VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV PROCESNÍHO INŽENÝRSTVÍ
INSTITUTE OF PROCESS ENGINEERING

PARAMETRICKÝ GENERÁTOR VÝPOČETNÍ SÍTĚ TRUBKOVÉHO SVAZKU VÝMĚNÍKU TEPLA PRO OPENFOAM

PARAMETRIC GENERATOR OF HEAT EXCHANGER TUBE BUNDLE MESH FOR OPENFOAM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. Martin Petrů
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. Vojtěch Turek, Ph.D.
SUPERVISOR

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav procesního inženýrství
Student:	**Bc. Martin Petrů**
Studijní program:	Procesní inženýrství
Studijní obor:	bez specializace
Vedoucí práce:	**doc. Ing. Vojtěch Turek, Ph.D.**
Akademický rok:	2021/22

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 diplomové práce:

Parametrický generátor výpočetní sítě trubkového svazku výměníku tepla pro OpenFOAM

1 Stručná charakteristika problematiky úkolu:

Open source software OpenFOAM pro CFD výpočty je propracovanou alternativou k různým komerčním CFD softwarům. V určitých ohledech je ale méně uživatelsky přívětivý – například specifikace výpočtových úloh se provádí pomocí skupiny textových konfiguračních souborů. Existence nástroje, který by na základě zadaných parametrů vygeneroval soubory s definicí kvalitní výpočetní sítě reprezentující trubkový prostor výměníku tepla se svazkem trubek v plášti, by tedy byla velmi přínosná.

2 Cíle diplomové práce:

(1) Seznámit se se základy CFD modelování (formou literární rešerše) a softwarem OpenFOAM.
(2) Vytvořit dle pokynů vedoucího práce počítačový kód, jehož výstupem bude zadaným parametrům odpovídající skupina konfiguračních souborů definujících kvalitní výpočetní síť trubkového prostoru výměníku tepla se svazkem trubek v plášti.
(3) Otestovat korektnost generovaných souborů zkušebním výpočtem v softwaru OpenFOAM.

3 Seznam doporučené literatury:

GREENSHIELDS, C. J. OpenFOAM User Guide, Version 7. London: The OpenFOAM Foundation, 2019.

VERSTEEG, H. K. a W. MALALASEKERA. An introduction to computational fluid dynamics: The finite volume method. 2nd ed. New York: Pearson Education, 2007. ISBN 978-0-13-127498-3.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2021/22

V Brně, dne

L. S.

prof. Ing. Petr Stehlík, CSc., dr. h. c. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

Abstrakt

Tato diplomová práce pojednává o základech výpočetní dynamiky tekutin (CFD), důvodech jeho použití a související nutnosti vytvářet výpočetní sítě. Další část práce se zaměřuje na vyhodnocování kvality výpočetních sítí a stručně na výpočetní metody CFD. Jsou také srovnány dostupné síťové generátory a poskytnut přehled o možnostech tvorby sítí. Dále jsou uvedeny základní informace o softwaru OpenFOAM a jeho možnostech, zejména tvorby sítě pomocí aplikace blockMesh. Následující část práce je věnována popisu vyvinuté aplikace a jejím funkcím pro generování kvalitní výpočetní sítě trubkového svazku výměníku tepla pro software OpenFOAM. V závěru je simulací ustáleného proudění ověřena správnost vytvořené sítě a její dostatečná kvalita.

Klíčová slova

CFD, výpočetní síť, CFD síťové generátory, OpenFOAM, blockMesh, kvalita sítě pro CFD, tvorba sítě

Abstract

This thesis deals with the basics of computational fluid dynamics (CFD), the reasons for its use, and the related necessity to create computational meshes. The next part of the thesis focuses on assessing the quality of computational meshes and, briefly, on computational methods of CFD. The available mesh generators are also compared, and an overview of meshing capabilities is provided. The following is basic information about the OpenFOAM software and its capabilities, in particular the creation of meshes using the blockMesh application. The following part of the thesis is focused on the description of the developed application and its functions for generating a high-quality computing mesh of a tube bundle heat exchanger for the OpenFOAM software. In conclusion, a steady flow simulation verifies the correctness of the created mesh and its sufficient quality.

Key words

CFD, computional mesh, CFD mesh generators, OpenFOAM, blockMesh, quality of mesh for CFD, mesh generation

Bibliografická citace

PETRŮ, Martin. Parametrický generátor výpočetní sítě trubkového svazku výměníku tepla pro OpenFOAM. Brno, 2022. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/139359. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav procesního inženýrství. Vedoucí práce Vojtěch Turek.

Čestné prohlášení

Prohlašuji, že jsem diplomovou práci Parametrický generátor výpočetní sítě trubkového svazku výměníku tepla pro OpenFOAM vypracoval samostatně pod vedením doc. Ing. Vojtěcha Turka Ph.D. a s použitím literatury uvedené v seznamu.

V Brně dne 20. května 2022

.....

Martin Petrů

Poděkování

Nejprve chci poděkovat svému Bohu, který se k nám ze své lásky přiblížil ve svém Synu Ježíši, když dobrovolně zemřel na kříži za naše viny. Děkuju mu, že přijetím Jeho odpuštění jsem tuto práci mohl psát ve svobodě s Ním. Také mu chci děkovat konkrétně za to, že mi trpělivě odpouštěl moji prokrastinaci, hněv na problémy při psaní, nedůvěru v Jeho načasování a i to, že jsem tuto práci psal ze zlých motivací. Žasnu nad tím, jak přes toto všechno mi ze své milosti dovolil tuto práci dokončit.

Velké díky patří mému skvělému vedoucímu doc. Ing. Vojtěchu Turkovi Ph.D., který mě v práci vedl s velkou trpělivostí a moudrostí. Děkuji mu za cenné rady, opravu mých mnohých chyb a upřímnost. Taky za to, že mi byl k dispozici doslova kdykoliv a za zpětnou vazbu.

V neposlední řadě bych chtěl poděkovat své rodině a přítelkyni, že mi po celý čas psaní této práce poskytovali zázemí a dávali lásku. Děkuji i svým přátelům, zejména Andrejce Pěchové za cenné rady a Petru Klímovi, který mi byl v těžkých chvílích při psaní oporou a dával mi motivaci. Chci mu poděkovat i za to, že je přítelem, který podrží.

Obsah

1 Úvod

Mnoho technologií jako například tepelné výměníky, motory nebo destilační kolony používalo v minulosti a používá i nyní tekutiny a plyny jako pracovní látky. V těchto zařízeních dochází k jejich proudění a jsou tak vystaveny různým účinkům zatížení, jako je například působení teploty a tlaku.

Pochopení sil a účinků dynamiky tekutin může vést ke zvýšené efektivitě a snížení rizik poruchovosti těchto zařízení. Správnými výpočty a simulacemi pomocí Computional fluid dynamics (CFD) lze předejít tomu, aby se počítače přehřívaly, součástky selhávaly a konstrukce se hroutily pod náporem silného větru. CFD se používá v případech, kdy řešení nelze dosáhnout konvenčními výpočty.

Prvotní snahou lidí zabývající se problematikou proudění bylo vytvoření rovnic, které by dynamiku tekutin správně popisovaly. Vznikly tak například nejznámější Navier–Stokesova rovnice. Tyto rovnice jsou však natolik složité, že oblast dynamiky proudění zažila posun ve svém vývoji až v době, kdy poprvé vznikla možnost využít pro výpočty obrovskou sílu počítačové techniky. Na základě toho vzniklo samotné CFD. [27]

Výpočet CFD požaduje velmi výkonný hardware. Požadavek pro dostatečně výkonný a zároveň dostupný hardware byl vyslyšen zejména v devadesátých letech minulého století, kdy se více rozšířil v průmyslovém odvětví. Složitosti výpočtů a jejich požadavků na výkon většinou odpovídá i cena. Ty se za daný hardware pohybují od 5 000 do 10 000 € a cena komerční licence pro uživatelsky přívětivý software se pohybuje až od 10 000 do 50 000 €. Naproti vysokým nákladům CFD nabízí vysoké množství detailů. Existují také nekomerční softwary pro výpočty CFD, jako například OpenFOAM, který nabízí konkurenceschopné nástroje jak pro tvorbu sítě, tak pro výpočet a post–processing. [4]

Vysokou přesnost řešení nabízí též experimentální metody. Ty jsou však rovněž velmi nákladné a mnohdy omezené rozměrově i dalšími parametry. Pomocí CFD je možné studovat proudění v libovolném průřezu geometrie a s různými softwarovými nástroji se lze o dynamice tekutin dozvědět mnohem více než z fyzických modelů. Zároveň propojením CFD se softwary pro 3D modelování lze rychle optimalizovat design a dosáhnout tak u zařízení vyšších výkonů.

Cílem této práce je krátce popsat CFD a vytvořit software, který tvoří dle zadaných parametrů kvalitní výpočetní síť trubkového výměníku pro CFD výpočty pro software OpenFOAM. Hlavní motivací pro vytvoření tohoto softwaru je tvorbu sítě automatizovat, jelikož manuální tvorba může být při větších počtech trubek problematická. Pro tvorbu geometrie byl použit rozšířený a mezi uživateli oblíbený programovací jazyk Python, který je také zdarma.

2 Výpočty dynamiky tekutin pomocí CFD

CFD je věda využívající metody numerické matematiky k řešení matematických rovnic pro simulaci proudění, přenosu tepla a hmoty a jevů s tím spojených. [1] Proudění je popsáno parciálními diferenciálními rovnicemi, které až na jednoduché případy nelze vyřešit analyticky.

Numerické metody se používají v případě, když je analytické řešení příliš komplikované nebo nemožné. Jejich cílem je pomocí vhodného algoritmu dojít k řešení dané numerické úlohy. Při jejich řešení se používají konkrétní číselné hodnoty. V dnešní době jsou závislé na práci s počítači, jelikož musí v rámci výpočtu pracovat s velkým množstvím rovnic. Počítač provádí pomocí algoritmu akce, které vedou k nalezení řešení. Výstupní hodnoty jsou závislé na definovaných vstupních parametrech. [5] Numerické metody, jako i jiné, obsahují chyby. Důležitým pojmem je konvergence a stabilita. Výpočet, který konverguje, se přibližuje k přesnému řešení. U stabilního výpočtu nedochází během iterací k nárůstu chyb, v ideálním případě dochází k jejich snižování. [3]

Pro geometrie malých objemů lze proudění vždy matematicky popsat a nalézt k němu přesné řešení. V případě velkých a složitých objemů je nutné danou geometrii pro nalezení přesného řešení rozdělit na menší útvary neboli geometrii diskretizovat. Tyto menší části se nazývají kontrolní objemy. [2] Diferenciální rovnice se aproximují na algebraické rovnice a aplikují se na jednotlivé kontrolní objemy. Mezi nimi se pak pomocí fyzikálních zákonů jako například zákona zachování hmoty a energie přepočítávají jednotlivé toky hmoty a energií. Tato metoda se nazývá "Metoda konečných objemů". [3]

Přesnost výpočetních CFD metod závisí hlavně na kvalitě diskretizované geometrie, neboli sítě. Její kvalita je ovlivněna více faktory, jako je například tvar jednotlivých kontrolních objemů nebo jejich velikost. Pro diskretizaci geometrie existují různé metody popsané v následujících kapitolách. [2]

2.1 Typy sítí pro CFD výpočty

Řídící rovnice proudění, které jsou buď v diferenciální nebo integrální formě, jsou diskrétní a předepisují se buď v bodě, prvku nebo v kontrolním objemu. Rovnice stejně jako geometrie se diskretizují. [7]

Pro vybraný řešič (solver), což je část kódu, jehož výstupem je řešení například tlakového nebo rychlostního pole a všech jeho neznámých proměnných zadaných na vstupu, je nutné zadat tyto vstupní parametry: [1]

- tvar a velikost geometrie, v rámci které jsou rovnice řešeny
- diskretizovat jak geometrii, tak i řídící rovnice v diferenciální nebo integrální formě do formy algebraických rovnic
- nastavit tzv. „okrajové podmínky"; vytvořená síť musí být ve formátu čitelném pro vybraný řešič.

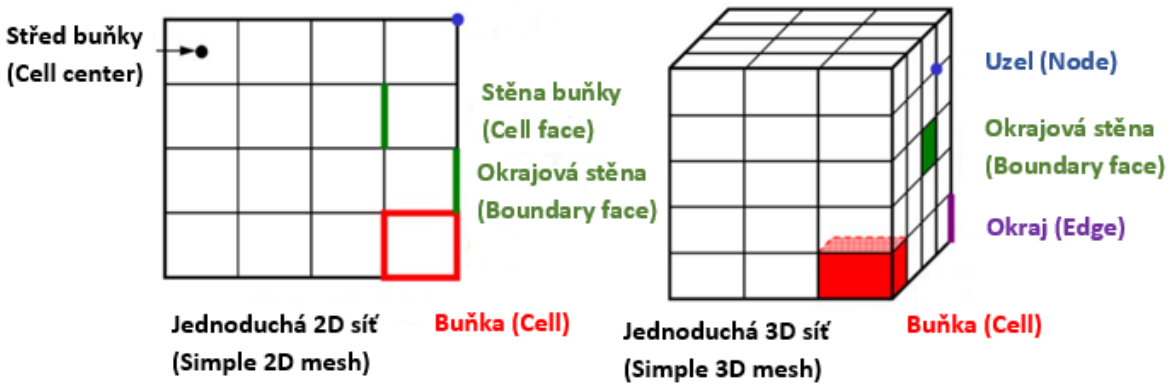
Okrajové podmínky (boundary conditions) jsou omezení, které udávají požadovanou hodnotu řešení hranici výpočetní domény. Jsou napsány pro dané stěny kontrolních objemů na okraji sítě

a definují chování proudění v interakci s touto stěnou. Při řešení CFD rovnic se odvíjejí od typu této stěny a povaze proudění, které může být buď turbulentní, nebo laminární. [8]

Tato kapitola se bude zabývat jednotlivými možnostmi diskretizace dané geometrie, což je začátkem řešení CFD.

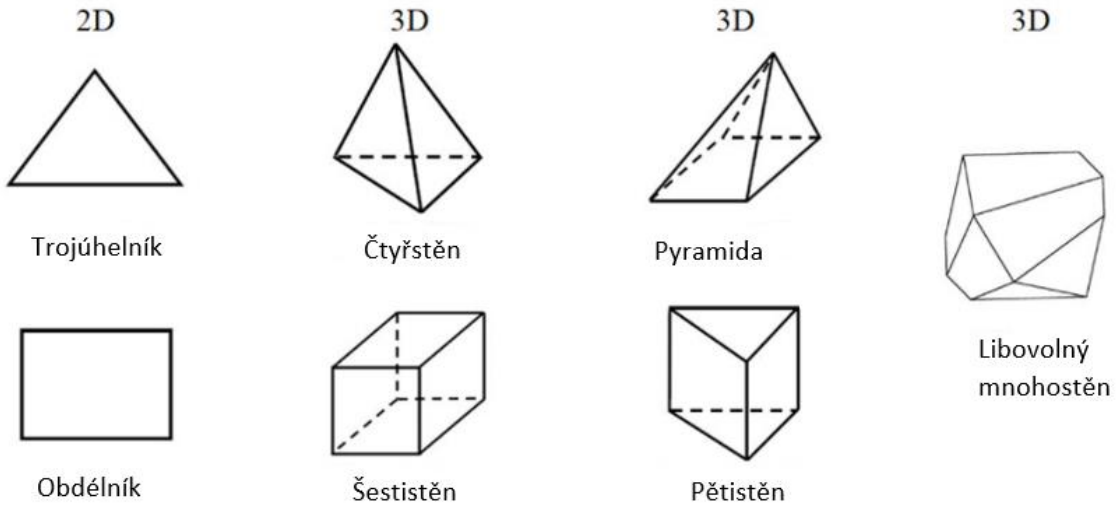
2.1.1 Základní pojmy CFD sítí

Tato kapitola se věnuje pojmenování základních parametrů sítě, tvaru kontrolních objemů a základnímu rozdělení síťových struktur. Na obr. 1 jsou pojmenovány jednotlivé části sítě.



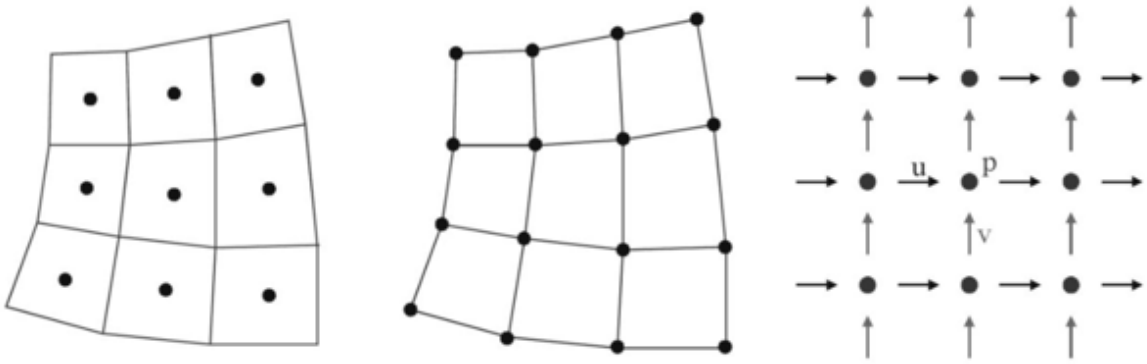
Obr. 1 Jednoduchá 2D a 3D síť [6]

Jak je vidět z obr.1, každá síť je složena z jednotlivých kontrolních objemů. Při generování sítě se s ohledem na vybraný řešič zvažuje jejich tvar. Ten je i jedním z parametrů stanovující kvalitu sítě. Řešiče strukturované sítě používají ve 2D čtyřúhelníky a ve 3D šestiboké hranoly. Řešiče nestrukturované sítě často používají trojúhelníky nebo čtyřstěny, ale novější řešiče mohou používat libovolné mnohostěnné tvary. [6] Uvedené tvary kontrolních objemů jsou spolu s dalšími tvary znázorněny na obr. 2.



Obr. 2 Používané tvary kontrolních objemů [6]

Hlavními informacemi o síti po jejím vytvoření jsou vrcholy kontrolních objemů a informace o jejich konektivitě. Podle toho s jakým typem úložiště řešič pracuje se rozlišují různé struktury sítě, které jsou ukázány na obr. 3. Některé pracují s jejich kombinacemi. Jeden řešič tak může používat dva různé způsoby ukládání hodnot. Záleží také na typu použité metody. [6]

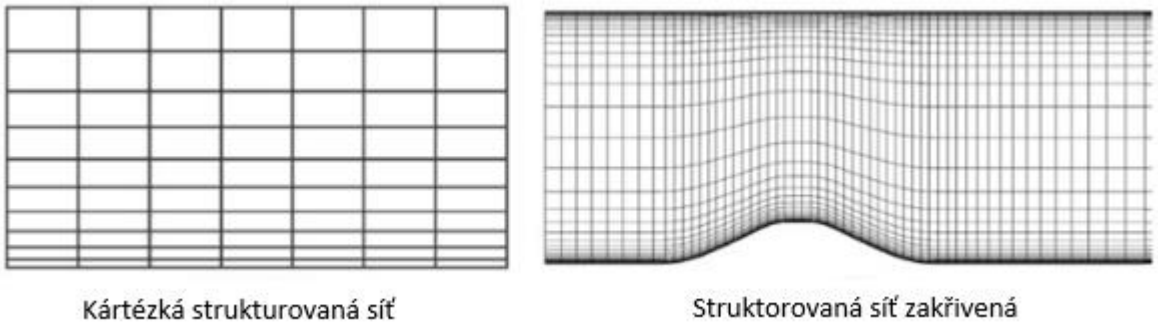


Obr. 3 Způsoby ukládání dat [6]

2.1.2 Strukturovaná síť

Rozlišují se dva typy strukturované sítě. Jedním z nich je kartézká. Je to nejjednodušší typ pro generování sítě, jelikož křížící se spojnice bodů mřížky jsou vždy na sebe vzájemně kolmé a rovnoběžné s osami kartézkého systému os. Každý kontrolní objem má tvar kvádru, což může poskytovat vysokou přesnost výpočtu. Druhý typ strukturované sítě je ten, který kopíruje zakřivené hrany dané geometrie. Oba typy jsou uvedeny na obr. 4.

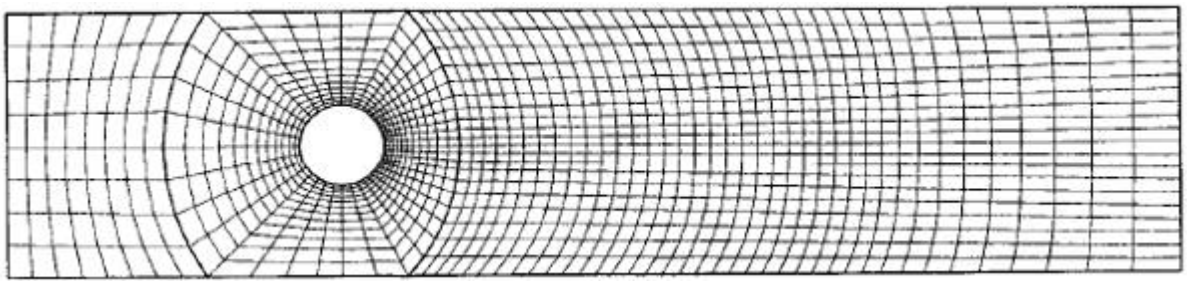
Každý uzel strukturované sítě má ve 2D čtyři a ve 3D šest sousedních uzlů. Vzájemná konektivita sousedních uzlů usnadňuje jejich řazení a zpřesňuje výpočty. Nevýhodou tohoto typu sítě může být aplikovatelnost pro pouze jednoduché geometrie a její náročnější tvorba. Další nevýhodou může být obtížné rozvržení jednotlivých uzlů, jelikož z důvodů přesnosti na jednom místě se mohou vytvořit zbytečně malé rozestupy kontrolních objemů. Tím se navýší i požadavek na výkon a čas výpočtu. [2]



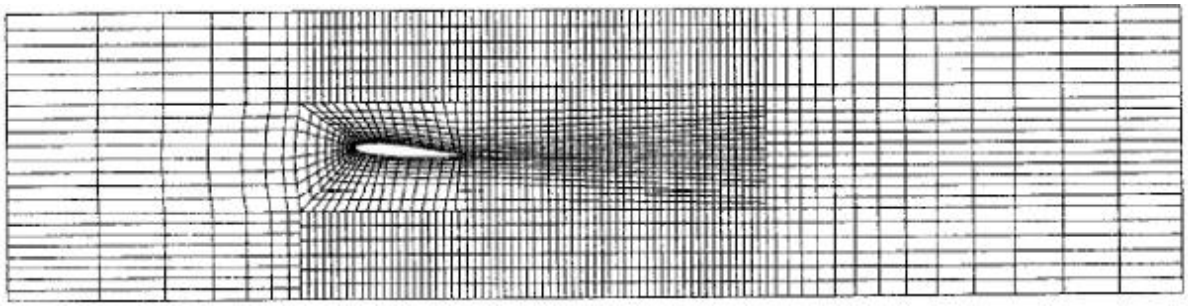
Obr. 4 Strukturovaná síť [9]

2.1.3 Blokově strukturovaná síť

U blokově strukturovaných sítí je geometrie rozdělena na menší počet oblastí neboli bloků. V každém bloku je strukturovaná síť, která mezi jednotlivými bloky na sebe může a nebo nemusí navazovat. V případě sítě, která na sebe navazuje, je počet kontrolních objemů na rozhraní stejný jako u bloku sousedního, jak je vidět na obr. 5. Ve druhém případě se počty kontrolních objemů na rozhraní neshodují, jak je vidět na obr. 6. Z těchto dvou možností je preferována první, jelikož nenavazující kontrolní objemy prodlužují výpočetní čas. [2]



Obr. 5 Blokově strukturovaná síť s navazujícími bloky [2]



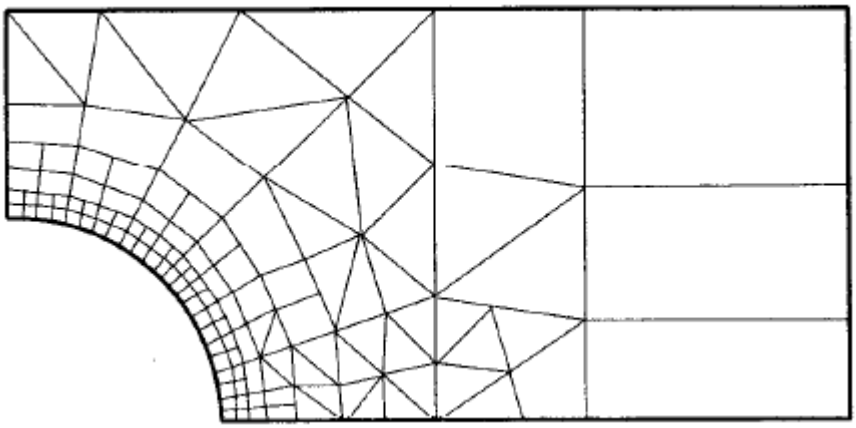
Obr. 6 Blokově strukturovaná síť s nenavazujícími bloky [2]

Síť s nenavazujícími bloky je používána pro geometrie se složitějšími hranicemi. Příkladem je obr. 6, kde se vyskytuje překážka v proudění. V takových případech je vyžadována vyšší přesnost v okolí překážky, které se dosahuje vyšší jemností sítě. Nenavazující síť umožňuje zjemění sítě bez ostrých přechodů mezi kontrolními objemy. Ostré přechody snižují její kvalitu. [2]

2.1.4 Nestrukturovaná síť

Tato síť je používána pro velmi komplexní geometrie, jelikož dokáže pokrýt jakoukoli jejich hranici. Může být použita pro každou diskretizační metodu rovnic, ale nejlépe se hodí pro metodu konečných objemů a konečných prvků. Bez závislosti na kvalitě sítě mohou mít kontrolní objemy jakýkoliv tvar a uzly mohou mít libovolný počet sousedních uzlů, jak je vidět na obr. 7.

