VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ENERGETICKÝ ÚSTAV
ENERGY INSTITUTE

OPENFOAM: MOŽNOSTI OPEN-SOURCE CFD

OPENFOAM: CAPABILITIES OF THE OPEN-SOURCE CFD

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE Jan Zbavitel
AUTHOR

VEDOUCÍ PRÁCE Ing. Jiří Kozák
SUPERVISOR

BRNO 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Energetický ústav
Student:	**Jan Zbavitel**
Studijní program:	Strojírenství
Studijní obor:	Základy strojního inženýrství
Vedoucí práce:	**Ing. Jiří Kozák**
Akademický rok:	2015/16

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Openfoam: možnosti open-source CFD

Stručná charakteristika problematiky úkolu:

Komerční CFD programy představují robustní a odladěný nástroj připravený pro koncového uživatele, tento komfort je ovšem vykoupen značnými pořizovacími náklady. Alternativu k těmto komerčním programům představují otevřené CFD programy. Nejvýraznějším představitelem této skupiny CFD programů je OpenFoam.
V první části bakalářské práce autor provede analýzu možností programu OpenFoam a obeznámí se se strukturou simulované úlohy.
Druhá část práce bude obsahovat řešení několika jednoduchých úloh. Tyto úlohy budou zpracovány formou návodu.

Cíle bakalářské práce:

Autor této práce provede analýzu možností CFD programu OpenFoam a seznámí se se strukturou řešené úlohy.
Bakalářská práce bude dále obsahovat řešení několika autorem zvolených úloh z oblasti hydrauliky. Tyto úlohy budou zpracovány formou návodu.

Seznam literatury:

Dokumentace OpenFoam

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2015/16

V Brně, dne

L. S.

doc. Ing. Jiří Pospíšil, Ph.D. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

Abstrakt

Bakalářská práce je zaměřena na základní analýzu možností open-source CFD programu OpenFoam a představení jeho klíčových vlastností. Získané vědomosti budou využity při tutoriální úloze analýzy Kármánovy vírové stezky.

Klíčová slova: Simulace proudění vody, CFD, OpenFoam, Kármánova vírová stezka

Abstract

The aim of the bachelor's thesis is to provide a basic survey of capabilities of the open-source CFD instrument OpenFoam and to introduce its major qualities. A collected knowledge will result in analysis of the von Karman vortex street tutorial case.

Keywords: Numerical simulation of water flow, CFD, OpenFoam, von Karman vortex street

BIBLIOGRAFICKÁ CITACE

ZBAVITEL, J. *Openfoam: možnosti open-source CFD*. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství. 2016. 49s. Vedoucí bakalářské práce Ing. Jiří Kozák.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně na základě svých znalostí, odborných konzultací, literatury uvedené v seznamu a pod vedením vedoucího bakalářské práce pana Ing. Jiřího Kozáka.

V Brně dne 27. 5. 2016 .....

PODĚKOVÁNÍ

Za podporu, trpělivost a cenné připomínky při zpracování této práce děkuji vedoucímu bakalářské práce panu Ing. Jiřímu Kozákovi.

Dále děkuji rodině a svým nejbližším za jejich pevnou podporu ve všech okamžicích mého studia.

OBSAH

ÚVOD

Uchopit v inženýrské praxi ono neviditelné proudění a pronést o něm soud vyžaduje jistou míru fantazie i zodpovědnosti; historie zaznamenala mnoho případů, kdy nenápadný a nepředvídaný efekt způsobil zkázu lidského úsilí či dokonce životů. Také proto je nutné co nejlépe rozumět nástrojům, jež se nabízí pro zkoumání kontinua, a získat pro ně dobrý cit.

Existuje mnoho situací, kdy experimentální metody nemohou samotné podat dostatečný popis proudění nebo je zájem snížit míru jejich využití kvůli úspoře času či nákladů. Pro tyto účely bylo v minulosti vyvíjeno množství metod a postupů, z nichž se následně zformovalo zcela nové odvětví fluidní mechaniky.

CFD (z anglického Computational Fluid Dynamics) je podle slov J. D. Andersona „umění nahrazovat integrály a parciální derivace v Navier-Stokesových rovnicích diskretizovanými algebraickými formami a řešit vzniklé soustavy lineárních rovnic s cílem nalézt hodnoty vlastností proudových polí v diskrétních bodech“ [7,14]. Jádrem této diskretizace je pak nejčastěji metoda konečných objemů, na niž navazují další numerická schémata a algoritmy řešení.

Mezi svobodnými CFD nástroji má významné postavení právě OpenFOAM, jenž své uplatnění nachází na nejširší akademické půdě i v různorodých odvětvích průmyslu. Nabídka prozkoumat možnosti tohoto softwaru byla proto zároveň výzvou, zdrojem obav o úspěch, ale i pokušením.

1. VÝPOČTOVÉ MODELOVÁNÍ PROUDĚNÍ

1.1 Úvod do CFD

Vzhledem k široké škále možných využití výpočtového modelování vyžadují konkrétní úlohy poměrně specifický přístup. Cílem této podkapitoly je proto vytvořit základní přehled různých druhů proudění tekutin a nastínit hlavní body jejich simulace pomocí CFD.

V prvé řadě je podstatné, v jakém vztahu jsou vůči sobě známé a hledané parametry či okolnosti. Hledáme-li popis proudění pro konkrétní geometrii a omezující podmínky[1], nazýváme úlohu *přímou*. Naopak v úloze *nepřímé* známe požadovaný tvar i vlastnosti proudění a tomu je potřeba podřídit příslušnou geometrii domény (např. návrh obtékaného profilu). Řešení nepřímé úlohy je možné manuální úpravou geometrie a opětovným provedením výpočtu či systematickou modifikací různými typy optimalizačních algoritmů, které svou složitost stupňují od heuristických až po evoluční metody.

Při řešení úloh přímých pak obvykle postup řešení prochází těmito hlavními body:

- rozbor problému, výpočet podobnostních čísel a charakteru proudění, odhad míst s vysokými gradienty
- tvorba geometrie a výpočtové sítě, definice okrajových a počátečních podmínek
- volba numerických schémat a algoritmů řešení algebraických rovnic, definice dalších funkcí
- výpočet a sledování konvergence řešení, případně úprava předchozích bodů nebo časového kroku
- vizualizace a vyhodnocení výsledků, porovnání s experimentálními daty

1.2 Matematický a fyzikální základ modelu

Matematický model definuje pole jednotlivých fyzikálních veličin jako závisle proměnné podle příslušných řídících diferenciálních rovnic. Pro jednoznačné určení modelované situace je proto dále zapotřebí popis počátečních a okrajových podmínek (dále v 1.2.2), jejichž počet je určen druhem a řádem příslušné rovnice.

1 Omezujícími podmínkami jsou míněny podmínky okrajové a počáteční.

1.2.1 Základní rovnice

Rovnice zachování jsou v mechanice tekutin vyjádřeny formálně jinak, než v mechanice klasické; mají však podobný fyzikální význam[2]:

- Rovnice kontinuity (pro neustálené prostorové proudění stlačitelné kapaliny)vyjadřuje zákon zachování hmotnosti, respektive hmotnostního toku. Pro ustálené proudění nestlačitelné tekutiny pak rovnice přechází do tvaru

$$\frac{\partial v_j}{\partial x_j} = 0$$

  který vyjadřuje, že zřídlovitost rychlosti částic je nulová (tj. že rychlost nikde nevzniká, může se pouze transformovat) a hmotnostní tok je konstantní.

- Navier-Stokesova pohybová rovnice

$$\frac{\partial v_i}{\partial t} + \frac{\partial v_i}{\partial x_j} v_j = a_{f(i)} - \frac{1}{\rho}\frac{\partial p}{\partial x_i} + \frac{\lambda + \mu}{\rho} \cdot \frac{\partial^2 v_j}{\partial x_i \cdot \partial x_j} + \frac{\mu}{\rho} \cdot \frac{\partial^2 v_i}{\partial x_j \cdot \partial x_j}$$

  popisuje zachování hybnosti pro viskózní stlačitelnou kapalinu. V případě ustáleného proudění stlačitelné kapaliny je člen s tzv. druhou (objemovou) viskozitou λ zanedbatelný, jeho vliv však významně narůstá s rostoucí frekvencí pulzací při neustáleném proudění [15]. Pro popis turbulentního proudění se dále využívá Reynoldsova rovnice

$$\frac{\partial v_i}{\partial t} + \frac{\partial (v_i \cdot v_j)}{\partial x_j} = a_{(f)i} - \frac{1}{\rho} \cdot \frac{\partial \left(\rho \cdot \overline{v'_i \cdot v'_j}\right)}{\partial x_j} + \frac{\eta}{\rho} \cdot \frac{\partial^2 v_i}{\partial x_j \cdot \partial x_j}$$

  kde $\left(\rho \cdot \overline{v'_i \cdot v'_j}\right)$ je turbulentní napětí.

- Rovnice pro přenos tepla vyjadřuje zákon zachování energie. V širokém pásmu řešených úloh nicméně nejsou jevy spojené s přenosem tepla podstatné a do matematického modelu proto často nemusí být zahrnuty.

2 uvedené rovnice a jejich úpravy vychází ze zdrojů [10] a [15]

1.2.2 Doplňkové podmínky

Konkrétní a jednoznačné řešení systému rovnic určují doplňkové podmínky. Ty mohou buď uzavírat výpočtový prostor definicí vnějších vlivů (takové nazýváme okrajovými podmínkami), nebo definovat jeho počáteční stav (počáteční podmínky).

Počáteční podmínky

Nastavením počátečních podmínek je určeno rozložení hodnot polí uvnitř domény pro výchozí čas simulace. Po spuštění výpočtu jsou tyto hodnoty postupně nahrazovány aktuálním řešením založeným na podmínkách okrajových [6], dokud není dosaženo požadované konvergence řešení. Čím více se tedy počáteční podmínky blíží konečnému řešení, tím rychlejší bude samotný výpočet.

Při modelování stacionárních úloh je často nutné vycházet z konstantních hodnot v celém objemu. Jelikož je však výsledkem ustálený stav systému, nemá tento nedostatek žádný vliv na kvalitu řešení. Naopak pokud je simulace v čase nestálá, může být důležité dostatečně přesně vystihnout počáteční stav, od kterého se výpočet odvíjí, a to především z důvodu náročnosti (či horší konvergence) výpočtu. Je proto běžné, že se jako výchozí základ nestacionárních simulací využívají hodnoty řešení ze simulací stacionárních.

Okrajové podmínky

Na hranicích výpočtové domény jsou definovány okrajové podmínky, které jsou v průběhu simulace neměnné a korespondují s požadovaným řešením. Stanovení těchto podmínek je nezbytné u všech typů úloh, neboť konkrétní řešení vlastně definují. Mohou to být kupříkladu:

- Podmínky na průtočných hranicích: Na vstupu (inlet) a výstupu (outlet) mohou být definovány například následující kombinace okrajových podmínek:

Inlet	*Outlet*
rychlost *u*	podmínka ustáleného proudu[3] $\frac{\partial \mathrm{u}}{\partial \mathrm{n}} = 0, \frac{\partial \mathrm{p}}{\partial \mathrm{n}} = 0, \frac{\partial \mathrm{T}}{\partial \mathrm{n}} = 0$
totální tlak	statický tlak[4]
rychlost *u*	statický tlak[5]

tab. 1: Volba podmínek na průtočných hranicích [12]

- Podmínky na stěně: Na styčných plochách definujeme rychlost kapaliny nulovou (pevná stěna) nebo s volným pokluzem. Případně se mohou předepisovat dynamické vlivy, drsnost, teplota a vlastnosti přestupu tepla.

