VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta strojního inženýrství
Ústav procesního inženýrství

Ing. Vojtěch Turek, Ph.D.

Efektivní analýzy distribuce toku v zařízeních na výměnu tepla

Efficient analyses of flow distribution in heat transfer equipment

HABILITAČNÍ PRÁCE

v oboru Konstrukční a procesní inženýrství

Brno

2020

Abstrakt

V procesních a energetických zařízeních na výměnu tepla má charakter proudění pracovních látek významný vliv na spolehlivost. Pro korektní návrh každého takového zařízení je proto nutné provést analýzu distribuce toku a na jejím základě zhodnotit pravděpodobnost výskytu vyšší míry zanášení, nerovnoměrného tepelného a mechanického namáhání trubek ve svazku a podobně. Běžnou inženýrskou praxí ovšem je předpokládat rovnoměrné rozdělení tekutiny. Důvodem pro zmíněné zjednodušení obvykle bývá vysoká výpočetní a časová náročnost simulací prováděných pomocí výpočtové dynamiky tekutin (CFD) a nutnost při hledání vhodné geometrie zařízení vyhodnotit mnoho různých variant. Přinejmenším v rané fázi návrhového procesu by přitom pro eliminaci zcela zřejmě nevhodných geometrií dostatečně dobře posloužil i jiný nástroj, byť méně přesný, zatímco detailní analýzy by stačilo provést u několika málo slibných konfigurací vzešlých z předběžného návrhu. Pomineme-li však různě vyspělé in-house softwary vyvíjené komerčními subjekty, nejsou patřičné nástroje obecně k dispozici.

V této práci jsou diskutovány dva způsoby provádění přibližných analýz procesních či energetických zařízení, konkrétně zjednodušené CFD modelování a modelování založené na analýze metodou konečných prvků (FEA). U prvního zmíněného přístupu jsou uvedena zjednodušení vedoucí k relativně rychlým výpočtům přijatelné přesnosti. Dále je popsáno dodatečné zefektivnění výpočtů vhodnou volbou maticových řešičů a metod předpodmínění a jsou prezentovány výsledky získané pomocí sady více než 50 tisíc testovacích úloh. Tyto potvrzují, že zjednodušené CFD modely jsou pro zamýšlenou aplikaci vhodné. Nakonec jsou nastíněny možnosti ohledně případného budoucího výzkumu, který by mohl vést k dalšímu zpřesnění a zrychlení výpočtů.

Motivací pro výzkum modelování založeného na analýze metodou konečných prvků je skutečnost, že u prostorově rozsáhlých (ačkoliv strukturou vcelku jednoduchých) zařízení, jako jsou například kotle na odpadní teplo, by se i zjednodušené CFD modely mohly ukázat jako výpočtově příliš náročné. Odpovídající výzkum je zatím v rané fázi a zde prezentované výsledky jsou pouze předběžné, nicméně pokud se patřičný přístup ukáže jako perspektivní, je výhledově cílem predikovat kromě charakteru proudění a přenosu tepla také výsledné mechanické namáhání trubkových svazků způsobené nerovnoměrným tepelným zatížením teplosměnných ploch.

Posledním diskutovaným tématem je numerická disipace v klasických CFD modelech, neboť tato se zde řešenou problematikou úzce souvisí a nezanedbatelnou měrou ovlivňuje přesnost získaných dat. Stěžejním bodem je popis implementace *a posteriori* odhadu chyb na základě údajů běžně poskytovaných CFD řešiči a vyvinutých uživatelsky definovaných funkcí (UDF), které lze využít v jednojádrových i vícejádrových simulacích v aplikaci ANSYS Fluent. Přesnost takového způsobu odhadu chyb je ovšem potřeba před nasazením v praxi ještě ověřit, což bude provedeno na základě dat získaných laserovou Dopplerovskou anemometrií (LDA), která jsou dostupná v literatuře.

Klíčová slova

proudění tekutin, CFD, FEA, numerická disipace

Abstract

Reliability of heat transfer equipment used in process and power industries is greatly affected by the character of fluid flow therein. In order to design any such apparatus properly, a flow distribution analysis is required, and the data thus obtained must then be used to evaluate the likelihood of increased rate of fouling, excessive thermal and mechanical loading of the tubes in the bundle, etc. In engineering practice, though, it is common to assume that the fluid is distributed uniformly. The reasons for this simplification usually are that computational fluid dynamics (CFD) simulations are very demanding in terms of both computing power and time and that a lot of different geometry options need to be evaluated while searching for a suitable design. However, a different, less accurate but much faster tool would still be sufficient in the initial phase of the equipment design process, while detailed analyses could be carried out for just the few options yielded by the preliminary evaluation. Yet – if we disregard in-house software packages of various maturities, which are developed by commercial companies – no such tools are generally available.

This thesis discusses two ways to approximately analyse process or power equipment, namely simplified CFD modelling and modelling based on finite element analysis (FEA). As for the former approach, the simplifications leading to relatively fast and sufficiently accurate computations are presented. Additional efficiency improvements via favourable selection of matrix solvers and preconditioning techniques are also described, and the results obtained using a set of more than 50 thousand test cases are presented. These confirm that simplified CFD models are suitable for the intended application. Finally, avenues of possible future research, which could lead to further improvements in accuracy and computational efficiency, are outlined.

The motivation for research of finite element analysis-based modelling is the fact that simplified CFD models could still prove to be too demanding in case of spatially extensive (although geometry-wise fairly simple) equipment such as waste heat recovery boilers. The respective research is yet in its beginning and, therefore, the results presented herein are only preliminary. Should this modelling approach turn out to be usable, it is planned for it to include in addition to fluid flow and heat transfer also mechanical loading of tube bundles resulting from non-uniform thermal loading of heat transfer surfaces.

The last matter discussed in this thesis is numerical dissipation in the classical CFD models because it is closely related to the overarching topic and significantly influences the accuracy of data obtained this way. The pivotal part is the description of the implementation of *a posteriori* error estimation using values commonly provided by CFD solvers and of the developed user-defined functions (UDF), which can be used in both serial and parallel simulations in ANSYS Fluent. Accuracy of this method, however, must first be verified before it is used in practice. This will be done using data obtained via laser Doppler anemometry (LDA), which are available in the literature.

Keywords

fluid flow, CFD, FEA, numerical dissipation

Mé poděkování náleží v prvé řadě prof. Ing. Petru Stehlíkovi, CSc., dr. h. c., na jehož popud jsem se kdysi – zcela neplánovaně – začal věnovat výpočtovému modelování proudění tekutin. Cením si všech jeho hodnotných podnětů a připomínek.

Dále děkuji Ing. Tomáši Létalovi, Ph.D., který přišel s nápadem využít ke zjednodušenému modelování proudění tekutin metodu konečných prvků a nemalou měrou se podílí na patřičném výzkumu.

Velký dík patří také Ing. Dominice Babičce Fialové za její neúnavnou pomoc při přípravě CFD modelů nutných k ověřování vznikajících matematických aparátů.

