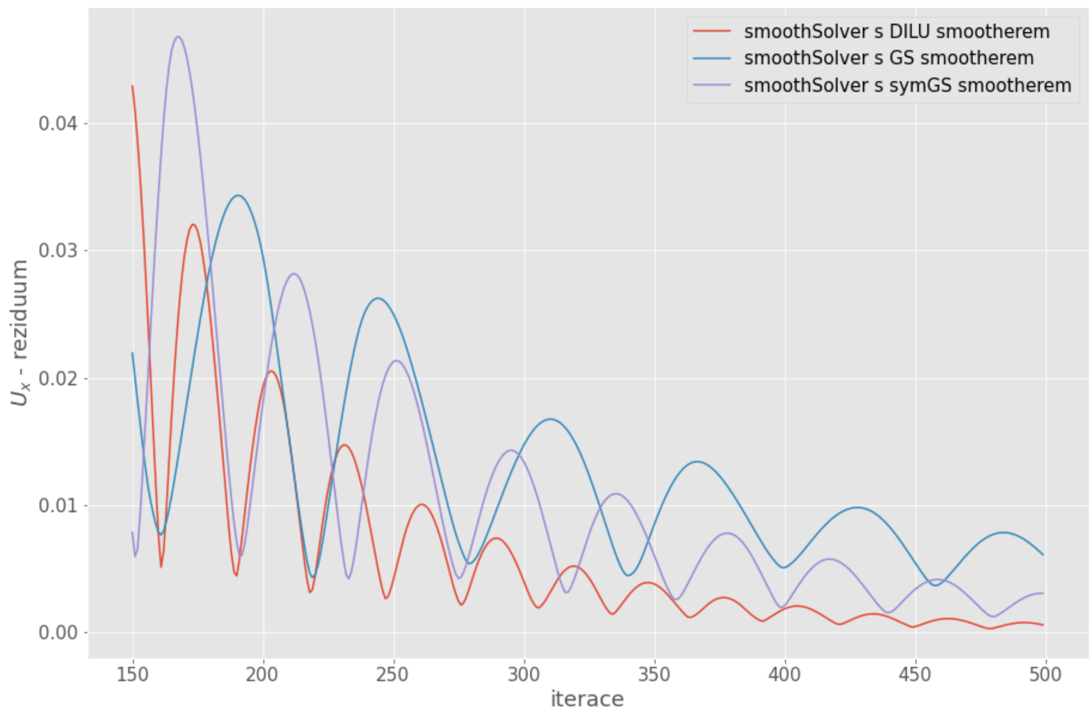
Tabulka 3.3: Nastavení lineárních řešičů pro jednotlivé simulace

Tlak p: **smoothSolver**		Rychlost U: **smoothSolver**			
Předpod.	Smoother	Předpod.	Smoother	# iter	Čas
—	DIC	—	GaussSeidel	500+	1136.21
—	DIC	—	DILU	500+	1039.73
—	GaussSeidel	—	GaussSeidel	500+	1011.42
FDIC	GaussSeidel	—	GaussSeidel	500+	1008.47
—	symGauss-Seidel	—	symGauss-Seidel	500+	1370.58

Na obrázcích 3.7 (pro tlak p) a 3.8 (pro rychlost U_x) je vidět proč smooth řešič nekonverguje, nebo konverguje velmi pomalu. V obou sledovaných veličinách reziduum osciluje a jeho hodnota se snižuje velmi pomalu.