3 podmínka ustáleného proudu pouze kontroluje, zda všechen hmotnostní tok vytekl ven; předpokládá, že profily veličin se dále nemění, což může vyžadovat dlouhou doménu (ovlivnění charakteru proudění)

4 poměrně nejednoznačná kombinace – není jak rozdělit p_{stat} a p_{dyn}, hrozí špatná konvergence

5 nejdoporučovanější rozložení podmínek – z hlediska matematiky je nejjednoznačnější

- Podmínky symetrie: Snižují náročnost výpočtu symetrických úloh redukcí řešené oblasti. Požaduje se nulová normálová rychlost a nulové gradienty všech hledaných veličin pro hraniční (řeznou) plochu [12].
    - periodické podmínky – uplatňují se pro translačně či rotačně periodické tvary proudění. Výpočet se často provádí jen v takové části objemu, která již nevykazuje další symetrické vlastnosti.
    - podmínky osové symterie – určují osu a 2D výpočtovou oblast s úplnou rotační symetrií proudu

Okrajové podmínky je nezbytné umístit co nejdál od míst, ve kterých požadujeme přesné řešení proudění, neboť jejich zjednodušený model ve svém blízkém okolí nevystihuje přesné chování tekutiny.

1.2.3 Metoda konečných objemů

Metoda konečných objemů poskytuje diskretizaci výpočtového objemu (domény), na který tak mohou být aplikovány diskretizované transportní rovnice (obvykle v integrální formě) [6]. Tím vznikne konečný počet lineárních rovnic, jejichž soustava je v maticích řešena příslušnými numerickými metodami.

Veličiny vyskytující se v těchto rovnicích je možné lokalizovat pro každý elementární objem v jednom nebo více bodech, což rozlišuje dvě následující strategie:

- soustředná mříž (co-located grid) definuje veškeré veličiny v jediném bodě elementu
- posunutá mříž (staggered grid) soustředí v centru buněk pouze veličiny objemového charakteru (např. tlak, hustota) a veličiny spojené s tokem (např. rychlost) lokalizuje na jejich stěny.

Určení hodnot veličin v jiných prostorových umístěních je zajištěno pomocí interpolace a řešení na soustředných mřížích tak vedou k větší míře využití těchto interpolačních operací.

1.2.4 Modelování turbulencí

Přestože v současné době neexistuje univerzálně platný model turbulence, nabízí se různé přístupy pro zjednodušení základních rovnic a jejich výsledkem je celá řada modelů vhodných pro různé typy úloh.

DNS (Direct Numerical Simulation)

Přímá numerická simulace řeší turbulentní proudění jako soustavu základních rovnic bez dalšího zjednodušení. Hustota výpočtové sítě proto musí řádově odpovídat velikosti nejmenších vírů před jejich disipací na teplo, což z této obvykle nejpřesnější metody činí zároveň také výpočtově nejnáročnější přístup. Jelikož zároveň s vyšší hodnotou Reynoldsova čísla roste též objem zpracovávaných dat (a tím nároky na operační paměť) [13], nemusí být výpočet při současném stavu výpočetní techniky realizovatelný.

RANS (Reynolds Averaged Navier-Stokes Equations)

Reynoldsovo časové středování Navier-Stokesových rovnic je běžným modelem pro většinu inženýrských úloh turbulentního proudění. Základem tohoto přístupu je rozdělení rychlostního pole na časově středované rychlosti a rychlosti fluktuační, které nejsou řešeny jako vírové struktury v diskretizovaném prostoru, ale modelovány zvlášť. Právě díky tomu při výpočtu postačuje relativně hrubá síť, což je zásadní výhoda z hlediska výpočtové náročnosti. Pro svou statistickou povahu jsou modely tohoto typu vhodné především při stacionárních simulacích nebo hrubých odhadech periodických dějů, zatímco v ostatních nestacionárních výpočtech dosahují relativně věrohodných výsledků spíše náročnější metody [13].

Modely RANS postavené na Boussinesquově hypotéze turbulentní viskozity počítají hodnoty v jejím poli přidanými rovnicemi [13]. Počet takto použitých diferenciálních rovnic rozděluje modely této skupiny na *nularovnicové* (použita je jedna algebraická rovnice), *jednorovnicové* (jedna diferenciální rovnice) a analogicky *dvourovnicové*. Mezi dvourovnicové modely patří například často používané větve k-ε a k-ω, kde jedna diferenciální rovnice slouží pro vyjádření turbulentní kinetické energie k a druhá pro určení rychlosti disipace ε, resp. víři-vosti ω.

Další podskupinou RANS jsou potom modely založené na metodě Reynoldsova napětí.

LES (Large Eddy Simulation)

Metoda velkých vírů rozlišuje dvě pásma velikostí vírových struktur. Velké víry, které lze zachytit výpočetní sítí, jsou řešeny vpodstatě přímou numerickou simulací, zatímco malé jsou parametrizovány tzv. subgridními modely a filtrací dále odstraněny z turbulentního pole [13]. Volba šíře pásma tohoto filtru ovlivňuje hustotu sítě, která musí být (především v blízkosti stěny) značně jemnější, než u modelů RANS [9].

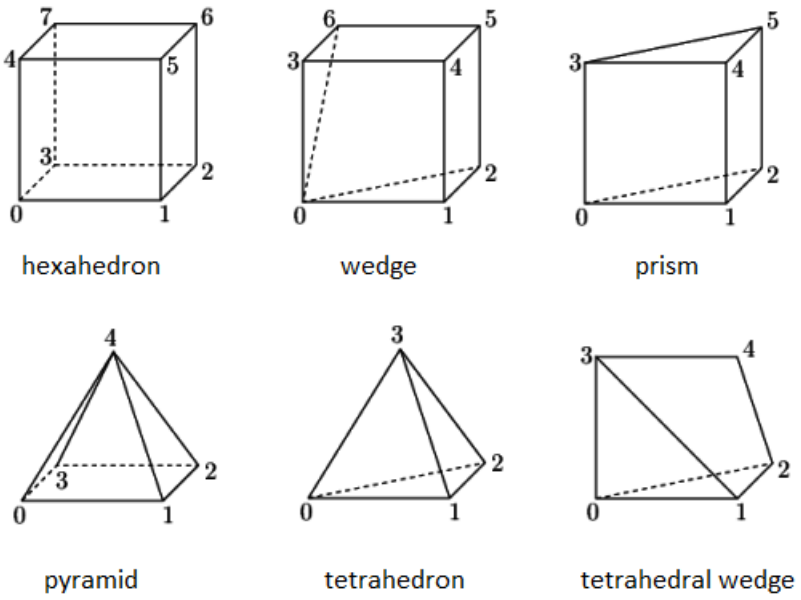
DES (Detached Eddy Simulation)

Jedná se o hybridní model kombinující výhodné vlastnosti metod LES a RANS. V blízkosti stěn využívá časové středování rovnic (nutný je model s výpočtem délkového měřítka), zatímco od definované hodnoty vzdálenosti přepíná na subgridní model [6].

1.3 Výpočtová síť

Prostorová diskretizace spočívá v rozdělení domény pro proudění na výpočtovou síť složenou z elementárních objemů (buněk) často různých tvarů. V rámci této práce bude uvažována pouze třírozměrná síť.

Pro dosažení lepších výsledků a konvergence řešení musí být síť zjemněna především v oblastech vysokých gradientů hodnot polí (např. mezní vrstvy) či zavíření, ideálně s plynulým přechodem hustoty buněk. V závislosti na minimálních rozměrech elementů sítě je pro udržení numerické stability řešení nutné zkrátit také časový krok simulace, což se zvláště v nestacionárních úlohách projeví navýšením délky doby výpočtu [9]. Značný význam pro kvalitu sítě mají také geometrické vlastnosti buněk, nejvíce pak míra jejich šikmosti (skewness) nebo poměr ploch prvků (aspect ratio) [12].



obr.1: Tvary elementů 3D sítě [1]

Pokud jsou v celém objemu použity pouze prvky typu hexahedron (obecný šestistěn), jedná se o *strukturovanou síť*. Ta poskytuje dobrou přesnost a stabilitu řešení i v oblastech vysokých gradientů, nelze ji však libovolně zhušťovat, neboť hranice (stěna) každého prvku musí navazovat na jedinou hranici prvku sousedního [12]. Výsledkem proto může být neúměrně jemná síť mimo podstatné oblasti proudění a tedy přílišné nároky na výpočet.

Naopak *nestrukturované sítě* využívají hexahedrické prvky pouze v okolí stěn, v ostatních oblastech (bez vysokých gradientů) zajistí snadnou změnu hustoty sítě zbývající typy prvků. Nevýhodou však může být vyšší numerická chyba způsobená nekonformním přechodem mezi elementy, případně také horší kontrolou nad šikmostí buněk.

2. PROSTŘEDÍ OPENFOAM

OpenFOAM (OF) je C++ knihovna k tvorbě spustitelných kódů - aplikací. Vznik knihovny se datuje k roku 1989, uveřejněna byla pod názvem OpenFOAM s open source licencí v roce 2004. V současnosti spravuje vydávání nových distribucí s licencí GPL společnost OpenCFD založená autorem Henrym Wellerem.

General Public Licence zajišťuje uživatelům z řad výzkumníků i komerčních společností právo kód svobodně spouštět, prohlížet, kopírovat a upravovat dle svých potřeb. Důsledkem otevřené (proto také bezplatné) licence je také řada výhod oproti obvyklým komerčním distribucím: možnosti jsou otevřeny například masivně paralelním výpočtům (licenční náklady nijak neomezují počet výpočetních stanic), úpravám systému řešení pro nestandardní požadavky a další. Unikátní podmínky a všestrannost OF se odráží v dynamickém růstu odborné komunity uživatelů, která se podílí na rozvoji softwaru i vzájemné podpoře. Profesionální tréninkové programy pak pro firmy nabízí sama společnost OpenCFD.

Flexibilita OF se odráží také v nabízených distribucích, instalaci lze provést na operačních systémech Linux, Mac OS X či Windows.

Pro potřeby následující části této práce byla využita oficiální kompilace OF 3.0.1 v operačním systému Linux Ubuntu, instalace v jiných operačních systémech či kompilace vlastní se budou v jistých ohledech lišit.

2.1 Rozbor prostředí

Distribuce OF obsahují množství předkompilovaných aplikací pro široký okruh úloh. Tyto aplikace se podle funkce rozlišují do dvou kategorií: [1]

- utility
- solvery

Utility zajišťují především podpůrné vstupní a výstupní úlohy (preprocessing, post-processing), jako je například tvorba či úprava výpočetní sítě, zpracování dat výsledků aj.

Každý solver je navržen pro iterační řešení specifického problému mechaniky kontinua. Při svém spuštění vyžaduje již kompletně nadefinovanou úlohu, výpočet provádí bez dalšího zásahu uživatele.

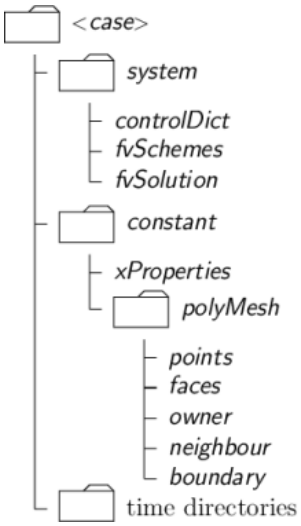
2.1.1 Programový jazyk

Objektově orientované programování v jazyce C++ je svou strukturou vhodné pro přirozenou formulaci postupů a řešení úloh matematicko-fyzikální povahy. Objekty stejného charakteru jsou zapsány pomocí tříd, jež obsahují souhrnné definice jejich vlastností. Kupříkladu objekt *rychlostní pole* je v OF definován svým názvem *U* a třídou *vectorField*, která v sobě slučuje vlastnosti tříd *vector* a *field*.

Třídy mohou být šablonové nebo v sobě zahrnovat dědičnost, což značně redukuje duplicitní tvorbu kódu [1]. V knihovně OF jsou aplikace složeny ze tříd a operací mezi nimi. Kódy aplikací jsou převážně procedurální povahy, jelikož úzce reprezentují algoritmy pro řešení rovnic.