Obsah

Seznam obrázků

Seznam tabulek

Seznam zkratek

AOCL AMD Optimizing CPU Libraries

API Application Programming Interface

BiCGstab(l) Bi-Conjugate Gradient Method Stabilized with Minimization of Residuals over l-Dimensional Subspaces

BLAS Basic Linear Algebra Subprograms

BSD Berkeley Software Distribution License

CFD Computational Fluid Dynamics

CG Conjugate Gradient Method

CGNR Conjugate Gradient Method on the Normal Residuals

CPU Central Processing Unit

DES Detached Eddy Simulation

DNS Direct Numerical Simulation

DOS Disk Operating System

DTBFMM Dense Tube Bundle Flow Modeller – Matrix version

EJML Efficient Java Matrix Library

FEA Finite Element Analysis

FIFO First-In, First-Out

GMRES Generalized Minimal Residual Method

GPL GNU General Public License

IC Incomplete Cholesky Factorization

ILES Implicit Large Eddy Simulation

ILU Incomplete Lower-Upper Factorization

ILUT Dual-Threshold Incomplete Lower-Upper Factorization

JIT Just-in-Time

LAPACK Linear Algebra PACKage

LDA Laser Doppler Anemometry

LES Large Eddy Simulation

LIFO Last-In, First-Out

LU Lower-Upper

MKL Intel Math Kernel Library

MTJ Matrix Toolkits Java

NaN Not a Number

PISO Pressure Implicit with Splitting of Operators

QMR Quasi-Minimal Residual Method

QUICK Quadratic Upstream Interpolation for Convective Kinetics

RAM Random-Access Memory

SAS Scale-Adaptive Simulation

SIMPLE Semi-Implicit Method for Pressure-Linked Equations

SIMPLEC SIMPLE Consistent

SIMPLER SIMPLE Revised

Spyder Scientific Python Development Environment

SSOR Symmetric Successive Over-Relaxation

TUI Text User Interface

TVD Total Variation Diminishing

UDF User-Defined Function

UDM User-Defined Memory

UJMP Universal Java Matrix Package

(U)RANS (Unsteady) Reynolds-Averaged Navier-Stokes equations

VTK Visualization Toolkit

Seznam symbolů

Kapitola 2 – Zjednodušené CFD modelování

A stěna kontrolního objemu

a koeficient v diskretizované transportní rovnici, $\mathrm{kg\,s^{-1}}$ †

C konstantní člen v diskretizované transportní rovnici pro skalární veličinu ϕ, $\mathrm{kg\,s^{-1}}[\phi]$

$\mathbf{c}$ vektor definovaný centroidy dvou sousedních buněk výpočetní sítě

D difuzní vodivost, $\mathrm{kg\,s^{-1}}$

d_1 vnitřní průměr trubky, m

d_2 vnější průměr trubky, m

$\mathbf{f}$ vektor definovaný centroidem buňky výpočetní sítě a centroidem některé její stěny

F konvektivní hmotnostní tok, $\mathrm{kg\,s^{-1}}$

F_q koeficient pro škálování měrného tepelného toku

I iterace CFD řešiče

i iterace maticového řešiče

i_max maximální povolený počet iterací maticového řešiče

k kinetická energie turbulence, $\mathrm{m^2\,s^{-2}}$

k_μ koeficient ve vztahu pro virtuální dynamickou viskozitu

L délka hrany náhradního čtvercového průřezu trubky ve zjednodušené výpočetní síti sestávající pouze z kvádrových buněk, m

l stupeň MR-polynomů („Minimal Residual polynomials“) v BiCGstab(l)

L_i délka hrany $i$té buňky ve směru od stěny ke středu náhradního čtvercového průřezu trubky ve zjednodušené výpočetní síti sestávající pouze z kvádrových buněk, m

N počet buněk napříč průřezem trubky ve zjednodušené výpočetní síti sestávající pouze z kvádrových buněk

n řád zaplnění neúplného rozkladu

† Vzhledem k charakteru diskutovaného způsobu modelování odpovídají všechny zde uvedené jednotky 3D úloze.

$\mathbf{n}$ normálový vektor stěny buňky výpočetní sítě

p tlak, Pa

Pe Pécletovo číslo

R skutečné (neškálované) reziduum pro veličinu ϕ, $[\phi]$

r škálované reziduum

$S(\phi)$ zdrojový člen v transportní rovnici pro skalární veličinu ϕ, $\mathrm{kg\,s^{-1}}[\phi]$

$\overline{S}$ měrná střední hodnota zdrojového členu $S(\phi)$, $\mathrm{kg\,m^{-3}\,s^{-1}}[\phi]$

S_c konstantní část linearizovaného zdrojového členu $S(\phi)$, $\mathrm{kg\,s^{-1}}[\phi]$

S_p koeficient hodnoty veličiny ϕ v linearizovaném zdrojovém členu $S(\phi)$, $\mathrm{kg\,s^{-1}}$

t čas, s

t_avg průměrný výpočetní čas potřebný k dosažení zkonvergovaného řešení, s

t_B výpočetní čas odpovídající kombinaci nepředpodmíněných řešičů, s

t_min minimální výpočetní čas pozorovaný s určitou kombinací výpočetní sítě a numerických metod, s

t_P výpočetní čas odpovídající kombinaci předpodmíněných řešičů, s

$\mathbf{u} = (u, v, w)$ vektor rychlosti proudění, $\mathrm{m\,s^{-1}}$

V kontrolní objem

W váha ve vážené metodě nejmenších čtverců

α koeficient v trendu pro relaxační faktor ω_R

β koeficient v trendu pro relaxační faktor ω_R

Γ difuzní koeficient, $\mathrm{kg\,m^{-1}\,s^{-1}}$

Δt délka časového kroku, s

ΔV objem kontrolního objemu V, $\mathrm{m^3}$

δx velikost kontrolního objemu ve směru osy x, m

$\hat{\epsilon}$ pomocná hodnota ve vztahu pro výpočet aktuální hodnoty relativní tolerance

ϵ_a absolutní tolerance

ϵ_b limit používaný pro detekci rozpadu řešiče

ϵ_d divergenční tolerance

ϵ_r relativní tolerance

θ_max maximální vnitřní úhel v buňce výpočetní sítě, °

θ_min minimální vnitřní úhel v buňce výpočetní sítě, °

κ růstový faktor

μ dynamická viskozita, Pa s, resp. střední hodnota v histogramu hmotnostních průtoků, $\mathrm{kg\,s^{-1}}$

$\hat{\mu}$ virtuální dynamická viskozita, Pa s

ρ hustota, $\mathrm{kg\,m^{-3}}$

σ směrodatná odchyka v histogramu hmotnostních průtoků, $\mathrm{kg\,s^{-1}}$

ϕ obecná skalární veličina

ω relaxační parametr v symetrické neúplné relaxaci

$\boldsymbol{\omega}$ vorticita, $\mathrm{s^{-1}}$

Indexy

0 hodnota z předchozího časového kroku

E ve východním uzlu

e na východní stěně

F pro dopředný chod symetrické neúplné relaxace

N v severním uzlu

n na severní stěně

nb pro sousední uzly

P v aktuálním uzlu

R pro zpětný chod symetrické neúplné relaxace

S v jižním uzlu

s na jižní stěně

W v západním uzlu

w na západní stěně

ϕ pro veličinu ϕ

Kapitola 3 – Modelování založené na principu FEA

A plocha příčného průřezu kanálu, m^2

I iterace FEA řešiče

$\mathbf{K} = (k_{ij})$ matice poddajnosti, m s

$\dot{m}$ hmotnostní tok, $kg\,s^{-1}$

$\dot{\mathbf{m}}$ vektor součtů hmotnostních toků, $kg\,s^{-1}$

$\dot{m}_{CFD}$ hmotnostní tok zjištěný detailním CFD modelem, $kg\,s^{-1}$

$\dot{m}_{corr}$ hmotnostní tok po korekčním kroku, $kg\,s^{-1}$

$\dot{m}_{max}$ horní mez pro hodnotu hmotnostního toku při užití metody bisekce, $kg\,s^{-1}$

$\dot{m}_{min}$ dolní mez pro hodnotu hmotnostního toku při užití metody bisekce, $kg\,s^{-1}$