Obrázek 3.7: Konvergence lineárních řešičů smoothSolver v různé konfiguraci pro tlak p



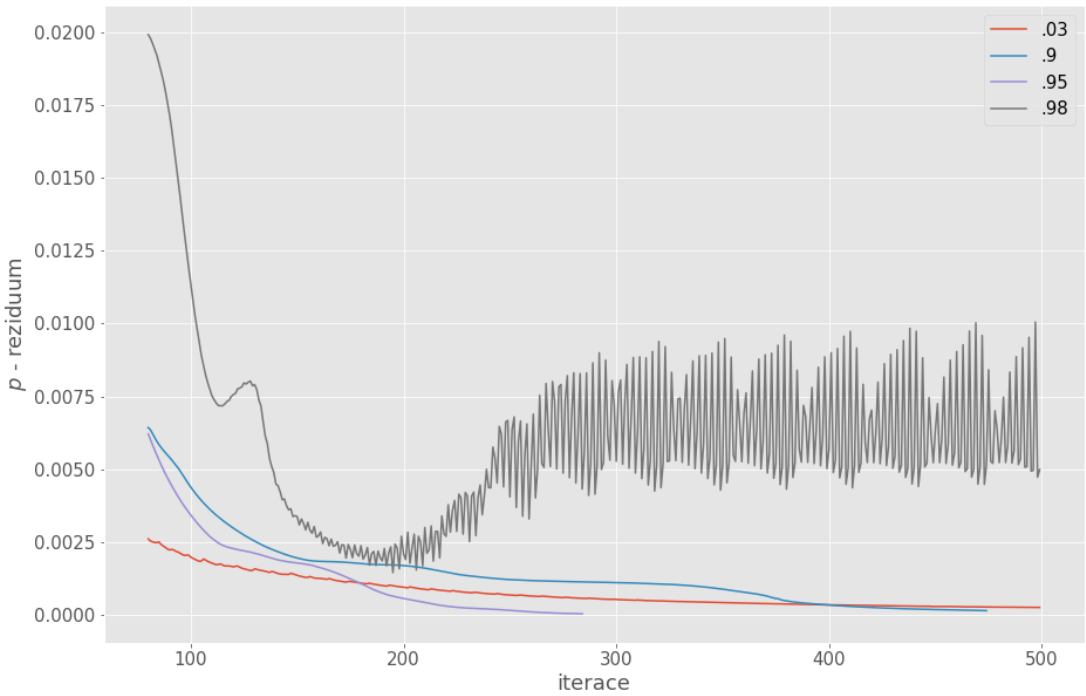
Obrázek 3.8: Konvergence lineárních řešičů smoothSolver v různé konfiguraci pro rychlost U_x

Na GAMG řešiči si ukážeme vliv hodnoty relaxace. Obecně platí, že vyšší relaxace zvyšuje rychlost konvergence, nižší metodu zestabilní. V tabulce 3.4 jsou výsledky pro různé hodnoty relaxačního parametru v porovnání s defaultní hodnotou 0.9.

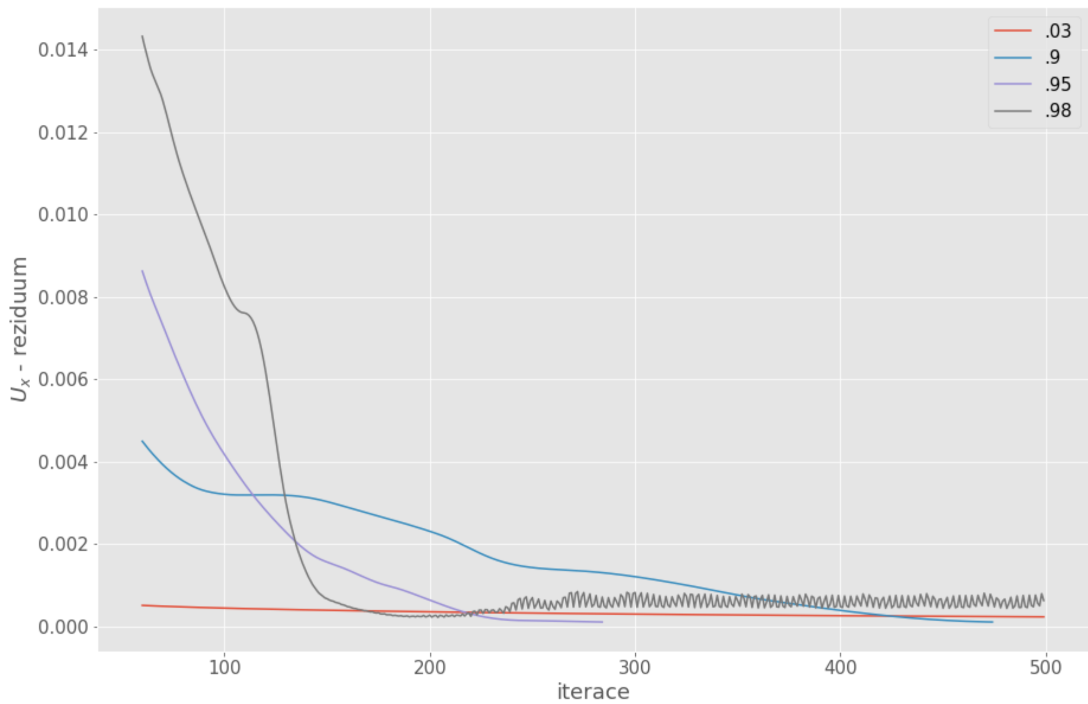
Tabulka 3.4: Nastavení lineárních řešičů pro jednotlivé simulace