2.1.2 Manipulace s daty

Aplikace jsou přímo v operačním systému spouštěny příkazem v terminálu. Po načtení vstupních dat ze souborů umístěných v adresáři úlohy *caseDir* provedou výpočetní operace a zpracovaná data uloží na patřičné místo v adresáři úlohy. Od uživatele je v této souvislosti vyžadováno dodržení základní struktury vkládání vstupních dat podle následujícího schématu:



obr. 2: Řazení složek v caseDir [1]

Adresář *system* je umístěním pro slovníky definicí numerických schémat *fvSchemes*, algoritmů *fvSolution* řešení a ovládání průběhu výpočtu *controlDict*. Tyto slovníky obecně řídí průběh a stabilitu celého výpočtu, proto jim bude dále zvlášť věnována třetí kapitola této práce.

Slovníky s časově neměnnými vstupy výpočtu jsou dále uloženy v adresáři *constant* (mezi takové patří například definice hodnot potřebných materiálových charakteristik). Umístění *constant/polymesh* pak obsahuje slovníky s definicemi bodů, stěn buněk a okrajových podmínek statické výpočetní sítě (viz 4.2.2).

Aktuální hodnoty polí proměnných jsou během výpočtu zaznamenávány do samostatných souborů uvnitř časových adresářů, jenž jsou v *caseDir* vytvářeny pro zápisový časový krok nastavený ve slovníku *controlDict* (není zde nutná shoda s výpočtovým časovým krokem). Výjimku tvoří počáteční časový adresář *0*, který obsahuje výchozí rozložení hodnot proměnných před spuštěním výpočtu úlohy. K jeho vytvoření lze využít příslušných utilit (kupříkladu *setFields*), často je však výhodnější na stejné síti spustit předběžnou stacionární simulaci s konstantním počátečním rozložením hodnot a její výsledky dále položit jako výchozí stav pro náročnější simulace nestacionární. Při přerušení výpočtu totiž časové adresáře zůstávají zaznamenány v adresáři úlohy, čehož lze jednoduše využít při změně nastavení simulace po určitém počtu iterací.

2.1.3 Grafické rozhraní

Základní distribuce OF nedisponuje žádným grafickým uživatelským rozhraním (GUI) využitelným pro nastavení aplikací a tvorbu sítě. Pro vizualizaci dat je proto distributorem doporučena open-source aplikace ParaView, která je vhodná pro analýzu značného objemu dat z distribuovaných zdrojů [5] a zároveň je pro svou hardwarovou nenáročnost využitelná i pro jednoduché výcvikové úlohy. ParaView nicméně není grafickým rozhraním pro obsluhu OF.

K dispozici jsou také plnohodnotné GUI pro pre-processing (včetně tvorby sítě), nastavení úlohy a spouštění výpočtu: kupříkladu open-source Helyx-OS vyvinutý společností Engys.

2.2 Tvorba výpočtové sítě

Diskretizace je v OF zajištěna metodou konečných objemů, jež zde využívá jedině soustřednou mříž (colocated grid) pro ukládání hodnot veličin ve středech elementárních objemů. Ty pak mohou nabývat různých tvarů či jejich kombinací.

OF podporuje (pouze) třírozměrné sítě strukturovaného i nestrukturovaného typu. Knihovna sama disponuje množstvím utilit pro tvorbu a úpravu sítě, uživatel má dále možnost importovat síť vytvořenou v externím programu (Fluent, Gambit, cfMesh, Salome aj.) pomocí sady k tomu určených utilit.

BlockMesh a snappyHexMesh poskytují základní možnosti tvorby sítě. Není však výjimkou, že potřebám uživatele lépe vyhovují komplexnější nástroje a zvolí proto cestu importu sítě z jiných zdrojů. Základní distribuce OF nicméně obsahuje také jiné užitečné utility, jako jsou foamyHexMesh (nově zavedený, pokročilejší nástroj pro tvorbu dominantně hexahedrální sítě), extrudeMesh (tvorba nové části sítě na síti již existující) a další.

2.2.1 **blockMesh**

Jedná se o v principu jednoduchou utilitu sloužící k vytvoření konformní blokově strukturované sítě. Své využití s výhodou nachází u jednoduchých geometrií a profilů, kde poskytuje dobrou kontrolu nad velikostí elementů sítě a jejich tvarem. Pro tvorbu sítí na geometricky složitých útvarech je však nevýhodná, neboť geometrii definuje vlastnoručními úkony sám uživatel a každé lokální zjemnění sítě výrazně zvyšuje náročnost její definice.

Blokově strukturovaná síť má dvě úrovně: první tvoří větší tvarové celky - bloky, druhá pak sestává z vnitřní strukturované sítě pro každý blok. Podmínkou je, aby na sebe stěny buněk na rozhraní bloků navazovaly a hranice těchto stěn byly tvořeny stejnými body (tím je také zaručena konformní síť i na těchto rozhraních). Strukturované sítě jsou zde tvořeny zvoleným typem buňek typu hexahedron, wedge, prism, pyramid, tetrahedron, či tetrahedral wedge [1].

2.2.2 **snappyHexMesh**

Tato utilita je určena k automatickému generování dominantně hexahedrálních sítí ze zdrojové geometrie ve formátu *triangulated surface geometry*. V první fázi je třeba vytvořit jednoduchou základní síť např. pomocí utility blockMesh tak, aby protínala minimálně jednou stěnou buňky importovanou geometrii. Síť se poté přizpůsobuje požadovanému povrchu a objemu iterativním zjemňováním počáteční sítě a přetvářením výsledné split-hexahedrální sítě.

Zjemnění sítě je možné docílit různými způsoby. Prvním je rozložení geometrie do více ploch (při exportu geometrie je nutné všechny části vyžadující různou úroveň zjemnění uložit zvlášť) a následné předepsání maximální velikosti elementů pro každou z nich. Druhým je určení počtu povrchových vrstev buněk a parametrů pro redukci jejich maximálních rozměrů.

2.2.3 **Převod a úprava sítě**

Absence grafického rozhraní pro tvorbu sítě v OF nemusí být nutně nevýhodou. K dispozici je celá řada pro tento účel vyvinutých komerčních i svobodných aplikací, mezi nimiž si může uživatel volit dle svých osobních preferencí. Podpora tohoto řešení je v knihovně značná, neboť současná distribuce obsahuje 31 utilit pro převod sítí z různých formátů či jejich dodatečné úpravy [2]. Tvorba a následná konverze sítí je tak možná například z editorů pro programy Fluent (*.msh*), CFX (*.geo*), STAR-CD či PROSTAR, GAMBIT (*.neu*) nebo I-DEAS (*.ans*).

Utility pro úpravu sítě jsou zde uvedeny jen úzkým příkladem bez doplňujících parametrů. Podrobný popis všech utilit a funkcí je volně dostupný v [2].

- *refineMesh* - zjemnění sítě rozdělením buněk v celém objemu nebo jeho části
- *renumberMesh* - optimalizace popisu buněk pro paralelní výpočet
- *checkMesh* - zkontroluje validitu sítě
- *polyDualMesh* - převod tetrahedrických sítí na polyhedrické

2.3 Výpočet

Hlavní periferií pro řízení průběhu výpočtu a souvisejících úkonů je slovník *controlDict* detailněji představený níže. Během samotné simulace pak solver poskytuje výpis reziduálů (případně dalších informací v závislosti na zvolených funkcích) pro každý časový krok. Tento výpis lze pro potřeby monitorování průběhu výpočtu zpracovat pomocí knihovny pyFoam nebo běžnými linuxovými prostředky, což zároveň usnadňuje nabídka nástrojů OF (*foamJob, foamLog*).

2.3.1 **controlDict**

Solvery začínají výpočet nastavením databáze, která dále v průběhu simulace řídí načítání vstupů a zápis výstupů. Všechny vstupní parametry potřebné k vytvoření této databáze pak v sobě zahrnuje slovník *controlDict*.

Z hlediska časového řízení simulace je nutné nastavit počátek výpočtu (libovolný uložený krok) a naopak jeho ukončení, délku časového kroku a interval zápisu výsledných hodnot. Samotná nastavení těchto parametrů mohou vycházet ze simulovaného času, počtu kroků, reálného času výpočtu či dalších a poskytují tak značnou variabilitu pro různé potřeby úlohy. Užitečnou možností je také volba automatické úpravy časového kroku během výpočtu.

Volba parametrů zápisu poskytuje kontrolu nad formátem, přesností či kompresí ukládaných výsledků; dále jsou podporovány výstupy pro monitoring průběhu konvergence řešení a velikostí reziduálů pomocí (linuxových) aplikací *gnuplot, Xmgr* nebo *JPlot*. Z důvodu úspory paměti úložiště potřebné pro ukládání výsledků rozsáhlých simulací je k dispozici funkce definující limitní počet časových složek (kroků zápisu), při jehož překročení jsou nejstarší výsledky cyklicky přepisovány adresáři aktuálními.

ControlDict dále poskytuje volbu, zda mají být slovníky načítány pouze před zahájením výpočtu či na začátku každého výpočetního kroku. Opětovné načítání zde poskytuje možnost modifikace všech obsažených parametrů v průběhu výpočtu. Zvláštní kategorií slovníku jsou pak uživatelem zaváděné funkce, jenž v průběhu simulace generují vlastní výstupy. Pro jejich připojení k výpočtu je zde nutné vypsat seznam načítaných přídavných knihoven a posléze i samotných funkcí s definicí potřebných parametrů.

2.3.2 **Solvery**

OpenFOAM disponuje značným množstvím prostředků pro řešení úloh v rámci nejrůznějších odvětví. Své využití tak nachází v hydraulice, aerodynamice, při simulacích chemických reakcí, spalovacích cyklů motorů, přenosu tepla, rozprašování či analýze pohybu částic různých fází v proudění, potlačování požárů v budovách a mnoha dalších technických aplikacích [1]. Vzhledem k zaměření práce budou dále uvedeny jen popisy některých solverů pro řešení jedno- a vícefázového proudění kapalin a plynů, získané ze zdrojů [1,2,4].

Jednofázové proudění nestlačitelné kapaliny

- *boundaryFoam* – stacionární 1D turbulentní proudění; obvykle slouží k vytvoření okrajových podmínek pro inlet
- *simpleFoam* – stacionární turbulentní/laminární proudění Newtonské či ne-Newtonské kapaliny, algoritmus SIMPLE
- *icoFoam* – nestacionární laminární proudění Newtonské kapaliny, algoritmus PISO
- *nonNewtonianIcoFoam* – *icoFoam* pro ne-Newtonskou kapalinu
- *icoDyMFoam* – *icoFoam* s podporou dynamické sítě
- *pisoFoam* – nestacionární turbulentní proudění, algoritmus PISO
- *pimpleFoam* – nestacionární turbulentní proudění s dlouhým časovým krokem, algoritmus PIMPLE (sloučení PISO+SIMPLE, viz 3.3.3)
- *pimpleDyMFoam* - *pimpleFoam* s podporou dynamické sítě

Jednofázové proudění stlačitelné tekutiny

- *rhoSimpleFoam* – *simpleFoam* pro stlačitelnou kapalinu *s* podporou přenosu tepla
- *rhoPimpleFoam* – *pimpleFoam* pro stlačitelnou kapalinu *s* podporou přenosu tepla
- *sonicFoam* – nestacionární trans- nebo supersonické laminární/turbulentní proudění plynů, algoritmus PIMPLE
- *sonicLiquidFoam* – modifikace *sonicFoam* pro supersonické laminární/turbulentní proudění kapaliny
- *sonicDyMFoam* – *sonicFoam* s podporou dynamické sítě

Proudění s kavitací

- *cavitatingFoam* – nestac. řešení společného proudění kapaliny a její plynné fáze založené na homogenním modelu stlačitelnosti jejich směsi.
- *cavitating DyMFoam* – *cavitatingFoam* s podporou dynamické sítě
- *interPhaseChangeFoam* – nestacionární izotermické turbulentní proudění dvou nestlačitelných, nemísitelných kapalin s fázovou přeměnou. Rozhraní fází je určeno metodou VOF (Volume Of Fluid), jenž sleduje podíl jednotlivých fází v konečném objemu a následně sestaví odpovídající hodnoty veličin pro dosazení do hybnostní rovnice. Sada modelů je primárně určena pro kavitaci, solver však podporuje i jiné mechanismy fázové přeměny.
- *InterPhaseChangeDyMFoam* – *interPhaseChangeFoam* s podporou dynamické sítě