$\dot{m}_{pre}$ hmotnostní tok z predikčního kroku, $kg\,s^{-1}$

p tlak, Pa

$\mathbf{p}$ vektor tlaků, Pa

R_A poměr ploch příčných průřezů kanálů ve složeném prvku

R_v poměr rychlostí proudění tekutiny jednotlivými kanály ve složeném prvku

v rychlost proudění tekutiny kanálem, $m\,s^{-1}$

Δp rozdíl mezi tlaky v koncových uzlech hrany výpočetní sítě, Pa

ϵ_R relativní chyba vůči datům z detailního CFD modelu, %

ρ hustota, $kg\,m^{-3}$

Indexy

I	v Ité iteraci FEA řešiče	$i, \ldots, n$	pro $i$tý, …, $n$tý uzel

Kapitola 4 – Numerická disipace v CFD modelech

A	stěna kontrolního objemu
$\overline{a}$	průměrná hodnota spočtená ze sady $\{a_1, a_2, \ldots, a_N\}$
E^{kin}	kinetická energie, kg m^2 s^{-2}
e^{kin}	měrná kinetická energie, m^2 s^{-2}
F^{ac}	akustický tok, kg m^2 s^{-3}
F^{kin}	tok kinetické energie, kg m^2 s^{-3}
F^{vis}	viskózní tok, kg m^2 s^{-3}
N	rozsah sady hodnot, z nichž je počítán průměr
$\mathbf{n} = (n_1, n_2, n_3)$	normálový vektor stěny
p	tlak, Pa
t	aktuální výpočtový čas, s
$\mathbf{u} = (u_1, u_2, u_3)$	vektor rychlosti proudění, m^2 s^{-1}
V	kontrolní objem
∂V	hranice kontrolního objemu V
W^{p}	práce tlakových sil, kg m^2 s^{-3}
y^+	bezrozměrná stěnová vzdálenost
ΔA	plocha stěny A, m^2
ΔE^{kin}	změna kinetické energie, kg m^2 s^{-2}
Δt	délka časového kroku, s
ΔV	objem kontrolního objemu V, m^3
δ_{ij}	Kroneckerovo delta
ϵ	hodnota disipační funkce, kg s^{-2}
ϵ^{n}	reziduum diskretizované integrální rovnice pro kinetickou energii, kg m^2 s^{-3}
ϵ^{vis}	viskózní disipace, kg m^2 s^{-3}
μ	efektivní dynamická viskozita, Pa s
ν	efektivní kinematická viskozita, m^2 s^{-1}
ν^{n}	numerická kinematická viskozita, m^2 s^{-1}

Indexy

A	na stěně A	$i, \ldots, k$	$i$tá, …, $k$tá složka vektoru
D	v subdoméně D	V	v kontrolním objemu V

Kapitola

1

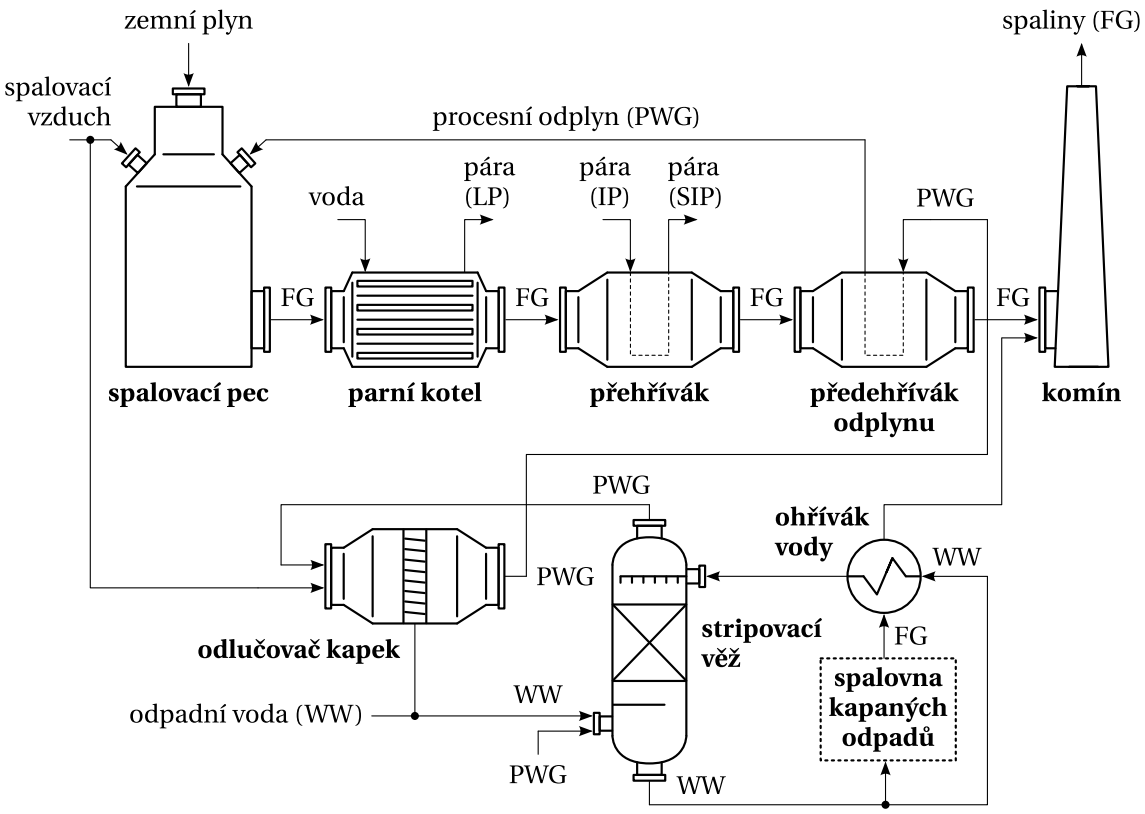
Úvod

Matematické modely proudění tekutin a přenosu tepla jsou z pohledu návrhu procesních či energetických zařízení naprosto nepostradatelnými nástroji. Typicky jsou přitom využívány softwary implementující modely standardizovaných aparátů (například HTRI Xchanger Suite [1], Aspen Exchanger Design & Rating [2] a podobně). Je-li ovšem potřeba navrhnout zařízení více či méně vzdálené standardním geometriím, nemusí být patřičné softwary použitelné buď vůbec, nebo pouze v omezené míře, a to z důvodu nutnosti aproximovat navrhovaný aparát nejbližším standardizovaným typem. Toto zcela zřejmě nemůže vést k příliš přesným výsledkům [A1]. V mnoha případech tedy nezbývá než využít univerzální, avšak výpočtově velmi náročné modelování pomocí výpočtové dynamiky tekutin („Computational Fluid Dynamics“, CFD).

V zařízeních na výměnu tepla bývá proud pracovní látky obvykle rozdělen do velkého počtu paralelních větví (například trubkového svazku), čímž se dosáhne zvýšení teplosměnné plochy a potažmo intenzity prostupu tepla. Kromě vhodnosti standardních softwarů pro návrh aparátů různých geometrických provedení je proto také potřeba uvážit skutečnost, že tyto softwary typicky předpokládají rovnoměrné rozdělení pracovní látky do všech jednotlivých paralelních větví. Takový předpoklad nemusí sice mít významný vliv na výsledný predikovaný tepelný výkon, ale nezohlednění skutečného rozložení průtoků může zásadním způsobem ovlivnit spolehlivost zařízení a jeho životnost. Důvod je přitom prostý – vlastní rozdělení proudu ovlivňuje nejen případný výskyt zón s nižší rychlostí proudění, které jsou pak náchylné k vyšší míře zanášení, ale také rozložení teplot napříč jednotlivými větvemi, což může mít za následek vyšší mechanické namáhání trubkového svazku, praskání trubek či dokonce jejich vytrhávání z trubkovnic. Zmíněný jev lze snadno demonstrovat na příkladu jednotky pro zneškodňování

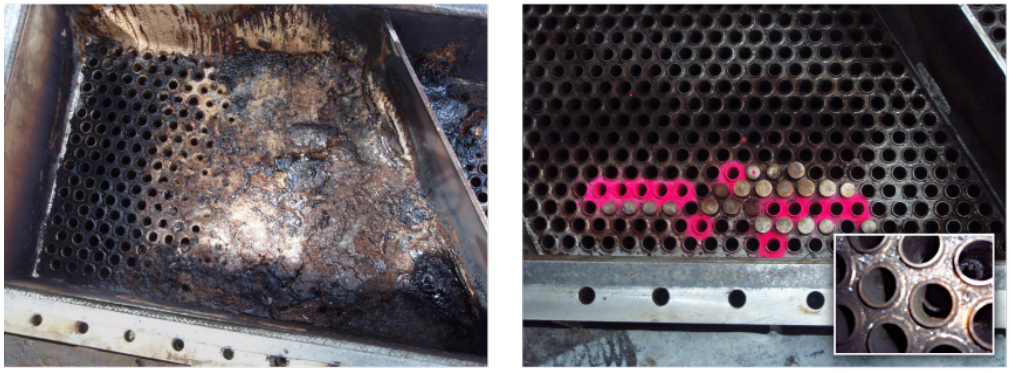
procesního odplynu z obrázku 1.1. Ve zde instalovaném předehříváku odplynu (viz také obrázek 1.2) totiž vlivem značně nerovnoměrné distribuce toku a nedostatečné funkce odlučovače kapek docházelo k silnému zanášení vstupní trubkovnice (obrázek 1.3a) a posléze vytrhávání okolních – dosud nezanesených – trubek (obrázek 1.3b), které byly chlazeny proudícím odplynem a dilatovaly proto méně. Detaily týkající se tohoto případu a odpovídajících nápravných opatření lze nalézt v článku [A2].