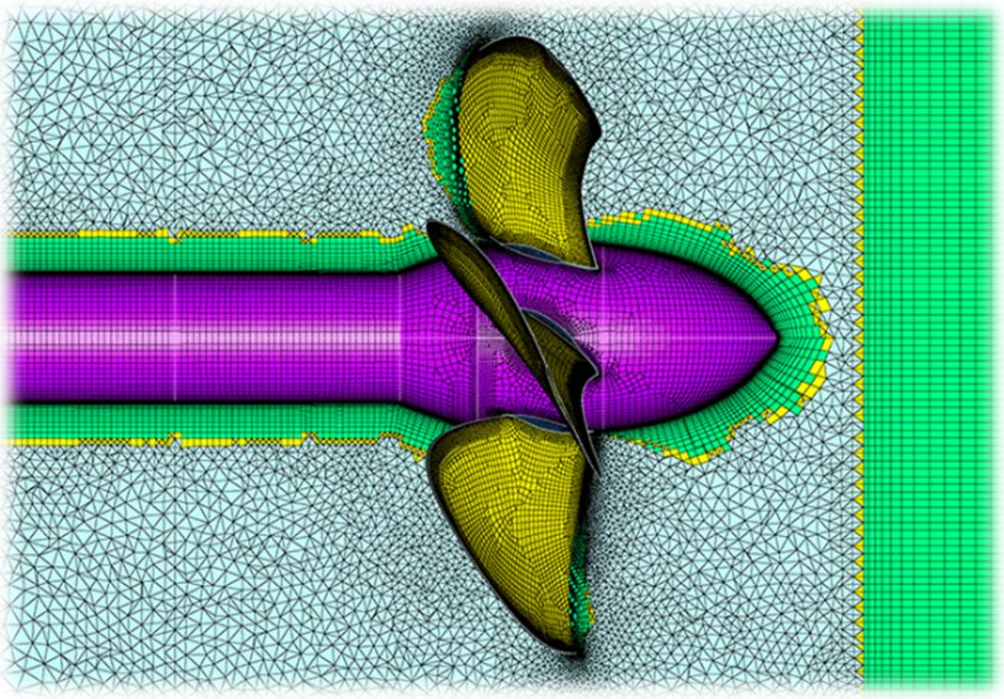
Nevýhodou této sítě je její nepravidelnost. Jednotlivé uzly a jejich konektivita s jejich jednotlivými sousedy musí být definována explicitně. Další nevýhodou je struktura algebraických rovnic, která není už tak pravidelná. Řešič je při řešení nestrukturované sítě obvykle pomalejší než u sítí pravidelného tvaru. Velmi komplexní jsou také síťové generátory. Prodlouží se i doba výpočtu, jelikož pro stejnou přesnost jakou má strukturovaná síť je zapotřebí většího počtu kontrolních objemů. [2]



Obr. 7 Nestrukturovaná síť [2]

2.1.5 Hybridní sítě

Šestistěny a čtvercové kontrolní objemy mohou dosahovat vyšší přesnosti než trojúhelníkové stěny a čtyřstěny. Na druhou stranu generování tvarem jednodušších kontrolních objemů nevyžaduje velké úsilí a šetří výpočetní čas. I z toho důvodu se v určitých případech v závislosti na složitosti geometrie typy sítí kombinují. Nazývají se hybridními sítěmi. Mohou vést k redukci celkového počtu kontrolních objemů, vyšší kvalitě sítě a kratšímu výpočetnímu času. Na obr. 8 je ukázka použití této šestistěnné a čtyřstěnné hybridní sítě pro zajištění nejvyšší možné přesnosti u konkrétního případu turbíny. Hybridní sítě využívají blokovou strukturu sítě. [6]



Obr. 9 Hybridní síť [45]

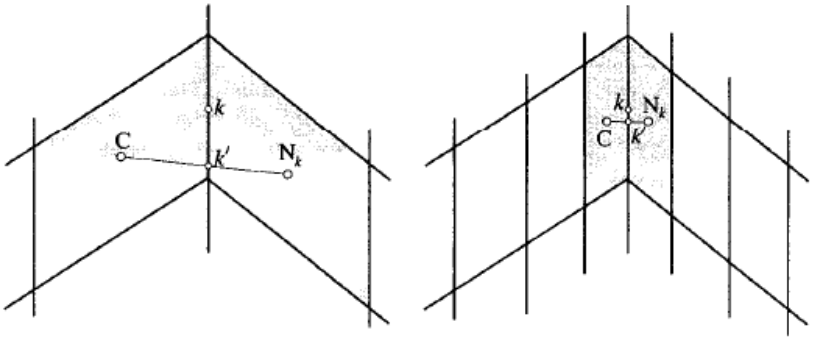
2.2 Vyhodnocování kvality sítí

Zvýšené kvality sítě je možné dosáhnout především jejím zjemňováním. Zároveň existují určité další aspekty sítě, které mají na její kvalitu také velký vliv. Tato kapitola je zaměřena na popis těchto aspektů. Navíc se zvyšujícím se zjemňování sítě rostou i požadavky na výpočetní výkon a čas. [2]

2.2.1 Velikost spojnice středů kontrolních objemů

Většina automatických síťových generátorů vytváří u blokově strukturovaných sítí ve vyčnívajících rozích velké vzdálenosti mezi body k a k', znázorněných na obr. 10 vlevo. Tato vzdálenost je dána středem spojnice mezi sousedními kontrolními objemy (bod k') a středem jejich rozhraní (bod k). Její velikost v poměru průměrné velikosti strany kontrolního objemu je kritériem kvality sítě.

Oblast kontrolních objemů, u nichž je velikost tohoto kritéria příliš velká, by se měla lokálně upravit. Ideálně by se velikost stran kontrolních objemů měla zmenšit tak, jak je uvedeno na obr. 10 vpravo, kde se tak zmenšila vzdálenost mezi body k a k'. [2]



Obr. 10 Úprava kontrolních objemů ve vyčnívajících rozích [2]

2.2.2 Poměr stran

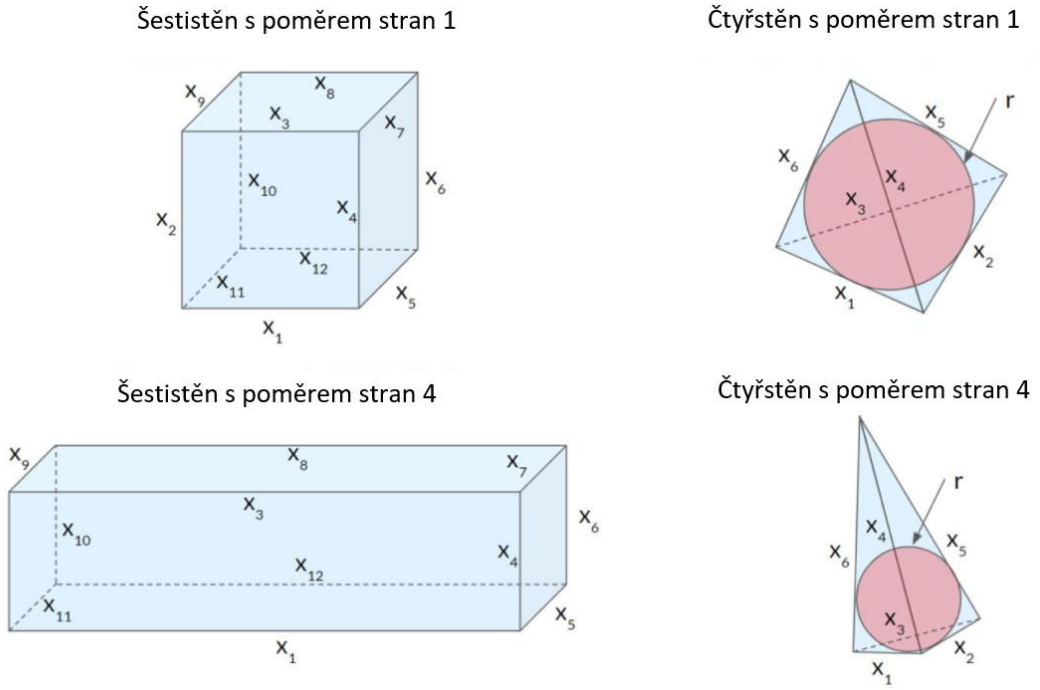
Dalším aspektem pro hodnocení kvality sítě je poměr stran. Jedná se o poměr vzdálenosti nejdelší strany kontrolního objemu s jeho nejkratší stranou. Avšak výpočetní vztah pro různé tvary kontrolního objemu se mírně odlišuje. Je zde uveden příklad některých z nich i s jejich znázorněním na obr. 11. [10]

Vzorec pro výpočet poměru stran šestistěnu, kde jednotlivé proměnné $x_1 - x_{12}$ jsou velikosti stran na obr. 11 vlevo:

$$Poměr\ stran = \frac{\max(x_1, x_2 \dots, x_{12})}{\min(x_1, x_2 \dots, x_{12})} \quad (1)$$

Vzorec pro výpočet poměru stran čtyřstěnu, kde jsou proměnné velikosti stran a poloměr na obr. 11 vpravo:

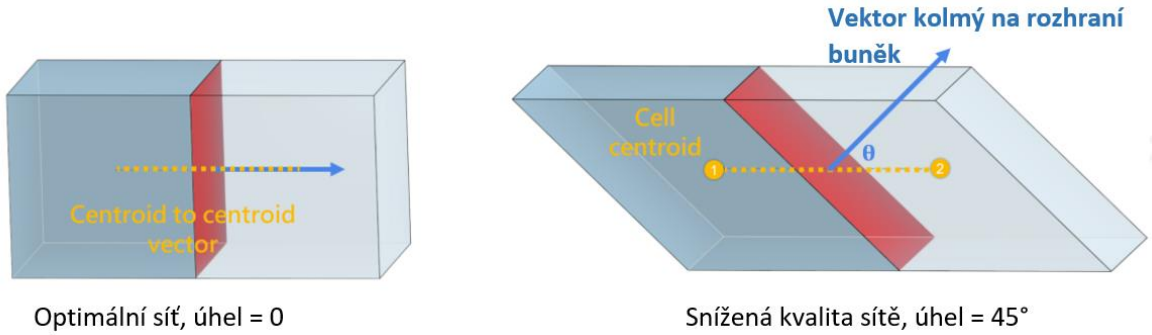
$$Poměr\ stran = \frac{\max(x_1, x_2 \ldots, x_6)}{2 \cdot \sqrt{6} \cdot r} \quad (2)$$



Obr. 11 Aspekt poměru stran kontrolního objemu [10]

2.2.3 Pravoúhlost

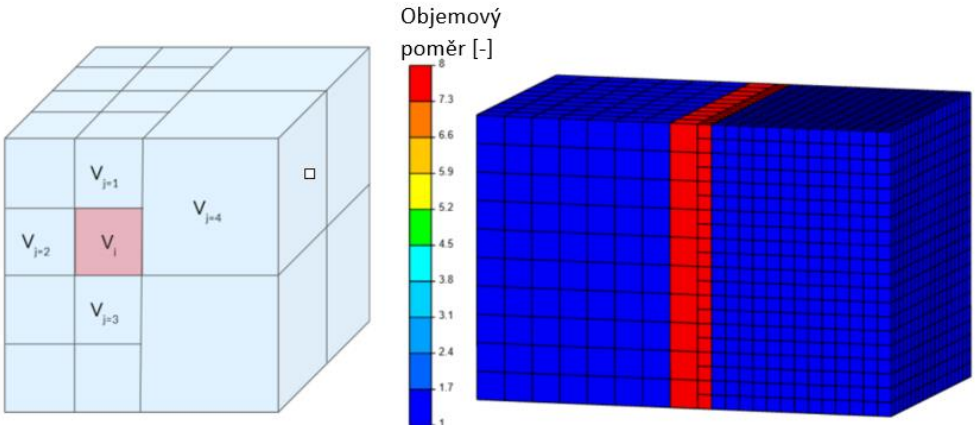
Úhel, který se posuzuje z hlediska kvality sítě se nachází mezi spojnicí středů dvou sousedních kontrolních objemů a vektorem kolmým na jejich rozhraní. Pokud je tento úhel roven nule, jak je znázorněno na obr. 12 vlevo, pak je síť optimální. Nejhorší případ, který může pro kvalitu sítě nastat, je, pokud by se tento úhel blížil 90°. Na obr. 12 vpravo je znázorněna síť snížené kvality, kdy je tento úhel roven 45°. Velké úhly se nepreferují, jelikož způsobují nestabilitu numerických výpočtů. Doporučená maximální hodnota úhlu, kterou je pro konvergenci dobré zachovat, je 70°. Výpočty budou pravděpodobně divergovat s úhlem větším než 85°. [10]



Obr. 12 Aspekt pravoúhlosti [10]

2.2.4 Objemový poměr

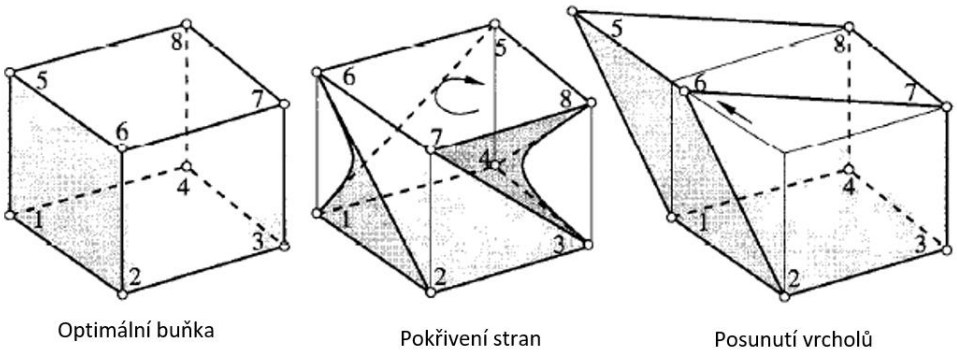
Objemový poměr je poměr objemů dvou sousedních kontrolních objemů. Ideální objemový poměr je roven jedné. Čím je tento poměr blíže jedné, tím je síť kvalitnější. Na obr. 13 je ukázka případu, kdy je větší objemový poměr sousedních kontrolních objemů jasně viditelný a tím se kvalita sítě snižuje. [10]



Obr. 13 Aspekt objemového poměru [10]

2.2.5 Zakřivení kontrolních objemů

Tento aspekt hodnotí různé pokřivení tvaru kontrolních objemů. Ideální kontrolní objem by měl mít stěny i hrany nepokřivené, dle základních tvarů kontrolních objemů vyjmenovaných např. v kapitole 2.1.1 této práce. Různé aspekty pokřivení jsou znázorněny na obr. 14., kde vlevo je optimální kontrolní objem, dále vpravo je snížená kvalita sítě v důsledku pokřivení stran (uprostřed) a v posledním případě v důsledku posunutí vrcholů. [2]



Obr. 14 Aspekt pokřivení kontrolních objemů [2]

Existuje i řada vzorců pro výpočet zakřivení pro různé typy kontrolních objemů. Pro příklad jsou zde uvedeny rovnice pro ty základní. Na obr. 15 je označen úhel, se kterým se zakřivení počítá. [10]

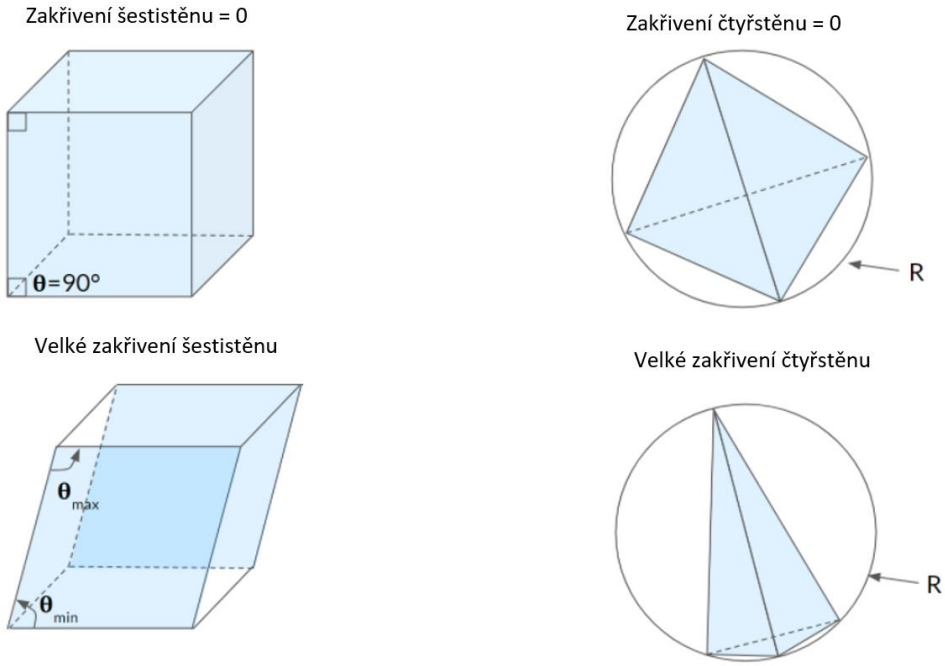
Rovnice výpočtu zakřivení pro šestistěnné kontrolní objemy:

$$Zakřivení = \max\left(\frac{\theta_{max} - 90}{180 - 90}; \frac{90 - \theta_{min}}{90}\right) \quad (3)$$

Rovnice výpočtu zakřivení pro čtyřstěnné kontrolní objemy:

$$V_{ideal.} = \frac{8 \cdot \sqrt{3} \cdot R^3}{27} \quad (4)$$

$$Zakřivení = \frac{\mathrm{V}_{ideal.} - V_{buňky}}{V_{ideal.}} \quad (5)$$



Obr. 15 Úhel a poloměr křivosti kontrolních objemů [10]

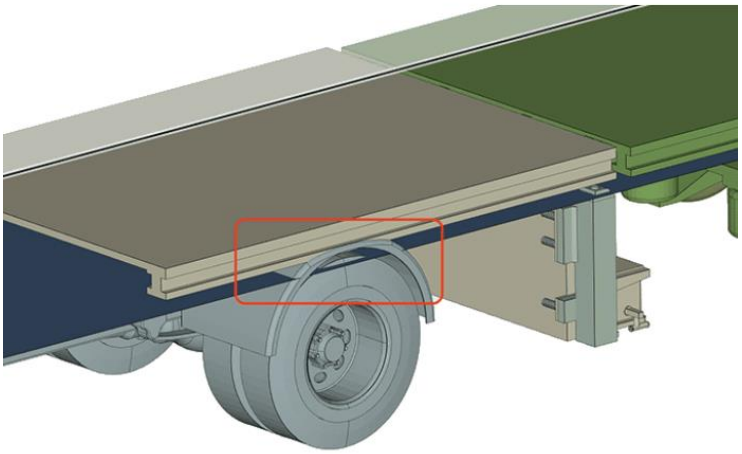
2.3 Optimalizace sítě

V minulé kapitole se vyhodnocovaly konkrétní aspekty kvalitní sítě převážně v požadavcích na velikost a tvar jednoho nebo dvou sousedních kontrolních objemů. Tato kapitola se bude zabývat globální optimalizací sítě v rámci celé geometrie a obecnými postupy, které vedou k přiblížení se optimální kvalitě sítě.

2.3.1 Tvar geometrie

Daná geometrie by měla být co nejjednodušší a měla by být uzavřená. Jednotlivé části geometrie by se neměly navzájem protínat, což umožní pro řešič rozlišovat mezi různými oblastmi toku. To je velmi důležité zejména pro simulace vnějšího toku. Na obr. 16 je možné vidět příklad takové

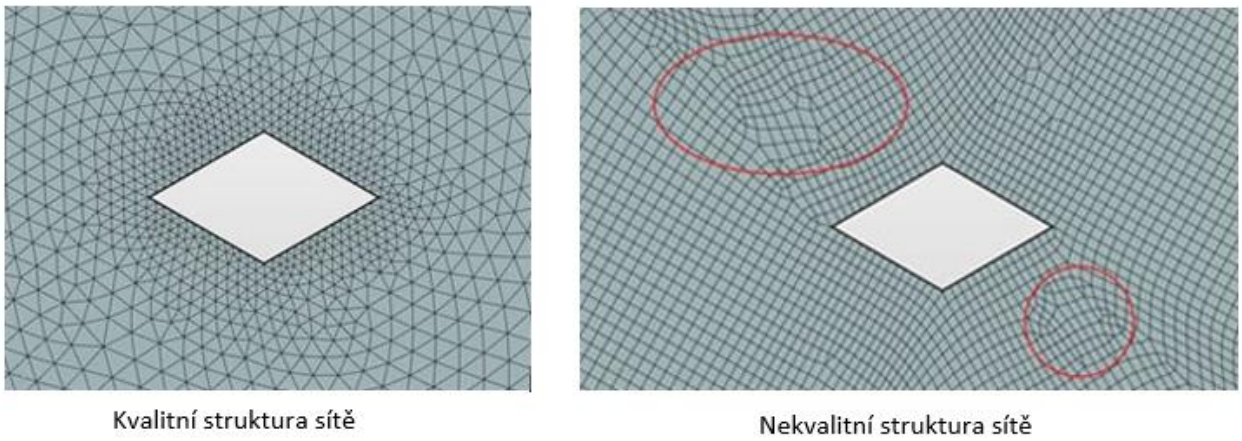
geometrie, která je příliš komplexní, protíná se a vytvořená síť bude v tomto případě nekvalitní. Pokud je to možné, mělo by se také vyhýbat ostrým rohům. [11]



Obr. 16 Nekvalitní protínající se geometrie [11]

2.3.2 Rovnoměrná struktura sítě

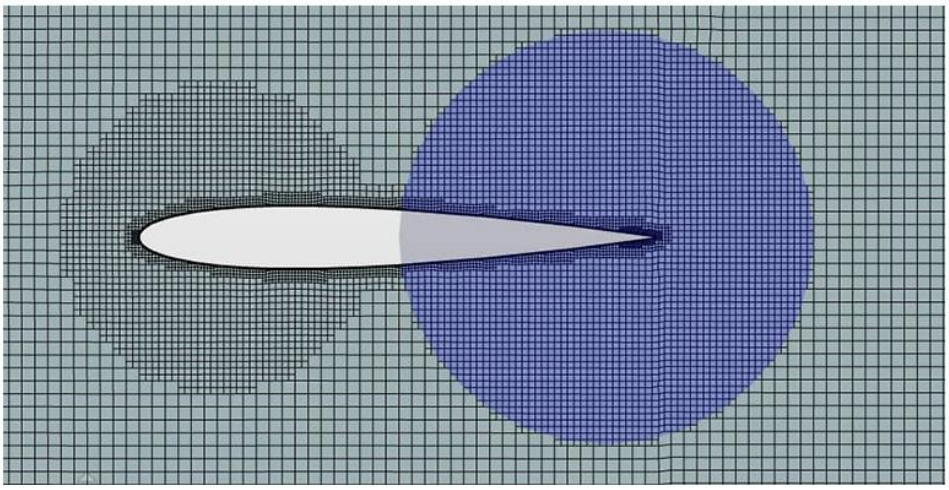
Na následujícím obr. 17 lze pouhým pohledem rozpoznat, která ze dvou typů sítí je kvalitnější. V mnoha případech lze takto posuzovat kvalitu po vytvoření sítě i vizuálně. Různé případy vyžadují různé poměry křivosti, ale mnohdy lze rozpoznat, kdy je míra zkřivení příliš velká. Existují také nástroje, které míru zakřivení mohou snížit definováním celkové velikosti všech kontrolních objemů. Nástroje tvořící sítě pro CFD výpočty budou rozebrány v dalších kapitolách. Jak lze z obr. 17 vidět, záleží také na volbě tvaru kontrolního objemu. [11]



Obr. 17 Rovnoměrné uspořádání a zakřivení kontrolních objemů [11]

2.3.3 Jemnost sítě v kritických oblastech

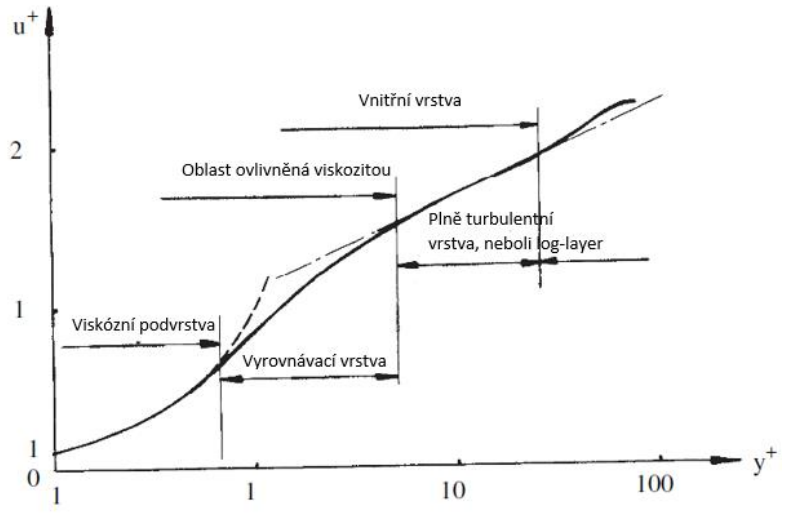
V některých konkrétních místech geometrie se může vyžadovat vyšší přesnost. Například je tomu tak v případech, kdy je žádoucí znát chování proudění kolem určité překážky. Přesnost je omezena, jak už bylo zmiňováno, vhodnou diskretizací spojitého prostředí. Vyjma aspektů optimální sítě je stále hlavním měřítkem přesnosti jemnost sítě. Jelikož jemnost zvyšuje požadavky na výkon a z toho důvodu si nelze dovolit mít celou geometrii pokrytou jemnou sítí, tak se síť zjemňuje pouze na konkrétních místech, kde je vyšší přesnost vyžadována. To je ukázáno na obr. 18. [11]



Obr. 18 Zjemnění sítě v kritických oblastech [11]

2.3.4 Tvorba sítě u hranice domény

Aby bylo možné zachytit vliv stěn geometrie na chování proudění, je nutno správně stanovit výšku vrstev kontrolních objemů hraničících s okraji. V blízkosti stěn, kde je proudění téměř laminární, hraje největší roli pro hybnost, přenos tepla a hmoty viskozita. Ve vzdálenějších vrstvách zase nejvíce záleží na turbulenci. Mezi těmito regiony je vliv obou faktorů stejně významný. Obr. 19 znázorňuje graf popisující jednotlivé vrstvy poblíž oblasti stěn. [3]



Obr. 19 Vrstvy ovlivněné faktory proudění u hranice domény [3]

Parametr y^+ na vodorovné ose na obr. 19 je bezrozměrná vzdálenost od stěny (dimensionless wall distance) daná vztahem:

$$y^+ = \frac{y \cdot u_t}{\nu} \tag{6}$$

Proměnná u_t je třecí rychlost, y je absolutní vzdálenost středu kontrolního objemu od stěny a ν je kinematická viskozita.

Na svislé ose na obr. 19 je u^+, což je taktéž bezrozměrná veličina reprezentující rychlost a je počítána vztahem:

$$u^+ = \frac{u}{u_t} \tag{7}$$

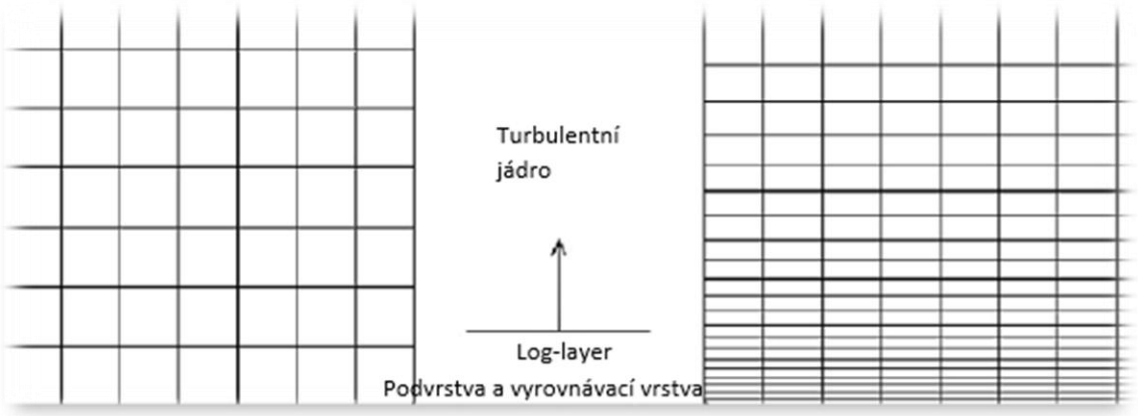
kde u je rychlost daná rychlostním polem. Třecí rychlost pak je určována pomocí:

$$u_t = \sqrt{\frac{\tau_w}{\rho}} \tag{8}$$

přičemž τ_w je smykové napětí a ρ je hustota proudícího média.

Veličina y^+ obvykle popisuje, v jaké vzdálenosti by jednotlivé kontrolní objemy na sebe měly být vrstvené a je závislá na typu proudění. Existuje i další způsob, jak modelovat proudění v blízkosti stěn. U druhého způsobu viskózní vrstva ani vyrovnávací vrstva není řešena, ale jsou použity „stěnové funkce", které vymezují viskózní napětí se stěnou v plně turbulentním regionu. Tento přístup se nazývá „high-Reynolds turbulence modeling". Oba přístupy jsou uvedeny na obr. 20.