Proudění s volnou hladinou

- *interFoam* – nestacionární izotermické proudění nemísitelných kapalin, metoda VOF
- *compressibleMultiphaseInterFoam* – *interFoam* pro *n* stlačitelných kapalin s podporou přenosu tepla
- *compressibleInterDyMFoam* – *interFoam* pro 2 stlačitelné kapaliny s podporou přenosu tepla a dynam. sítě (včetně změny topologie a adaptivního znovunačtení sítě)

Proudění mísitelných tekutin

- *interMixingFoam* – nestacionární proudění 3 nestlačitelných kapalin (z nichž 2 jsou mísitelné), metoda VOF
- *twoLiquidMixingFoam* – nestacionární proudění 2 mísitelných nestlačitelných kapalin
- *twoPhaseEulerFoam* – nestacionární proudění 2 fází stlačitelné kapaliny s přenosem tepla, z nichž jedna fáze je rozptýlená (např. v podobě bublin)

Rozprašování, sledování částic

- *sprayFoam* – nestacionární solver pro stlačitelné, laminární/turbulentní proudění s rozstřikem, algoritmus PIMPLE
- *reactingParcelFilmFoam* – nestacionární, laminární/turbulentní proudění stlačitelné kapaliny s lagrangeovskými částicemi a modelováním vzniklého povrchového filmu

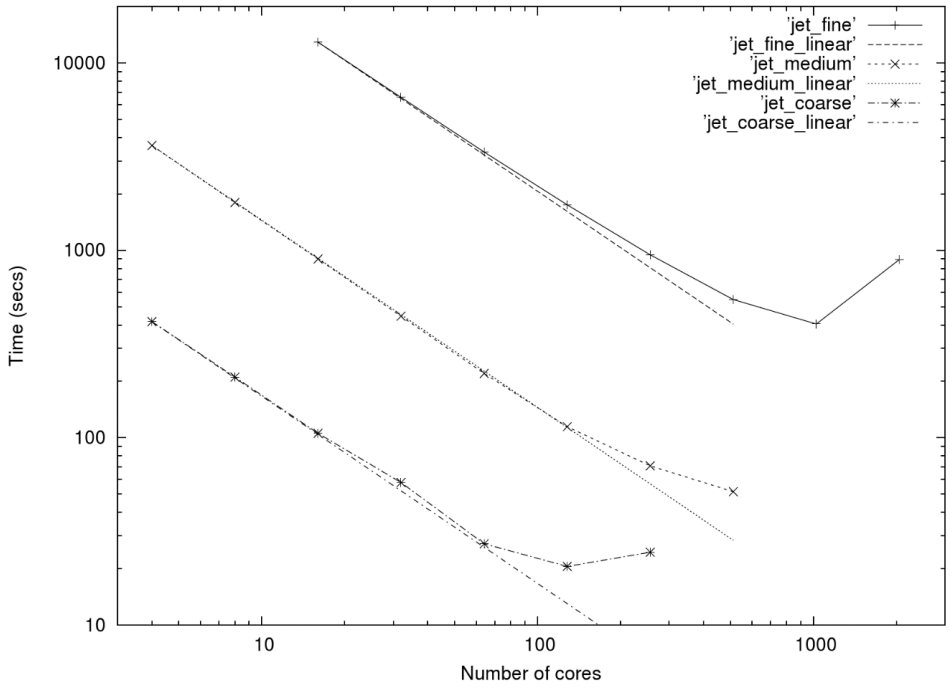
2.3.3 **Turbulence**

V knihovně OF může být každý z přístupů uvedených v (1.2.4) použit v téměř libovolném solveru zahrnujícím modelování turbulencí. Pro vysoké hodnoty Reynoldsova čísla se mezní vrstva stává tak tenkou, že je velmi obtížné zachytit sítí jevy s ní spojené [9]. Proto jsou využívány stěnové funkce, které chování této turbulentní mezní vrstvy modelují. Jejich definice se nachází v kategorii *boundaryField* polí turbulentních proměnných, stejně jako další specifické turbulentní okrajové podmínky (například intenzita turbulentní kinetické energie na inletu).

2.3.4 **Paralelizace výpočtu**

OpenFOAM byl již od začátku vyvíjen s důrazem na efektivní paralelizaci výpočtů. Tato schopnost je integrována do samotného jádra systému a nové (či uživatelské) aplikace proto nemusí obsahovat nijak specifický kód [2]. Ve výsledku je díky tomu možné spouštět paralelně téměř každou aplikaci ve všech krocích řešení úlohy, jako je tvorba sítě, preprocessing, samotná simulace i postprocessing.

Rozdělení sítě a polí na požadovaný počet úseků provádí utilita *decomposePar*. Každý úsek připadá na jedno výpočetní jádro (vlákno) procesoru, maximální míra paralelizace tedy záleží především na vlastní konfiguraci stanice. Aplikace jsou spouštěny zvlášť pro každý úsek (jádro), přičemž hodnoty na společných hranicích jsou mezi procesory předávány komunikačním protokolem MPI (Message Passing Interface) [2]. Distributor udává, že OpenFOAM dokáže v komplexních úlohách efektivně zhodnotit přinejmenším 1000 výpočetních jader [2].



obr. 3: Efektivita paralelních výpočtů[2]

3. ŘÍZENÍ STABILITY VÝPOČTU

3.1 Podobnostní čísla

Podobnost proudění může být geometrická (poměr rozměrů, úhly, drsnost), kinematická (poměr rychlostí a zrychlení) nebo dynamická (poměr působících sil). Pro její vyjádření jsou využívána podobnostní čísla.

Reynoldsovo číslo

$$Re = \frac{U.L}{\nu} \approx \frac{setrvačné\ síly}{odporové\ síly}$$

je bezrozměrné číslo, které dává do souvislosti charakter proudění se vzájemným poměrem setrvačných a odporových sil. *U* ve vzorci je rychlost proudění, *L* charakteristický rozměr a *ν* kinematická viskozita. Užívá se mimo jiné pro vyjádření hranice mezi laminárním a turbulentním prouděním.

Courantovo číslo

$$Co = \Delta t \cdot \sum_{i=1}^{n} \frac{v_{x_i}}{\Delta x_i} \leq Co_{max}$$

nebo také Courant-Friedrichs-Lewyova podmínka vymezuje míru stability numerického řešení nestacionárního proudění. Maximální hodnota bezrozměrného Co_{max} je rozdílná pro různé metody řešení rovnic: pro explicitní solvery je $Co_{max} = 1$, implicitní solvery obvykle tolerují hodnotu vyšší [18].

Pécletovo číslo

$$Pe = \frac{U \cdot L}{D} \approx \frac{konvektivní\ rychlost}{difusní\ rychlost}$$

je opět bezrozměrné číslo, kde *L* je charakteristický rozměr, *U* lokální rychlost proudění a *D* koeficient difuse. Peckletovo číslo také může být vyjádřeno jako součin lokálního Reynoldsova Schmidtova čísla [19].

Strouhalovo číslo

$$St = \frac{f \cdot L}{U}$$

udává frekvenci periodicky vznikajících vírů za obtékaným tělesem o charakteristickém rozměru *L* při rychlosti proudění *U* [17].

3.2 Numerická schémata

Numerická schémata jsou využívány metodou konečných objemů pro diskretizaci členů rovnic matematického modelu a interpolaci hodnot mezi výpočtovými uzly. Všechny definice potřebných vstupů jsou soustředěny ve slovníku *fvSchemes* umístěném v adresáři *system* a musí odpovídat požadavkům formulace matematického modelu solveru.

Schémata jsou ve slovníku rozloženy do kategorií neboli podslovníků, které odpovídají členům matematického modelu. Každý z těchto podslovníků pak poskytuje možnost definice schématu dané operace pro všechna (*default*) nebo pro jednotlivá pole modelu zvlášť.

3.2.1 **ddtSchemes**

Tato schémata určují diskretizaci časově závislých členů. Mohou podstatně ovlivňovat stabilitu výpočtu při nestacionárním proudění, pro stacionární úlohy se však volí nejčastěji vstup *steadyState*, který časově závislé členy v modelu neuvažuje. Výchozím schématem pro první časové derivace je stabilní Eulerovo schéma prvního řádu přesnosti, které může být (obvykle až v průběhu výpočtu) nahrazeno specifičtějším.

schéma	*vlastnosti*
Euler	první řád, omezené, implicitní/explicitní
CrankNicolson	druhý řád, omezené, využívá předchozí pole i čas. derivaci, implicitní
backward	druhý řád, využívá dvě předchozí hodnoty polí, implicitní
CoEuler	první řád, ustálené proudění, krok dle max. Co, implicitní/explicitní
localEuler	první řád, ustálené proudění, lokální časový krok, implicitní/explicitní
steadyState	nezahrnuje časové změny

tab. 2: Přehled vybraných časových schémat [1,4]

3.2.2 **interpolationSchemes**

Interpolace hodnot ze středů buněk na jejich stěny je (jako jeden z důsledků využití strategie *colocated grid*) v OF častou operací: v matematickém modelu představuje nezbytný nástroj pro vyjádření hodnot rychlostí na stěnách buněk a dále je v rámci třech členů Navier-Stokesovy rovnice využívána ve spojení s Gaussovou diskretizací. V tomto smyslu může vést výběr interpolačních schémat vyšších řádů k lepším výsledkům, zároveň však s sebou přináší vyšší nároky na výpočtový čas a případně horší stabilitu řešení. K dispozici jsou nicméně také limitované verze některých schémat, které pak mají stabilnější charakter.

schéma	*vlastnosti*
centrální schémata	
linear	druhý řád, lineární interpolace (centrální diference), stabilní pro Pe≤2
cubicCorrection	třetí řád, kubické schéma
midPoint	druhý řád, lineární interpolace symetricky vyvážená
protiproudá schémata (Upwind)	
upwind	první řád, stabilní, nepřesné (vhodné spíše pro začátek výpočtu)
linearUpwind	druhý řád
skewLinear	druhý řád, korekce šikmosti
filteredLinear2	druhý řád, filtrace vysokofrekvenčních oscilací
limitovaná schémata (Total Variation Diminishing)	
limitedLinear	druhý řád, limiter
vanLeer	vyhlazený van Leer limiter
MUSCL	druhý řád, MUSCL limiter
limitedCubic	třetí řád, limiter
normalizovaná schémata (Normalized Variable Diagram)	
SFCD	druhý řád, filtrované

tab. 3. Ukázka některých interpolačních schémat [1,4]

Každá kategorie schémat má své charakteristické vlastnosti [8,9]. Centrální schémata mohou být numericky méně stabilní, zatímco protiproudá mohou proudění uměle vyhlazovat. Limitovaná TVD schémata jsou pak vhodná pro stabilizaci proudění s podstatnými tlakovými změnami.