Tlak p: **GAMG**		Rychlost U: **GAMG**				
Předpod.	Smoother	Předpod.	Smoother	Relaxace	# iter	Čas
—	GaussSeidel	—	GaussSeidel	0.9	475	176.99
—	GaussSeidel	—	GaussSeidel	0.3	500+	151.49
—	GaussSeidel	—	GaussSeidel	0.95	285	114.15
—	GaussSeidel	—	GaussSeidel	0.98	500+	257.28

Ideální hodnotou pro nejrychlejší konvergenci je 0.95, úloha dokoverguje v 285 iteracích za přibližně 114s. Hodnota relaxace 0.3 je moc nízká na to, aby úloha dokonvergovala v maximálním počtu 500 iterací. Hodnota 0.98 je naopak moc vysoká a reziduum úlohy se rozkmitá a řešení nekonverguje, jak je vidět z obrázků 3.9 a 3.10.



Obrázek 3.9: Konvergence lineárních řešičů smoothSolver v různé konfiguraci pro rychlost U_x



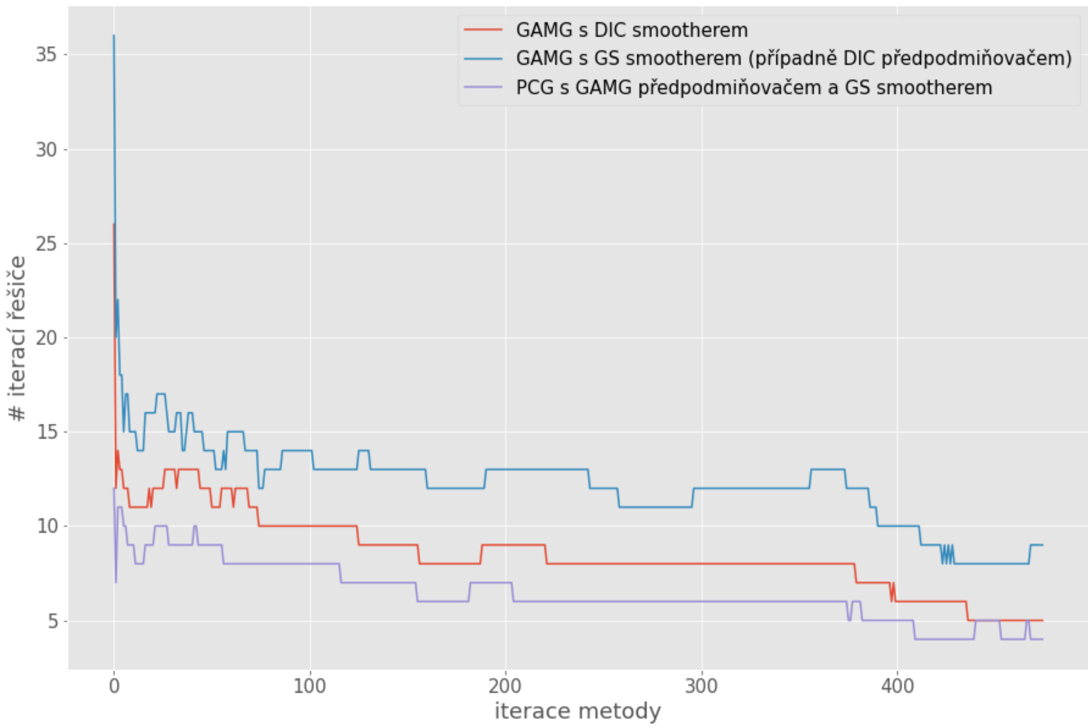
Obrázek 3.10: Konvergence lineárních řešičů smoothSolver v různé konfiguraci pro rychlost U_x