Případ sítě vlevo je používán zejména v průmyslových procesech, jelikož nemá vysoké výkonnostní požadavky. Lze ho ale použít jen pro určité typy proudění, pro které existují funkce, jejichž předpoklady pro chování proudění u stěn korespondují s daným typem proudění. Parametr y^+ však u tohoto typu nemusí dosahovat takových hodnot, které znamenají vysoce zahuštěné vrstvy na okrajích stěn. U případu vpravo se mezní vrstva modeluje přímo s využitím dostatečně jemné sítě. [3]



Obr. 20 Metody vrstvení kontrolních objemů u hranice domény [3]

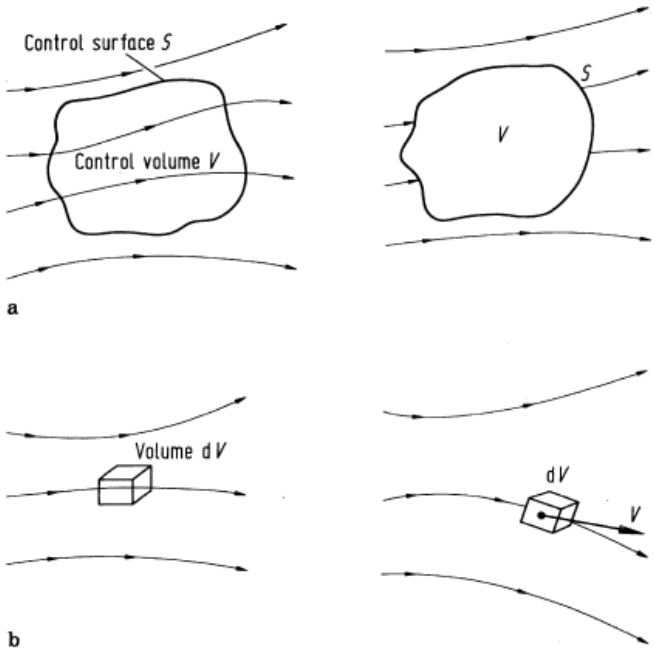
2.3.5 Studium konvergence

Studium konvergence začíná s vygenerováním sítě o vyšší hrubosti. V dalších krocích se daná síť neustále zjemňuje. Chyby způsobené diskretizací se tak postupně snižují. Cílem je dosažení takové jemnosti sítě, jejíž hodnoty nepřesnosti řešení budou nižší než tolerance chyby řešení dané uživatelem. Tedy do doby, kdy uživatelem požadovaná přesnost je dosažena a další zjemňování by pouze zvyšovalo nároky na výkon a čas výpočtu. Tímto způsobem se tak dosahuje optimální jemnosti sítě. To je důležité zejména pro rozsáhlé a komplexní sítě, kde výpočetní časy mohou dosahovat velmi vysokých hodnot. [11]

2.4 Řídící rovnice proudění

Aby mohly být představeny metody, které diskretizují řídící rovnice proudění, bude se tato kapitola krátce věnovat základním rovnicím popisující proudění.

Formulace rovnic se v základu dělí na dvě formy. První, konzervativní forma, aplikuje fyzikální principy a popis rovnic proudění na kontrolní objem, který je fixovaný v prostoru. Proudící médium vstupuje do jeho prostoru skrze jeho hranice a následně vystupuje. Druhá, nekonzervativní forma, aplikuje zákony fyziky na kontrolní objem, který se pohybuje společně s proudem. Obě formulace jsou znázorněny v obr. 21. [3]



Obr. 21 a) Konzervativní forma b) Nekonzervativní forma [3]

2.4.1 Rovnice kontinuity

V celém proudovém poli musí být splněn zákon o zachování hmoty. Kapalina nikde nevzniká ani nezaniká. Její matematická formulace může být vyjádřena vztahem: (9)

$$\frac{\partial \rho}{\partial t} + div(\rho \cdot \vec{v}) = 0 \tag{9}$$

kde druhý člen rovnice představuje zřídlovost vektoru rychlosti. Tato forma je v diferenciálním tvaru v nekonzervativní formě platná pro jakýkoliv bod v proudovém poli. [3]

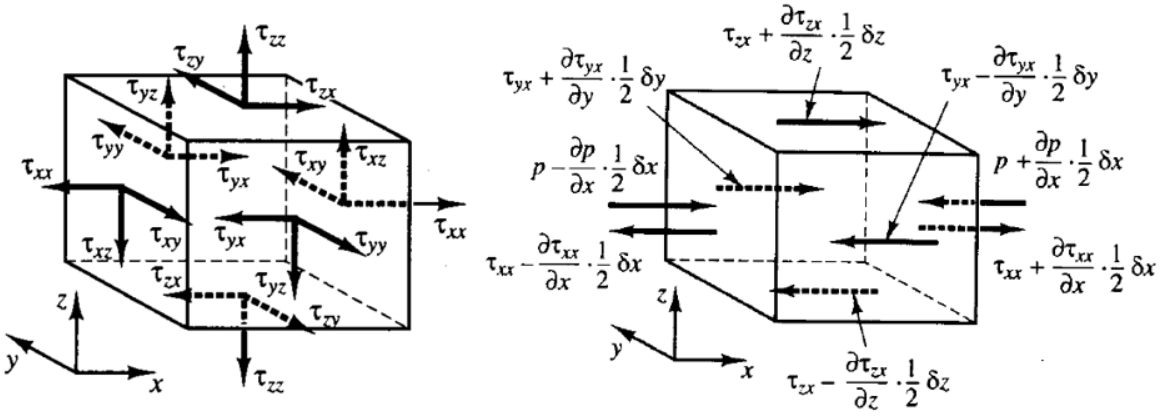
2.4.2 Zákon zachování hybnosti

Rovnice hybnosti, též označována jako Navier–Stokesova, používá druhý Newtonův zákon pro popis rovnováhy sil působících na částici kapaliny v závislosti na změně její hybnosti. Rozlišují se dva typy sil. Povrchové síly (způsobené tlakem a viskozitou) a objemové síly působící na těleso (gravitační, odstředivé, elektromagnetické a Coriolisova síla). Na stěny částice působí tlak označený jako „p" a třecí síly způsobené vizkozitou označované jako „τ". Jejich směry, index a znaménka jsou znázorněny na obr. 22 a řídí se dle směru daného souřadnicového systému. Každý tenzor napětí τ_{ij} je vztažen na danou částici $\partial(x, y, z)$. Objem částice je pak $\delta x \delta y \delta z$. [4]

V těžišti elementu působí vnější objemové zrychlení popsané vztahem:

$$\vec{A} = \vec{\imath} \cdot Ax + \vec{\jmath} \cdot Ay + \vec{k} \cdot Az \tag{9}$$

kde $\vec{A}$ je objemové zrychlení a $Ax, Ay\ a\ Az$ pak jeho jednotlivé složky působící v směrech souřadnicového systému.



Obr. 22 Napětí sil působících na částici tekutiny [4]

Obecná rovnice hybnosti po aplikování silové rovnováhy a odvození může být napsána ve tvaru: [12]

$$\frac{\partial \vec{v}}{\partial t} + \vec{v} \cdot grad\vec{v} = \vec{A} - \frac{1}{\rho} \cdot grad\ p + \nu \cdot \Delta\vec{v} \tag{10}$$

kde $\frac{\partial \vec{v}}{\partial t}$ je lokální zrychlení, $\vec{v} \cdot grad\vec{v}$ je konvektivní zrychlení, $\frac{1}{\rho} \cdot grad\ p$ je zrychlení způsobené tlakovým spádem a $\mu \cdot \Delta\vec{v}$ je zrychlení potřebné k překonání vizkózního tření tekutiny. Tento vztah platí pouze pro nestlačitelné kapaliny.

2.4.3 Zákon zachování energie

Rovnice energie je odvozena z prvního zákona termodynamiky, který stanovuje míru změny energie částice tekutiny rovnu míře jí přidaného tepla a práci na ní vykonané. Obecně je tato změna její energie v čase dána vztahem: [4]

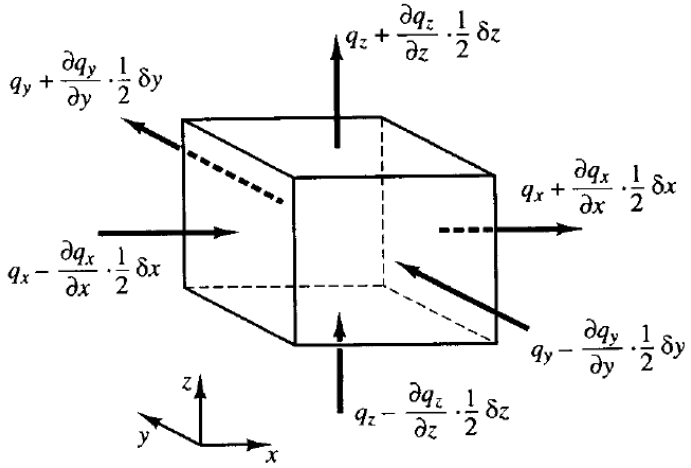
$$\rho \frac{DE}{Dt} \tag{11}$$

Energii částice lze dále rozvést jako součet veškeré vnitřní tepelné energie „i", kinetické energie a potenciální gravitační energie. Tato definice počítá s částicí, která tuto potenciální gravitační energii obsahuje. Je ale též možné brát tuto energii jako sílu konající na tuto částici práci při pohybu v gravitačním poli. Tento vliv bude v rovnici zahrnut ve zdrojovém členu „S_E".

Rovnice by se tedy mohla napsat ve tvaru:

$$\rho\frac{DE}{Dt} = -div(p\cdot u) + \left[\frac{\partial(u\tau_{xx})}{\partial x} + \frac{\partial(u\tau_{yx})}{\partial y} + \frac{\partial(u\tau_{zx})}{\partial z} + \frac{\partial(u\tau_{xy})}{\partial x} + \frac{\partial(u\tau_{yy})}{\partial y} + \frac{\partial(u\tau_{zy})}{\partial z} + \frac{\partial(u\tau_{xz})}{\partial x} + \frac{\partial(u\tau_{yz})}{\partial y} + \frac{\partial(u\tau_{zz})}{\partial z}\right] + div(k\cdot grad\ T) + S_E \quad (12)$$

kde člen $div(k\cdot grad\ T)$ zde představuje míru přidané tepelné energie v důsledku kondukce přes hranice částice tekutiny. Tok energie v důsledku kondukce je znázorněn na obr. 23.



Obr. 23 Tepelný tok energie v důsledku kondukce [4]

Zbylý člen $-div(p\cdot u) + \left[\frac{\partial(u\tau_{xx})}{\partial x} + \frac{\partial(u\tau_{yx})}{\partial y} + \frac{\partial(u\tau_{zx})}{\partial z} + \frac{\partial(u\tau_{xy})}{\partial x} + \frac{\partial(u\tau_{yy})}{\partial y} + \frac{\partial(u\tau_{zy})}{\partial z} + \frac{\partial(u\tau_{xz})}{\partial x} + \frac{\partial(u\tau_{yz})}{\partial y} + \frac{\partial(u\tau_{zz})}{\partial z}\right]$ představuje veškerou práci vykonanou na částici vlivem povrchových napětí. [4]

2.5 Diskretizační metody pro řešení rovnic proudění

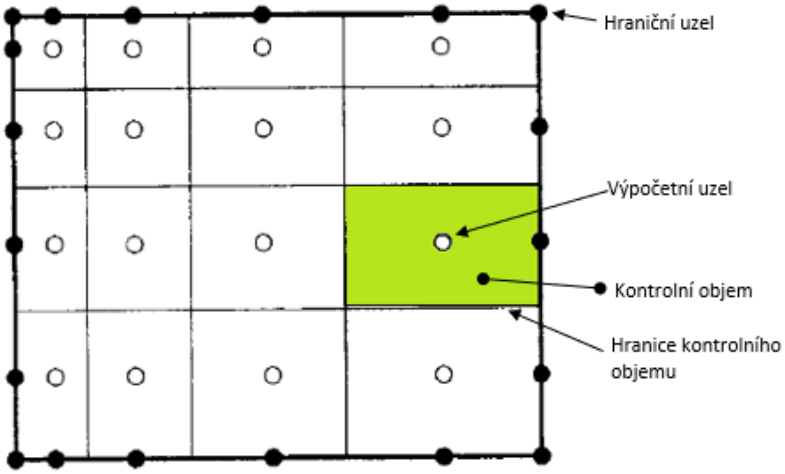
Metody řešení se liší případ od případu. Odvíjí se od typu sítě a charakteru proudění. Existují tři metody pro diskretizaci diferenciálních rovnic, které tato kapitola stručně popíše.

2.5.1 Metoda konečných objemů

Tato metoda patří v CFD mezi ty nejpoužívanější. Celá geometrie je rozdělena na konečný počet malých kontrolních objemů. Každý kontrolní objem obsahuje ve svém centru uzel, ke kterému se proměnné počítají. Řídící rovnice proudění se aplikují v integrální formě na povrchy kontrolního

objemu a interpolačně se vztahují k uzlu v jeho středu, který výpočtově reprezentuje celý kontrolní objem. Jednotlivé pojmy jsou znázorněny na obr. 24. [13]

Tato metoda je aplikovatelná na jakýkoliv typ sítě, takže je vhodná i pro komplexní geometrie. Síť definuje pouze hranice kontrolních. Nevýhoda této metody je ve složitosti aproximace rovnic nutné k interpolaci hodnot, diferenciaci a integraci. [2]



Obr. 24 Metoda konečných objemů [2]

2.5.2 Metoda konečných diferencí

Jedná se o nejstarší metodu, jejíž použití je nejjednodušší pro jednoduché geometrie. Byla představena Eulerem v osmnáctém století.

Metoda používá konzervativní rovnice proudění v diferenciálním tvaru. Celá doména je pokryta sítí uzlů. Pro každý uzel je následně diferenciální rovnice aproximována nahrazením parciálních derivací aproximacemi z hlediska uzlových hodnot funkce. Výsledkem pak je pro každý uzel jedna algebraická rovnice, ve které se objevuje hodnota proměnné a několik sousedních uzlů jako neznámé.

Principiálně může být tato metoda aplikovaná na jakýkoliv typ sítě. Je však nejvíce používaná pro strukturované sítě, kde její linie slouží jako souřadnicové čáry. Pro tento typ sítě je tato metoda velmi snadná a efektivní. Její omezení je u komplexního proudění. [2]

2.5.3 Metoda konečných prvků

Tato metoda je podobná metodě konečných objemů, kdy se geometrie rozdělí na kontrolní objemy. Obvykle se v 2D používá trojúhelníkových tvarů kontrolních objemů, zatímco v 3D čtyřstěnů nebo šestistěnů. Rozdíl spočívá v roznásobení rovnic váhovou funkcí (weight function). Jde o funkci, kdy se prvkům přiřadí určitá váha. V nejjednodušší formě je metoda v rámci každého kontrolního objemu aproximována ve tvaru lineární funkce tak, aby byla zajištěna kontinuita řešení přes jejich hranice. Takovou funkci lze sestavit z jejich hodnot v jednotlivých uzlech kontrolních objemů.

Tato aproximace je poté dosazena do vážené integrální formy (nebo váženého integrálu) zákonu zachování. Rovnice k řešení jsou odvozeny tak, že se požaduje, aby derivace integrálu vztažená ke každé hodnotě v uzlu byla nulová. To koresponduje s vybráním nejlepšího řešení z množiny možných funkci (to s minimálním residuem). Výsledkem je množina nelineárních algebraických rovnic. Výhodou této metody je schopnost řešit libovolné tvary geometrií. [2]

3 Nástroje pro tvorbu sítí

Softwarů pro diskretizaci geometrie, tedy tvorbu sítě v oblasti CFD preprocesingu je celá řada. Asi nejvýraznější rozdíl mezi nimi udává cena. Existují komplexní a plně vybavené softwary vyvinuté specializovanými společnostmi na CFD a open-source softwary, nabízející do jisté míry konkurenceschopné síťování zdarma. Tato kapitola se jim bude odděleně věnovat a popíše, co které softwary aktuálně nabízí.

3.1 Komerční softwary pro tvorbu sítí

Standardem těchto komerčních programů je obvykle možnost importování geometrie daného modelu ze známých CAD softwarů. Generování sítě je poměrně snadné, avšak v pozadí se skrývají poměrně složité procesy. Tvorba optimální sítě tedy od uživatele vyžaduje určitou míru odbornosti a znalosti těchto procesů. Cena školení a samotný software je tedy ve výsledku nabízen za poměrně vysokou cenu. Avšak výsledky těchto CFD softwarů dosahují velmi vysoké kvality a v složitosti dnešních technologií se většinou větším firmám takový komplexní nástroj pro simulace a modeling vyplatí.

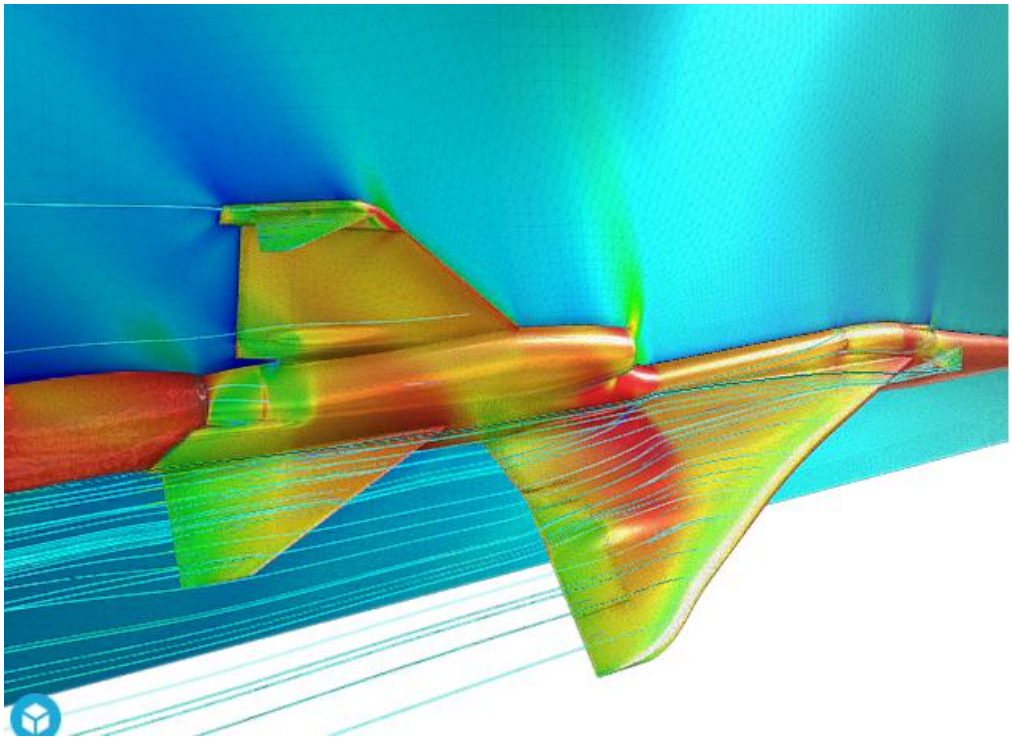
3.1.1 Ansys ICEM CFD

Tento software je produktem od firmy Ansys a vytváří síť pro další procesing ve Fluentu, což je nástroj od téže firmy pro řešení nestacionárních Navier–Stokesových rovnic metodou konečných objemů. ICEM umožňuje načtení a opravu CAD modelů a tak následnou tvorbu kvalitní výpočtové sítě s minimální námahou. Zároveň vytváří kvalitní hexaedrální sítě pomocí blokových schémat a obsahuje automatizované a interaktivní nástroje pro jejich úpravu. Tyto nástroje vytvářejí uživatelsky přívětivé prostředí pro rychlou tvorbu a práci i se složitými geometriemi. [14]

Další pozitivní vlastností u tohoto nástroje je parametrizovatelnost blokových schémat, která jsou zároveň skriptovatelná. To dává prostor pro vysokou variabilitu testování daných modelů. Nevýhodou je jeho cena, která se s celým balíčkem od Ansysu pro CFD modeling může pohybovat okolo 60 000 USD. Na obr. 25 je uveden příklad 3D modelu letadla z programu ICEM, na kterém je vidět komplexnost a možnosti Ansysu ICEM. [15]

Souhrn vlastností softwaru ICEM:

- import a oprava CAD modelů, rychlá a nenáročná tvorba sítě
- komplexní tvroba hexaedrálních sítí (i rychlá tvorba sítí jiných typů) pomocí blokových schémat
- parametrizovatelnost a skriptovatelnost blokových schémat
- nástroje pro diagnostiku a úpravu sítí
- výstup pro širokou škálu CFD řešičů v různých formátech.



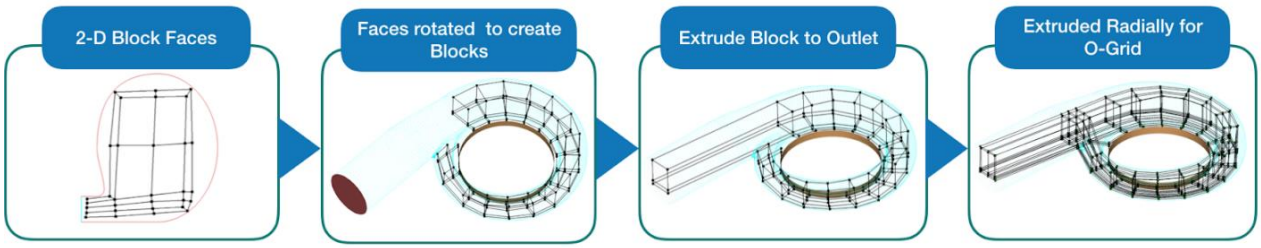
Obr. 25 Ansys ICEM model letadla [16]

3.1.2 GridPro

Tento software je od společnosti Program Development Company a řadí se mezi komerční. Jeho cena je podobně jako u Ansysu vyšší. Při generování sítě je používán topologický přístup, který studuje vlastnosti geometrie, jenž se při spojitých transformacích nemění. [18]

Geometrický model lze importovat, jelikož GridPro spolupracuje s běžnými CAD programy. Stejně jako ICEM používá pro tvorbu sítě blokových schémat. Oproti ICEM nabízí GridPro jemnější strukturu sítě, avšak zkušenosti uživatelů říkají, že cena za vyšší kvalitu sítě je jejich náročnější tvorba. Také nemusí na všechny geometrie fungovat stejně dobře a proto uživatelé doporučují si tvorbu sítě nejprve otestovat na vlastní geometrii. [17]

Pokud je vstup CAD model, GridPro obsahuje nástroje, které můžou vytáhnout či rotovat 2D plochy do prostoru pro tvorbu blokových schémat. Tento postup je popsán na obr. 26. [19]

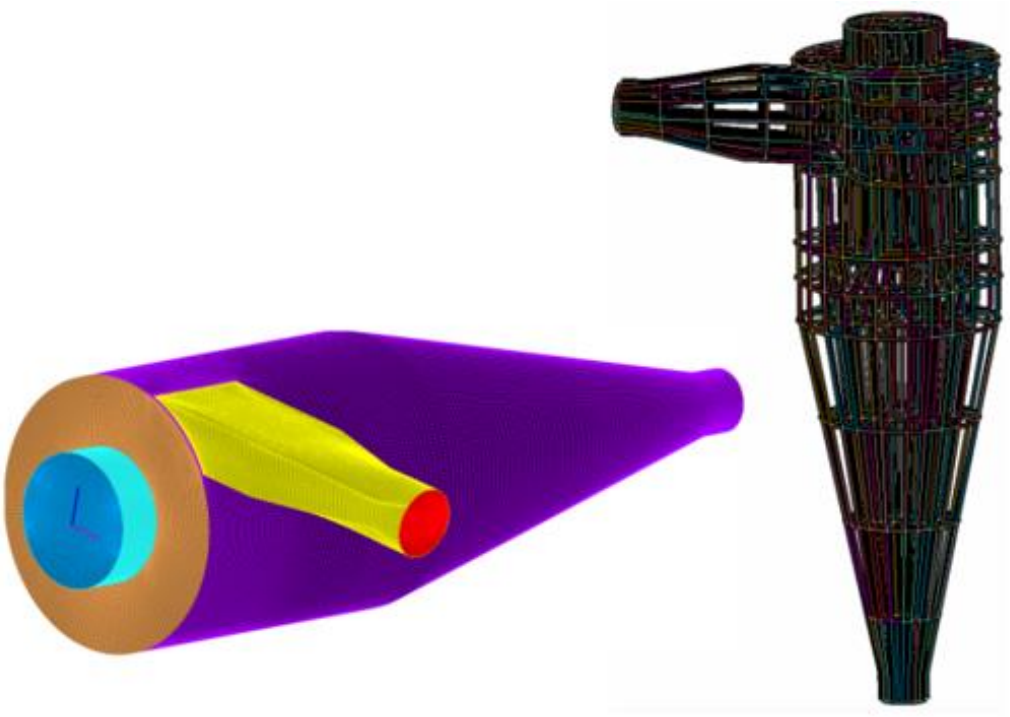


Obr. 26 Tvorba blokových schémat v GridPro [19]

Souhrn vlastností softwaru GridPro:

- jednoduchá tvorba sítě z CAD modelů pomocí blokových schémat nebo vlastních nástrojů
- parametrizovatelnost pro rychlé změny modelu
- účinný topologický nástroj pro tvorbu sítě
- při pečlivější práci velmi jemná a kvalitní síť
- nástroje pro úpravu a kontrolu kvality sítí

Nástroj GridPro má široké praktické uplatnění. Je používán v oblastech leteckého, chemického a automobilní průmyslu, také v kosmonautice, turbozařízeních nebo petrochemickém průmyslu. Na obr. 27 je uveden příklad modelu cyklónového separátoru a jeho bloková struktura. [18]



Obr. 27 Cyklónový separátor v GridPro [18]

3.1.3 Souhrn vlastností dalších komerčních softwarů

Mezi komerční softwary s podobnými funkcemi jako ICEM nebo GridPro se řadí např. Pointwise, Hypermesh, Ansa a Centauri. Tato kapitola stručně popíše hlavní charakteristiky těch nejznámějších.

Pointwise [20]

- silné exportní a importní síťové nástroje s předem vybudovanými vazbami na další řešič se schopností integrace do téměř jakéhokoliv CFD workflow
- schopnost generovat samotný geometrický model nebo importovat CAD modely
- moderní nástroj zapojený do řady CFD projektů
- řada užitečných nástrojů umožňující testování

- rychlá tvorba strukturovaných i nestrukturovaných sítí velmi dobré kvality v uživatelsky přívětivém rozhraní
- výstup v různých formátech pro téměř jakýkoliv řešič
- možnost definování okrajových podmínek pro zajištění maximální integrity během převádějícího procesu na vhodný formát pro vybraný řešič

Hypermesh [21]

- použití metody konečných prvků
- obsahuje pokročilé nástroje pro práci s různými materiály včetně laminátových kompozitů
- vysoká kvalita automaticky generované sítě na CAD modelech
- možnost parametrizace modelu
- práce se složitými kompozitními strukturami

3.2 Open-source softwary

Open-source softwary pro tvorbu sítí umožňují vytvoření sítě pro CFD řešič zdarma. Nevýhodou oproti placeným programům je obvykle zvýšená náročnost tvorby sítě, chybějící funkce pro kontrolu a editaci sítě nebo import z CAD modelů. Nemají tak pokročilé funkce jako komerční programy a to často pro úpravu a parametrizaci sítí vede k nutnosti skriptování. Samotné skriptování však může být pro některé typy geometrií velkou výhodou, jelikož umožňuje snadnou a rychlou reprodukovatelnost a parametrizovatelnost modelu při zachování stejné kvality sítě.