Znalost rozložení proudu již ve fázi návrhu zařízení je tedy klíčová. Výhradní používání detailních CFD modelů ovšem není řešením, a to s ohledem na jejich výpočtovou náročnost. V prvotní fázi návrhu zařízení, při tvarové optimalizaci a podobně, kdy je nutné u velkého počtu přípustných geometrií zařízení alespoň přibližně vyhodnotit rozložení tekutiny v obsažených distribučních systémech, ostatně ani není vysoká přesnost v naprosté většině případů vyžadována. Naopak bohatě dostačují i o něco méně přesné, avšak rychlé a konzistentním způsobem provedené analýzy, na jejichž základě lze vyřadit většinu nevhodných variant a původní obsáhlou sadu přípustných provedení tak omezit



Obrázek 1.1. Schéma jednotky pro zneškodňování procesního odplynu; za povšimnutí stojí netradiční uspořádání výměníků pro výrobu nízkotlaké (LP) a přehřáté středotlaké (SIP) páry

Obrázek 1.2. Předehřívák procesního odplynu; odplyn („Process Waste Gas“, PWG) vstupuje do výměníku v horní části a po průchodu U-trubkovým svazkem odchází do spalovací pece (na fotografii úplně vlevo), spaliny („Flue Gas“) z pece prochází výměníkem horizontálně



(a) boční část zanesené trubkovnice

(b) již zaslepené a nově vytržené trubky ve střední části trubkovnice

Obrázek 1.3. Boční část zanesené vstupní trubkovnice (vlevo) a již zaslepené trubky, resp. nově vytržené trubky ve střední části trubkovnice, které byly identifikovány při čištění výměníku (vpravo)

na několik málo možností. Detailní CFD modely pak jsou nasazeny až u těchto zbylých geometrií, kde je již vysoká přesnost namístě a zvýšená výpočtová náročnost je tudíž opodstatnitelná.

1.1 Předmět a cíle práce

Výpočtové modelování procesních a energetických zařízení na výměnu tepla lze rozdělit do tří různých úrovní, a to na:

(i) bilanční výpočty,

(ii) tepelně-hydraulické výpočty a

(iii) pevnostní výpočty.

Při uvážení dříve uvedených informací je ihned zřejmé, že z pohledu korektního návrhu jsou bilanční výpočty – na rozdíl od tepelně-hydraulických a pevnostních analýz – pouze podpůrným nástrojem. Předložená habilitační práce se přitom zaměřuje výhradně na tepelně-hydraulickou úroveň, tedy na praktickou aplikaci matematického modelování proudění tekutin a přenosu tepla. Zároveň je kladen důraz na distribuci toku a na skutečnost, že ne vždy je možné použít výpočtově a časově náročné detailní 3D CFD modely. V zásadě se tedy dá říci, že zde popsaný výzkum vychází z potřeb projektantů a provozovatelů zmíněných typů aparátů, resp. nutnosti řešit u patřičných zařízení různé provozní potíže (zanášení teplosměnných ploch, nadměrné tepelné a mechanické namáhání trubkových svazků v důsledku nerovnoměrné distribuce pracovních látek či tepla apod.). Veškeré softwary vznikající v rámci výzkumu pak také nejsou vyvíjeny pouze pro potřeby ověření odpovídajících matematických modelů, ale jsou naopak dále využívány i při řešení projektů smluvního výzkumu pocházejících od zadavatelů z průmyslové sféry, kteří se s provozními potížemi obracejí na pracoviště autora.

Primárním cílem této práce je popsat dva hlavní způsoby zjednodušeného modelování zmíněných jevů, které lze vzhledem k požadované rychlosti výpočtů a přesnosti výsledných dat aplikovat v inženýrské praxi. Značná pozornost je proto věnována zajištění vysoké míry výpočtové efektivity a robustnosti modelů skrze vhodnou volbu numerických metod. S přesností a kvalitativním charakterem predikcí pak souvisí druhé téma diskutované v této habilitační práci, a sice numerická disipace v detailních CFD modelech, které jsou využívány pro analýzy proudění v geometriích zbylých po počátečním „eliminačním“ kroku. Účelem zde je nastínit budoucí výzkum specifické implementace odhadu míry disipace vzniklé v důsledku nutné diskretizace výpočetní domény a použitých rovnic. Tato pak může po provedení potřebného srovnání s experimentálními daty sloužit jako měřítko přesnosti dat získávaných pomocí CFD výpočtů, resp. k odhadu řešení nezávislého na výpočetní síti.

1.2 Použité metody

Na základě výsledků popsaných v autorově disertační práci [A3] by se mohlo zdát, že i značně zjednodušené modelování s využitím kvazi-1D[†] výpočetních sítí a algebraických rovnic zahrnujících tlakové korekce je vhodným nástrojem pro přibližné analýzy proudění a distribuce toku ve výměnících tepla či jiných běžných procesních a energetických zařízeních. Toto však platí pouze v případě, že patřičný model byl pro určitou třídu aparátů předem „naladěn“. Pokud jsou naopak modely využívající tlakové korekce použity bez předchozího „naladění“, nedokáží obvykle poskytnout vůči prostému předpokladu rovnoměrného rozdělení tekutiny v trubkovém svazku žádnou podstatnou informaci [3].

Dalším faktorem, který je nutné u zmíněným modelů vzít do úvahy, je obtížná reprezentace komplexnějších geometrií distribučních systémů. Tlaková a rychlostní pole totiž nejsou zjišťována maticovými výpočty, jak je běžné například u CFD, ale postupným řešením značně nelineárních algebraických rovnic pro jednotlivé hrany výpočetní sítě. Jestliže bychom se tedy namísto vzájemně propojených jednorozměrných podsítí rozhodli použít propojené jednorozměrné a dvourozměrné podsítě (typicky 1D pro trubky svazku a 2D pro trubkovnice), vyvstala by otázka, jak v iteračním výpočtu provádět korekce hodnot veličin. Článek [A4] sice ukazuje, že nejde o neřešitelný problém, avšak rozhodně také nejde o situaci, kdy by aplikace určitého předem daného postupu vždy vedla k získání zkonvergovaného řešení. Toto je ovšem v příkrém rozporu s požadavky inženýrské praxe, kde je cílem mít k dispozici robustní výpočtové nástroje.

Opomeneme-li tedy velmi jednoduché distribuční systémy (trubkové svazky obsahující pouze jednu řadu trubek apod.), resp. systémy, pro které je k dispozici dostatek dat nutných k prvotnímu „naladění“ modelu, pozbývá z pohledu procesního inženýra použití diskutovaných modelů smyslu. Postupy popsané v následujících kapitolách se proto zaměřují na jiné způsoby provádění simulací, a to s využitím zjednodušených 3D modelů založených na výpočtové dynamice tekutin a kvazi-3D[‡] modelů založených na analýze metodou konečných prvků („Finite Element Analysis“, FEA).