Zdálo by se, že použít relaxaci na „opravení“ smooth řešiče je dobrý nápad. Z tabulky 3.5 je ale vidět, že se výsledky nijak nezlepší.

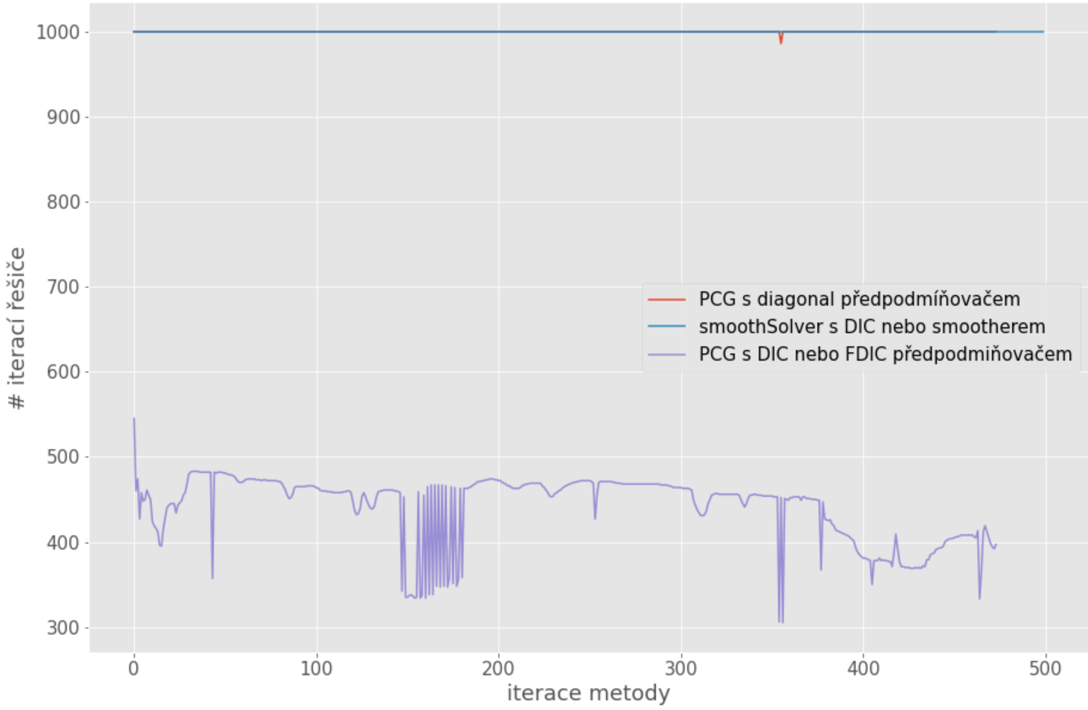
Tabulka 3.5: Nastavení lineárních řešičů pro jednotlivé simulace

Tlak p: **smoothSolver**		Rychlost U: **smoothSolver**				
Předpod.	Smoother	Předpod.	Smoother	Relaxace	# iter	Čas
—	GaussSeidel	—	GaussSeidel	0.9	500+	1011.42
—	GaussSeidel	—	GaussSeidel	0.3	500+	951.71
—	GaussSeidel	—	GaussSeidel	0.95	500+	1086.7
—	GaussSeidel	—	GaussSeidel	0.98	500+	1222.24
FDIC	GaussSeidel	—	GaussSeidel	0.9	500+	1008.47