Nevýhodou pro daný software může také být nedostatek studijních materiálů. Používají se většinou pro akademické nebo čistě uživatelské jednodušší geometrie. V této kapitole bude uveden přehled a popis některých z nich.

OpenFOAM jako open-source software, pro který byla zpracována geometrie tepelného výměníku jako praktické zadání této práce, obsahuje dva vlastní síťové generátory. BlockMesh a snappyHexMesh.

3.2.1 Gmsh

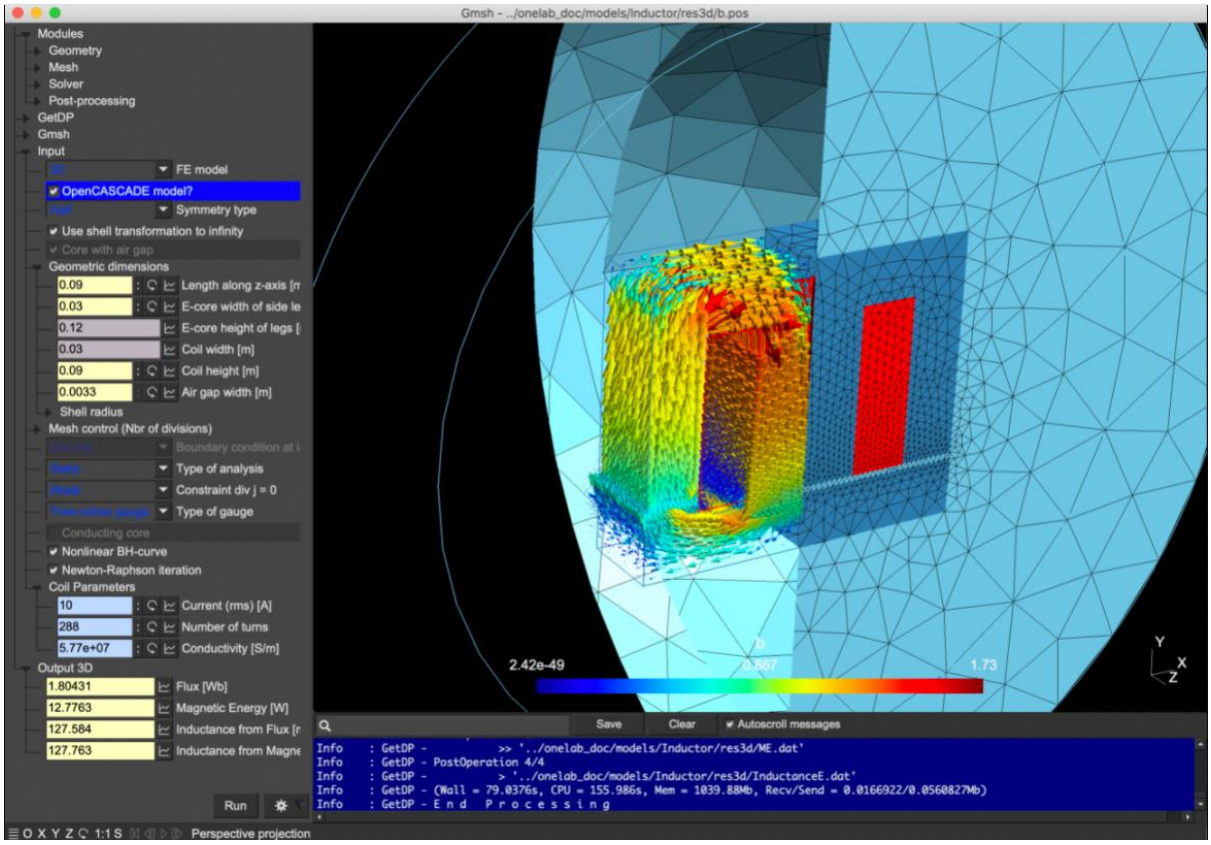
Gmsh je 3D generátor sítě pracující s metodou konečných prvků, která není po hardwarové stránce tolik náročná a umožňuje tak rychlou práci se složitějšími geometriemi i na běžném počítači.

Má v sobě zabudovaný CAD nástroj a postprocesor, takže umožňuje jak tvorbu geometrie, tak i sítě s následnou možností výpočtu a postprocesingu. Jeho účelem je poskytnout rychlé a snadné řešení s velmi rychle si pro uživatele osvojujícími nástroji. Rovněž umožňuje parametrický vstup a obsahuje pokročilé vizualizační možnosti. Uživatel může tvořit geometrii jak s pomocí grafického rozhraní, tak i s pomocí skriptovacího jazyka, který je zdarma. Tento software je určen primárně pro akademické účely. [22]

Jednou z jeho nevýhod je nutnost skriptování, jelikož možnosti grafického rozhraní jsou omezené a omezená je k němu podle uživatelských zkušeností i dokumentace. Na obr. 28 je ukázka jeho uživatelského rozhraní znázorňující analýzu řešené simulace. [23]

Souhrn vlastností softwaru Gmsh:

- použití metody konečných prvků – rychlá a nenáročná tvorba sítě
- CAD nástroj pro tvorbu geometrie
- možnost skriptování
- parametrizace vstupu
- uživatelsky přívětivé nástroje
- primárně pro akademické účely v rámci jednoduchých modelů.



Obr. 28 Gmsh uživatelské rozhraní – analýza [22]

3.2.2 EnGrid

Tento software pro tvorbu geometrie umožňuje spolupráci s programem Blender, což je program pro 3D modeling objektů. Jelikož EnGrid vytváří síť primárně z čtyřstěnů, umožňuje tak snadno vytvořit síť na libovolném modelu vytvořeném v Blendru, ale i u modelů ve STL formátu. Na druhou stranu kvalita sítě už nemusí být pro dosažení přesných výsledků dostatečná. Častým problémem je též vytvoření prizmatických vrstev na okrajích geometrie. Pro tvorbu těchto vrstev a kontrolních objemů používá enGrid knihovnu Netgen. Program umožňuje převod čtyřstěnných kontrolních objemů na mnohostěnné. Výsledek však nemusí být vždy optimální, jelikož může způsobovat křivost a zhoršení kvality kontrolních objemů.

EnGrid také vytváří výstupní síť pro program OpenFOAM, do kterého musí být exportována, jelikož není přímou součásti tohoto softwaru jako například blockMesh nebo snappyHexMesh.

EnGrid lze při tvorbě sítě s těmito aplikacemi kombinovat za účelem dosažení vyšší kvality sítě. [24, 25]

Souhrn vlastností softwaru EnGrid:

- spolupráce s grafickým softwarem Blender
- import STL souborů
- knihovna Netgen - tvorba čtyřstěných kontrolních objemů
- oproti jiným softwarům nižší kvalita sítě
- export souborů pro OpenFOAM, spolupráce s jeho vlastními síťovými generátory

3.2.3 Salome

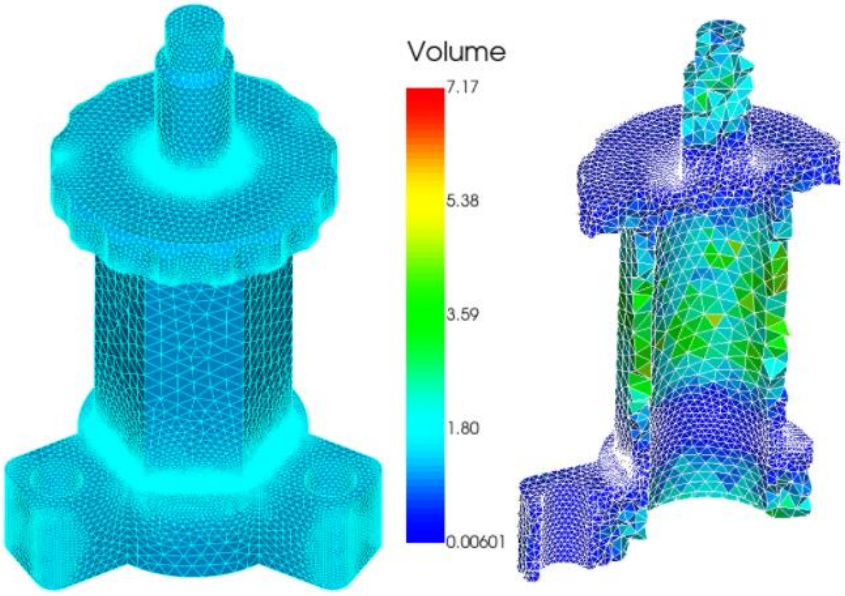
Salome používá širokou škálu algoritmů pro síťování geometrie obzvláště pro metody konečných prvků a objemů. Obsahuje nástroj pro přidání skupin nebo popisků pro jednotlivé oblasti geometrie, aby se případně mohla rozlišit jiná struktura sítě. Ty také slouží k usnadnění určování okrajových podmínek, vizualizace a následného postprocesingu.

Tvorba geometrie může být plně zpracována v rámci skriptování. Modul SMESH v Salome nabízí nástroje pro úpravu kvality sítě, které také umožňují úpravu modelu v rámci grafického rozhraní.

Tvar kontrolních objemů může být čtyř nebo šestistěnný, nabízí také tvorbu vrstev u okrajů geometrie. Obsahuje také nástroje od Netgen a Gmsh. Modul SMESH je doplněn o modul HOMARD, který provádí lokální úpravy sítě pro splnění požadavků na přesnost a výkon. Dovoluje zpřesňující operace pro přizpůsobení sítě v závislosti na numerických chybách simulace tak, aby byl zajištěn účinný kompromis mezi jemností sítě a nízkými výpočetními náklady. Na obr. 29 je možné vidět vytvořenou síť v programu Salome. [26]

Souhrn vlastností softwaru Salome:

- různé algoritmy pro tvorbu sítě
- použitá metoda konečných prvků a objemů
- možnost skriptování geometrie
- kvalitní vizualizace a nástroje pro editing sítě
- moduly pro optimalizace sítě.



Obr. 29 Vytvořená síť v programu Salome [26]

4 Stručný popis aplikace OpenFOAM

OpenFOAM je knihovna spustitelných aplikací pro CFD psaná v C++, což je objektově orientovaný programovací jazyk. OpenFOAM vytvořil v roce 1989 Henry Weller pod názvem FOAM a později, roku 2004 byl poprvé pod svým nynějším názvem zveřejněn. Nyní je vlastněn OpenFOAM Foundation a distribuován pod General Public License (GPL), zajišťující použití tohoto softwaru všem uživatelům zdarma a je každý rok aktualizován. [28]

OpenFOAM obsahuje širokou škálu řešičů pro řešení nejrůznějších typů úloh počínaje turbulentním prouděním, chemickými reakcemi, až po akustiku nebo elektromagnetismus.

Zajištění kvality je založeno na přísném testování. Proces vyhodnocování, ověřování a validace kódu zahrnuje několik stovek denních jednotkových testů. Testy jsou určeny k posouzení využití paměti, výkonu kódu a škálovatelnosti.

OpenFOAM lze instalovat na systémy Linux, Mac OS X i Windows, přičemž původní verze je vytvořena pro Linux Ubuntu. [29]

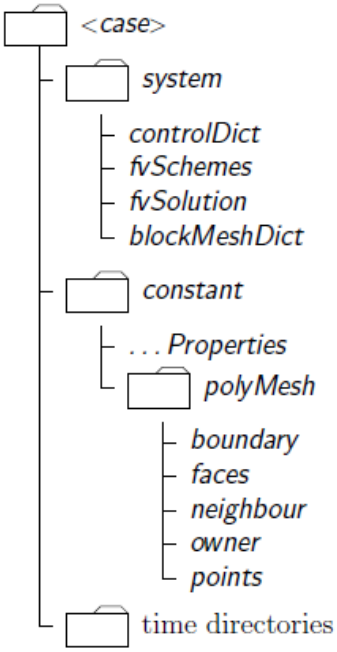
4.1 Struktura aplikací a adresářů OpenFOAM

Aplikace se v OpenFOAM rozdělují na dvě kategorie. Do první kategorie spadají řešiče, které jsou navrženy pro řešení problémů mechaniky kontinua a nevyžadují už žádnou manipulaci s daty a zásah uživatele. Druhé jsou utility, které jsou navrženy k vykonávání úkolů a manipulaci s daty definovanými uživatelem. Utility spadají z větší části do oblasti pre-procesingu jako je například tvorba sítě a nastavení okrajových podmínek.

Pro post-procesing používá OpenFOAM paraFoam, což je skript, který načte Paraview. Paraview je vizualizační software zdarma, který umožňuje analýzu dat.

Všechny vstupy v adresářích pro vybrané aplikace jsou definovány v textových souborech a klíčová slova musí odpovídat definicím slovníku OpenFOAM, aby byly pro aplikace čitelné. Textové soubory jsou velkou výhodou z hlediska snadné čitelnosti a možných úprav. Soubor pro aplikaci blockMesh je obvykle tvořen pomocí skriptování. To vše umožňuje variabilitu a rychlou možnost změny vstupních dat pro výpočet.

Na obr. 30 je znázorněna struktura adresáře výpočtové úlohy (case) v OpenFOAM, jenž obsahuje minimální počet souborů nutných pro spuštění simulace. [30]



Obr. 30 Adresář OpenFOAM pro jeden case [30]

4.1.1 Adresář system

Tento adresář je určen k nastavení kritérií pro samotný výpočet. Obsahuje nejméně tři soubory, které jsou popsány v následujících kapitolách.

4.1.2 ControlDict

Slouží pro nastavení kontrolních parametrů výpočtu pro vytvoření databáze, která je zdrojem pro načítání vstupů a zápis výstupů. Každý výpočetní iterační krok probíhá s určitým časovým krokem, který má svůj počátek a konec. Tyto informace o časových údajích jsou součástí této databáze. Funkce timeFormat definuje formu zapisovaného času ve výstupních souborech. [30]

V této databázi se také nachází údaje o zapisování výstupních souborů výpočtu. Funkce writeControl určuje způsob zapisování časových intervalů. Například pomocí timeStep se data zapisují v každém nastaveném časovém intervalu. Pomocí runTime zase v každém intervalu v sekundách simulovaného času. Funkce purgeWrite svojí hodnotou udává počet časových adresářů, které se během výpočtu zapíší. WriteFormat pak definuje formát výstupních dat. [47]

ControlDict obsahuje i nastavení, které upravuje časový krok podle maximální hodnoty Courantova čísla, které se zde také definuje. Courantovo číslo je bezrozměrná veličina a v CFD indikuje, jak velkým poměrem jednoho kontrolního objemu proteče proud za jeden časový interval. Je dáno vztahem:

$$C = \frac{U \cdot \Delta t}{\Delta h} \tag{13}$$

kde proměnná U znamená rychlost proudění, Δt časový interval a Δh charakteristický rozměr kontrolního objemu. [46]

Výhodou je automatické čtení této databáze, které umožňuje změny časového kroku během výpočtu. [30]

4.1.3 FvSchemes

V tomto souboru jsou popsány všechny definice související s diskretizací a interpolací rovnic, které se objevují v aplikacích. Jejich definování se liší dle toho, jaký řešič se použije. Ten je definován v souboru fvSolution a je použit v rámci metody konečných objemů. Jednotlivá schémata rozřazená dle použití se v souboru řadí do podslovníků, kde je pomocí klíčových slov volí uživatel. Jejich základní rozdělení je popsáno níže. [30]

DdtSchemes jsou časová schémata, které obsahují derivace časově závislých členů rovnic. Obsahují speciální koeficient, díky čemuž mohou zvyšovat stabilitu výpočtu. Jako výchozí schéma je nastaveno Eulerovo schéma pro neustálený stav. [31]

GradSchemes jsou schémata, které počítají gradienty vyskytující se v rovnicích použité aplikace. [31]

DivSchemes jsou použity k diskretizaci divergenčních členů v rovnicích použité aplikace.

LaplacianSchemes jsou schémata pro diskretizaci Laplaceových operátorů. Jedinou možností pro diskretizaci těchto operátorů je Gaussovo schéma.

InterpolationSchemes jsou schémata pro interpolaci hodnot ze středů kontrolních objemů na jejich stěny.

snGradSchemes jsou schémata pro diskretizaci gradientu ve směru kolmém k hranici řešeného modelu. [30]

4.1.4 FvSolution

Soubor fvSolution obsahuje základní nastavení pro zvolený řešič, tolerance i kontrolní a pomocné algoritmy vztahující se k výpočtu. Nacházejí se zde také relaxační faktory, které jsou popsány níže. [30]

Řešiče

OpenFOAM neobsahuje obecný řešič aplikovatelný pro všechny typy úloh. Uživatel musí řešič dle typu dané úlohy specifikovat. Všechny typy řešičů jsou obsaženy v adresáři $FOAM_SOLVERS, který může být rychle spuštěn pomocí příkazu app napsaném v příkazovém řádku. Tento adresář je dále rozdělen dle jednotlivých typů úloh na například nestlačitelné proudění, přenos tepla, vícefázové toky nebo spalování. Každý řešič je specificky pojmenován. Jejich jména ukazují také na typ použitého algoritmu. Například simpleFOAM používá metodu SIMPLE. Často jejich názvy odrážejí fyzikální model nebo typ problému, pro který je daný řešič přímo navržen. Například cavitatingFOAM řeší úlohy s kavitací. V tabulce 4.1 je uveden stručný přehled používaných řešičů a jejich hlavní použití. [48]

Tab. 4.1 Řešiče a jejich použití [48]

Řešič	Nestacionární proudění	Stlačitelné kapaliny	Turbulentní proudění	Přenos tepla	Vztlaková síla	Spalování	Vícefázový tok
buoyantPimpleFoam	x	x	x	x	x		
chemFoam	x			x		x	
engineFoam	x	x	x	x		x	
icoFoam	x						
pimpleFoam	x		x				
pisoFoam	x		x				
rhoPimpleFoam	x	x	x	x			
simpleFoam			x				
reactingFoam	x	x	x	x		x	
reactingParcelFoam	x	x	x	x	x	x	

Zápis pro vstup pro vybraný řešič používá klíčových slov vztahujících se k proměnné, vyskytující se v rovnicích pro daný řešič. Pokud se v rovnicích pro vybraný řešič počítá rychlost a tlak, klíčové slova – nebo v tomto případě písmena – budou „U“ a „p“. Každá proměnná obsahuje i tolerance přesnosti řešení. [31]

SIMPLE a PISO algoritmy

Tyto algoritmy jsou použity pro většinu řešičů a iteračním způsobem řeší rovnice rychlostí a tlaku. PISO se používá pro neustálené úlohy a vyžaduje více jak jednu korekci, zatímco SIMPLE pro ustálený stav a pro dosažení přesných výsledků počítá jen s jednou korekcí rychlosti a tlaku. [31]

Tolerance

Tolerance zapsané v tomto souboru jako vstupní údaj určují požadovanou přesnost výpočtu. Výpočet je dokončen, pokud jsou po konečné iteraci škálovaná rezidua menší než tyto tolerance, nebo počet iterací překročí pevně stanovený limit.

Reziduum je velikosti rozdílu pravé a levé strany rovnic. Udává chybu řešení dané iterace a při konvergenci výpočtu se jeho hodnota s každou novou iterací zmenšuje. Čím je tedy tato hodnota menší, tím je výsledek přesnější. Tolerance musí být definovány pro každý řešič a maximální iterační limit je volitelný. [30]

Relaxace řešení

Používají se při výpočtu lineární kombinace řešení z předchozí a aktuální iterace za účelem zvýšení stability, příp. zrychlení konvergence. Podrelaxace, kdy je relaxační faktor menší než jedna, dává limitní hodnotu pro změnu dané proměnné. Činí tak například pomocí modifikace matice výpočtu. Tím zpomaluje rychlost konvergence, ale zároveň zvyšuje stabilitu výpočtu.

Relaxační faktor větší jak jedna se používá v případě, kdy je požadováno urychlení konvergence. V běžných případech se však preferuje stabilita výpočtu s co možná nejvyššími hodnotami relaxačního faktoru, tedy hodnotami blížící se jedné. Hodnoty menší jak 0,2 jsou považovány za neúnosně omezující. [30]

4.1.5 Adresář constant

Ve složce polyMesh se nachází celkový popis sítě dané geometrie. Další soubory ve složce constant obsahují informace o fyzikálních vlastnostech určených pro danou aplikaci. Tyto parametry definující výpočetní síť a další soubory s ní spojené generuje aplikace blockMesh.

Adresáře Time directories obsahují počáteční a okrajové podmínky pro daný problém a výstupní data zapsaná řešičem pro konkrétní výpočetní časy v rámci řešené úlohy. [30]

4.2 BlockMesh

BlockMesh je základní generátor pro tvorbu sítě v programu OpenFOAM. Vytváří blokově strukturovanou síť s možností zakřivení hran a nestejnoměrného dělení („grading").

Soubor obsahující popis bloků pro tvorbu sítě aplikací blockMesh má název blockMeshDict a je automaticky čten z adresáře system. BlockMesh vytvoří model i se sítí a zapíše data do souborů body, kontrolní objemy a hranice do stejného adresáře.

Uživatel tedy vytváří bloky dle dané geometrie, které popíše jednotlivými uzly, objemy a povrchy. V rámci jednotlivých bloků definuje počet kontrolních objemů, „grading" i možné zakřivení hran celého bloku. Při vytváření se musí držet několika základních pravidel popsaných v manuálu OpenFOAM a má na výběr použití libovolného skriptovacího jazyka, jelikož vstupem pro blockMesh je vždy textový soubor. [30]

Skriptování je výhodné z hlediska variability, možnosti parametrizace při zachování kvality sítě nebo kopírování již existujících částí modelu. Například pro úpravu počtu trubek a jejich řazení ve svazku tepelného výměníku je to rychlejší nástroj než CAD modelování. Zároveň již vytvořená síť kopírováním zůstává z hlediska kvality a struktury nezměněna.

4.2.1 Definování souboru pro tvorbu sítě

Jednotky

Definování jednotek se provádí pomocí příkazu scale. Pro scale rovno 0,001 platí, že veškeré uvedené hodnoty budou v tisícinách metru (základní jednotky SI), tedy v milimetrech.

Uzly

Uzly definují vrcholy jednotlivých bloků ve formě souřadnic a zapisují se do kulatých závorek na začátku souboru. Bloky jsou vždy šestistěnné s možností výjimky v případě rovnoběžného ztotožnění dvou hran se sousedními za vzniku pětistěnu s trojúhelníkovou podstavou. Až na tuto výjimku je jeden blok složen vždy z osmi uzlů.

Jednotlivé souřadnice uzlů mají své identifikační číslo dané pořadím, ve kterém byly zapsány. Pomocí tohoto čísla jsou následně přesněji určeny do parametrů pro bloky, stěny, hrany, vstupy a výstupy (pro proudění). Jelikož je OpenFOAM psán v jazyku C++, zachovává jeho konvenci s počátkem indexování v nule. Výpis souřadnic uzlů jednoho bloku je znázorněn na obr. 31. [32]

```
vertices
(
    (0 0 0)     vertex 0
    (1 0 0)     vertex 1
    (1 1 0)     vertex 2
    (0 1 0)     vertex 3
    (0 0 0.1)   vertex 4
    (1 0 0.1)   vertex 5
    (1 1 0.1)   vertex 6
    (0 1 0.1)   vertex 7
);
```

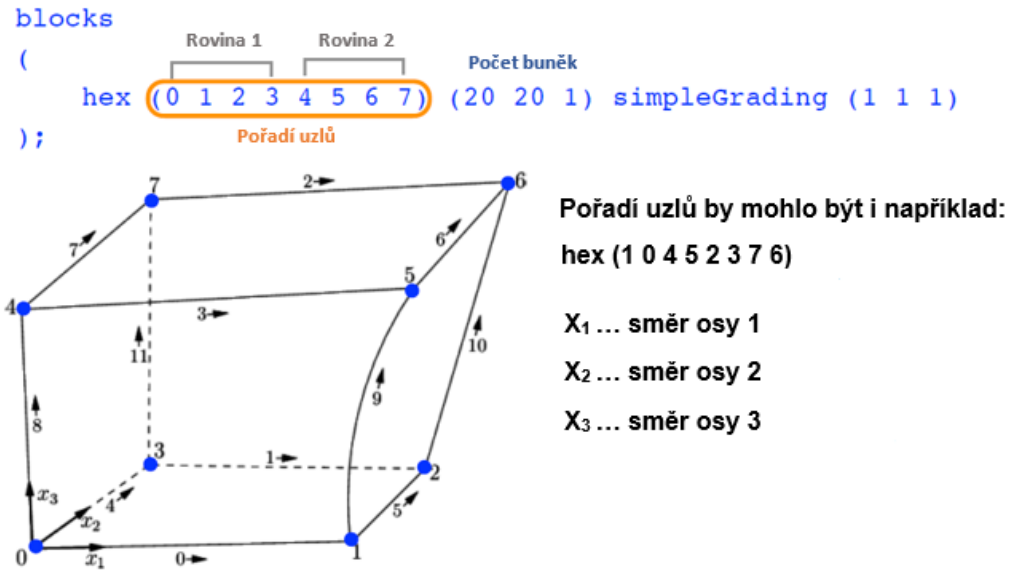
Obr. 31 Výpis souřadnic uzlů v blockMeshDict [32]

Objemy

Klíčové slovo pro definování jednoho bloku je v blockMesh „hex”. Za ním následuje závorka s vždy osmi identifikačními čísly uzlů v přesně definovaném pořadí. První čtyři uzly tvoří jednu rovinu a zbylé uzly rovinu druhou. Pravidlem pro jejich řazení je pravidlo pravé ruky tak, aby palec u první roviny směřoval dovnitř bloku. Druhá rovina je řazena dle první roviny ve stejném směru. [32]

Pořadím uzlů při řazení bloku je definován i jeho lokální souřadnicový systém. První osa je určena prvním (tedy s indexem nula) uzlem a druhým uzlem. Druhá osa je určena druhým a třetím. Poslední osa je pak určena prvním a pátým uzlem.