V případě zjednodušeného 3D CFD modelování je účelem získat univerzální nástroj pro rychlé – avšak stále dostatečně přesné – simulace v prvotních fázích návrhů zařízení či implementace v optimalizačních algoritmech. Modelování založené na metodě konečných prvků pak k řešení úloh přistupuje diametrálně odlišným způsobem. Oproti zjednodušeným CFD modelům je výpočet výrazně méně náročný a potažmo také výrazně rychlejší. Jeho nevýhodou ovšem je aplikovatelnost pouze na některé typy distribučních systémů – typicky jednodušší trubkové svazky například v kotlích na odpadní teplo – a nižší přesnost výsledných dat.

Údaje týkající se rozložení proudu tekutiny v rámci distribučního systému obvykle nejsou pro provozovatele zařízení kritické a není proto ani prováděno jejich měření.

[†] Kvazi-1D síť je složena ze vzájemně propojených jednorozměrných podsítí reprezentujících jednotlivé části modelovaného systému – například distribuční kanály, trubky ve svazku a podobně.

[‡] Kvazi-3D modely pracují se sítěmi složenými ze vzájemně propojených podsítí různých – obecně vyšších – dimenzí.

Vzniklé matematické modely tudíž zatím byly vždy ověřovány pomocí výsledků z detailních CFD výpočtů. Co se týče testovacích výpočtů sloužících k hodnocení numerické efektivity, kromě používání vhodných knihoven pro lineární algebru bylo v případě implementace v jazyce Java [4] dbáno také na to, aby výsledky nebyly zkresleny principem fungování odpovídajícího běhového prostředí (Just-in-Time kompilace, optimalizace strojového kódu prováděné za běhu aplikace atd.).

1.3 Struktura dokumentu

Tato habilitační práce popisuje tři hlavní oblasti aktuálního výzkumu autora a vlastní text je proto rozdělen do tří kapitol. První oblastí – diskutovanou v kapitole 2 – jsou možné způsoby zjednodušení CFD modelů za účelem jejich následného použití pro rychlé analýzy proudění v procesních a energetických zařízeních, resp. implementace takových modelů v optimalizačních algoritmech. Kapitola 3 oproti tomu popisuje modelování proudění a distribuce toku založené na metodě konečných prvků. Výsledky dosažené v oblasti odhadu míry numerické disipace jsou pak shrnuty v kapitole 4. Tyto tři kapitoly jsou nakonec doplněny závěrečným shrnutím (kapitola 5).

Kapitola

2

Zjednodušené CFD modelování

Standardní CFD modely vynikají svou univerzálností a relativní přesností poskytovaných dat, ale jsou velmi náročné z pohledu výpočetního času a obecně výkonu výpočetního hardwaru. Jejich použití je tedy v inženýrské praxi omezeno spíše na situace, kdy je potřeba provést detailní analýzy několika málo zařízení či jejich částí. Naopak využití takových modelů například v prvotní fázi návrhu tepelného výměníku ke zjištění, která z možných geometrií trubkového prostoru nejlépe splňuje požadavky budoucího provozovatele, by zcela zřejmě nebylo časově ani ekonomicky přijatelné. Jinak řečeno, vzhledem k náročnosti standardních CFD modelů je jejich hromadné nasazení v oblasti návrhu zařízení na výměnu tepla problematické, stejně jako jejich aplikace v optimalizačních algoritmech jakéhokoliv druhu.

Pro potřeby počáteční rozvahy však není nutné provádět detailní analýzy všech možných konfigurací. Stejně tak v této fázi není nutné disponovat velmi přesnými daty o rychlostních, tlakových a teplotních polích či dalších veličinách. Je však otázkou, jaké konkrétní úpravy lze uvažovat, aby výsledný model stále byl dostatečně přesný a přitom poskytoval výsledky dostatečně rychle. Hlavním cílem výzkumu popisovaného v této kapitole tudíž bylo nalézt zjednodušení[†] vedoucí k prakticky použitelným CFD modelům určeným k výše zmíněnému účelu. Toto zahrnovalo nejen navržení vhodného způsobu tvorby výpočetní sítě, ale i posouzení vhodnosti různých numerických metod pro řešení

[†]Odsud také pochází název „zjednodušené CFD modelování“.

soustav linearizovaných rovnic a souvisejících metod předpodmínění, rozvahu diskretizačních schémat a podobně.

Veškerá zde diskutovaná zjednodušení byla testována pomocí vlastního CFD kódu, ve kterém byla zajištěna kompletní kontrola nad celým modelem. Tato skutečnost byla podstatná, neboť zaručovala, že nedojde k ovlivnění výsledků testovacích výpočtů různými uživateli skrytými faktory. Praktická implementace patřičných postupů však samozřejmě je do určité míry proveditelná i v běžně dostupných CFD softwarech, byť hodně záleží právě na možnostech té které aplikace. Informace zmíněné v této práci tedy mohou být velmi užitečné i pro procesní inženýry či projektanty, od kterých lze těžko očekávat vývoj vlastního CFD řešiče, neboť pro dotyčné mohou představovat určitý návod, jak s komerčními CFD softwary získat při výrazné úspoře času přibližné modely poskytující data přijatelné kvality.

2.1 Hlavní výhody a nevýhody

Primární výhodou zjednodušených CFD modelů je jejich výpočtová nenáročnost a potažmo krátké výpočetní časy. Představme si například relativně nekomplikovaný trubkový prostor vzduchového chladiče, tj. dva hlavní kanály (distributor a kolektor) spojené několika málo řadami přímých trubek. Patřičná standardní CFD úloha může zcela realisticky vyžadovat výpočetní síť s jednotkami milionů buněk. Její vyhodnocení pak může i při paralelním běhu v šestnácti či více vláknech na výkonném výpočetním hardwaru trvat jednotky až desítky hodin. Odpovídající zjednodušená CFD úloha přitom obvykle vystačí se sítí o jednotkách desítek tisíc buněk a její vyhodnocení v jednom vlákně i na běžném kancelářském počítači běžně zabere nejvýše jednotky až desítky minut. Druhou výhodou zmíněných modelů pak je výrazně snadnější vytváření výpočetních sítí, neboť jejich struktura je jednodušší a celá operace je proto lépe automatizovatelná.

Hlavní nevýhodou zjednodušených CFD modelů naopak je snížená přesnost výsledných dat. Byť tedy lze takové modely použít pro srovnání jednotlivých geometrií zařízení v rámci konzistentně vyhodnocované sady, rozhodně není vhodné je považovat za plnohodnotnou náhradu standardních CFD modelů. Mezi další nevýhody pak patří větší náchylnost k numerickým potížím, což je dáno převážně relativně hrubou výpočetní sítí, a dále také – vzhledem k charakteru těchto modelů – zpravidla i *de facto* nutnost znalosti základů programování uživatelem.

2.2 Požadavky kladené na použité výpočtové metody

Od numerických metod a algoritmů použitých ve zjednodušených 3D CFD modelech očekáváme splnění dvou hlavních požadavků, a sice co možná nejvyšší výpočetní efektivitu a přijatelnou robustnost. Potřeba efektivních metod je zřejmá, neboť hlavními cílovými použitími zmíněných modelů jsou předběžné analýzy velkých sad možných geometrií a tvarová optimalizace. Je tedy nutné vhodně zvolit nejen vlastní metody pou-

žívané pro řešení soustav linearizovaných rovnic, ale i nastavení interních limitů (počet iterací, relativní tolerance atd.) patřičných řešičů a odpovídající metody předpodmínění. Ostatně, soustavy linearizovaných rovnic je vždy cílem řešit pouze „dostatečně přesně“, nikoliv co možná nejpřesněji. Rozhodně také není nejmenší důvod snažit se v průběhu počáteční fáze iteračního procesu CFD řešiče, kdy jsou jednotlivá rychlostní pole a další zájmové veličiny daleko od svých konečných stavů, o získání přesného řešení postaveného na nepřesných vstupních datech. Jiné interní limity numerických řešičů tedy použijeme na začátku výpočtu a jiné pak v případě, že se jednotlivá sledovaná CFD rezidua postupně blíží nastaveným limitním hodnotám pro ukončení výpočtu.