3.2.3 **gradSchemes**

Tento podslovník poskytuje volbu schémat pro výpočet gradientů vyskytujících se v modelu daného solveru. Volba schémat nejvíce ovlivňuje výsledky simulace v místech vysokých gradientů polí proměnných nebo také při použití tetrahedrální topologie sítě [9]. Volit lze mezi následujícími schématy:

schéma	*vlastnosti*
Gauss<*interpol. schéma*>	druhý řád, využívá interpolace na stěny a Gaussovu větu
leastSquares	druhý řád, metoda nejmenších čtverců
fourth	čtvrtý řád, metoda nejmenších čtverců
cellLimited<*grad. schéma*>	limituje extrapolované hodnoty na stěnách mezi maximum a minimum určené ze středů dané buňky a buněk sousedících, limit aplikuje na všechny směry gradientu
faceLimited<*grad. schéma*>	limituje extrapolované hodnoty na stěně mezi hodnoty ve středech touto stěnou sousedících buněk, limit aplikuje na všechny směry gradientu

tab. 4. Přehled vybraných gradientních schémat [1,4]

3.2.4 **snGradSchemes**

Surface normal gradient je komponenta gradientu hodnot ležících ve středech dvou buněk, která je normálová ke stěně, jež tvoří jejich rozmezí. Její význam v matematickém modelu bude dále popsán v 3.2.*6 (laplasianSchemes).*

schéma	*popis*
corrected	druhý řád, explicitní korekce neortogonality
uncorrected	druhý řád, bez korekce neortogonality
limitedΨ	druhý řád, limitovaná korekce neortogonality
fourth	čvrtý řád

tab. 5. Vybraná schémata snGradSchemes [1,4]

3.2.5 **divSchemes**

Základem výpočtu divergencí je Gaussova věta a získané hodnoty gradientů na stěnách buněk. Gaussovo diskretizační schéma je zde nutným parametrem, volí se pouze interpolační schéma pro závislé pole rychlostí *U*.

schéma	*charakter*
linear	druhý řád, neomezené
skewLinear	druhý řád, neomezené, korekce šikmosti
cubicCorrected	čtvrtý řád, neomezené
linearUpwind	první/druhý řád, omezené
QUICK	první/druhý řád, omezené

tab. 6: Numerický charakter vybraných interpolačních schémat v divSchemes [1,4]

3.2.6 **laplacianSchemes**

Pro člen zahrnující působení třecích sil v Navier-Stokesově rovnici je v matematickém modelu knihovny OF ekvivalentní následující zápis:

$$\nabla \cdot (\nu \nabla U) \equiv laplacian(nu, U)$$

Stejně jako v *3.2.3* je jedinou možností pro diskretizaci Gaussovo schéma, které dále vyžaduje volbu interpolačního schématu (pro určení kinematické viskozity *ν* na stěnách buněk) a schématu pro výpočet gradientů ve směru normály stěn (pro gradient rychlosti *<nabla>U*). Interpolační schéma je typicky doporučeno *linear*[1], výběr *snGradScheme* se pak stává hlavním činitelem výsledného numerického chování členu.

schéma	*charakter*
corrected	druhý řád, neomezené, konzervativní
uncorrected	první řád, omezené, nekonzervativní
limitedΨ	smíšení <corrected> a <uncorrected>
fourth	čtvrtý řád, neomezené, konzervativní

tab. 7: Numerický charakter vybraných interpolačních schémat v laplacianSchemes [1,4]

3.3 Řízení a algoritmy výpočtu

3.3.1 Řešení lineárních soustav

Podslovník *solvers* umístěný uvnitř *fvSolution* určuje metodu řešení systému lineárních rovnic. Tyto metody rozlišují symetrické a nesymetrické matice, přičemž symetrie matic záleží na struktuře rovnic. Volba metody je tedy omezena tvarem matic v rovnicích, OF však dokáže sám rozpoznat, zda nebyla zvolena metoda nesprávná a případně vypíše popis chyby (výpočet se tedy nespustí) [1].

Metoda	Parametr
Preconditioned (bi-)conjugate gradient	PCG/PBiCG†
Solver using a smoother	smoothSolver
Generalised geometric-algebraic multigrid	GAMG
Diagonal solver for explicit systems	diagonal

†PCG pro symetrické, PBiCG pro nesymetrické matice

tab. 8: Přehled doporučených metod řešení lineárních rovnic [1]

Řešení soustav rovnic je prováděno iterativně a je založené na redukování jejich reziduálů. Reziduál je určen vyhodnocením rozdílu levé a pravé strany pro aktuální iteraci a je tedy měřítkem chyby řešení. Aby bylo dosaženo nezávislosti na měřítku řešeného problému, reziduály jsou normalizovány [1]. Solver zastaví výpočet, pokud je splněna alespoň jedna z následujících podmínek:

- všechny reziduály jsou menší, než definovaná tolerance řešení *tolerance*
- poměr všech velikostí aktuálního ku reziduálu z předchozí iterace se sníží pod definované relativní zpřesnění *relTol*
- počet iterací dosáhne svého maximálního počtu *maxIter* (volitelný parametr)

Tolerance by měla reprezentovat úroveň, na které jsou reziduály natolik malé, aby bylo možné pokládat řešení za dostatečně přesné. V nestacionárních úlohách se dále obvykle volí *relTol 0*, aby řešení konvergovalo až k *tolerance* pro každý časový krok.

3.3.2 **Relaxace**

Relaxace (nebo také podrelaxace) je technika využívaná pro zlepšení stability výpočtu, zvláště při řešení ustálených úloh. V principu jde o omezení změn hodnot proměnné z jedné iterace na druhou, čehož je dosaženo buď modifikací matice výsledků a zdroje určujícího řešení pro dané pole, nebo změnou tohoto výsledného pole samotného [1].

Faktor podrelaxace α určuje sílu podrelaxace a platí pro něj následující tvrzení [1]:

- $0 < \alpha \leq 1$
- pokud α není definováno, není použita relaxace
- pro $\alpha = 1$ je předpokládána diagonální dominance matic (tedy vysoká stabilita řešení i bez relaxace)
- pro nižší α se zvyšuje účinek relaxace
- pro $\alpha = 0$ se řešení s dalšími iteracemi nemění (je zamezeno změně hodnot)

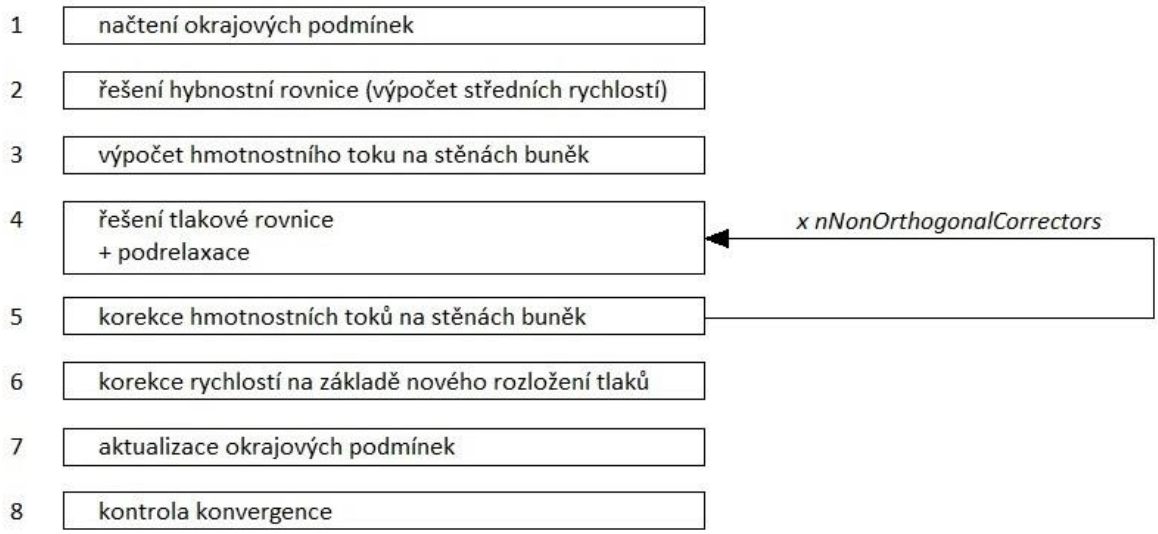
Optimální volbou je tedy α dostatečně nízké pro zajištění stability výpočtu, zároveň však dostatečně vysoké pro rychlý postup iteračního procesu. Uživatel může určit α jako výchozí pro všechna pole proměnných, nebo pro každé partikulární pole (např. U) zvlášť.

3.3.3 **Algoritmy PISO, SIMPLE a PIMPLE**

Řešení Navier-Stokesových rovnic nevede přímo k výsledku, jelikož neposkytují explicitní vyjádření tlaku. Rovnice pro tlak proto musí být odvozena zvlášť a to vyjádřením členu divergence rychlosti z Navier-Stokesovy rovnice a jeho dosazením do rovnice kontinuity [2]. Samotný systém řešení pak řídí zvolený algoritmus, jenž musí být spolu s jeho parametry definován ve slovníku *fvSolution*.

SIMPLE

Semi-Implicit Method for Pressure-Linked Equations je algoritmus poskytující iterativní proceduru výpočtu tlaku s využitím podrelaxace ve stacionárních úlohách. Navier-Stokesova hybnostní rovnice i odvozená rovnice pro tlak jsou řešeny implicitně, korekce rychlostí pak explicitně (odtud také vychází samotný název).

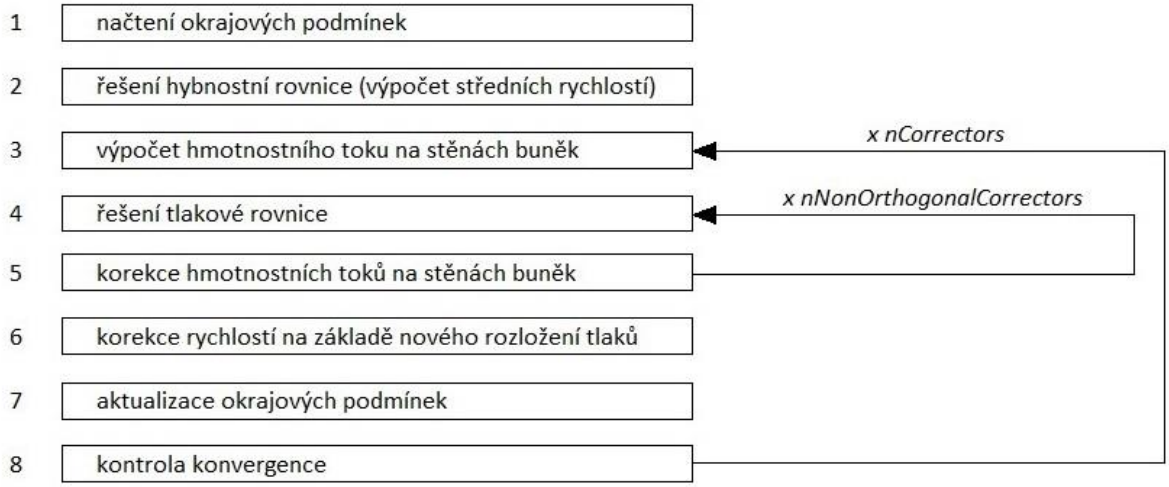


obr. 4: Základní schéma algoritmu SIMPLE [2]

Ve slovníku *fvSolution* musí být definován také parametr *nNonOrthogonalCorrectors*, jenž udává počet požadovaných smyček pro řešení tlaku s podrelaxací a korekci toků na stěnách. Parametry podrelaxace jsou pak určeny ve vlastní kategorii téhož slovníku.

PISO

Pressure-Implicit with Splitting Operators je obdobná metoda pro řešení hybnostních rovnic v nestacionárních úlohách. Oproti algoritmu SIMPLE zde však není využita podrelaxace a pro jeden časový krok je vícekrát provedena korekce rychlostí. Tento postup dosahuje nižší stability řešení a vyžaduje proto výrazně kratší časový krok, neboť musí splňovat Courantovu podmínku [2].

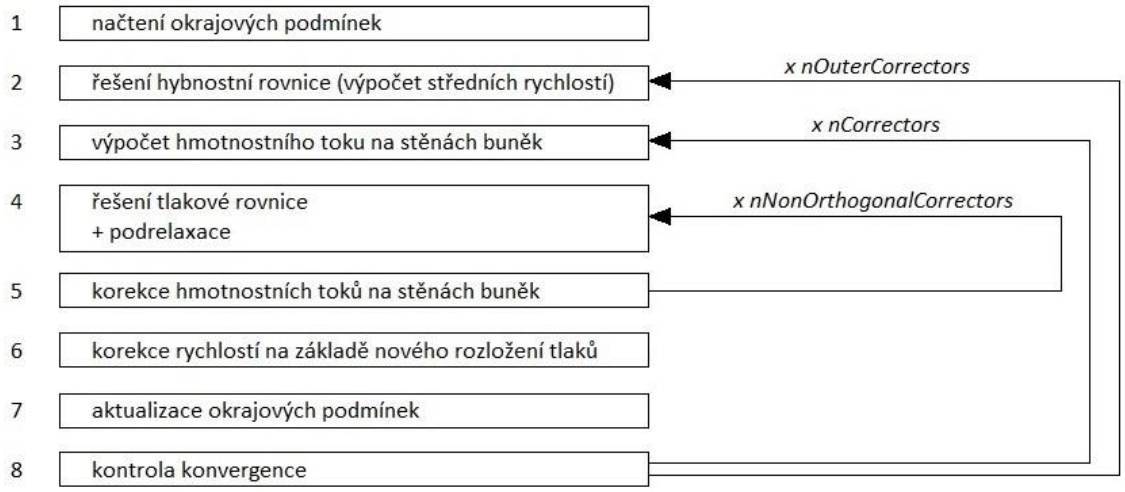


obr. 5: Základní schéma algoritmu PISO [2]

Smyčky pro výpočet tlaku (*nNonOrhogonalCorrectors*) jsou oproti předchozí metodě v PISO řešeny bez využití podrelaxace. Dále je nutné zvolit požadovaný počet PISO smyček pomocí parametru *nCorrectors*.