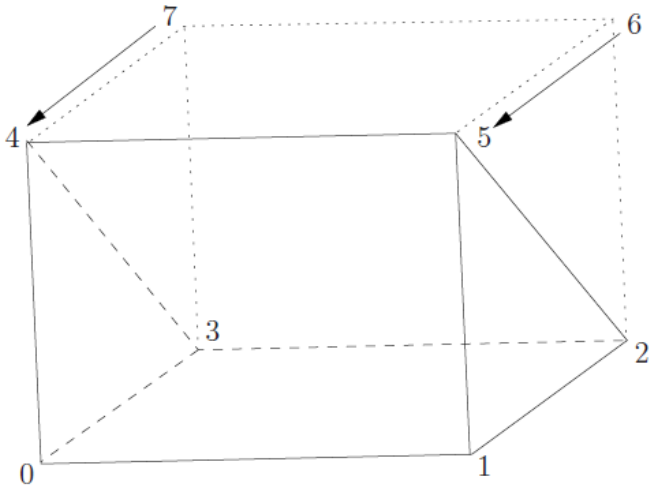
Za závorkou s pořadím uzlů následuje závorka s hodnotami pro počet kontrolních objemů v pořadí jednotlivých os a dál závorka pro funkce grading. Na obr. 32 je pro ukázku znázorněn popis jednoho bloku. [32]



Obr. 32 Tvorba bloku v blockMesh [32]

BlockMesh také umožňuje „zborcení" jedné hrany do protilehlé za účelem vytvoření pětistěnu s trojúhelníkovou podstavou. Tato změna je možná změnou pořadí uzlů při tvorbě bloku. Daný blok obsahuje stále stejný počet uzlů a ty vrcholy, do kterých se posunuly ty „zborcené", se místo nich opakují. Příklad tohoto způsobu tvorby pětistěnu je znázorněn na obr. 33. [30]

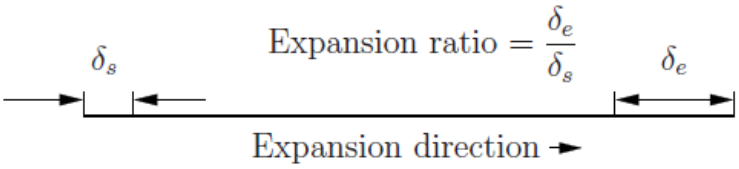
```
hex (0 1 2 3 4 5 5 4)
```



Obr. 33 Tvorba pětistěnu v blockMesh [30]

SimpleGrading

SimpleGrading obsahuje tři hodnoty, kde každá z nich pro daný směr definuje poměr velikosti posledního kontrolního objemu k tomu prvnímu. Pro přesnější definici je použit obrázek 34. [30]



Obr. 34 Poměr velikosti prvního a posledního kontrolního objemu [30]

Celkové zastoupení kontrolních objemů může být pro každý směr definováno i podrobněji. Zapisuje se do kulatých závorek se třemi hodnotami dle počtu částí, které byly zvoleny. První z nich udává procentuální délku vybrané části. Další z nich definuje procentuální zastoupení počtu kontrolních objemů ve vybrané části a poslední hodnota udává velikost expanzního poměru.

Hrany (edges)

BlockMesh obsahuje více možností, jak definovat křivost hrany (např. arc, simpleSpline, polyLine, nebo polySpline), z nichž nejpoužívanější je metoda s klíčovým slovem „arc" vytvářející kruhový oblouk. Tato metoda vyžaduje definování dvou indexů uzlů, které jsou vrcholy dané hrany. Následuje závorka se souřadnicemi interpolačního bodu, který bude oblouk protínat. Zápis se provádí způsobem, který je na obrázku 35. [30]

```
edges
(
    arc 1 5 (1.1 0.0 0.5)
);
```

Obr. 35 Definování zakřivené hrany pomocí „arc" [30]

Stěny na hranici domény (faces)

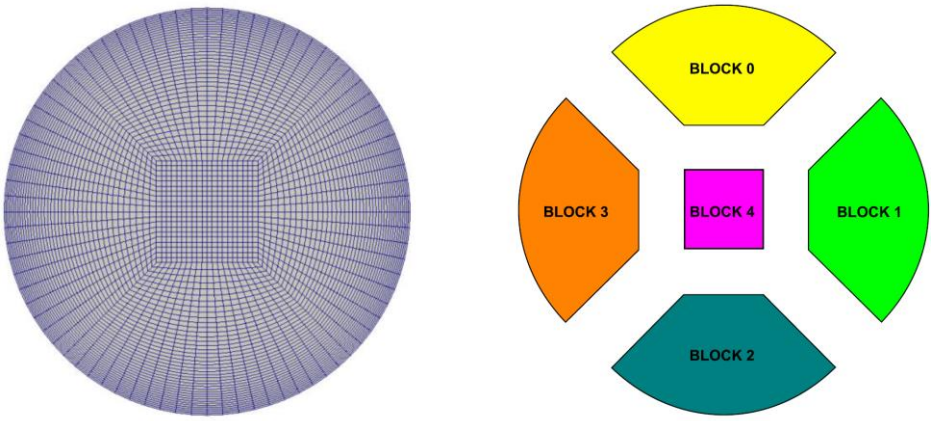
Počátek definování hranic začíná klíčovým slovem „boundary". Následně je hranice rozdělena na jednotlivé regiony (patches) s definovanými názvy jako klíčovými slovy. Jedná se například o definování vstupu (inlet) a výstupu (outlet) proudícího média. Další vlastnosti jednotlivého regionu definuje slovník obsahující informace o typu daného regionu. Dle použitého typu jsou zároveň s ním definovány i okrajové podmínky. Například pro pevnou stěnu je použito klíčové slovo „wall". BlockMesh následně aplikuje okrajové podmínky dle zvoleného typu „wall" tak, aby se region při výpočtu choval jako pevná, pro použité médium nepropustná stěna. [30]

Každý takto definovaný region se skládá z jednotlivých stěn zvolených bloků. Stěny jsou určeny pořadím uzlů tak, aby z vnitřního pohledu bloku byl směr pořadí uzlů shodný se směrem hodinových ručiček. Dle pravidla pravé ruky palec vždy směřuje od dané stěny do prostoru. Pro

příklad si vězměme obr. 27.1. Jeho vybraná horní stěna by měla uzly definované v následujícím pořadí: (4 5 6 7). V pořadí nezáleží na počátečním bodu. [30]

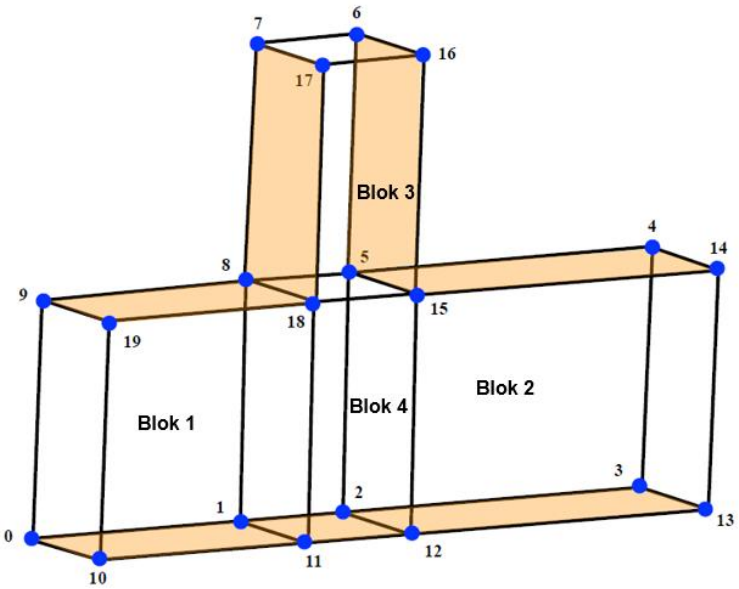
Spojení dvou bloků

Když se geometrie skládá ze dvou a více sousedních bloků, existují dvě možnosti, jak bloky v blockMesh spojit. Na obr. 36 je uveden příklad sítě válcové geometrie z pohledu příčného řezu, který je složen z pěti jednotlivých bloků. Tento první případ, kdy jednotlivé bloky na sebe navzájem navazují a mají sousední uzly totožné, je tvořen automaticky. Uzly, které mají bloky navzájem totožné, se však nesmí v úvodním seznamu uzlů vyskytovat duplicitně. [33]



Obr. 36 Spojení navazujících bloků v blockMesh [33]

Druhou možností je funkce mergePatchPairs, kdy jednotlivé bloky na sebe svými uzly nenavazují, ale dotýkají se pouze svými rovinami. Funkce mergePatchPair umožňuje spojení takových rovin, jejich kvalita bude ale horší než u sítě s navazujícími bloky (viz. kap. 2.1.3 této práce). Z toho důvodu, pokud to tvar geometrie umožňuje, se tato funkce nepoužívá a místo toho se přistupuje k první možnosti spojení. Pro příklad je uveden obr. 37, kde se docílilo první možnosti spojení přidáním nového bloku. Spodní část je tak rozdělena z jednoho na tři bloky tak, aby blok čtyři svými uzly navazoval na blok tři a byla zajištěna optimální kvalita sítě. [32]



Obr. 37 Varianta přidání bloku pro zajištění optimální kvality sítě [32]

5 Tepelné výměníky se svazkem trubek v plášti (TVT)

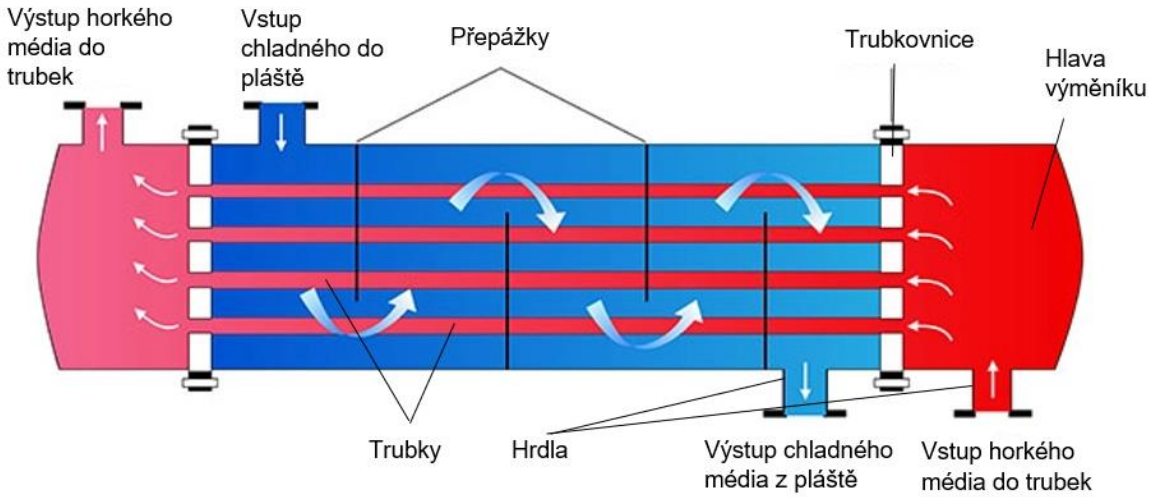
Výměníky tepla jsou zařízení umožňující přenos tepla mezi dvěma i více látkami rozdílné teploty, které jsou v tepelném kontaktu. Svoje uplatnění mají v mnoha technologických odvětvích, přičemž výměníky tepla se svazkem trubek v plášti patří mezi ty nejpoužívanější. Používají se například v chemickém průmyslu, a to hlavně díky snadnému čištění a vyměnitelnosti svých částí. [35]

Z procesního hlediska se tento typ výměníku používá pro střední až vysoké tepelné výkony (stovky kW až jednotky MW), nebo nízké průtoky pracovních látek, a pro velké pracovní tlaky. Také pro tento typ existuje mnoho návrhových metod a výpočtů. Pro trubky je k dispozici celá řada materiálů. Řadí se mezi nejspolehlivější typy výměníků. [36]

Tato kapitola je zaměřena na jejich základní konstrukci a pojmenování jednotlivých částí.

5.1 Konstrukce TVT

Existuje několik konstrukčních typů trubkového výměníku se svazkem trubek v plášti. Jedním z hlavních kritérií jejich rozdělení je počet chodů na straně trubek. Výměník tak může být jednochodý s rovnými trubkami nebo dvouchodý s trubkami ve tvaru písmene U nebo s plovoucí hlavou vyrovnávající teplotní roztažnost trubek. Počet chodů udává kolikrát médium projde skrz zařízení výměníku. V rámci této práce byla vytvořena geometrie pro program OpenFOAM trubkového prostoru jednochodého výměníku s přímými trubkami, jehož základní části konstrukce jsou popsány pomocí obr. 38 níže. [37]



Obr. 38 Výměník tepla se svazkem trubek v plášti [37]

Svazek trubek

Plocha výměny tepla mezi látkou proudící trubkovým prostorem a látkou proudící v plášti je dána svazkem trubek. Trubky jsou na obou stranách uchycené do trubkovnice a mohou být buď hladké nebo žebrované pro zesílení výkonu. [34]

Obvykle se pro zvýšení přenosu tepla a snížení zanášení navrhuje turbulentní proudění, k čemuž slouží turbulizátory, což jsou zařízení, která víří proudící médium a vytváří turbulentní proudění.

Ve většině případů se umisťují do trubek v podobě drátů nebo spirál. Na straně pláště se může použít zmiňované žebrování, které slouží především k zvětšení plochy výměny tepla. [35]

Trubkovnice

Je deska s otvory pro uchycení trubek a drážkami pro umístění těsnění. Obsahuje i otvory pro vodící tyče, které slouží k upevnění přepážek a zajištění geometrické přesnosti zavedení trubek. Trubky se do trubkovnice upevňují buď válcováním, nebo při větších pracovních tlacích vnějším svařováním. [34]

Plášť

Plášť tvoří mezitrubkový prostor výměníku a nachází se v něm obvykle látka s nižším tlakem, než v trubkovém prostoru z důvodu nižší materiálové pevnosti pláště oproti trubkám. Ve většině případů je strana pláště limitující stranou přenosu tepla a z toho důvodu jsou v ní umístěny přepážky, které usměrňují tok média a intenzifikují přenos tepla. [34]

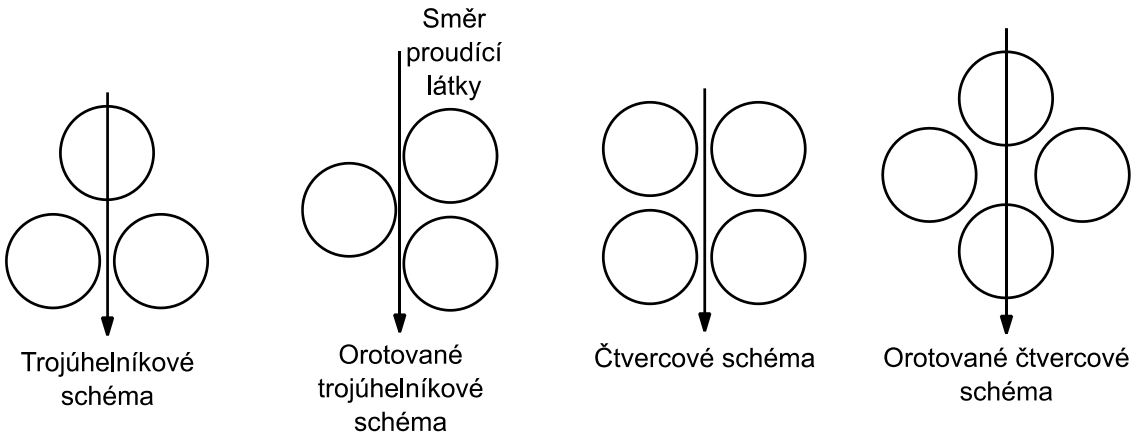
Hrdla

Tvoří vstup a výstup pracovních látek a pro jejich zhotovení se používají normalizované bezešvé trubky, které se k plášti přivařují. [34]

V rámci této práce byla vytvořena geometrie pro CFD analýzu strany trubkového prostoru. Tedy svazek trubek s hlavami výměníku a vstupními i výstupními hrdly. Následující kapitola proto obsahuje popis uspořádání svazku trubek a jeho vliv na funkci tepelného výměníku

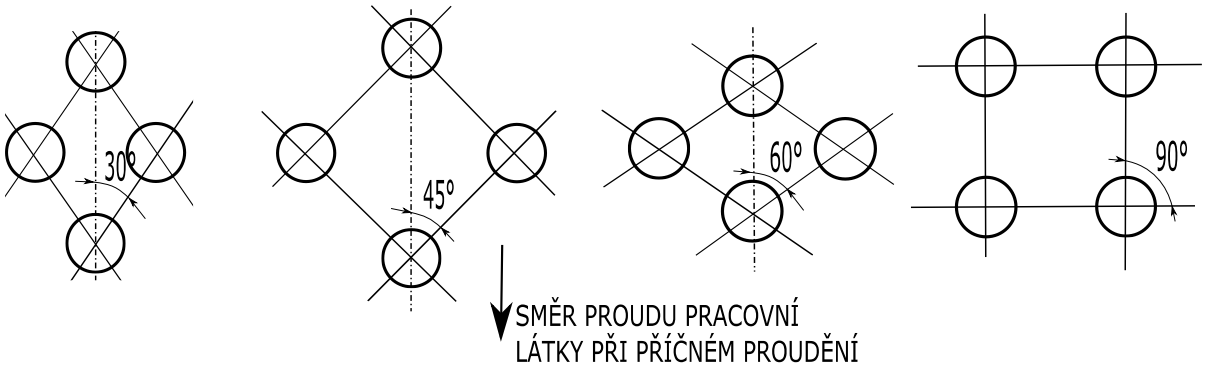
5.2 Hlavní charakteristiky geometrie svazku trubek

Pro uspořádání trubek ve svazku existují standardy popsané v Tubular Exchangers Manufacturers Association (TEMA) [49]. Typická uspořádání jsou znázorněna na obr. 39. [38]



Obr. 39 Základní používané uspořádání svazku trubek [38]

Čtvercové uspořádání svazku trubek se nazývá „v zákrytu" a trojúhelníkové „prostřídané". Úhel, který spojnice středů trubek mezi sebou svírají může být různý. Běžně používané varianty jsou znázorněny na obr. 40. Z těchto variant se nejvíce používá třicet nebo pětačtyřicet stupňů. [34]



Obr. 40 Základní používané uspořádání svazku trubek dle odlišných úhlů [34]

Z obr. 39 je patrné, že trojúhelníkové schéma dokáže v celkovém uspořádání pojmout více trubek než schéma čtvercové. Také napomáhá tvorbě vyššího turbulentního proudění a z toho důsledku zvyšuje součinitel přestupu tepla, což se pozitivně promítá na tepelném výkonu. Na druhou stranu způsobuje vyšší tlakové ztráty a není vhodné k použití pro zanášivé látky, jelikož mechanické čištění svazku je pak obtížnější. [39]

Ze studie prováděné Petinrinem a Dareovou vyplývá, že trojúhelníkové schéma s úhlem 60° oproti původnímu s úhlem 30° nepřináší z hlediska tlakových ztrát ani součinitele přestupu tepla žádné výhody, a proto se používá velmi málo. [38]

Z hlediska provozních nákladů je výhodnější uspořádání „v zákrytu", jelikož způsobuje nižší pokles tlaku, což má pozitivní vliv na výkon výměníku a šetří energie na jeho provoz. Zároveň „prostřídané" uspořádání zvyšuje celkový součinitel prostupu tepla, což vede ke snížení požadované plochy na přenos tepla a tím pádem i vstupních investic. Pro výběr uspořádání svazku trubek tak musí být rozhodnuto na základě kompromisu prostřednictvím termoekonomické analýzy. [40]

6 Aplikace pro tvorbu výpočtové sítě trubkového prostoru TVT

Pro vytvoření aplikace, která generuje textový soubor výpočetní sítě trubkového prostoru TVT pro OpenFOAM byl použit programovací jazyk Python. Textový soubor, který je praktickým výstupem této práce, je vstupem pro aplikaci blockMesh, která soubor automaticky čte, generuje síť a zapisuje důležitá data pro výpočet do adresáře system.

Aplikace umožňuje vytvářet libovolný počet řad s různým počtem trubek pro uspořádání „v zákrytu" a uspořádání „prostřídané" pro pětačtyřicet stupňů. Rozměry trubek, hlav výměníku a dalších geometrických parametrů pro blokové uspořádání jsou volitelné.

Výhodou této aplikace je její dostupnost, jelikož Python i OpenFOAM jsou zdarma a jsou mezi uživateli hojně používány. Další je možnost parametrizace, jelikož například změna počtu trubek nebo velikosti jejich průměrů spočívá pouze v přepsání několika málo proměnných hodnot, přičemž se nezmění kvalita sítě. Oproti komerčním aplikacím je tak parametrizace snazší a rychlejší. Jak samotný skript, tak textový soubor jsou pro uživatele zároveň ve své zdrojové formě dostupný k nahlédnutí a úpravám, což dává možnost úprav dle aktuálních potřeb toho kterého uživatele.

6.1 Python a struktura vytvářené aplikace

Python je interpretovaný, objektově orientovaný programovací jazyk, který je velmi populární mezi uživateli. Je používán u částí kódů webových služeb jako Facebook, Spotify nebo Instagram. Obecně se používá pro tvorbu webových aplikací, softwarů, výpočtových skriptů, počítačových her, marketingových aplikací, datových analýz a dalších.

Syntaxe jazyku se řadí mezi ty snadnější a dobře čitelné, což snižuje náklady na údržbu aplikací a je i tak pro méně znalé uživatele v tomto ohledu dostupnější. Python podporuje moduly a balíčky a také obsahuje vlastní standarizovanou knihovnu (obsahující například Matplotlib, Numpy nebo SciPy), dostupnou ve zdrojovém kódu, což uživateli umožňuje její implementaci do vytvářené aplikace a tím usnadnění její vlastní tvorby. Je zároveň dostupný pro různé operační systémy jako Windows, Mac, Linux a Unix. [41]

Aplikace byla vytvářena z velké části pomocí tříd. Třídy jsou prostředkem pro spojování dat a funkcí dohromady. Vytvořením třídy vznikne nový typ objektu, který umožňuje vytvářet další instance stejného typu. Každá instance může mít dále určité vlastnosti definované uživatelem. Tyto vlastnosti mohou být také proměnné, které se mohou pomocí vytvořených metod v dané třídě dále modifikovat. [42]

Objektově orientované programování neboli vytváření tříd slouží k lepšímu uspořádání a snadnějšímu použití opakovaných funkcí a jejích čtení v kódu. Tento typ programování může být například použit při tvorbě aplikace na úpravu obrázků. Objekt by v tomto případě byl obrázek. Vlastnosti, kterými je objekt popsán a mohou se dále upravovat mohou být velikost obrázku, jeho jméno nebo pixely. Upravit tak tento objekt je možné změnou jména, odebráním pixelů (ořezáním

obrázku) nebo otočením (změnou polohy pixelů). Každý objekt má tedy vlastnosti a obsahuje metody, které tyto vlastnosti upravují. [43]

Typy dat v Pythonu

Typ dat je pro každý programovací jazyk důležitým aspektem, jelikož určuje způsob, jakým jsou hodnoty v proměnných zapsány, tedy jaké mají vlastnosti a jak mohou být dále upravovány. V této kapitole je uveden seznam základních typů, které byly použity při tvorbě této aplikace.

Numerické typy

Do numerických typů dat patří celá čísla (integers) a desetinná čísla (floating point numbers). Celá čísla mohou být jakékoliv délky a jsou omezena pouze úložnou pamětí. Desetinná čísla mohou mít přesnost až na patnáct desetinných míst. Tyto typy dat mají tedy konkrétní číselné hodnoty.

Textové typy

Textovým typem je v Pythonu řetězec (string). Řetězec je posloupnost znaků Unicodu. Pro zapsání textového typu se používají jednoduché, nebo dvojité uvozovky.

Kolekce

Kolekce je v pythonu označena jako list a tuple. List je seřazená sekvence jednotlivých položek numerického nebo textového typu oddělená čárkami. List se zapisuje do hranatých závorek a tuple se zapisuje do závorek kulatých. Tuple nejde na rozdíl od listu po jeho zapsání dále modifikovat. [44]

6.2 Tvorba aplikace

Aplikace se skládá ze dvou hlavních částí, jímž je tvorba svazku trubek a tvorba vstupní a výstupní hlavy výměníku s hrdly. Geometrie svazku i hlav byly rozděleny do jednolivých bloků, jelikož blockMesh vytváří blokově strukturovanou síť. Spojení jednotlivých bloků používá první, automatickou možnost napojení, kdy uzly sousedních bloků jsou totožné.

Z toho důvodu se všechny uzly tvořící síť trubek nachází v jedné ze čtyř rovin hlav výměníku a od tvorby jejich bloků se následně odvíjí bloková struktura zbylé části geometrie hlav. Nejprve se tedy definují parametry tvořící svazek trubek a podle těchto parametrů se následně tvoří bloky jednotlivých hlav. Ty musí pro zajištění kvality sítě a přesnosti výpočtu, jako i pro splnění první podmínky napojení, na jednotlivé bloky svazku trubek svými uzly navazovat.

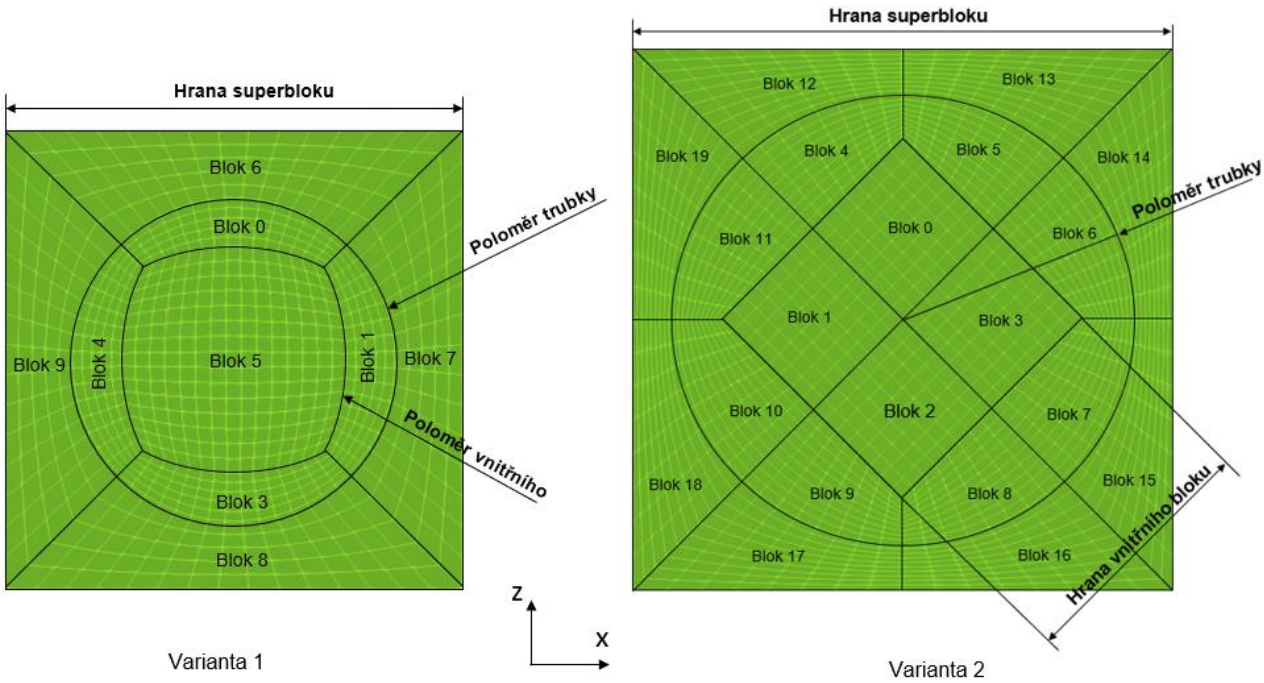
Postup v obou částech při tvorbě aplikace byl podobný. Nejprve se zvažovalo rozdělení geometrie do blokové struktury, následně vytvoření jednotlivých uzlů daných bloků a nakonec jejich řazení do bloků, stěn, oblouků a zapsání do textového souboru.

Následující kapitoly se věnují tvorbě daných částí aplikace, popisem jednotlivých tříd, jejich funkcí a vstupních parametrů. Většina názvů vyskytujících se v kódu byla napsána česky, ale bez použití diakritiky. Pro přehlednost budou názvy v této práci psány bez diakritiky tak jako v Pythonu.

6.2.1 Tvorba uzlů svazku trubek

Vytvoření jedné trubky

Pro tvorbu trubky bylo nejprve nutné diskretizovat její geometrii na jednotlivé bloky tak, aby byla kvalita její výsledné sítě dostatečná. Byly uvažovány dvě varianty, znázorněné na obr. 41. První varianta je mezi uživateli běžně používaná. Zmíníe, že jedna trubka se celkově skládá z pěti bloků. Druhá zvolená varianta je pro tvorbu aplikace o něco náročnější, svojí blokovou strukturou však umožňuje uspořádání „prostřídané", jelikož je celkový blok (pojmenovaný jako superblok) rozdělen na čtvrtiny. Díky tomu jednotlivé superbloky v tomto uspořádání na sebe ve svazku trubek navazují. Je tak zároveň zachována kvalita sítě.