Co se týče robustnosti, tento požadavek je primárně motivován skutečností, že použitá zjednodušená kvádrová výpočetní síť (která bude podrobněji popsána dále v kapitole 2.3) je prostě na poměry standardních CFD modelů příliš hrubá a nekvalitní. S takovou značně neideální sítí lze nevyhnutelně očekávat potíže s konvergencí, zvláště pak v průběhu prvních několika iterací. Je proto nutné, aby použité numerické metody dokázaly tuto fázi překlenout, aniž by došlo k divergenci nebo rozpadu numerického řešiče. Částečně sice lze situaci vyřešit výše zmíněným vhodným nastavením interních limitů, nicméně se nejedná o dokonalý a vždy funkční postup. Některé numerické metody tudíž v zásadě nelze vůbec použít, zatímco jiné mohou být stabilní (v numerickém smyslu, tedy z pohledu generování a šíření diskretizačních a dalších numerických chyb) a i se značně nepřesným počátečním odhadem se vyrovnají bez větších potíží. Tato problematika je detailněji diskutována v kapitolách 2.4, 2.5 a 2.7.

Kromě toho je namístě se zamyslet, zda musí zjednodušený 3D CFD model nutně implementovat některý ze standardních modelů turbulence, nebo zda si můžeme dovolit využít nějaké vhodné zjednodušení. Běžně užívané modely turbulence totiž typicky zahrnují řešení dvou či více dalších soustav rovnic, což představuje nezanedbatelné zvýšení výpočtové náročnosti. Nadto je otázkou, zda by byla implementace standardního modelu žádoucí, neboť se dá očekávat, že do stávajícího výpočtu se sníženou numerickou stabilitou přinese dodatečné numerické obtíže. Zmíněné téma je rozebráno v kapitole 2.6.

V neposlední řadě je potřeba zohlednit způsob, jakým jsou jednotlivé využité numerické metody a další algoritmy implementovány. Jinými slovy, musíme zajistit nejen to, aby byly odpovídající maticové operace prováděny vhodným způsobem, ale také efektivitu procesu výpočtu jednotlivých koeficientů linearizovaných rovnic (resp. efektivitu dalších souvisejících operací). Snadno se totiž může stát, že výpočetní čas vyžadovaný samotnými procesy řešení matic reprezentujících soustavy rovnic bude výrazně kratší než souhrnná doba nutná pro dokončení ostatních operací, tedy že situace bude obrácená vůči tomu, co by se dalo rozumně očekávat. Tomuto tématu se podrobněji věnují kapitoly 2.8 a 2.9.

2.3 Výpočetní síť

Výpočetní síť nabízí hned několik cest ke zvýšení rychlosti výpočtu. V případě zjednodušených CFD modelů se intuitivně nabízí v prvé řadě použití hrubší sítě. Dodatečného

snížení počtu buněk v síti bez výrazného snížení kvality dat pak lze dosáhnout tvorbou sítě pomocí růstového faktoru. Kromě toho je také možné sítě sestavovat pouze z kvádrových buněk, čímž se zjednoduší a významně zrychlí proces výpočtu koeficientů v jednotlivých diskretizovaných rovnicích.

Vzhledem k relativní složitosti procesních a energetických zařízení je výhodnější pracovat s nestrukturovanou sítí. Procházení sítě při generování potřebných matic je potom sice méně přímočaré (nestačí prostý cyklus přes dva či tři indexy v případě dvourozměrných, resp. trojrozměrných simulací), avšak lze snadněji definovat i tvarově komplikovanější výpočetní doménu. Nadto se také u složitějších geometrií obecně sníží výsledná paměťová náročnost, jelikož odpovídající datové struktury nemusí zohledňovat „hluchá“ místa, kde se v doméně nic nenachází, ale pro která ve strukturované síti přesto musí existovat odpovídající buňky.

2.3.1 Snížení počtu buněk

První a zcela logickou cestou ke zkrácení výpočetního času je snížení počtu buněk na minimální akceptovatelnou úroveň. Tím se sníží velikosti jednotlivých řešených soustav lineárních rovnic a potažmo dojde i ke snížení paměťových nároků. Značným limitujícím faktorem zde je skutečnost, že procesní a energetická zařízení v praxi velmi často obsahují trubkové svazky, což – jak bude zřejmé z kapitoly 2.3.3 – významně ovlivňuje rozsah možných jemností výsledné výpočetní sítě, má-li tato být sestavena rozumně. V zásadě tedy je nutné vyvážit přesnost modelu (danou jemností sítě), výpočetní zdroje, které má procesní inženýr k dispozici, a požadavky týkající se rychlosti výpočtu (typicky u úloh optimalizujících tvar určité části zařízení).

2.3.2 Využití růstového faktoru

Druhou cestou, vedoucí nejen ke snížení výpočtové náročnosti výsledného modelu, ale i k částečnému zlepšení konvergence, je plynulá změna velikostí buněk v určitých podoblastech sítě. Této strategie se ostatně často využívá i u sítí ve standardních CFD modelech. Poměr velikostí libovolných dvou sousedních buněk potom v předem daném směru zůstává konstantní a nazývá se růstovým faktorem (κ, „growth factor“).

Mají-li se rozměry buněk měnit například ve směru souřadné osy x a buňky v první řadě vybrané podoblasti mají patřičné hrany délky x_1, pak odpovídající hrany buněk v druhé řadě takové podoblasti budou dlouhé $x_2 = \kappa x_1$, ve třetí řadě $x_3 = \kappa x_2 = \kappa^2 x_1$ atd. Takto lze zajistit, aby v místech s většími gradienty veličin byly rozměry buněk v odpovídajících směrech menší (lokálně jemnější síť) a naopak tam, kde jsou změny méně významné, byla síť hrubší. Cílem zmíněných úprav je získání sítě rozumného rozsahu, na které však zároveň lze dostatečně dobře popsat modelované děje. Ideálně by přitom mělo platit $\kappa \leq 1{,}2$ [5].

Typickým příkladem použití růstového faktoru jsou sítě reprezentující trubky ve svazku tepelného výměníku. V okolí ústí trubek, kde dochází k významným změnám směru proudění, jsou vyžadovány menší buňky, zatímco dále od ústí, kde už bývá proudění

v radiálním směru zanedbatelné, lze použít buňky mnohonásobně větší délky. Analogicky lze pomocí růstového faktoru upravit strukturu sítě nad trubkovnicí či obecně v okolí stěn. Ukázka sítě se změnami velikostí buněk ve směru proudění je na obrázku 2.1.

2.3.3 Síť pouze z kvádrových buněk

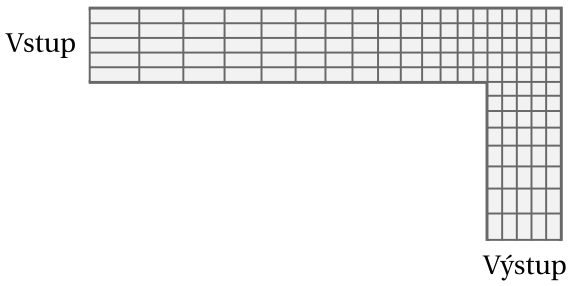
Třetí cestou ke zkrácení výpočetního času je sestavení sítě výhradně z kvádrových buněk. Takové buňky sice v naprosté většině případů nedokáží popsat původní geometrii stejně věrně jako sítě sestavené z obecných mnohostěnů (nejčastěji čtyřstěnů, pyramidových buněk a trojbokých a čtyřbokých hranolů), avšak přináší jedno velmi významné zjednodušení v podobě snadnějšího výpočtu gradientů. Zatímco u obecných mnohostěnů je při výpočtech koeficientů diskretizovaných rovnic nutné gradienty získávat přepočtem z normálových stěnových toků, u kvádrových buněk nic takového není potřeba.