PIMPLE

Sloučení algoritmů SIMPLE a PISO umožňuje mezi časovými kroky dosáhnout poměrně rychlé konvergence využitím SIMPLE smyčky a podrelaxace. To poskytuje lepší stabilitu řešení i při delších časových krocích (a tedy vyšším Courantově čísle), což sebou přináší úsporu výpočtového času v rozsáhlých či silně nestacionárních úlohách [2]. Zároveň je ale nutné dobře zvážit fyzikální podstatu dějů, neboť v rámci příliš dlouhého časového kroku nemusí být výpočtem nutně pokryty některé velmi rychlé změny (např. rychlé pulzace tlaků) [2].



Obr. 6: Základní schéma algoritmu PIMPLE [2]

Definované parametry jsou pak následující [2]:

- *nNonOrthogonalCorrectors* spolu s parametry podrelaxace (viz SIMPLE)
- *nCorrectors* určuje počet PISO smyček, obvykle se volí nižší (1→3)
- *nOuterCorrectors* udává počet cyklů pro řešení hybnostních rovnic a dále tlaku (smyčka PIMPLE) - zde se volí vysoký počet cyklů (50→200)
- *residualControl* umožňuje zkrátit výpočet přerušením smyčky PIMPLE při dosažení požadované konvergence

4. KÁRMÁNOVA VÍROVÁ STEZKA

Kármánova vírová stezka je periodicky kmitající vírová struktura, jež za určitých podmínek vzniká při obtékání objektů neaerodynamického tvaru (tzv. bluff bodies). Tyto objekty se vyznačují tím, že jejich odporová síla způsobená tlakovými rozdíly při proudění je větší, než síla vyvozená účinky viskozity.

4.1 Úvod do problému

4.1.1 Popis úlohy

Jako základ úlohy byla vybrána studie proudění okolo válcového profilu od autorů Masami Sato a Takaya Kobayashi [6]. Jejich simulace v komerčním software Abaqus/CFD se opírají o experimentální vizualizaci proudění i vyhodnocení silových účinků na objekt, práce tedy poskytuje dobré možnosti srovnání s výpočtem vlastním.

Modelováno je proudění vody okolo válcového tělesa při následujících parametrech:

teplota celého systému	T=273K
hustota vody	ς=998,204kg·m^{-3}
kinematická viskozita vody	ν=1,004·10^{-6}m^{2}·s^{-1}
průměr válce	d=8mm

Rychlost proudění bude vypočítána pro Reynoldsova čísla o hodnotách 26, 50 a 195 vyjádřením ze vztahu uvedeného ve třetí kapitole:

$$U = \frac{\mu \cdot Re}{\varsigma \cdot D}$$

Při těchto podmínkách ještě lze uvažovat laminární charakter proudění, ten se podle autorů postupně mění v turbulentní až pro Reynoldsova čísla okolo 200. Zároveň je možné modelovat vodu jako nestlačitelnou kapalinu bez výrazné chyby výsledků, neboť absolutní tlakové změny způsobené prouděním jsou velmi malé.

Účelem simulací bude získat vizualizaci tvaru proudění, rozložení rychlostí a tlaků, velikost sil působících na objekt a při vzniku vírové stezky pak frekvenci kmitů.

4.1.2 Volba solverů

Při velmi nízkých nátokových rychlostech lze uvažovat časový průběh proudění stacionární, neboť v makroskopickém měřítku nebudou vznikat žádné nestabilní struktury a separované proudy se v určité vzdálenosti za objektem znovu spojí. Naopak při vyšších rychlostech lze za obtékaným tělesem předpokládat také rostoucí míru zavíření či nestabilit - například z důvodu odtrhávání proudů na zadní části jeho povrchu a lokálních poklesů tlaků.

Jako stacionární solver pro *Re* 26 a 50 je jedinou odpovídající volbou *simpleFoam*. Nabídka nestacionárních solverů je širší, neboť s turbulentním modelem *laminar* by bylo možné využít i solvery navržené primárně pro turbulentní proudění; nejvíce přímou cestu k řešení však představuje *icoFoam*, který bude použit pro *Re* 50 (ověření ustáleného proudění) a 195.

4.2 Pre-processing

4.2.1 Tvorba a import sítě

Obtékání štíhlého tělesa bude zobrazeno v rovině řezu kolmé na jeho délku. Pro takovou simulaci je vhodné z důvodu úspory počtu buněk a tedy výpočetního času využít 2D geometrii, matematický model OF však vyžaduje třetí rozměr. V tomto směru bude šířka buněk nenulová, bude však použita jen jediná jejich vrstva a dvojrozměrné chování bude zaručeno definicí příslušných okrajových podmínek.

Pro rozměry výpočtové domény okolo obtékaného profilu je doporučena (Zdravkovich, 1997) minimální délka ve směru nátoku $5 \cdot d$, ve směru odtoku pak $15 \cdot d$a ve směru kolmém $10 \cdot d$. Zaručena je tak přijatelná míra ovlivnění výsledků okrajovými podmínkami definovanými na vnějších hranicích domény.

Geometrie i samotná síť byly vytvořeny v programu Ansys 16.2 a exportovány jako *.msh* soubor definující topologii sítě. Měřítko vytvořené geometrie neodpovídalo požadovaným rozměrům z důvodu prezentace rozšířených možností importu sítě.

Po vložení souboru sítě do projektové složky (*caseDir*) je možné zahájit samotný import. Pro geometrii formátu *.msh* k tomu slouží utilita *fluentMeshToFoam*, která musí být spuštěna právě v terminálu s načteným umístěním *caseDir*. Změny měřítka je pak možné docílit například pomocí příkazu *transformPoints -scale '(x x x)'*, kdy utilita *transformPoints* vynásobí rozměry (již importované) sítě v jednotlivých směrech zvolenými parametry (*x,y,z*).

Dále je vhodné stejným způsobem spustit utilitu *checkMesh*, jež kontroluje topologii sítě (např. proti překrývání buněk) a poskytuje další informace o síti, jako jsou její vnější rozměry, počet buněk, statistiky velikosti či šikmosti buněk a další (viz příloha 1).

Z výpisu je zřejmé, že rozměry sítě dosáhly po úpravě měřítka požadovaných hodnot. Importovaná síť má 155 524 buněk, což je pravděpodobně z hlediska požadované přesnosti postačující. Než bude možné otevřít a vizuálně zkontrolovat síť v aplikaci *Paraview*, je nutné přejmenovat časovou složku *0* (např. na *0.org*), neboť prozatímní definice (především názvy a stěny buněk) okrajových ploch domény v souborech převzatých z tutoriální úlohy se neshodují s definicemi právě importované sítě ve slovníku */constant/polyMesh/boundaries* a způsobily by pád aplikace.

4.2.2 Okrajové a počáteční podmínky

Okrajové plochy byly včetně názvů, typu a přiřazení stěn příslušných buněk definovány již při tvorbě geometrie a sítě. Slovník boundaries je tedy spolu s ostatními definičními slovníky sítě generován již při jejím importu, poskytuje však možnost jednotlivé definice dodatečně upravit.

Okrajové podmínky je však nutné určit i pro všechna pole proměnných v časové složce *0*. Názvy definovaných ploch se musí shodovat s názvy ve slovníku boundaries, neboť těmto plochám dále přiřazují předepsané vlastnosti jednotlivých polí. Každé z těchto ploch jsou dále předepsány požadované vlastnosti, viz ukázka pro pole rychlostí *U* (obdobně pak pro pole tlaků *p*).

```
dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform (0.00627 0 0);
    }

    outlet
    {
        type            zeroGradient;
    }

    symmetryplane1
    {
        type            symmetryPlane;
    }

    wall
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }

    symmetryplane2
    {
        type            symmetryPlane;
    }

    frontAndBackPlanes
    {
        type            empty;
    }
}
```

```
6
(
    inlet
    {
        type            patch;
        nFaces          223;
        startFace       310105;
    }
    outlet
    {
        type            patch;
        nFaces          254;
        startFace       310328;
    }
    symmetryplane1
    {
        type            symmetryPlane;
        inGroups        1(symmetryPlane);
        nFaces          599;
        startFace       310582;
    }
    wall
    {
        type            wall;
        inGroups        1(wall);
        nFaces          211;
        startFace       311181;
    }
    symmetryplane2
    {
        type            symmetryPlane;
        inGroups        1(symmetryPlane);
        nFaces          599;
        startFace       311392;
    }
    frontAndBackPlanes
    {
        type            empty;
        inGroups        1(empty);
        nFaces          311048;
        startFace       311991;
    }
)
```

obr. 7: Okrajové a počáteční podmínky pole U (vlevo), slovník boundaries (vpravo)

Definice ve slovníku mají následující význam:

dimensions	udává fyzikální rozměr pole v jednotkách SI, tedy $m \cdot s^{-1}$
internalField uniform	definuje počáteční hodnotu rychlostí v celém objemu
fixedValue uniform	definuje stálou rychlost na vstupní ploše proudu (inlet)
zeroGradient	určuje nulové konvektivní zrychlení na výstupu (konstantní rychlost)
symmetryPlane	nulová rychlost ve směru normály, v ostatních směrech volný pokluz
empty	zjednodušuje modelované proudění na 2D (vynechá výpočet rychlostí v normálovém směru)

Během celé úlohy je nutné mít na paměti, že veličina „*p*“ je v rovnicích OF definována jako tlak podělený hustotou. Tomu je nutné podřídit nejen velikost zadávaných hodnot, ale také rozměry v soustavě SI a především vyhodnocení výsledků simulace.
Dále jsou nastaveny fyzikální vlastnosti (včetně jejich rozměrů v soustavě SI) kapaliny ve slovníku*/constant/transportProperties*, zde pouze kinematická viskozita ν.

Jelikož je *simpleFoam* solver pro řešení turbulentního proudění, je nutné zvolit také model turbulence v */constant/turbulentProperties*. Pro laminární proudění je zvolen typ modelu *laminar*.