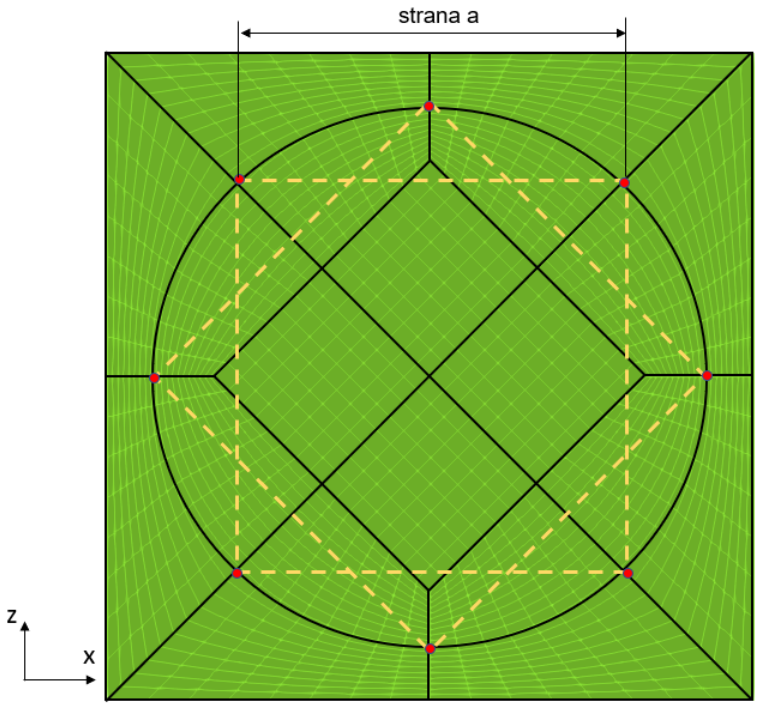
Obr. 41 Bloková struktura výpočtové sítě trubek

Z obr. 41 je zřejmé, že se jeden superblok tvořící geometrii trubky a jejího okolí v rovinách hlav výměníku skládá z celkem dvaceti bloků. Vstupními parametry pro tvorbu celého superbloku jsou poloměr trubky (polomer_trubky), délka úhlopříčky vnitřního bloku (uhlopricka_vnitrniho_bloku) a délka jeho vnější hrany (hrana_superbloku).

Superblok je v poměru délek svých hran čtverec, aby se mohly jednotlivé trubky řadit do uspořádání „v zákrytu", nebo „prostřídané" a uzly jednotlivých bloků na sebe navazovaly. Z důvodu složitosti geometrie nebyla pro tvorbu trubky napsána třída, ale byla použita Python knihovna Numpy usnadňující práci s maticemi, do kterých se zapisovaly jednotlivé souřadnice uzlů.

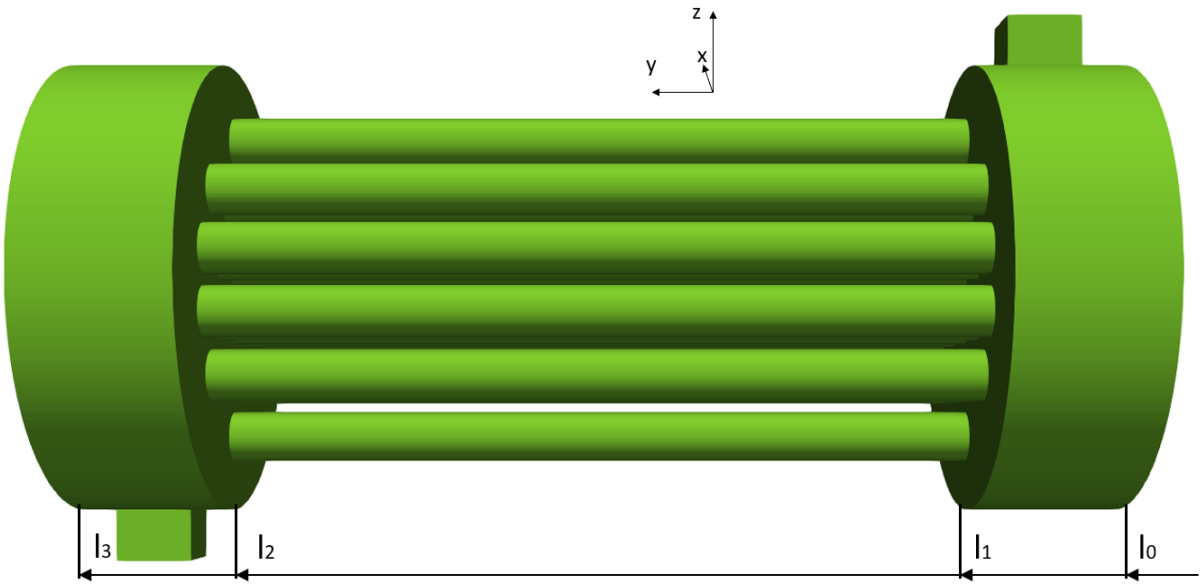
Pro vytvoření uzlů superbloku byla napsána funkce „vytvor_blok", která vytváří jeden celý blok s osmi uzly v 3D. Jednotlivé bloky s uzly tak nebyly tvořeny samostatně, ale v rámci jednotlivých

bloků a jejich natočením. Příklad této funkce je z pohledu příčného řezu znázorněn na obr. 42, kde je ukázána tvorba dvou bloků o stejných rozměrech (pouze jeden je natočený o 45°).



Obr. 42 Tvoření uzlů superbloku

Vstupními parametry pro funkci na tvorbu bloků svazku trubek jsou délky stran čtverce daného bloku, které vychází ze vstupních parametrů pro celý superblok a délky, ve kterých se jednotlivé roviny hlav výměníku nachází (l_0, l_1, l_2, l_3). Tyto roviny jsou znázorněny i na obr. 43. Například strana „a" na obr. 42 vychází ze vstupního parametru poloměru trubky. Tímto způsobem jsou vytvořeny všechny uzly bloků tvořících trubku a celý superblok v první a druhé rovině vstupní hlavy. Pro vytvoření uzlů v dalších rovinách slouží funkce „translace", která kopíruje uzly vstupní hlavy a vytváří uzly protilehlých bloků hlavy výstupní.



Obr. 43 Válcové roviny hlav výměníku

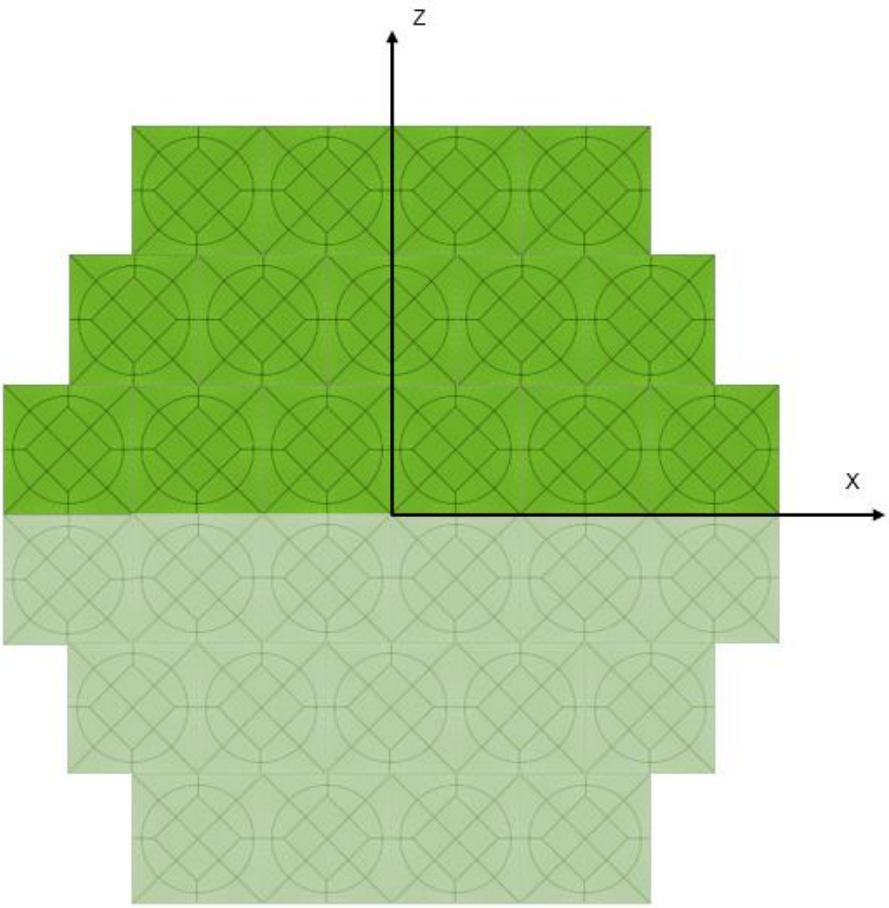
Souřadnice uzlů tvořící bloky vstupní hlavy se zapisují do matice s názvem „uzly_svazku_1". Dále kopie uzlů jednoho superbloku pro výstupní hlavu výměníku se s již vytvořenými uzly zapisují do matice „uzly_svazku_2". Tvorba jedné trubky pak spočívá ve správném seřazení uzlů, které leží v rovinách hlav ve vzdálenostech l_1 a l_2. Tak se dle pravidel blockMesh vytvoří bloky tvořící geometrii trubek mezi těmito rovinami.

Vytvoření svazku trubek

Kopírování uzlů superbloků v matici „uzly_svazku_2" vykonává i pro vytváření a řazení trubek do svazku funkce „translace". Na rozdíl od vytváření druhé roviny hlavy se pro kopírování trubek používá proměnná v textové podobě s názvem „pridej". Podle této proměnné funkce zvolí druhý postup, který superbloky kopíruje a zároveň nevytváří žádné duplicitní. Souřadnice posunutí v osách x a z jako další vstupní údaj pak posouvají všechny souřadnice uzlů superbloku o vzdálenost vepsanou v listu těchto souřadnic.

Pro vytvoření souřadnic posunutí byla napsána funkce „serad_trubky". Jejím vstupem je počet řad a počet trubek a jejím výstupem je list souřadnic o velikostech posunutí tak, aby se jednotlivé superbloky řadily buď do uspořádání „v zákrytu", nebo „prostřídané".

Počet řad se stejně jako počet trubek zapisuje za sebou do listu, jako vstupní údaj pro funkci „serad_trubky". Funkce je pro zjednodušení vstupních údajů psána tak, aby řady s trubkami zrcadlila podle vodorovné osy. Příklad možného uspořádání trubek ve svazku je znázorněn na obr. 44.



Obr. 44 Vytvoření svazku trubek

Vstupní údaje jako je počet trubek a počet řad se pod tímto názvem zapisují do listu. Pro uspořádání trubek z obr. 44 by vstupní údaje byly zapsány následovně:

Pocet_trubek = [6, 5, 4]

Pocet_rad = [0, 1, 2]

Nové superbloky se zapisují do matice s názvem „uzly_svazku_3". Je to ale pouze pomocná matice, která se přidá do matice „uzly_svazku_4", kam se vepisují všechny nově vytvořené superbloky svazku trubek.

Pokud by uživatel požadoval jiné uspořádání, lze funkci „serad_trubky" vynechat a do listu souřadnic posunutí s názvy „x_trans" a „z_trans" zapsat vlastní požadované souřadnice posunutí trubek. Pro správnou funkci vytvářené aplikace musí uživatel dodržet pravidlo návaznosti jednotlivých bloků.

Jelikož sousední uzly superbloků jsou totožné, tak při jejich kopírování a posouvání vzniknou uzly duplicitní. Ty se vymažou funkcí „duplicita".

6.2.2 Tvorba uzlů hlav výměníku

Při vytváření této části kódu byly použity třídy a pro další zjednodušení kódu byly uzly bloků nejprve tvořeny v jedné rovině ve 2D a následně se zkopírovaly souřadnice uzlů pro další roviny. Pro přehlednost použitých metod na dané objekty generuje vytvářená aplikace 2D obrázek toho, co bylo vytvořeno a jednotlivými metodami upraveno. Tento je ve formátu „svg". Metoda, která vytváří zmíněný grafický výstup se nazývá „namaluj_obrazec" a je možné ji spustit po zahájení třídy „Pole_Ctvercu". Tímto zahájením se vytváří daná třída, pro kterou je nutné definovat její vstupní parametry.

Diskretizace geometrie

Jelikož jednotlivé uzly bloků na sebe musí navazovat a jelikož hlavními parametry jsou parametry svazku trubek, musela se diskretizace geometrie hlav výměníku do bloků odvíjet od vytvořených bloků svazku trubek (superbloků).

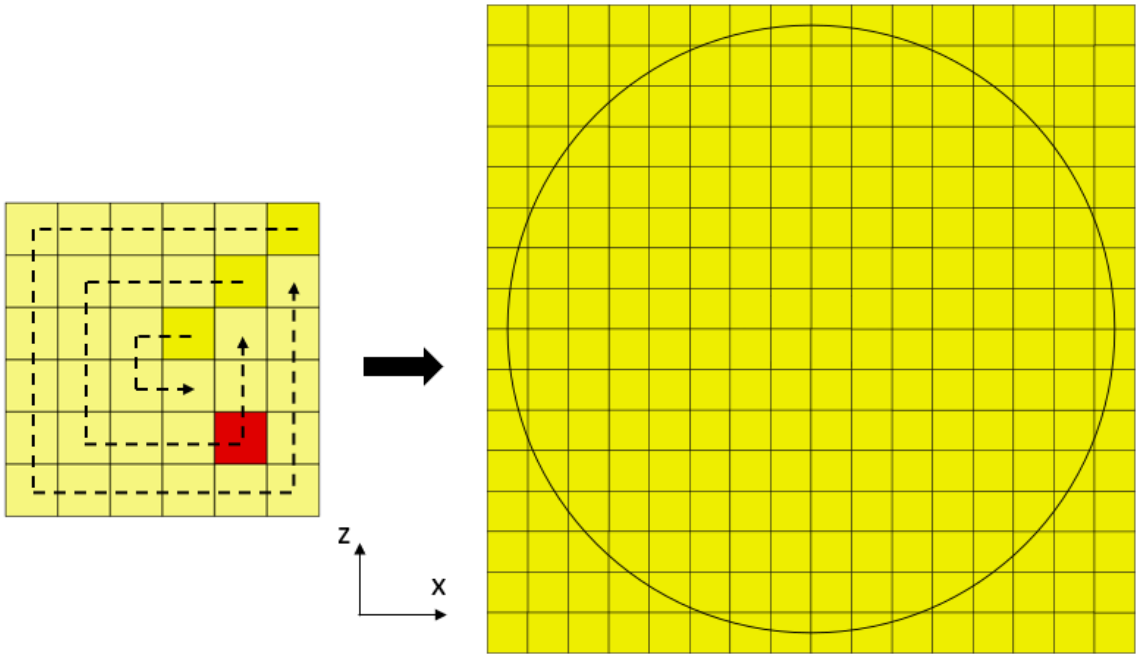
Jeden superblok je po svém obvodu rozdělen na osminy tak, aby v návaznosti bloků umožňoval i uspořádání „prostřídané". Každá jeho vnější hrana je tedy rozdělena na dvě poloviny. Počet bloků hlav na jednu hranu superbloku je tak roven dvěma. Na jeden celý superblok pak rozměrově připadají čtyři bloky hlav výměníku. Jak superblok, tak bloky hlav jsou čtverce.

Tvorba bloků hlav výměníku začala v jedné rovině, tedy v 2D vytvořením čtverců. V první fázi bylo pro vytvoření všech uzlů hlav v jedné rovině vytvořeno celé pole těchto čtverců. Další postup se odvíjí od průměru hlav výměníku. Čtverce v poli čtverců musely být dle této kružnice ořezány, aby byly hlavy výměníku válcové. Celé bloky hlav výměníku byly vytvořeny v 3D až v poslední fázi postupu, kdy se vytvořené a upravené uzly bloků hlav zkopírovaly do každé z rovin hlav výměníku (l_0, l_1, l_2, l_3).

Pole čtverců

Tvorba čtverců začíná od počátku souřadnicového systému v kladném kvadrantu a postupuje proti směru hodinových ručiček. Hlavním parametrem určujícím jeho velikost je proměnná s názvem „pocet_vrstev". Jednotlivé vrstvy se tvoří s počátkem v pozicích sytě žlutých čtverců a dále pokračují proti směru hodinových ručiček. První vrstva tedy vytvoří čtyři čtverce kolem počátku souřadnicového systému, druhá vrstva dále vytvoří čtverce kolem první vrstvy (jak je znázorněno na obr. 45 vlevo). Vrstvy se nepočítají od nuly, ale od jedné. Na obr. 45 vlevo je tedy počet vytvořených vrstev roven třem. Jednotlivé vrstvy objektu pole čtverců jsou tvořeny metodou „vytvor_vrstvu".

Dalšími parametry nutnými pro inicializaci třídy PoleCtvercu je „radius_kruznice”, což je požadovaný poloměr hlavy výměníku a pomocný parametr „pocet_ctvercu_na_hranu_trubky”, který udává počet čtvercových bloků hlav na jednu hranu superbloku (což jsou dva bloky). Pro vytvářenou aplikaci je důležité, aby hrany bloků vytvořených čtverců měly průsečíky s vytvořenou kružnicí hlavy výměníku. Počet vrstev se tedy odvíjí od velikosti poloměru hlavy, jak je znázorněno na obr. 45 (kružnice nesmí pole čtverců přesahovat).



Obr. 45 Pole čtverců

Souřadnice

Souřadnice popisující uzel, které tvoří každý čtverec, jsou vlastnostmi samostatné třídy s názvem „Souradnice”. Jedna souřadnice uzlu (jako objekt) tak obsahuje vlasnosti jako je název, který pro lepší orientaci popisuje polohu bodu ve čtverci (např. „Pravy_dolni”), prostorové souřadnice x, y a z.

Čtverec

Čtverec je tvořen čtyřmi souřadnicemi (objekty vytvořené třídou „Souradnice”), z nichž každá představuje polohu jednoho uzlu a je zapsána v listu „souradnice_list”, který je vlastností třídy „Ctverec”. Celé pole čtverců je pak typem objektu třídy s názvem „PoleCtvercu” (dále už jen pole čtverců). Vstupní parametry pro inicializaci dané třídy jsou definovány v závorce za zkratkou „__init”. Hlavními vlastnostmi objektu „Ctverec” (dále pouze jako čtverec) ve vytvořeném poli je jeho poloha v souřadnicovém systému. Tu určují jeho vlastnosti s názvy „column” a „row”.

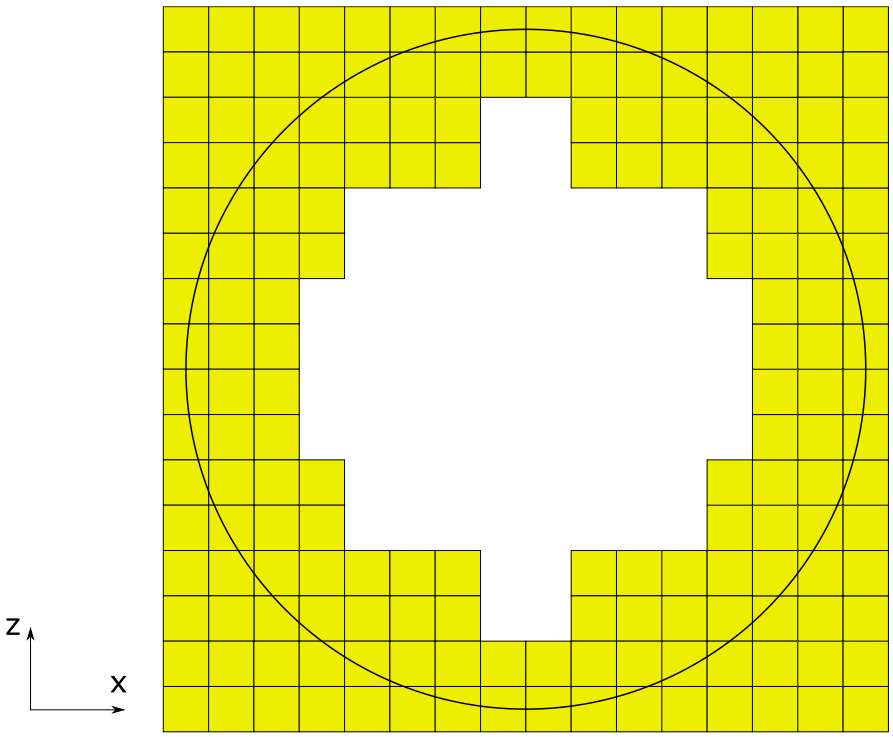
Například červený čtverec na obr. 45, má vlastnost „column” rovnou jedné, jelikož se nachází v prvním sloupci. Tvorba prvního čtverce začíná dle konvence Pythonu v nule. První vytvořený

čtverec se tak nachází v nultém sloupci a nultém řádku. Na obr. 38 je to sytě žlutý čtverec nejblíže středu. Dle tohoto systému má uvedený červený čtverec dále vlastnost „row" rovnu mínus dvěma.

Další vlastností čtverce je délka jeho hrany, která je rovna polovině délky hrany superbloku svazku trubek. V dalších metodách se čtverce ořezávají dle kružnice hlavy výměníku a pro jejich tvarové odlišení slouží vlastnost „tvar", která je nastavena na „ctverec" (s datovým typem string). Při ořezávání může vzniknout i trojúhelník nebo pětiúhelník. Vlastnost „tvar" odliší jednotlivé bloky dle jejich tvaru tak, aby byl nastavitelný počet jejich kontrolních objemů. Různé tvary mají odlišnou velikost a při stejném počtu kontrolních objemů by byl poměr jejich objemů velmi rozdílný, což zhoršuje kvalitu sítě. K různé velikosti kontrolních objemů dojde v každém případě. Tato vlastnost slouží spíše k redukci této chyby. Jedná se o omezení dané aplikací blockMesh, která je zpracovaná v kapitole 6.3.

Odstranění vytvořených bloků svazku trubek

Vytvořené pole čtverců je svými uzly z části duplicitní s uzly bloků svazku, které jsou již vytvořené. Metoda, která vymaže ty čtverce, které se překrývají s bloky svazku se nazývá „vymaz_svazek_trubek". Její vstupní parametry jsou totožné s parametry pro vytvoření svazku, tedy počet trubek a počet řad v listu. Její výstup je znázorněn na obr. 46, kde chybějící čtverce uvnitř pole představují vytvořené bloky svazku trubek.

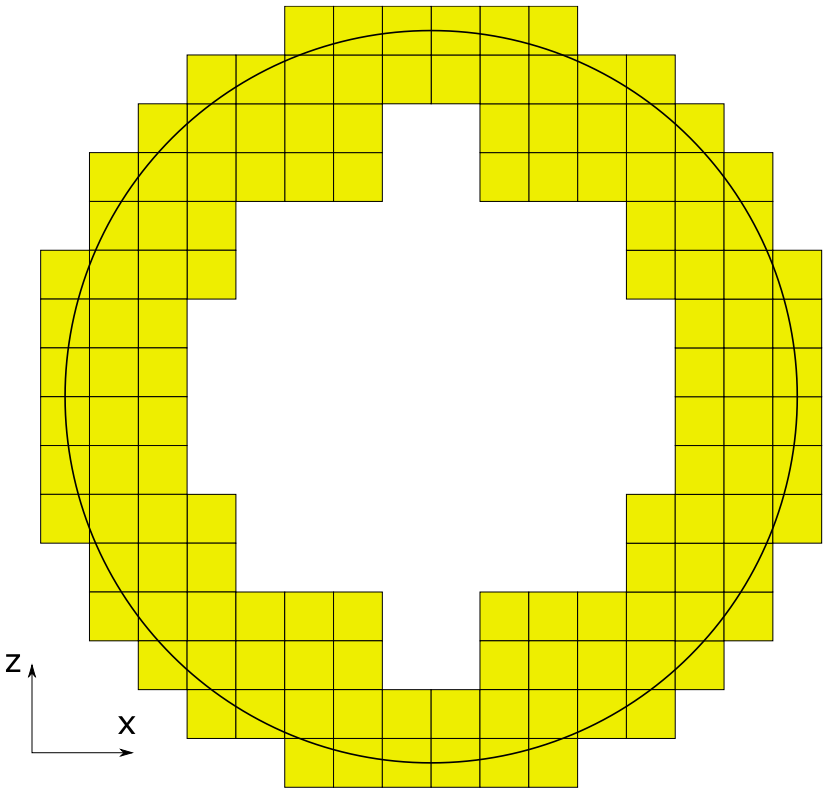


Obr. 46 Metoda „vymaz_svazek_trubek"

Odstranění čtverců vně kružnice hlavy výměníku

Dalším krokem pro vytvoření válcové hlavy výměníku z jednotlivých bloků je odstranění čtverců z vytvořeného pole, které leží vně poloměru hlavy výměníku a tato kružnice hlavy je neprotíná. Tato metoda má název „smaz_vse_za_kruznici".

Metoda používá metodu třídy „Ctverec" s názvem „nejmensi_vzdalenost_od_stredu", jejímž výstupem je vzdálenost uzlu, který je nejblíže počátku souřadnicového systému. Pokud je tato vzdálenost nejbližšího uzlu daného čtverce větší než poloměr hlavy, metoda čtverec smaže. Výstup této metody je znázorněn na obr. 47.



Obr. 47 Metoda „smaz_vse_za_kruznici"

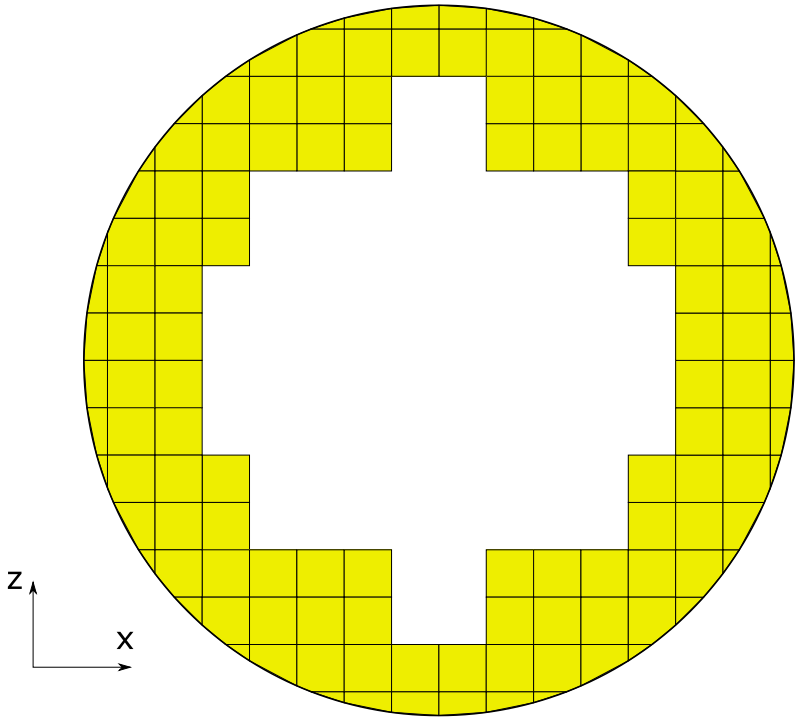
Oříznutí čtverců dle kružnice

Pro oříznutí čtverců dle kružnice je metoda „nahrad_body_vne_kruznice", která prochází všechny čtverce a pomocí dalších metod najde dva body, na jejichž spojnici existuje průsečík s kružnicí a body ležící vně kružnice nahradí těmito průsečíky.