Jediným omezením výpočetních sítí složených pouze z kvádrových buněk, pokud to v kontextu zjednodušených CFD modelů vůbec lze omezením nazvat, je nutnost zachovávat ekvivalentní průtočné průřezy a tepelné toky [A5]. V praxi to znamená, že například při modelování trubkového svazku nelze trubky o vnitřním průměru d_1 na jejich průřezech nahradit $N \times N$ buňkami o celkové ploše d_1^2, ale je nutné délku hrany nového čtvercového příčného průřezu dopočítat pomocí

$$L = \frac{\sqrt{\pi} d_1}{2}. \tag{2.1}$$

V případě nevyužití růstového faktoru je délka hrany libovolné buňky dána triviálním vztahem L/N. Jinak (tj. při $\kappa > 1{,}0$) lze za předpokladu středově symetrického rozložení velikostí buněk po průřezu trubky odvodit vztah pro délku hran stěnových buněk v radiálním směru,

$$L_1 = L \left[2 \sum_{i=0}^{\lfloor N/2 \rfloor} \kappa^i - \left(1 + \max\left\{0, (-1)^N\right\} \right) \kappa^{\lfloor N/2 \rfloor} \right]^{-1}, \tag{2.2}$$



Obrázek 2.1. Řez rovinou symetrie sítě ostrého 90° kolena, ve kterém se velikosti buněk mění podél směru proudění pomocí růstového faktoru $\kappa = 1{,}1$

kde $\lfloor \cdot \rfloor$ značí funkci „dolní celá část“ („floor“). Délky hran buněk v dalších vrstvách směrem ke středu trubky poté jsou

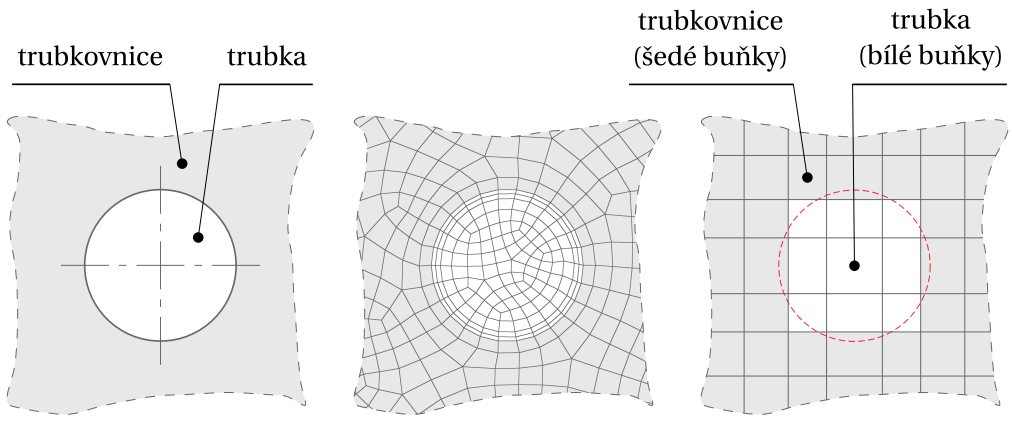
$$L_n = \kappa^{n-1} L_1, \quad n = 2, 3, \ldots \tag{2.3}$$

Příklad kvádrové sítě se třemi buňkami napříč průřezem trubky, která byla získána pomocí $\kappa > 1{,}0$, je spolu s typickou CFD sítí na obrázku 2.2. Co se týče ekvivalentního tepelného toku, vzhledem k odlišné ploše stěny náhradní geometrie trubky je nutné poměrově upravit okrajové podmínky na teplosměnných plochách, tj. např. vynásobit měrný tepelný tok koeficientem

$$F_\mathrm{q} = \frac{\pi d_2}{4L}, \tag{2.4}$$

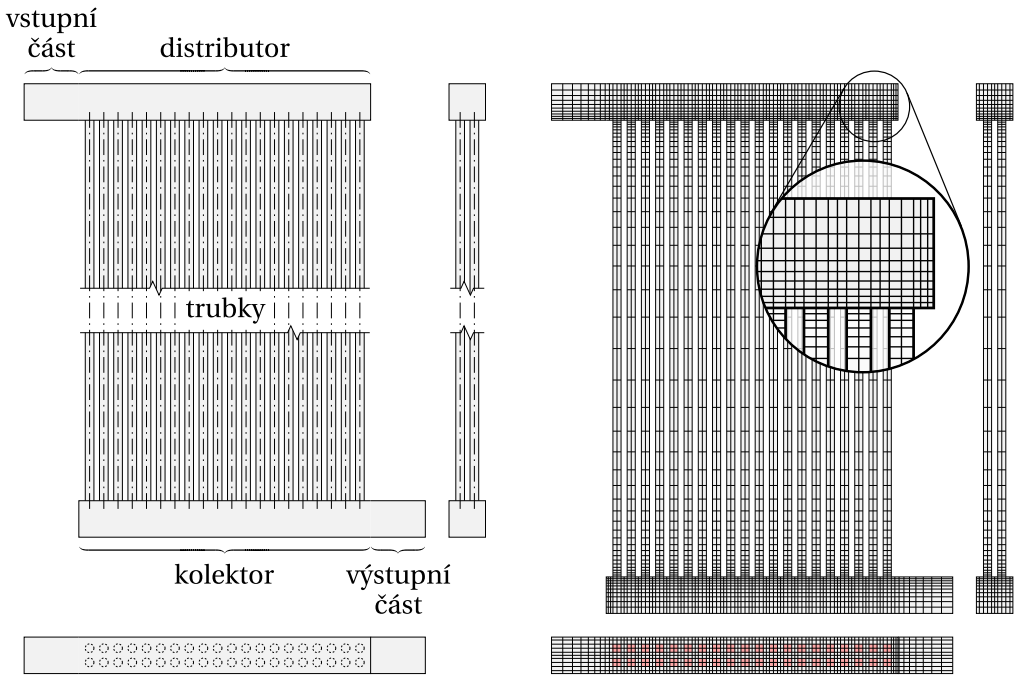
kde d_2 značí vnější průměr původní trubky a L celkovou délku hrany příčného průřezu náhradní geometrie z rovnice 2.1.

Z výše uvedeného je také patrné, že nepovolíme-li dělení stěn buněk („face splitting“), podřizuje se struktura celé sítě reprezentující trubkový prostor zařízení pouze dvěma parametrům, a sice počtu buněk napříč průřezem jedné trubky, N, a růstovému faktoru (či lokálním růstovým faktorům), κ. Místa mezi trubkami a u okrajů trubkovnic jsou pak pokryta buňkami o velikostech co možná nejbližších těm z vnitřků trubek tak, aby byl v případě větších mezer respektován růstový faktor, resp. aby byly buňky u stěn distribučních kanálů a v místech s očekávanými většími gradienty veličin dostatečně malé. Jak taková síť může vypadat, je znázorněno na obrázku 2.3.



(a) původní geometrie (b) typická CFD síť (c) kvádrová síť

Obrázek 2.2. Půdorysný pohled na původní geometrii okolí ústí trubky a odpovídající typickou CFD výpočetní síť a kvádrovou výpočetní síť; při tvorbě kvádrové sítě byl využit růstový faktor $\kappa > 1{,}0$, tj. buňky nemají jednotnou velikost (adaptováno z [A5])



Obrázek 2.3. Distribuční systém vzduchového chladiče s uspořádáním „Z“ (vlevo) a jemu odpovídající zjednodušená výpočetní síť složená výhradně z kvádrových buněk (vpravo, ~25 tis. buněk); červené buňky v půdorysném pohledu značí místa, kde se nachází příčné průřezy trubek. Při tvorbě sítě byl využit růstový faktor $\kappa = 1{,}15$ (adaptováno z [A6]).