4.2.3 Volba numerických schémat a algoritmů řešení

Volbu numerických schémat poskytuje slovník */system/fvSchemes*. Protože se simulaci snažíme přiblížit pouze přibližným experimentálním výsledkům, nebude nutné v průběhu výpočtu tuto volbu nijak měnit a snažit se o přesnější napodobení fyzikální reality.

```
ddtSchemes
{
    default         steadyState;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default         none;
    div(phi,U)      bounded Gauss linearUpwind grad(U);
    div(phi,nuTilda) bounded Gauss linearUpwind grad(nuTilda);
    div((nuEff*dev2(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

wallDist
{
    method meshWave;
}
```

```
ddtSchemes
{
    default         Euler;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default         none;
    div(phi,U)      Gauss limitedLinearV 1;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}
```

obr. 8: Použité nastavení slovníků pro stacionární (vlevo) a nestacionární (vpravo) výpočet

ddtSchemes
Jedinou volbou pro stacionární řešení (solver *simpleFoam*) je parametr *steadyState*, viz (3.2). Pro stále ještě velmi nízká Reynoldsova čísla nestacionárního proudění nebudou časové změny rychlostí vysoké a při dostatečně jemném časovém kroku proto stačí Eulerovo schéma prvního řádu.

gradSchemes
Gaussovo diskretizační schéma druhého řádu pro výpočet gradientních členů vyžaduje zároveň volbu příslušného schématu interpolačního. Vzhledem k neaerodynamickému (či nehydrodynamickému) tvaru obtékaného objektu lze při rostoucích rychlostech konvekce předpokládat výskyt vyšších tlakových gradientů.

divSchemes
Pro výpočet divergencí je Gaussovo diskretizační schéma jedinou volbou. Podobně jako u předchozí kategorie vyžaduje volbu interpolačního schématu, který se v tomto případě mezi oběma solvery liší.
Pro stacionární řešení bylo zvoleno omezené (*bounded*) protiproudé schéma druhého řádu, které pro své omezení nemusí být konzervativní, zato však (především na počátku simulace) poskytuje vyšší stabilitu řešení. Parametr *grad(U)* specifikuje tokové pole, na kterém je schéma založené.
V solveru *icoFoam* pak bylo použito TVD schéma *limitedLinearV* s limiterem formulovaným pro zahrnutí směru vektoru. Číselný parametr za vektorem nastavuje úroveň limiteru, kdy pro volbu *0* dosahuje schéma nejvyšší přesnosti a pro volbu *1* nejlepší konvergence.

laplacianSchemes
Skladba parametrů je *Gauss <interpolační schéma> <snGrad schéma>*. Interpolační schéma je standardně voleno *linear*, volba *corrected* pak poskytuje neomezené, konzervativní chování tohoto členu zahrnujícího vliv třecích sil.

interpolationSchemes
Také pro interpolaci hodnot polí ze středů buněk na jejich stěny je využito schéma druhého řádu *linear*. Schéma neupřednostňuje žádný směr konvekce, stabilní je však pouze pro jemnou síť (Pécletovo číslo $Pe \leq 2$).

snGradSchemes
SnGrad je složka gradientu v normálovém směru stěny buňky. Pro neortogonální elementy sítě však často neleží přímo na spojnici středů buněk, z nichž je gradient počítán, což způsobuje chybu řešení tohoto členu. Korekci neortogonality poskytuje například zde použité schéma *corrected*.

4.2.4 Nastavení výpočtu

Slovník */system/controlDict* obsahuje parametry pro řízení průběhu výpočtu. Vlastní možnosti nastavení počátku simulace či frekvence zápisu výsledků jsou poměrně široké a detailně popsané v OF: User guide [1].

Nastavení časového kroku ve stacionárních výpočtech má vliv pouze na označení časových složek a je tedy výhodné používat délku kroku 1. Jelikož není nezbytné ukládat průběžně výsledky výpočtu, je zde také obvykle volen delší interval zápisu z důvodu úspory kapacity úložiště.

```
application     simpleFoam;
startFrom       latestTime;
startTime       0;
stopAt          endTime;
endTime         20000;
deltaT          1;
writeControl    timeStep;
writeInterval   500;
purgeWrite      0;
writeFormat     ascii;
writePrecision  6;
writeCompression off;
timeFormat      general;
timePrecision   6;
runTimeModifiable true;
functions
{
    forces
    {   (...)
```

```
application     icoFoam;
startFrom       latestTime;
startTime       0;
stopAt          endTime;
endTime         35;
deltaT          0.001;
writeControl    runTime;
writeInterval   0.1;
purgeWrite      0;
writeFormat     ascii;
writePrecision  6;
writeCompression off;
timeFormat      general;
timePrecision   6;
maxCo           0.6;
runTimeModifiable true;

adjustTimeStep  yes;
```

obr. 9: Nastavení slovníků controlDict

V nestacionárních simulacích je délka časového kroku významným parametrem z hlediska stability a zčásti také přesnosti řešení. Solver icoFoam používá při výpočtu algoritmus PISO, jehož konvergence je podmíněna Courantovou podmínkou $Co_{max} \leq 1$ (viz 3.1) a při trvajícím překročení této podmínky výpočet kolabuje.

Hodnotu maximální délky časového kroku lze získat pokusem nebo je možné použít hrubý odhad pro jednorozměrné proudění ve směru x

$$\Delta t_{max} \leq \frac{\Delta x}{U_x}$$

kde dosadíme U jako vstupní rychlost na inletu a Δx jako nejmenší rozměr elementů sítě (viz 4.2.1). Dále lze předpokládat, že lokální rychlost v místě obtékání profilu (kde je zároveň nejjemnější síť) nezanedbatelně vzroste. Počáteční délka časového kroku je proto s bezpečnou rezervou zvolena <0,0001>s a v průběhu výpočtu bude dále upravena.
Užitečnou funkcí je také automatická regulace časového kroku *adjustableTimeStep*, jež případně zkrátí délku kroku v závislosti na zvolené maximální hodnotě Courantova čísla *maxCo*. Možnost změny nastavení všech slovníků v průběhu výpočtu (tedy bez jeho přerušení) pak představuje volba *runTimeModifiable*, která předepisuje nové načítání databází ze slovníků na začátku každého časového kroku.

Kategorie volitelných funkcí (*Function objects*) v OF obsahuje rozsáhlou nabídku dostupnou opět v User guide [1]. V této úloze byly zvoleny funkce *forceCoeffs* (určení silových účinků na objekt), v solveru icoFoam pak dále *probes* (sondy pro bodové hodnoty, konkrétně tlakového pole) a *fieldAverage* (průběžné ukládání průměrných hodnot nestacionárních polí).

```
functions
{
    forces
    {
        type forceCoeffs;
        functionObjectLibs ( "libforces.so" );
        outputControl timeStep;
        outputInterval 1;

        patches
        (
            wall // název okrajové plochy tělesa
        );

        pName p; // názvy polí proměnných
        UName U;
        rhoName rhoInf; // nestlačitelná kap.
        log yes;
        rhoInf 998.204; // konstantní hodnota
        liftDir (0 1 0); // určení směrů koef. sil
        dragDir (1 0 0);
        CofR (0.05 0.05 0); // vztažný bod pro sil. moment
        pitchAxis (0 0 1); // směr osy rotace
        magUInf 0.0245; // vel. rychlosti volného proudu
        lRef 1; // měřítko ref. délky tělesa
        Aref 3.58e-05; // ref. plocha tělesa
    }
```

obr. 10: Definice volitelné funkce forceCoeffs

4.3 Výpočet

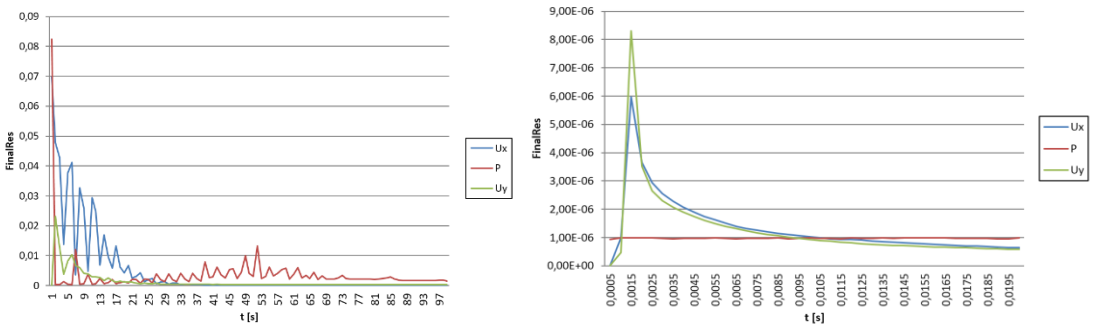
Pro paralelizaci výpočtu je nutné zavést slovník */system/decomposePar*, který obsahuje parametry pro dekompozici sítě na požadovaný počet úseků v jednotlivých směrech. Proudění v každém jednotlivém úseku potom bude řešeno paralelně spuštěným solverem na jednotlivých výpočetních jádrech (vláknech) procesoru.

Proces dekompozice je dále automatizován a proveden spuštěním utility *decomposePar*. Dále je vhodné (zvláště pro rozsáhlé sítě) přepsat číslování buněk sítě utilitou *renumberMesh* z důvodu úspory šířky pásma znaků potřebných pro čtení a zápis hodnot. Jelikož už je síť rozdělena na jednotlivé úseky, musí být dále (až do jejich opětovného sloučení) spouštěny všechny aplikace paralelně přes protokol *mpi*. Všechny paralelní výstupy aplikací jsou ukládány do nově vzniklých složek pro každý procesor zvlášť a před následným zpracováním dat (například v aplikaci Paraview) musí být opět sloučeny pomocí utility *reconstructPar*.

Podoba zmíněných příkazů pro paralelní spuštění aplikací je tedy následující:

- *mpirun -np 2 renumberMesh -overwrite -parallel*
- *foamJob mpirun -np 2 simpleFoam -parallel*

Především na začátku výpočtu je důležité monitorovat průběh konvergence řešení. Výpis hodnot reziduálů během řešení je standardně zobrazován pouze v okně terminálu, je však možné jej přesměrovat do textového souboru linuxovým příkazem > *název_souboru*. Další možností je využít skript*foamJob*, která přesměruje výpis do souboru *log*. Průběh konvergence výsledků jednotlivých polí pak ze souboru extrahuje skript *foamLog*, jehož výstup lze graficky zobrazit například pomocí linuxové aplikace *Gnuplot*.



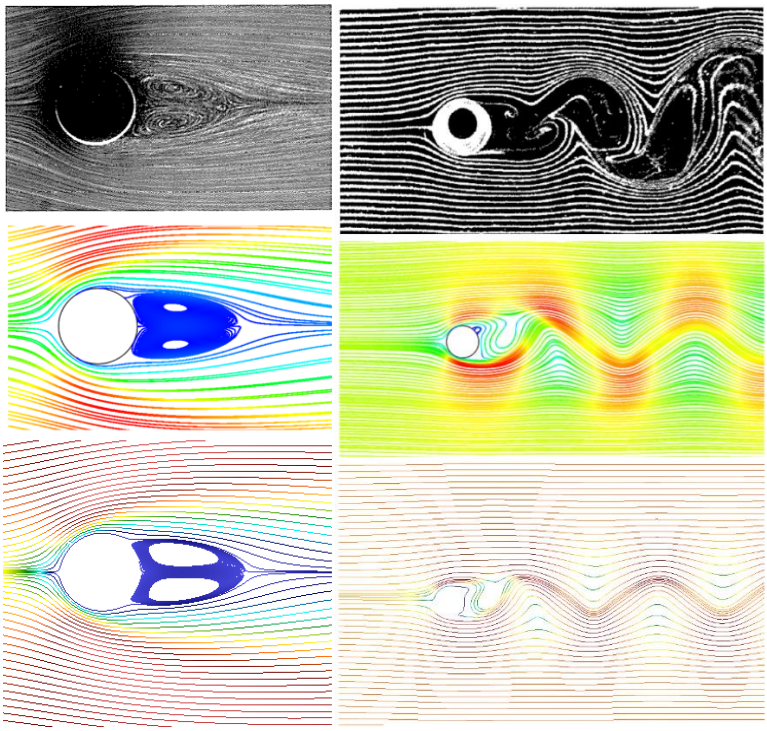
obr. 11: Vykreslení konvergujících výpočtů
(vlevo simpleFoam, Re 50; vpravo icoFoam, Re 195)

4.4 Post-processing a porovnání výsledků

4.4.1 Paraview

V rámci vyhodnocení výsledků simulací budou použity grafické nástroje pro vizualizaci proudění v aplikaci Paraview. S ohledem na poměrně dlouhý interval vzorkování hodnot polí není vhodné využít některé funkce tohoto prostředí (např. *probes* pro sledování lokálního vývoje hodnot proměnných v čase), ty však byly nahrazeny pomocí výpisů *function objects* uvedených níže. Značné zjednodušení načtení simulace poskytuje v OF funkce *paraFoam*, jenž po zavolání z terminálu spustí Paraview a otevře aktuální úlohu.