Jejím vstupním parametrem je poloměr kružnice. Tato metoda prochází jednotlivé uzly a vybere dva, které nejsou totožné a zároveň mají totožnou jednu ze souřadnic x nebo z. Tím se zajistí to, že se průsečík vyhledává na jedné z hran daného čtverce a ne na jeho neexistujících úhlopříčkách.

Metoda najde průsečík po splnění podmínky, kdy jeden uzel leží vně kružnice a druhý uzel téhož čtverce leží uvnitř kružnice. Pro tuto podmínku používá metodu třídy „Souradnice" s názvem „vzdalenost_od_stredu", která počítá vzdálenost každého bodu od počátku souřadnicového systému.

Pro konkrétní vyhledání souřadnic průsečíku je použita funkce „prusecik_line_circle”. Jejími vstupními parametry jsou souřadnice dvou bodů a poloměr kružnice. Souřadnice průsečíku se zapisují do proměnné „prusecik_xz”, která je pak vstupem pro vytvoření nové souřadnice daného čtverce. Metoda pak uzly vně kružnice nahradí těmito novými souřadnicemi. Výstup této metody je znázorněn na obr. 48.



Obr. 48 Metoda „nahrad_body_vne_kruznice”

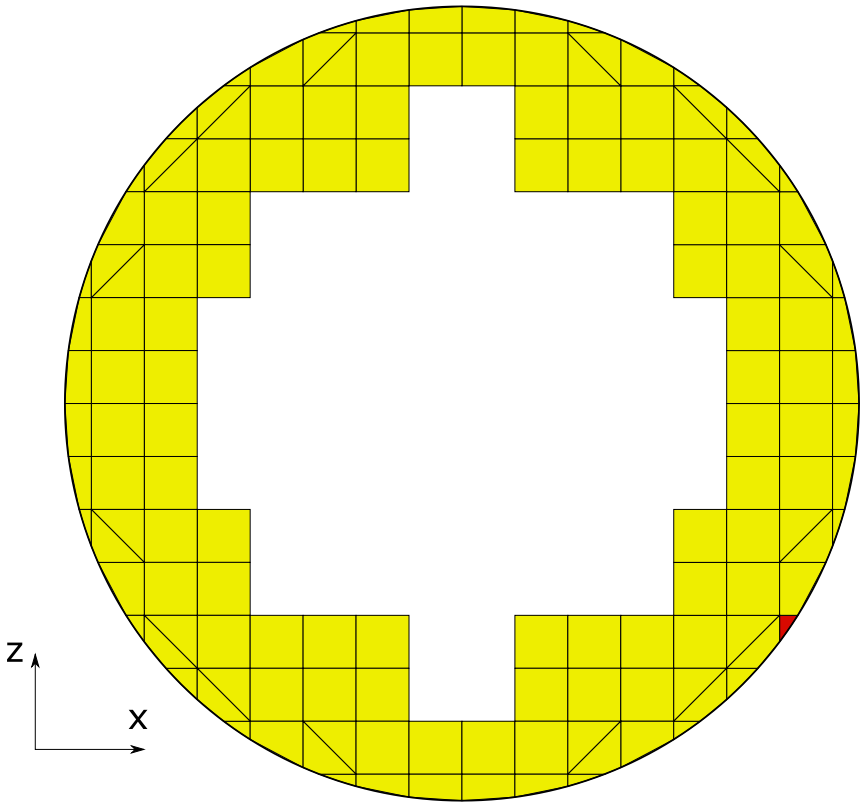
Úprava počtu uzlů jednoho bloku

Kružnice hlav výměníku může protínat jednotlivé čtverce (typ objektu třídy „Ctverec”) v různých místech. Jejich oříznutím dle této kružnice může vzniknout i pětiúhelník nebo trojúhelník. Všechny tyto tvary jsou podstavami následně vytvořených bloků sítě. Jelikož aplikace blockMesh neumí vytvářet bloky s podstavou pětiúhelníku, musí se případně vzniklé pětiúhelníky rozdělit spojnicí dvou bodů na jeden trojúhelník a jeden čtverec (které z pohledu geometrie nemusí striktně být trojúhelníky či čtverce, ale je tak na ně pohlíženo kvůli struktury vytvořeného kódu). Jejich tvarové odlišení je dále dáno názvem ve vlastnostech daného objektu „Ctverec”. Trojúhelník je ve vlastnosti „tvar” pojmenován jako „trojuhelnik” a čtverec jako „ctverec z petiuhelniku”. V případě vzniklého trojúhelníku oříznutím je pro lepší kvalitu sítě doprostřed jeho přepony přidán jeden uzel, čímž se vytvoří čtverec. Jeho název ve vlastnosti „tvar” je „ctverec z trojuhelniku”.

K rozdělení pětiúhelníků a přidání uzlu trojúhelníkům slouží metoda „rozrad_ctverce”. Metoda prochází listy souřadnic všech bloků a sčítá počty uzlů. Pokud je počet uzlů daného útvaru roven třem, pak to znamená, že daným útvarem je trojúhelník. Následně na to je použita metoda „pridej_bod_trojuhelniku”.

Tato metoda použije metodu „vzdalenost_od_stredu” a odstraní bod, který je nejbližší středu souřadnicového systému. Tím vzniknou dva body ležící na kružnici jako vstupní data pro metodu „votvor_bod_na_kruznici_mezi_body”. Ta středový bod ležící na kružnici mezi danými body vytvoří. Výstupem zmiňovaných metod je list souřadnic s přidaným uzlem trojúhelníku, a je vytvořen čtverec.

Pokud je dále v metodě „rozrad_ctverce” počet souřadnic roven pěti, daná metoda dále pracuje s metodou třídy „Ctverec” s názvem „rozdel_petiuhelnik”. Tato metoda používá také metodu „vzdalenost_od_stredu”, aby odstranila dva nejvzdálenější uzly a vytvořila ze zbývajících tří uzlů trojúhelník. Zároveň přidá čtyři nejvzdálenější uzly do nového listu souřadnic tak, aby vznikl nový čtverec (typ objektu třídy „Ctverec”). Výsledný výstup použité metody třídy „PoleCtvercu” s názvem „rozrad_ctverce” je znázorněn na obr. 49, kde jsou vidět všechny rozdělené pětiúhelníky a vpravo dole jeden vybraný trojúhelník vzniklý oříznutím s přidaným bodem (vybarvený červeně).



Obr. 49 Metoda „rozrad_ctverce”

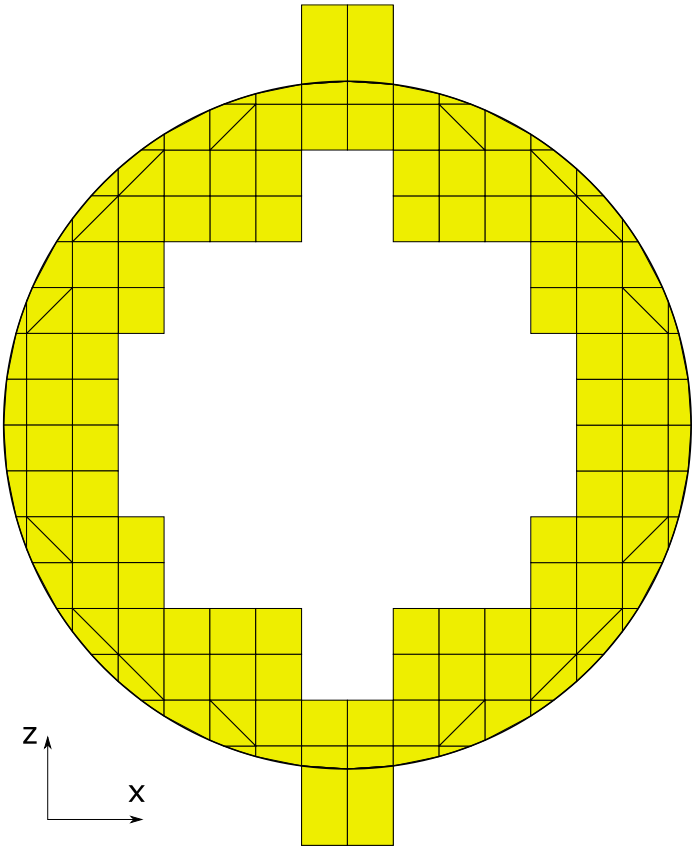
Takto vytvořené uzly v přední rovině se ve své konečné podobě pomocí metody „vytvor_body_pole” zkopírují do dalších tří rovin hlav výměníku. Rovina, ve které byly uzly v 2D vytvářené byla první rovinou ze čtyř rovin hlav výměníku ve vzdálenosti l_0. Celkově tak vzniknou čtyři roviny uzlů tvořící bloky hlav, které svým uspořádáním vyplňují prostor mezi svazkem trubek a vnější kruhovou plochou hlavy výměníku (další roviny se nacházejí ve vzdálenostech l_1, l_2 a l_3).

Uzly vstupního a výstupního hrdla

Vytvářené uzly vstupního a výstupního hrdla navazují na bloky, které mají souřadnice jednoho z uzlů totožné s osou z. Vstupní hrdlo se v základním nastavení nachází v kladném směru osy z a výstupní v záporném. K nalezení těchto navazujících bloků slouží metoda „najdi_ctverce_pro_input_output". Vstup a výstup pracovní látky se nachází na hrdlech, které mají tvar hranolů. Tvar hrdel je dán omezením aplikace blockMesh a snahou o návaznost jednotlivých bloků sítě tak, aby byla její kvalita dostatečná.

Postup tvorby bloků hrdel je stejný jako u tvorby pole čtverců. Nejprve se vytvoří čtverce ve 2D pomocí metody „vytvor_ctverce_pro_input_output_2D", která zkopíruje uzly navazujících bloků, které leží na kružnici metodou „zrcadli_body_ctvercu" o vzdálenost definovanou uživatelem. Tato vzdálenost neboli velikost hrany hrdel se definuje ve vstupních parametrech v proměnné s názvem „velikost_inletu_outletu". Nově vytvořené čtverce se ukládají do proměnných „ctverce_input" a „ctverce_output".

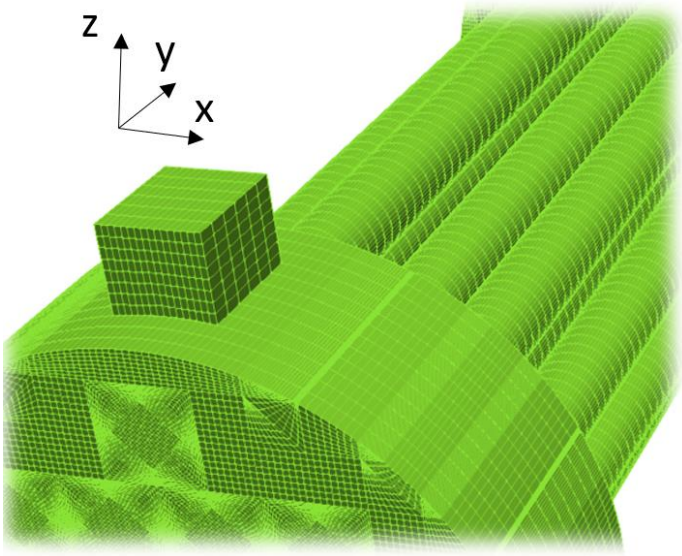
Na vytvoření celých bloků hrdel je metoda „vytvor_ input_output_3D". Jejím vstupním parametrem jsou roviny, do kterých se uzly hrdel zkopírují pomocí metody „zrcadli_ctverec". Jejich souřadnice se zapisují do proměnné „body_vymeniku", která obsahuje souřadnice všech uzlů a pro vykreslení 2D obrazového výstupu i do pole čtverců. Její výstup je znázorněn na obr. 50, kde je vykresleno v jedné rovině i výstupní hrdlo.



Obr. 50 Vytvoření hrdel výměníku

Odsazení hrdel od hrany hlavy výměníku bylo řešeno pomocí funkce mergePatchPairs. Je tomu tak z důvodu nenavazujících uzlů na žádnou ze čtyř rovin hlav (v ose y), ve kterých jsou tvořeny všechny bloky (vyjma bloků trubek). Délka odsazení je jedním ze vstupních parametrů, které definuje uživatel.

Další možností byla nová diskretizace geometrie na více rovin dle rovin hrdel. Počet bloků by ale byl čtyřnásobně větší a na kvalitu sítě by to při dodržení návaznosti kontrolních objemů nemělo vliv, a taktéž by se zbytečně prodloužil čas výpočtu. Návaznost kontrolních objemů bloků hlav a hrdel je tak řešena výpočtem. Výpočet z definované šířky hlav výměníku a šířky hrdel stanoví počet kontrolních objemů hrdel a vypíše je uživateli po spuštění vytvářené aplikace v textové podobě. V případě vypsání desetinného čísla musí uživatel vynásobit délku kontrolních objemů nejbližším celým číslem a uzpůsobit tak šířku hrdel počtu kontrolních objemů. Další možností je přizpůsobit počet kontrolních objemů požadované šířce hrdel. Konečné řešení hrdel je znázorněno na ukázkovém příkladu na obr. 51.



Obr. 51 Vstupní a výstupní hrdla hlav výměníku

Interpolační body pro napojení na válcové povrchy hlav

Bloky vytvořených čtverců mají ploché stěny. Pro vytvoření válcových hlav je zapotřebí okrajové plochy tvořící stěnu válcových hlav zaoblit. Oblouky jsou vytvořeny pomocí metody v blockMesh, která využívá klíčové slovo „arc" a u které je nutné definovat interpolační bod oblouku. Pro vytvoření těchto interpolačních bodů je metoda s názvem „vytvor_interpolacni_body". Ta pomocí metody „najdi_krajovy_body" vybere z listu souřadnic dva uzly, které jsou počátku souřadnicového systému nejvzdálenější a leží tak na kružnici hlavy výměníku. Dále metodou „vytvor_bod_na_kruznici_mezi_body" vytvoří pomocí úhlu s osou x mezi vybranými uzly interpolační bod.

Jediný rozdíl nastává u hrdel, u kterých je potřeba zaoblit jejich spodní hrany, tedy vytvořit interpolační bod mezi uzly daných bloků, které jsou počátku souřadnicového systému nejblíže. Ve funkci je tak vytvořena tato výjimka.

6.2.3 Řazení uzlů svazku trubek

Všechny vytvořené souřadnice uzlů se bez duplicity pomocí metody „vymaz_duplicitu_bodu" zapisují do listu s názvem „body_vymeniku". Metoda, která hledá uzly dle zadaného pořadí v listu „body_vymeniku" a jejímž výstupem je vždy index vyhledaného bodu má název „najdi_index_bodu".

Pro vyhledávání a řazení uzlů svazku trubek do bloků, stěn a oblouků slouží metoda „najdi_indexy_bodu_svazku". Jejími vstupy jsou „blok", pro který se mají jednotlivé uzly řadit do požadovaného směru. Dále souřadnice posunutí s názvy „x_j", „y_j", „z_j", určující polohu každého superbloku a „plocha" v textové podobě, jako zvolený typ řazení. Funkce vyhledává pomocí zadaných souřadnic uzel, jehož souřadnice jsou pak pro hledání následujícího uzlu daného bloku brány jako počáteční. K jeho souřadnicím jsou tak pro vyhledání dalšího uzlu přičteny pro každý směr vzdálenosti dle jeho vlastní polohy. Tyto vzdálenosti jsou zapsány v listech s názvy „xp" a „zp". Názvy posunutých souřadnic znamenají délky posunutí v osách x a z. Délky posunutí jsou vztaženy k počátku souřadnicového systému, proto pro jejich korelaci v rámci jednotlivých superbloků, které nemají svůj střed v počátku souřadnicového systému, slouží vstupní parametry posunutí s názvy „x_j", „z_j" a pro jednotlivé roviny (kolmé na osu výměníku) „y_j".

Vytvoření jednotlivých bloků svazku trubek, jejich ploch a zaoblení jejich hran umožňuje metoda „vytvor_hex_face_arc_svazku". Ta prochází jednotlivé bloky, zapisuje jejich pořadí a vytváří také stěny a oblouky hran trubek. Jednotlivé bloky jsou rozděleny dle svého tvaru a polohy tak, aby byl počet kontrolních objemů a definování poměru velikosti počátečního kontrolního objemu ke konečnému („simpleGrading") nastavitelný uživatelem a aby tak síť byla dostatečné kvality. Čísla jednotlivých bloků jsou uvedena na obr. 41 vpravo (na str. 45) a dle parametru v cyklu „for" s názvem „blok" je možné určit, které bloky se do jednotlivých listů bloků, stěn a oblouků zapisují.

6.2.4 Řazení uzlů hlav výměníku

Pro řazení uzlů v jednotlivých blocích do objemů a stěn slouží funkce „vytvor_hex_a_face_hlav". Pokud je „tvar" čtverce roven „ctverec", pak se jednotlivé uzly řadí do bloků a stěn podle své vlastnosti „nazev". Vlastnost „nazev" představuje v textové podobě názvy jednotlivých vrcholů čtverce. Pro vytváření bloků musí funkce hledat uzly ve dvou rovinách, ve kterých leží jejich podstavy. K tomu slouží vstupní parametr „y_pozice_rovin", který obsahuje souřadnice polohy jednotlivých rovin hlav výměníku v ose y.

Pokud čtverec nemá „nazev" roven „ctverec", pak se jednotlivé uzly řadí za sebou tak, jak jsou zapsány v listu „souradnice_list". Pro zajištění správného směru řazení uzlů v upravených blocích (jako je například vytvoření čtyřhranu přidáním jednoho uzlu trojúhelníku) slouží funkce „serad_body", která jako referenční bod najde ten nejblíže počátku souřadnicového systému a dále body seřadí po směru hodinových ručiček. Pro řazení uzlů do jednotlivých stěn jsou pak body z funkce pro tvoření bloků zapsány v žádaném pořadí tak, aby byla splněna pravidla aplikace blockMesh.

Pro vytvoření stěn obvodu hlav byla napsána metoda „vytvor_face_obvodu_hlav”. Ta používá metodu „vyhledej_okrajovy_ctverec” pro nalezení všech čtverců, jejichž některé ze souřadnic leží na válcové stěně hlavy výměníku. Metoda dále používá metodu „najdi_krajovy_body” pro vybrání pouze těch bodů, které leží na válcové stěně hlavy výměníku. Dále je pomocí vstupního parametru „y_pozice_rovin” řadí do stěn jednotlivých bloků tvořící pak celkovou obvodovou stěnu hlav výměníku.

6.2.5 Tvorba textového souboru blockMeshDict

Úvodní část textového souboru je ve většině případů až na definovaný počet kontrolních objemů a „simpleGrading” stejná a vypisuje se pomocí metody „nacti_start_vystupu”. Ve vytvářeném souboru s definicí bloků sítě jsou používány i proměnné, jejichž názvy jsou uvozeny znakem dolaru ($).

Souřadnice obsažené v listu „body_vymeniku” se vypisují do souboru pomocí metody „vystup_souradnic”. Metody „vystup_blocks”, „vystup_arc” a „vystup_boundary” zapisují do souboru jednotlivé bloky rozdělené dle blokové struktury, stěn a oblouků. Všechny tyto metody jsou vyvolané metodou „vytvor_txt_vystup”.

„SimpleGrading” a počet kontrolních objemů

Zápis bloků je rozdělen podle jejich tvarů a velikostí tak, aby byl u každého typu bloku nastavitelný počet kontrolních objemů. Základní nastavení zajišťuje vzájemnou návaznost kontrolních objemů tak, aby byl výpočet co možná nejpřesnější a časově nenáročný.

Základní nastavení počtu kontrolních objemů a „simpleGrading” se nachází v počátečním textovém souboru s názvem „nacti_start_vystupu” a je možné ho dále upravit, jako i další parametry v samotné aplikaci. Pojmenování jednotlivých zkratek se nachází v tabulce 6.

Tab. 6 Proměnné a jejich výchozí hodnoty pro nastavení počtu kontrolních objemů a „simpleGrading”

Význam zkratky	Název zkratky	Nastavená hodnota [-]
Počet kontrolních objemů v ose x pro všechny bloky	xx	8
Počet kontrolních objemů v ose z pro všechny bloky	xzv	8
Počet kontrolních objemů v ose y pro bloky hlav výměníku	yf	12
Počet kontrolních objemů v ose y pro bloky hrdel (hodnota odvíjející se od odsazení hrdel)	yfio	6
Počet kontrolních objemů v ose y pro bloky trubek	yft	60

Simple grading v ose x pro všechny bloky	gx	1
Simple grading v ose z pro všechny bloky	gr	1
Simple grading v ose y pro bloky hlav výměníku	gy	1
Délka pro simple grading v % pro bloky trubek v ose y (okraje trubek)	tco	0,3
Délka pro simple grading v % pro bloky trubek v ose y (střed trubek)	tcs	0,4
Počet kontrolních objemů v % pro bloky trubek v ose y (okraje)	tpo	0,36
Počet kontrolních objemů v % pro bloky trubek v ose y (střed)	tps	0,28
Simple grading pro bloky trubek v ose y (okraj trubek blíže vstupní hlavy výměníku)	tgov	1,4
Simple grading pro bloky trubek v ose y (okraj trubek blíže výstupní hlavy výměníku)	tgom	0,6
Simple grading pro bloky trubek v ose y (střed trubek)	tgs	1

Nastavení „simpleGrading" pro bloky hlav výměníku v ose y je rovno jedné z důvodu nutnosti návaznosti kontrolních objemů hrdel. Nastavení kontrolních objemů pro bloky trubek je rozděleno na třetiny tak, aby byl jejich počet u vstupu do trubek a jejich výstupu vyšší, než v jejich středu. Je tomu tak z důvodu požadované vyšší míry přesnosti v těchto kritických místech.

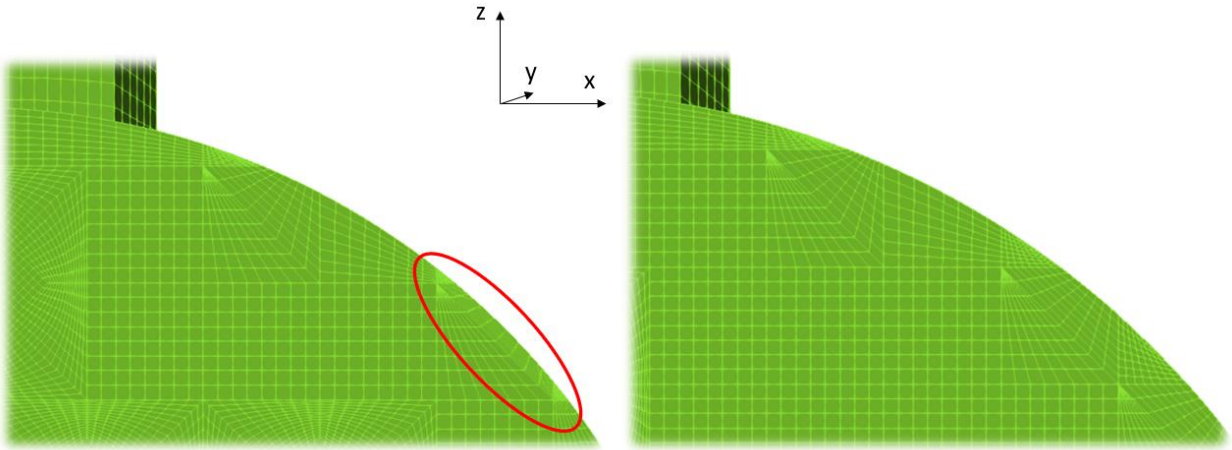
6.3 Test vytvořené aplikace

Správnost vytvořené sítě pro CFD software OpenFOAM je třeba ověřit. Pro ověření byl vytvořen model se zkušebními parametry definovanými níže v tabulce 6.3.

Tab. 6.3 Vstupní parametry zkušebního modelu

Název parametru	Hodnota parametru	Jednotky	Datový typ
Počet řad	[0, 1, 2]	[-]	list
Počet trubek	[5, 4, 1]	[-]	list
Délka trubek	1500	[mm]	int
Šířka hlav výměníku	300	[mm]	int
Úhlopříčka vnitřního bloku	80	[mm]	int
Poloměr trubek	50	[mm]	int
Hrana superbloku	120	[mm]	int
Poloměr hlav výměníku	450	[mm]	int
Šířka hrdel	150	[mm]	int
Výška hrdel	100	[mm]	int

Pro uspořádání svazku trubek byla zvolena kombinace obou možností „v zákrytu" a „prostřídané". Poloměr hlav byl zvolen tak, aby bloky, které jsou dle válcových stěn hlav výměníku ořezány, byly pro kvalitu sítě vůči ostatním blokům dostatečně velké. Na obr. 52 vlevo je možné vidět sníženou kvalitu sítě danou většími rozdíly ve velikostech bloků hlav výměníku u válcových stěn vůči blokům, které jsou dále od těchto stěn. Na obr. 52 vpravo je pak vidět lepší volba poloměru hlav, která zachová nižší rozdíl velikosti těchto bloků a vede k dostatečné kvalitě celé sítě. Jedná se o omezení dané aplikací blockMesh.



Obr. 52 Vliv poloměru hlav výměníku na blokovou strukturu sítě

Kvalitu sítě je pak možné ověřit aplikací OpenFOAM s názvem checkMesh. Byl otestován model jak bez odsazení hrdel od hrany hlav, tak s jejich odsazením a použitím funkce mergePatchPairs. Bylo zjištěno, že na kvalitu sítě tento parametr odsazení nemá vliv (při zachování návaznosti kontrolních objemů). Výsledky kontroly kvality sítě u zkušebního modelu je možné vidět na obr. 53.

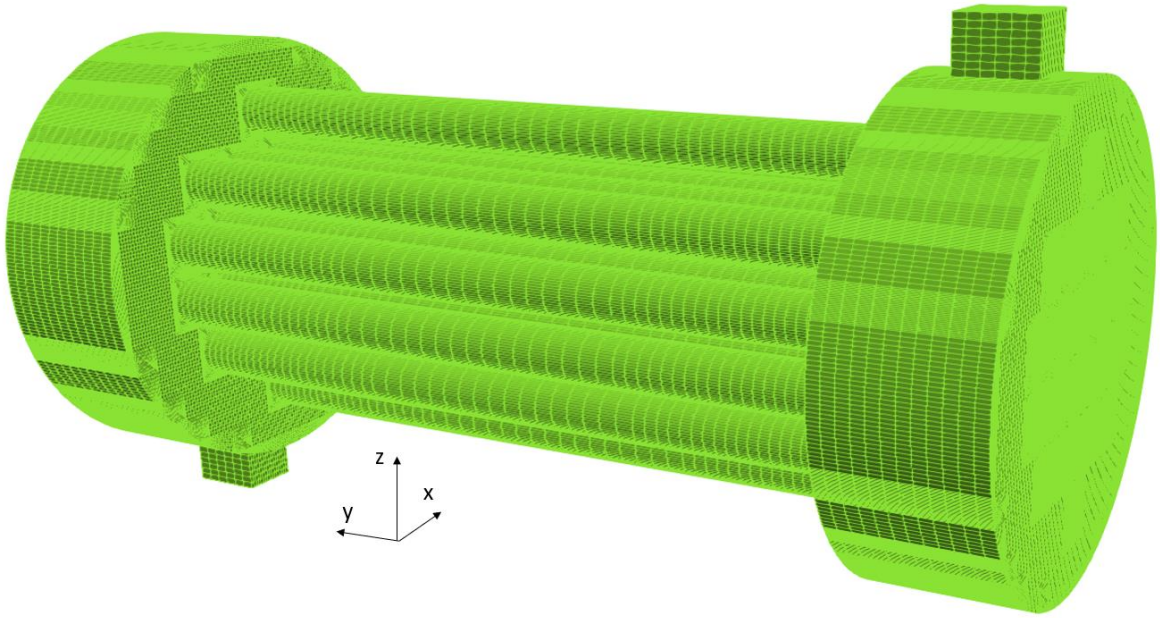
```
Checking geometry...
    Overall domain bounding box (-0.45 0 -0.55) (0.45 2.1 0.55)
    Mesh has 3 geometric (non-empty/wedge) directions (1 1 1)
    Mesh has 3 solution (non-empty) directions (1 1 1)
    Boundary openness (1.22801e-16 2.81521e-15 1.12635e-16) OK.
    Max cell openness = 2.89121e-16 OK.
    Max aspect ratio = 144.729 OK.
    Minimum face area = 8.0191e-07. Maximum face area = 0.000325056.  Face area magnitudes OK.
    Min volume = 2.00478e-08. Max volume = 2.42615e-06.  Total volume = 0.620538.  Cell volumes OK.
    Mesh non-orthogonality Max: 81.4189 average: 10.8951
   *Number of severely non-orthogonal (> 70 degrees) faces: 4416.
    Non-orthogonality check OK.
  <<Writing 4416 non-orthogonal faces to set nonOrthoFaces
    Face pyramids OK.
    Max skewness = 1.98317 OK.
    Coupled point location match (average 0) OK.

Mesh OK.

End
```

Obr. 53 Ověření kvality vytvořené sítě

K vizualizaci byl použit nástroj paraview. Vytvořená síť geometrie zkušebního modelu je znázorněna na obr. 54.