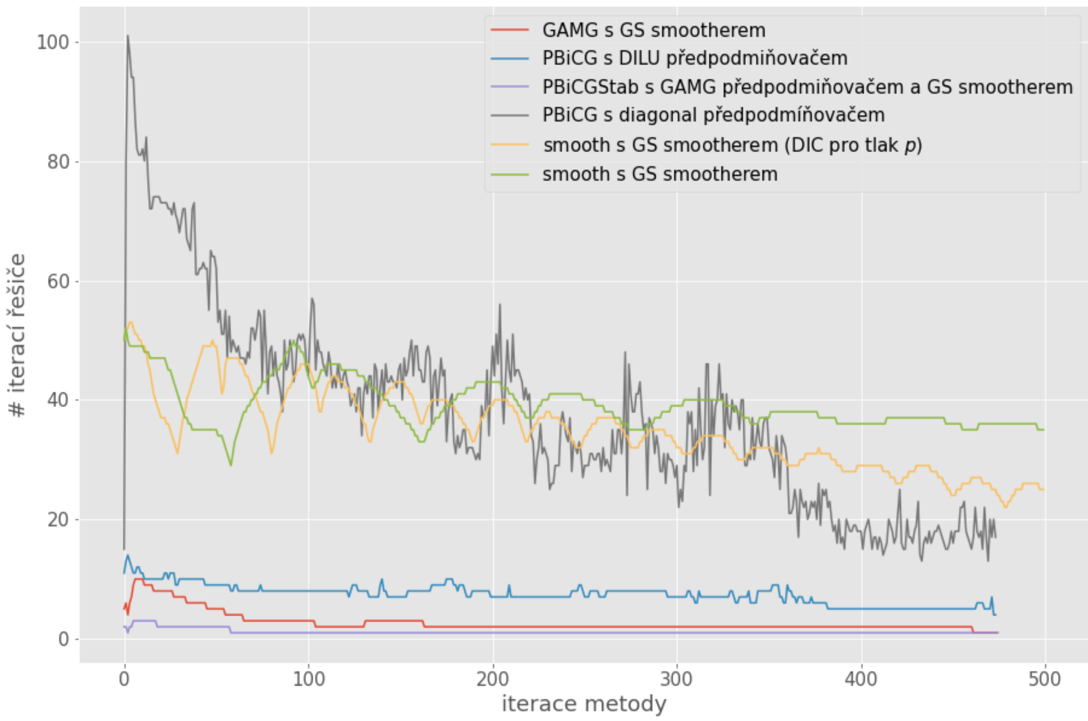
Další parametr, který popíše „úspěšnost“ lineárního řešiče při řešení úlohy je počet iteračních kroků lineárního řešiče v rámci časových kroků metody. V každém kroku řeší lineární řešič soustavu a zastaví iterování v chvíli, kdy dosáhne nastavené tolerance. Na obrázcích 3.11 a 3.12 je vidět tato závislost pro řešení tlaku p, na obrázku 3.13 pak pro rychlost U_x. Ze všech tří grafů opět vyplývá, že GAMG řešič dosahuje nejlepší konvergence. Naopak smoothSolver naráží ve všech časových iterací metody na maximální možnou hodnotu iterací řešiče 1000.



Obrázek 3.11: Počet iterací lineárního řešiče v závislosti na iteraci metody pro tlak p



Obrázek 3.12: Počet iterací lineárního řešiče v závislosti na iteraci metody pro tlak p



Obrázek 3.13: Počet iterací lineárního řešiče v závislosti na iteraci metody pro rychlost U_x

4 | Závěr

Ukázali jsme, jakými parametry lineárních řešičů se dá ovlivnit výsledná konvergence úlohy. Obecně nelze říci, který řešič je nejvhodnější a to ani pro konkrétní úlohu. Pro naši ukázkovou úlohu proudění v lopatkové mříži a pro konkrétní síť je nejvhodnější GAMG řešič s GaussSeidel smootherem a relaxací nastavenou na 0.95. Tato volba ale nemusí být optimální pro jinou síť, jinak nastavené parametry turbulence nebo i jiný hardware, na kterém úlohu řešíme. Předchozí text by tedy měl sloužit přinejlepším jen jako přehled a ukázka řešičů dostupných v OpenFOAM.

A | blockMeshDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|  \\    /   O peration     | Version:  v2006
|   \\  /    A nd           | Website:  www.openfoam.com
|    \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
version         2.0;
format          ascii;
class           dictionary;
object          blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

scale   1;

vertices
(
(0 0    0)
(1 0    0)
(3 0    0)
(3 0.43 0)
(1 0.43 0)
(1 0.57 0)
(3 0.57 0)
(3 1    0)
(1 1    0)
(0 1    0)
(0 0.57 0)
(0 0.43 0)
```

```
(0 0    0.1)
(1 0    0.1)
(3 0    0.1)
(3 0.43 0.1)
(1 0.43 0.1)
(1 0.57 0.1)
(3 0.57 0.1)
(3 1    0.1)
(1 1    0.1)
(0 1    0.1)
(0 0.57 0.1)
(0 0.43 0.1)


);

blocks
(
hex (0 1 4 11 12 13 16 23) (200 86 1) simpleGrading (0.2 1 1)
hex (1 2 3 4 13 14 15 16) (400 86 1) simpleGrading (5 1 1)
hex (11 4 5 10 23 16 17 22) (200 28 1) simpleGrading (0.2 1 1)
hex (10 5 8 9 22 17 20 21) (200 86 1) simpleGrading (0.2 1 1)
hex (5 6 7 8 17 18 19 20) (400 86 1) simpleGrading (5 1 1)
);

edges
(
);

boundary
(
frontAndBack
{
type empty;
faces
(
(0 1 4 11)
(1 2 3 4)
(11 4 5 10)
(10 5 8 9)
(5 6 7 8)

(12 13 16 23)
(13 14 15 16)
```

```
(23 16 17 22)
(22 17 20 21)
(17 18 19 20)
);
}

top
{
type cyclic;
neighbourPatch bottom;
faces
(
(9 8 20 21)
(8 7 19 20)
);
}

bottom
{
type cyclic;
neighbourPatch top;
faces
(
(0 1 13 12)
(1 2 14 13)
);
}

fixedBalk
{
type wall;
faces
(
(5 6 18 17)
(4 5 17 16)
(4 3 15 16)
);
}

outlet
{
type patch;
faces
(
```

```
(6 7 19 18)
(2 3 15 14)
);
}

inlet
{
type patch;
faces
(
(0 11 23 12)
(11 10 22 23)
(10 9 21 22)
);
}
);
mergePatchPairs
(
);

// ************************************************************** //
```

B fvSolution

```
/*---------------------------*- C++ -*-----------------------------*\
| =========                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|  \\    /   O peration     | Version:  v2006
|   \\  /    A nd           | Website:  www.openfoam.com
|    \\/     M anipulation  |
\*-------------------------------------------------------------------*/
FoamFile
{
version     2.0;
format      ascii;
class       dictionary;
location    "system";
```

```
object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
p
{
solver          smoothSolver;
smoother        GaussSeidel;
tolerance       1e-06;
relTol          0;
}

"(U|k|epsilon|omega|f|v2)"
{
solver          smoothSolver;
smoother        GaussSeidel;
tolerance       1e-06;
relTol          0;
}
}

SIMPLE
{
nNonOrthogonalCorrectors 0;
consistent      yes;

residualControl
{
p               1e-3;
U               1e-4;
"(k|epsilon|omega|f|v2)" 1e-4;
}
}

relaxationFactors
{
equations
{
U               .95;
".*"            .95;
}
}
```

```
// ************************************************************** //
```