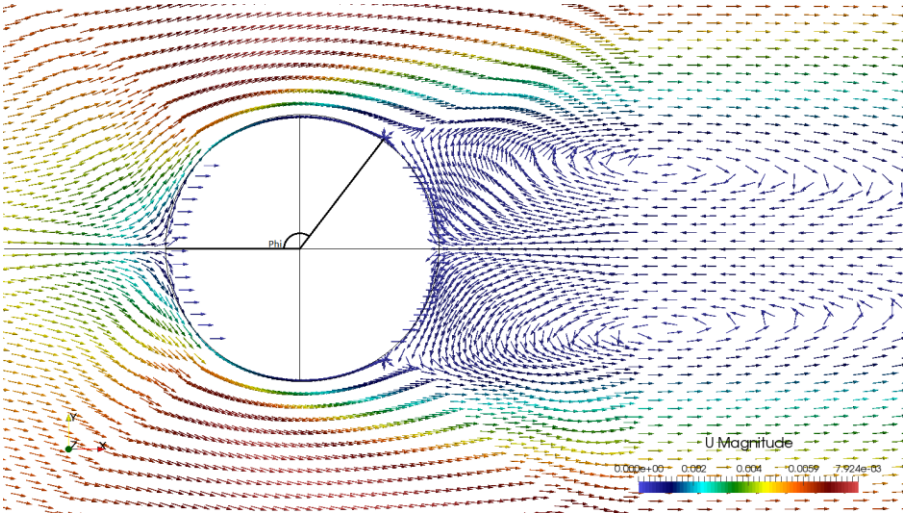
Pro korektní dvourozměrné zobrazení je vhodné provést řez domény zvolenou rovinou (*slice*). Jako zdroj určující polohu proudnic (*streamlines*) v prostoru je v následujících ukázkách zvolena úsečka rovnoběžná se směrem *y*, po jejíž délce je pak rovnoměrně rozložen předepsaný počet výchozích bodů proudnic.



obr. 12: Porovnání proudnic pro Re 26 (vlevo) a Re 195 [17]

Výsledné tvary proudění si v rámci simulací odpovídají, vidíme však zřetelný nesoulad s vizualizací experimentální. To může být dáno unášením a vizualizačního média proudem (vizualizace tedy neodpovídá definici proudnice, nýbrž zobrazuje tzv. *streaklines*) a také nesplněnou podobností proudění v ohledech jiných, než srovnává Reynoldsovo číslo.

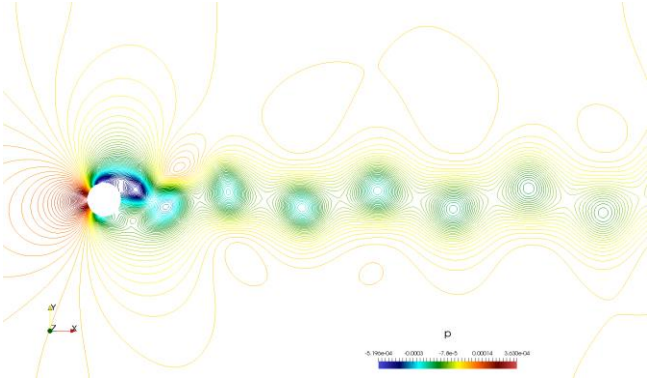
Vizualizace směrů vektorů rychlostí je možná pomocí funkce *glyph*. Škálování měřítka šipek v závislosti na velikosti rychlostí není v ukázce povoleno z důvodu zřetelné vizualizace místa odtržení proudu od tělesa. Informaci o velikosti vektorů podává standardní zobrazení hodnot pole rychlostí *U*.



obr. 13: Zobrazení směrů rychlostí pro Re 50 pomocí funkce glyph

Zjištěný úhel odtržení proudu *Phi* dosahuje ve výsledcích solveru icoFoam velikosti 126°, což odpovídá závěrům studie [16]. Zároveň bylo ověřeno, že

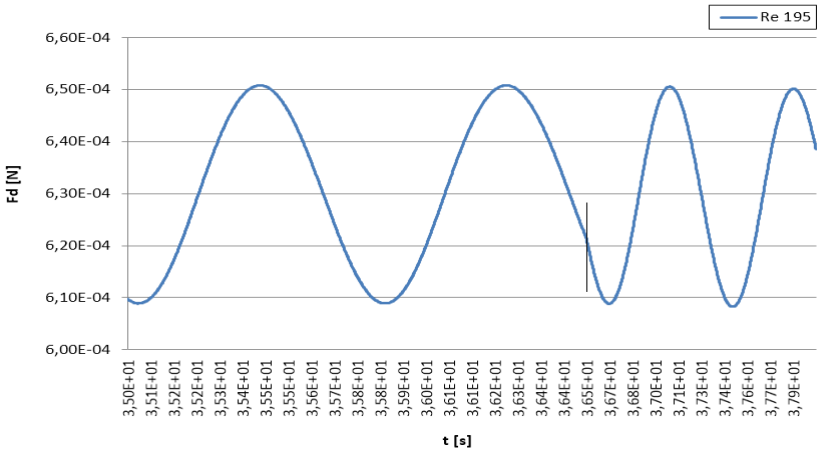
Pomocí izočar (*contour*) tlakového pole je pak dobře viditelná stezka lokálních tlakových poklesů v proudu za tělesem, jenž vzniká v důsledku stále rotujících vírových struktur.



obr. 14: Kontury tlakového pole

4.4.2 Vyhodnocení volitelných funkcí

Soubory s výpisy hodnot *function objects* se nachází v příslušných podsložkách vytvořeného adresáře *postProcessing*. Výstup *forceCoeffs* poskytuje hodnoty odporového součinitele pro jednotlivé iterace a nepřímo tak rovněž výsledek odporové síly působící ve směru proudu na objekt. Pokud při obtékání tělesa vznikne oscilující vírová stezka, osciluje rovněž odporová síla (změna křivky grafu v čase 36,5s je dána manuální změnou časového kroku v průběhu výpočtu). Střední hodnota součinitele odporu pro modelovaný profil byla vypočtena jako 1,44 a koresponduje s výsledky studie. Frekvence kmitů byla určena na 1,22Hz.



obr. 15: Časový průběh působení odporové síly na objekt

5. ZÁVĚR

OpenFOAM je velice komplexní nástroj poskytující široké možnosti využití. Jeho hranice nemohou být pevně stanoveny snad v žádném technickém odvětví, neboť aktivní uživatelská komunita a taktéž otevřená komunita developerů přináší stále nové podněty i řešení. Svobodný kód zde jednoznačně otevírá dveře dynamickému rozvoji.

Snad právě otevřené modulární rysy tohoto systému mohou být pro začátečníka v oblasti CFD poněkud odstrašující. Překonání tohoto počátečního nepohodlí však nakonec může vést k rychlejšímu propojení či poznání jednotlivých souvislostí, také práce s textovými soubory místo dialogových oken přináší časem nemalé množství výhod. OF se takto jeví jako mocný prostředek v rukou zkušeného uživatele i jako kvalitní lektor, který propustí jen velmi málo chyb.

V průběhu práce byla pozornost soustředěna především na jednoduché úlohy proudění kapalin a náležitosti s nimi spojené. Téměř bez povšimnutí tak zůstal značný rozsah využití této rozmanité knihovny, jenž sice nepodléhá rámci práce, bezpochyby však zaslouží přinejmenším ekvivalentní pozornost.

Silnou stránkou knihovny je podpora masivně paralelních výpočtů. Systém je tak nakloněn jednotlivcům, univerzitám i firmám, jenž zde nachází značnou úsporu licenčních poplatků. Je však nutné zdůraznit, že tyto poplatky tvoří jen část nákladů spojených s CFD výpočty a například značné náklady na profesionální školení, koupi i údržbu výpočetních stanic a v neposlední řadě odměny zaměstnancům zůstávají.

6. SEZNAM POUŽITÉ LITERATURY

[1] OpenFOAM User Guide [online]. CFD Direct, c2015 [vid. 20. května 2016]. http://cfd.direct/openfoam/user-guide/

[2] OpenFOAMWiki [online]. https://openfoamwiki.net/index.php/Special:Categories

[3] OpenFOAM – OpenCFD [online]. http://www.openfoam.com

[4] Official OpenFOAM repository [online]. https://github.com/OpenFOAM

[5] Paraview (oficiální web) http://www.paraview.org/overview/

[6] SATO, M., KOBAYASHI, T. *A fundamental study of the flow past a circular cylinder using Abaqus/CFD*.

[7] ANDERSON, J.D. *Computational Fluid Dynamics: The basics with applications.* McGraw Hill. 1995.

[8] VERSTEEG, H.K., MALALASEKERA, W. *An introduction to computational fluid dynamics: The finite volume method*. Longman Group Ltd. 1995.

[9] MARIĆ T., HÖPKEN J., MOONEY K. *The OpenFOAM Technology Primer*. Sourceflux UG, 2014. 458 p.

[10] DRÁBKOVÁ, S. a kol. *Mechanika tekutin* [online]. 1.vyd. Ostrava: VŠB-TU, 2007. 257 s. http://www.338.vsb.cz/PDF/DrabkovaMechanikatekutin.pdf. ISBN 978-80-248-1508-4

[11] KOZUBKOVÁ, Milada, Sylva DRÁBKOVÁ a Pavel ŠŤÁVA. *Matematické modely nestlačitelného a stlačitelného proudění: metoda konečných objemů*. 1. vyd. Ostrava: VŠB-TU, 1999. ISBN 80-7078-709-0.

[12] KOZUBKOVÁ, Milada. *Modelování proudění tekutin FLUENT, CFX* [online]. 1. vyd. Ostrava: VŠB-TU, 2008. 142 s. http://www.338.vsb.cz/PDF/Kozubkova-Fluent.pdf. ISBN 978-80-248-1913-6.

[13] BLEJCHAŘ, T. *Turbulence, Modelování proudění – CFX* [online].1. vyd. Ostrava: V3B-TU, 2008. 263 s. http://www.338.vsb.cz/PDF/Turbulence_ESF_v4.pdf

[14] HÁJEK, J. *Aplikace výpočtové dynamiky tekutin v oblasti procesního průmyslu.* Brno: VUTIUM, 2008. 26 s. ISBN 978-80-214-3598-8.

[15] ŠTIGLER, J. Pohybové rovnice - podklady pro předmět Hydromechanika [online]. http://studyenergyweb.fme.vutbr.cz/file/512

7. PŘÍLOHY

7.1 Výpis utility checkMesh

```
Overall number of cells of each type:
    hexahedra:     155524
    prisms:        0
    wedges:        0
    pyramids:      0
    tet wedges:    0
    tetrahedra:    0
    polyhedra:     0

Checking topology...
    Boundary definition OK.
    Cell to face addressing OK.
    Point usage OK.
    Upper triangular ordering OK.
    Face vertices OK.
    Number of regions: 1 (OK).

Checking patch topology for multiply connected surfaces...
    Patch               Faces    Points   Surface topology
    inlet               223      448      ok (non-closed singly
connected)
    outlet              254      510      ok (non-closed singly
connected)
    symmetryplane1      599      1200     ok (non-closed singly
connected)
    wall                211      422      ok (non-closed singly
connected)
    symmetryplane2      599      1200     ok (non-closed singly
connected)
    frontAndBackPlanes  311048   312934   ok (non-closed singly
connected)

Checking geometry...
    Overall domain bounding box (-5.48874e-18 0 -0.00223607) (0.2
0.1 0.00223607)
    Mesh has 2 geometric (non-empty/wedge) directions (1 1 0)
    Mesh has 2 solution (non-empty) directions (1 1 0)
    All edges aligned with or perpendicular to non-empty
directions.
    Boundary openness (1.70542e-17 -1.40788e-16 -6.67709e-19) OK.
    Max cell openness = 2.24632e-16 OK.
    Max aspect ratio = 3.74596 OK.
    Minimum face area = 1.00312e-08. Maximum face area =
2.05615e-06.  Face area magnitudes OK.
    Min volume = 4.48608e-11. Max volume = 9.24114e-10.  Total
volume = 8.9218e-05.  Cell volumes OK.
    Mesh non-orthogonality Max: 44.875 average: 5.61909
    Non-orthogonality check OK.
    Face pyramids OK.
    Max skewness = 1.37382 OK.
    Coupled point location match (average 0) OK.

Mesh OK.
```