Obr. 54 Vizualizace sítě zkušebního modelu

Ze zvizualizované sítě modelu je vidět dle vlastního definování blokové rozdělení sítě. Je zároveň vidět, že jednotlivé bloky svými kontrolními objemy na sebe navazují. Správný je také výsledný tvar modelu a nastavený„simpleGrading". Síť ani po vizualizaci nevykazuje žádné známky snížené kvality.

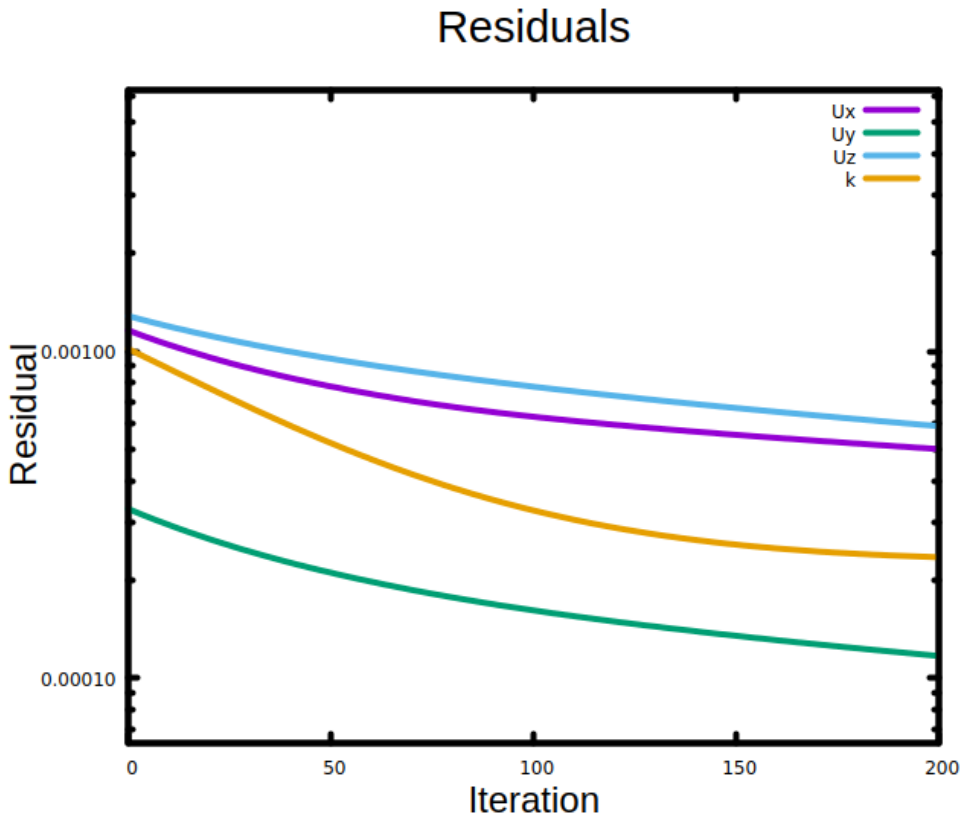
Po vytvoření sítě pomocí aplikace blockMesh bylo možné provést zkušební simulaci proudění. Pro zvolené rozměry byl zvolen vstupní průtok vody 4,5 kg / s tak, aby rychlost ve vstupním hrdlu při hustotě pracovní látky 1000 kg / m^3 byla 0,25 m / s. Všechny zvolené parametry pro výpočet se nachází v tabulce 6.4.

Tab 6.4 Základní zvolené parametry pro zkušební simulaci proudění

Název parametru	Hodnota	Jednotky
Vstupní hmotnostní tok	4,5	[kg/s]
Hustota média	1000	[kg/m^3]
Relaxační faktor pro tlak	0,3	[-]
Relaxační faktor pro rychlost	0,7	[-]
Zvolený řešič	simpleFOAM	[-]
Čas výpočtu	15	[s]
Časový krok	0,05	[s]
Tolerance proměnných	$1e^{-6}$	[-]

U simpleFOAM, který simuluje ustálené proudění se časem výpočtu společně s časovým krokem definuje počet iterací.

Zkušební výpočet konvergoval. Výsledné hodnoty reziduí rychlostí v jednotlivých složkách dle směru (Ux, Uy, Uz) a turbulentní kinetické energie (k) je možné vidět na obr. 55.

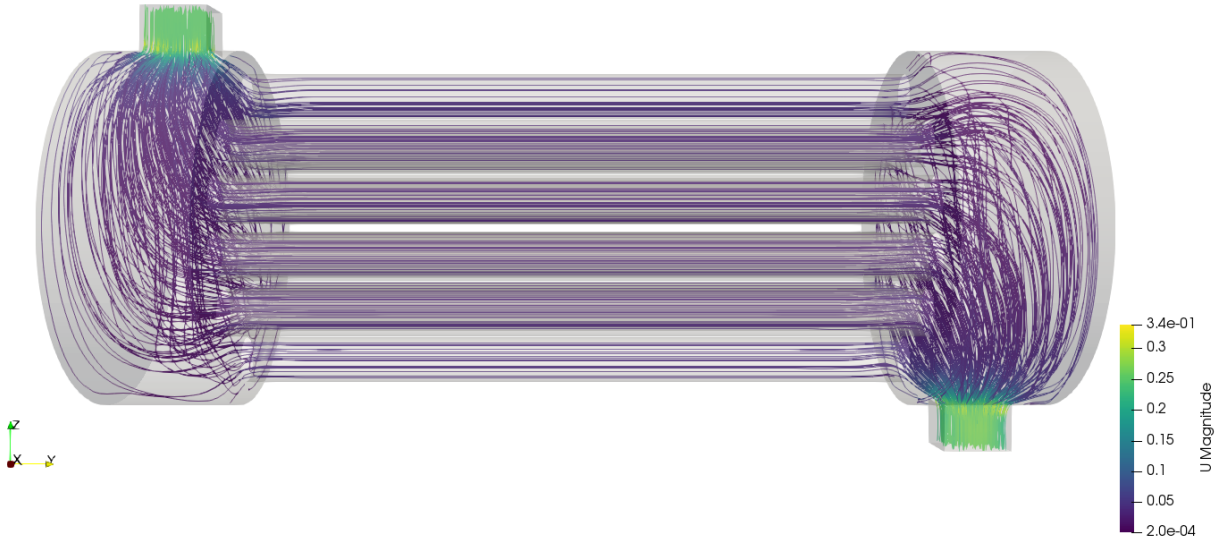


Obr. 55 Výsledné hodnoty reziduí výpočtu zkušebního modelu

Simulovalo se ustálené proudění. Pro ověření kvality sítě se analyzovalo výsledné rychlostní pole a charakter proudnic. Zjišťovalo se, jestli jsou správně definovány hranice domény, jestli proudící látka skrz hranice neprotéká nebo není zastavena stěnou v prostoru, kudy má protékat.Výsledné rychlostní pole je v podélném řezu výměníku znázorněno na obr. 56.

Obr. 56 Rychlostní pole zkušebního modelu

Na obr. 57 jsou na zkušebním modelu znázorněny proudnice.



Obr. 57 Proudnice zkušebního modelu

Z výsledného rychlostního pole na obr.56 je vidět, že nejvyšší rychlost proudící látky je v zúžených místech hrdel a rychlost je vyšší i v oblasti přechodu z hlavy výměníku do trubek. Nízká rychlost je v místě stěny hlavy výměníku po přechodu ze vstupního hrdla do válcové hlavy výměníku. Z charakteru proudnic na obr. 57 je vidět, že proudící látka proudí všemi trubkami a ze vstupního hrdla do výstupního bez výskytu anomálií. Proudnice neprostupují nikde skrz hranice domény ani

nejsou přerušeny nebo nemění svůj tvar neočekávanou překážkou. Z analýzy vyplývá dostatečná kvalita výpočetní sítě.

7 Závěr

Úvodní část této práce pojednává o základech CFD, důvodech jeho použití a související nutnosti vytvářet výpočetní sítě. V první podkapitole jsou definovány základní typy sítí a oblasti jejich obvyklého využití. Další podkapitoly se zaměřují na vyhodnocování kvality sítí a to zejména z pohledu tvaru a velikosti jednotlivých kontrolních objemů. Následně je rozvinuta problematika kvality sítí z pohledu jejich komplexní optimalizace. Obsah těchto kapitol byl výchozím bodem pro vlastní tvorbu kvalitních výpočetních sítí pro software OpenFOAM. Jsou také stručně popsány výpočetní metody CFD, kde každá metoda je vhodná pro jiný typ sítě a jinak diskretizuje hlavní řídící rovnice proudění, které jsou před popisem těchto metod stručně uvedeny.

Druhá část práce je věnována srovnání dostupných síťových generátorů a krátkému popisu jejich funkcí. Je tak poskytnut přehled o možnostech tvorby sítí a v případě placených softwarů i jejich přibližná cena. Většina placených softwarů spolupracuje s CAD softwary umožňujícími generování 3D modelů pomocí grafického rozhraní. Volně dostupné softwary využívají pro generování modelu spíše skriptování. Ve srovnání se skriptováním je použití CAD softwarů výhodné hlavně u geometrií složitějších tvarů. Pro geometrie jednodušších tvarů složené z velkého počtu podobných částí, jako je například trubkový svazek výměníku tepla, je rychlejší a snazší danou geometrii vytvořit pomocí skriptování. Rychlejší a snazší je i parametrizace modelu, která se narozdíl od CAD modelování provádí pouze změnou omezeného počtu vstupních parametrů.

Dále jsou v práci uvedeny základní informace o softwaru OpenFOAM a jeho možnostech. OpenFOAM obsahuje pro řešení různých typů úloh (turbulentní proudění, kavitace, přenos tepla a podobně) vlastní knihovnu řešičů. Kromě toho je popsána aplikace blockMesh a její možnosti a způsoby generování výpočetní sítě. Jejím hlavním omezením je možnost generovat pouze blokové sítě s kontrolními objemy ve tvaru čtyřbokých nebo trojbokých hranolů. Toto omezení muselo být vzato do úvahy při následné tvorbě počítačového kódu, jelikož u složitých geometrických tvarů může vést nevhodně zvolená bloková struktura k nižší kvalitě výpočetní sítě. Například síť reprezentující každou jednotlivou trubku v modelovaném trubkovém prostoru výměníku musela být na příčném řezu rozdělena na celkem dvacet bloků, aby tvary a návaznosti kontrolních objemů zajistily dostatečnou kvalitu sítě.

Následující část práce je věnována popisu vyvinuté aplikace. Tato je napsána v programovacím jazyce Python a síť generuje ve tvaru, který lze následně využít jako vstupní data pro aplikaci blockMesh (součást OpenFOAM). Většina kódu je vytvořena pomocí objektově orientovaného programování. To umožnilo variabilní vytváření sítě a parametrizaci, tedy možnost snadno volit uspořádání svazku trubek (v „zákrytu" a „prostřídané"), počet řad ve svazku s počtem trubek v jednotlivých řadách atd. Výhodou je také možnost upravovat vyvinutou aplikaci dle požadavků toho kterého uživatele, neboť tato je poskytnuta ve formě zdrojového kódu. Další výhodou je cílení aplikace na software OpenFOAM, který je zdarma. Vytvořená aplikace generuje po spuštění kromě samotného datového souboru pro blockMesh i obrázek ve 2D, který uživateli umožňuje rychlou vizuální kontrolu rozměrů, uspořádání trubkového svazku a velikostí jednotlivých bloků. Zmíněný obrázkový výstup měl velkou hodnotu i při vytváření dané aplikace, kde sloužil primárně pro odhalování chyb během ladění kódu, a může být stejně nápomocný v případě, že se uživatel rozhodne kód upravit dle svých požadavků.

Vzhledem k blokové struktuře sítě je diskretizace geometrie, až na hrdla výměníku, aplikací provedena nejprve ve čtyřech příčných rovinách hlav výměníku. Tím se zajistí návaznost jednotlivých kontrolních objemů mezi bloky, což je pro dostatečnou kvalitu sítě důležité. Ze stejného důvodu je svazek trubek tvořen pomocí čtvercových superbloků, popsaných v kapitole 6.2.1, které jsou rozděleny tak, aby na sebe navazovaly v obou uvažovaných typech uspořádání trubek („v zákrytu“, „prostřídané”). Analogicky jsou pro zajištění návaznosti rozděleny hlavy výměníku, přičemž adekvátní ořezání bloků po obvodech hlav je opět provedeno za dodržení požadavků aplikace blockMesh.

Nakonec je pozornost věnována kontrole správnosti sítě vytvořené pro vhodně zvolenou testovací geometrii trubkového prostoru výměníku tepla. Zde bylo ověřeno jak vizuálně, tak pomocí nástroje checkMesh, že kvalita výpočtové sítě je přijatelná. Zkušební simulace proudění v OpenFOAM také neodhalila žádné potíže (výpočet konvergoval dle očekávání, v proudění se nevyskytovaly žádné anomálie), čili je zřejmé, že výpočetní síť opravdu byla vytvořena korektně.

8 Reference

[1] Introduction to CFD Analysis [online]. Fluent, 2002 [cit. 2022-05-12]. Dostupné z: https://vdoc.pub/documents/fluent-introduction-to-cfd-analysis-5dndnbgipos0

[2] FERZIGER, Joel H. a Milovan PERIC. Computional Methods for Fluid Dynamics. 3rd edition. NewYork: Springer Verlag Berlin Heidelberg, 2002. ISBN 3-540-42074-6.

[3] WENDT, John F., John D. ANDERSON, Joris DEGROOTE, Gérard DEGREZ, Erik DICK, Roger GRUNDMANN a Jan VIERENDEELS. Computional Fluid Dynamics: An Introduction. 3rd edition. Berlin: Springer Verlag Berlin Heidelberg, 2009. ISBN 978-3-540-85055-7.

[4] VERSTEEG, H. K. a W. MALALASEKERA. An Introduction to computional fluid dynamics: The finite volume method. Anglie: Longman Group Ltd 1995, 1995. ISBN 0-582-21884-5.

[5] ČERMÁK, Libor a Rudolf Hlavička. Numerické metody. Brno: AKADEMICKÉ NAKLADATELSTVÍ CERM, s.r.o. Brno, 2016. ISBN 978-80-214-5437-8.

[6] All there is to know about different mesh types in CFD! ManchesterCFD group [online]. Menchester: Masoud Meskin, 2021 [cit. 2022-05-11]. Dostupné z: https://www.manchestercfd.co.uk/blog

[7] Basics of Grid Generation for CFD Analysis. LearnCAx [online]. Pune (India): Centre for Computational Technologies (CCTech) [cit. 2022-05-11]. Dostupné z: https://www.learncax.com/knowledge-base/blog/by-category/cfd/basics-of-grid-generation-for-cfd-analysis

[8] Wall Boundary Condition. SimScale [online]. Mnichov: SimScale, 2022-02-27 [cit. 2022-05-11]. Dostupné z: https://www.simscale.com/docs/simulation-setup/boundary-conditions/wall/

[9] Methods for Grid Generation in CFD Simulations. Cadence [online]. Cadence [cit. 2022-05-11]. Dostupné z: https://resources.system-analysis.cadence.com/blog/msa2021-methods-for-grid-generation-in-cfd-simulations

[10] Mesh Quality. SimScale [online]. Mnichov: SimScale, 2021 [cit. 2022-05-11]. Dostupné z: https://www.simscale.com/docs/simulation-setup/meshing/mesh-quality/

[11] 5 Tips on How to Create a High-Quality Mesh. SimScale [online]. Mnichov: Barry Ho, 2021 [cit. 2022-05-11]. Dostupné z: https://www.simscale.com/blog/2018/03/tips-high-quality-mesh/

[12] Navier–Stokesova rovnice. Předmět Hydromechanika [online]. VUT Brno: Fluidní inženýrství, 2009, 5 str. [cit. 2022-05-11]. Dostupné z: https://moodle.vut.cz/

[13] Finite volume. CFD Online [online]. CFD Online [cit. 2022-05-11]. Dostupné z: https://www.cfd-online.com/Wiki/Finite_volume

[14] Finite volume. Stéphane Sanchi Computational Fluid Dynamics Solutions [online]. Préverenges (Švýcarsko): Stéphane Sanchi [cit. 2022-05-11]. Dostupné z: http://stephanesanchi.ch/index.php/other-tools/ansys-icem-cfd

[15] Ansys ICEM CFD. TechSoft Engineering [online]. Česká Republika: TechSoft Engineering [cit. 2022-05-11]. Dostupné z: https://www.techsoft-eng.cz/software/ansys-icem-cfd

[16] Ansys Fluent Fluid Simulation Software. Ansys [online]. Pensylvánie (USA): Ansys [cit. 2022-05-11]. Dostupné z: https://www.ansys.com/products/fluids/ansys-fluent#tab1-2

[17] GridPro. CFD Online [online]. CFD Online [cit. 2022-05-11]. Dostupné z: https://www.cfd-online.com/Forums/gridpro/12510-gridpro.html

[18] Program Development Company. GridPro [online]. New York (USA): Program Development Company, 2022 [cit. 2022-05-11]. Dostupné z: https://www.gridpro.com/About/

[19] Volute Optimization Study CAESES. CFD support [online]. Praha: CFD support, 2022 [cit. 2022-03-15]. Dostupné z: https://www.cfdsupport.com/download/Volute-Optimization-Study-CAESES-GridPro-TCFD.pdf

[20] Fidelity Pointwise for Computational Fluid Dynamics Meshing. Cadence [online]. Cadence [cit. 2022-05-11]. Dostupné z: https://www.cadence.com/en_US/home/tools/system-analysis/computational-fluid-dynamics/pointwise.html#mesh-types

[21] HyperMesh. INDIELEC [online]. Valencie (Španělsko): INDIELEC [cit. 2022-05-11]. Dostupné z: https://www.indielec.com/hypermesh-cms-4-51-418/

[22] GEUZAINE, Christophe a Jean-Francois REMACLE. Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. International Journal for Numerical Methods in Engineering [online]. John Wiley & Sons, Ltd. , 2009, 24 str. [cit. 2022-05-11]. Dostupné z: http://gmsh.info//doc/preprints/gmsh_paper_preprint.pdf

[23] Open-source code with lots of scripting options. In: G2 - Business software review [online]. Alfonso S, 2021 [cit. 2022-05-11]. Dostupné z: https://www.g2.com/products/gmsh/reviews/gmsh-review-4886379

[24] Second OpenFOAM User Conference Berlin Oct 7-9, 2014. ESI Group [online]. Paříž: ESI Group, 2014, 3 str. [cit. 2022-05-11]. Dostupné z: https://www.esi-group.com/sites/default/files/news-release/486/openfoam_conference_2014_pr_en.pdf

[25] Welcome to the enGrid wiki! GitHub [online]. Mark Olesen, 2017 [cit. 2022-05-11]. Dostupné z: https://github.com/enGits/engrid/wiki/

[26] Meshing with SMESH. Salome [online]. Salome, 2021 [cit. 2022-05-12]. Dostupné z: https://www.salome-platform.org/?page_id=374

[27] The CFD Story. Autodesk [online]. Autodesk, 2019 [cit. 2022-05-12]. Dostupné z: https://knowledge.autodesk.com/support/cfd/learn-explore/caas/CloudHelp/cloudhelp/2014/ENU/SimCFD/files/GUID-79AA9A74-FCAC-42CE-B835-0BF3DEA36EFA-htm.html

[28] About OpenFOAM. CFD Direct the Architects of OpenFOAM [online]. Caversham (UK): CFD Direct [cit. 2022-05-12]. Dostupné z: https://cfd.direct/openfoam/about/

[29] OpenFOAM. OpenFOAM [online]. OpenFOAM, 2022 [cit. 2022-05-12]. Dostupné z: https://www.openfoam.com/

[30] OpenFOAM: The Open Source CFD Toolbox (User Guide). Verze v2006. OpenCFD Limited, 2020.

[31] MARIC, Tomislav, Jens HOPKEN a Kyle MOONEY. The OpenFOAM Technology Primer. Sourceflux UG, 2014. ISBN 978-3-00-046757-8.

[32] CARDIFF, Philip. Introduction to Meshing in OpenFOAM [online]. Dublin (Irsko): ResearchGate, 2017, 73 str. [cit. 2022-05-12]. Dostupné z: https://www.researchgate.net/publication/325218270_Introduction_to_Meshing_in_OpenFOAM

[33] Mesh generation using blockMesh. In: Mesh generation using blockMesh [online]. Genoa (Itálie): Wolf Dynamics, 66 str. [cit. 2022-05-12]. Dostupné z: http://www.wolfdynamics.com/wiki/meshing_OF_blockmesh.pdf

[34] STEHLÍK, P., J. KOHOUTEK a J. NĚMČANSKÝ. Tepelné pochody: Výpočet výměníku tepla. Brno: AKADEMICKÉ NAKLADATELSTVÍ CERM, s.r.o. Brno, 1991. ISBN 802-1403632.

[35] All About Shell And Tube Heat Exchangers - What You Need To Know. Thomas [online]. USA: Thomas [cit. 2022-05-12]. Dostupné z: https://www.thomasnet.com/articles/process-equipment/shell-and-tube-heat-exchangers/

[36] JEGLA, Zdeněk. Úvod do předmětu KNP, navrhování trubkových výměníků tepla: Přednáška č. 1 předmětu: „Navrhování procesních a energetických systémů“. 2022.

[37] KEEPING YOUR COOL: MANAGING HEAT REJECTION REQUIREMENTS FOR YOUR AIR COMPRESSOR SYSTEM. In: VMAC - Vehicle Mounted Air Compressors [online]. Nanaimo (Kanada): VMAC - Vehicle Mounted Air Compressors [cit. 2022-05-12]. Dostupné z: https://www.vmacair.com/blog/keeping-cool-managing-heat-rejection-requirements-air-compressor-system/

[38] PETINRIN, Moses Omolayo a Ademola DARE. Performance of Shell and Tube Heat Exchangers with Varying Tube Layouts. Anglie: British Journal of Applied Science & Technology, 2015. ISSN 2231-0843.

[39] RAI, Durgesh a Sohail BUX. To study of experimental analysis of tube bundle geometry in STHE. Indie: P.G Research Scholar, Department of Thermal Engineering, Agnos college of Technology, RKDF University Bhopal, M.P, India, 2015. ISSN 2395-4396.

[40] BAHRAMPOURY, Rasool, Ali BEHBAHANINIA, Abdollah SHADARAM a Ehsan TAGHDIRI. Comparison between In-line and Staggered Arrangements in Heat Recovery Steam Generators [online]. 1st edition, USA: American University of Sharjah, 2009, 6 str. [cit. 2022-05-12]. Dostupné z: https://www.researchgate.net/publication/263853515

[41] What is Python? Executive Summary. Python [online]. Python [cit. 2022-05-12]. Dostupné z: https://www.python.org/doc/essays/blurb/

[42] Classes. Python 3.10.4 documentation [online]. Python, 2022 [cit. 2022-05-12]. Dostupné z: https://docs.python.org/3/tutorial/classes.html

[43] Co je to OOP a proč se to mám učit? ENGETO Academy [online]. Brno: Dominik Remetei, 2022 [cit. 2022-05-12]. Dostupné z: https://engeto.cz/blog/programovani/co-je-to-oop/

[44] Python Data Types. Programiz: Learn to Code for Free [online]. Programiz [cit. 2022-05-12]. Dostupné z: https://www.programiz.com/python-programming/variables-datatypes

[45] Hybrid Meshing Key to Improving CFD Simulation Efficiency | Pointwise Case Study. Cadence [online]. Texas (USA): Cadence [cit. 2022-05-12]. Dostupné z: https://www.pointwise.com/case-studies/5faaf33404d997423c0ac57c

[46] Courant Number. IdealSimulations [online]. Londýn (UK): IdealSimulations [cit. 2022-05-12]. Dostupné z: https://www.idealsimulations.com/resources/courant-number-cfd/

[47] OpenFOAM v6 User Guide: 4.3 Time and data input/output control. CFD Direct the Architects of OpenFOAM [online]. Caversham (UK): CFD Direct [cit. 2022-05-12]. Dostupné z: https://cfd.direct/openfoam/user-guide/v6-controldict/

[48] Standard solvers. OpenFOAM [online]. OpenFOAM [cit. 2022-05-12]. Dostupné z: https://www.openfoam.com/documentation/user-guide/a-reference/a.1-standard-solvers

[49] BYRNE, Richard C. STANDARDS OF THE TUBULAR EXCHANGER MANUFACTURERS ASSOCIATION [online]. 10th edition. New York: Tubular Exchanger Manufacturers Association, 2019 [cit. 2022-05-20]. Dostupné z: https://www.ingeduca.com/wp-content/uploads/2020/11/TEMA-10th-Edition-2019.pdf

9 Seznam použitých zkratek a symbolů

Symbol	Význam	Jednotka
y^+	bezrozměrná vzdálenost od stěny	[-]
y	absolutní vzdálenost středu kontrolního objemu od stěny	[m]
v	kinematická viskozita	[m^2/s]
u_t	třecí rychlost	[m/s]
τ_w	smykové napětí	[MPa]
ρ	hustota média	[kg/m^3]
v	rychlost proudění	[m/s]
t	čas	[s]
A	objemové zrychlení	[m/s^{-2}]
p	tlak	[Pa]
E	Energie tekutiny	[J]
τ	třecí síla	[N]
$\partial(x,y,z)$	vzdálenost v jednotlivých směrech (x, y, z) vztažené na kontrolní objem	[mm]
u	kinetická energie	[J]
C	Courantovo číslo	[-]
h	charakteristický rozměr kontrolního objemu	[mm]
$l_{(0,1,2,3)}$	vzdálenosti příčných rovin hlav výměníku na ose y	[mm]
$U_{(x,y,z)}$	rychlost pracovní látky ve složkách jednotlivých směrů (x, y, z)	[m/s]
k	turbulentní kinetická energie	[J]

10 Seznam příloh

Generátor_sítě_trubkového_svazku_ pro_OpenFOAM.py

blockMeshDict_start.txt