Na tomto místě je dále vhodné poznamenat, že při využití metod pracujících s přesazenou sítí (například různé verze metody SIMPLE – viz kapitola 2.4) je doporučeno volit $N \geq 2$. V případě $N = 1$, tj. pokud je průřez každé trubky tvořen pouze jednou buňkou, by totiž přenos hybnosti z hlavních kanálů do trubek (resp. opačně) byl limitovaný a potažmo by došlo k výraznějšímu snížení přesnosti výpočtu. Pracovat s hodnotami parametru $N > 5$ však také není ideální, neboť pak výrazně narůstá počet buněk, což je v rozporu s cíli zjednodušeného modelování.

2.3.4 Vlastnosti zjednodušené sítě

V předchozích odstavcích byly popsány způsoby, jakými použití zjednodušené sítě zrychluje výpočet. Jedna ze standardních metod akcelerace výpočtu, tedy opakované řešení CFD úlohy na sadě sítí různých jemností („multigrid solution method“), však zmíněna nebyla. Důvod je prostý – u uvažovaných zjednodušených sítí tento postup v zásadě nelze aplikovat, jelikož z nich není jak vyrobit hrubší verze.

Nevyhnutelnou vlastností zjednodušené sítě je její nižší kvalita, byť buňky samotné bývají z pohledu standardních kritérií zmíněných obvykle v uživatelských příručkách

k CFD softwarům (např. v příručce aplikace ANSYS Fluent [6, kap. 23.2.4]) kvalitativně přijatelné:

- Míra zkosení buněk („normalized equiangular skewness“) je rovna nule a buňky jsou tudíž nejkvalitnější možné, neboť všechny vnitřní úhly jsou pravé, stejně jako vnitřní úhly ideální buňky:

$$\max\left\{\frac{\theta_{\max}-90}{90},\frac{90-\theta_{\min}}{90}\right\}=0. \tag{2.5}$$

- Míra ortogonality („orthogonal quality“) je také nejvyšší možná, protože normálové vektory, $\mathbf{n}$, stěn buněk, A, jsou vždy rovnoběžné s vektory definovanými centroidem buňky a centroidy těchto stěn, $\mathbf{f}$, resp. centroidem buňky a centroidy odpovídajících sousedních buněk, $\mathbf{c}$. Jinak řečeno, normalizované skalární součiny těchto vektorů nabývají pro libovolnou stěnu libovolné buňky hodnoty 1:

$$\min_{A}\left\{\frac{\mathbf{n}\cdot\mathbf{f}}{|\mathbf{n}||\mathbf{f}|},\frac{\mathbf{n}\cdot\mathbf{c}}{|\mathbf{n}||\mathbf{c}|}\right\}=1. \tag{2.6}$$

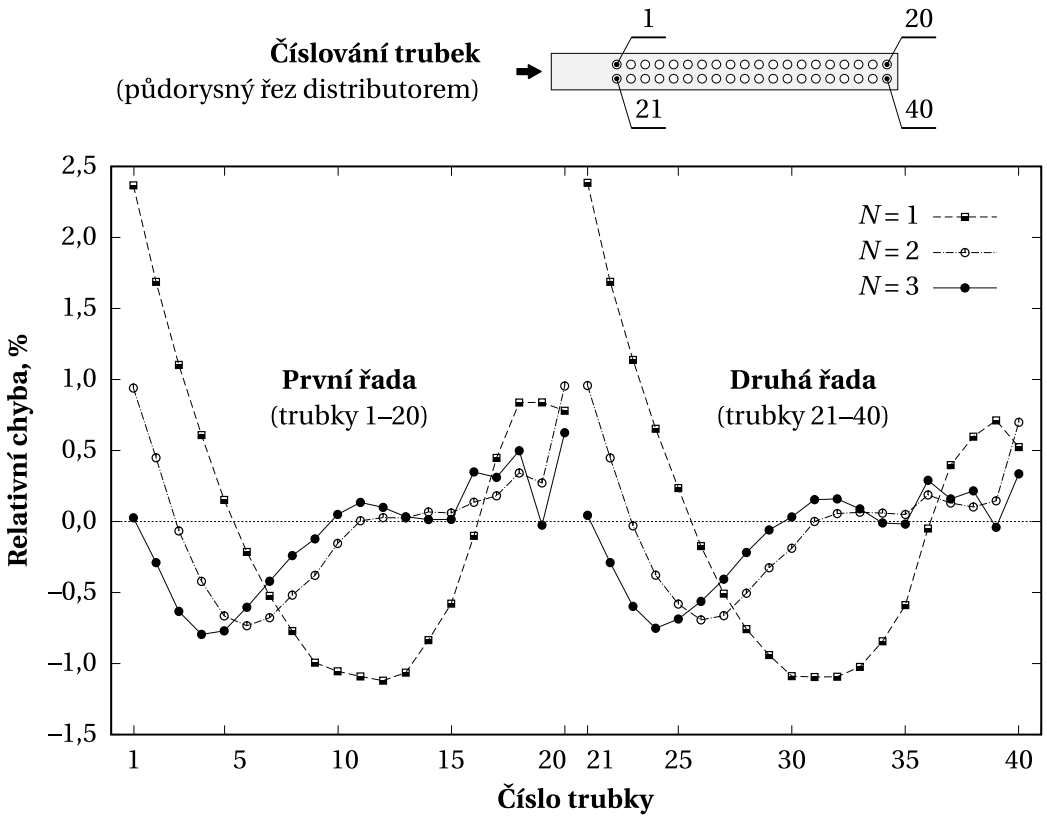
- Poměr délek nejdelší a nejkratší hrany („aspect ratio“), stejně jako hladkost změn velikostí buněk, jsou v oblastech, kde má význam tato kritéria uvažovat, při rozumně zvoleném růstovém faktoru obvykle přijatelné. Jedinou výjimku zde tvoří případ, kdy je nutné vložit do sítě velmi úzký pás buněk (např. v distribučním kanálu, na který je napojen trubkový svazek s těsným prostřídaným uspořádáním).

Velikost buněk však je z pohledu řešení CFD úlohy naprosto zásadní a použití zjednodušené sítě má proto za následek horší konvergenci. Typicky je tudíž potřeba CFD řešič více podrelaxovat – například ve vyvinutém softwaru [A5], který je podrobněji popsán v kapitole 2.10, jsou výchozí hodnoty relaxačních faktorů poloviční oproti hodnotám, které se běžně používají v komerčních CFD softwarech. Na základě výsledků publikovaných v článcích [A5–A7] však lze konstatovat, že použití výpočetní sítě složené pouze z kvádrových buněk nepředstavuje vážný problém.

Vzhledem k velmi omezenému počtu buněk je zřejmé, že testy nezávislosti výsledků na síti („mesh independence studies“) zde pozbývají téměř veškerého významu. Nejen že by případné výraznější zjemnění sítě nemuselo kvůli uvažovaného tvaru buněk nutně vést k adekvátnímu zvýšení přesnosti, ale nadto by bylo kontraproduktivní z pohledu rychlosti výpočtu. Primárním cílem zjednodušených CFD výpočtů ostatně není maximální přesnost, ale přesnost „dostatečná“ (což v různých situacích může znamenat různé věci) při co nejkratších možných výpočetních časech. Z výsledků prezentovaných v článku [A5] (resp. tabulce 2.1 a obrázku 2.4) však vyplývá, že určitého zpřesnění dosáhnout lze, byť to s sebou samozřejmě přináší nezanedbatelný nárůst výpočetního času.

Tabulka 2.1. Celkové počty buněk, průměrné výpočetní časy potřebné ke zkonvergování ustálených úloh a průměrné velikosti relativní chyby vůči detailním CFD výpočtům pro případy z obrázku 2.4; zjednodušené CFD modely používaly metodu SIMPLEC a diskretizační schéma „power law“ (viz kapitoly 2.4 a 2.5) a byly řešeny na jednom jádře CPU (data převzata z [A5])

N	Počet buněk	Výpočetní čas	Průměrná velikost relativní chyby
1	6 686	4,760 s	0,84 %
2	27 920	64,59 s	0,33 %
3	72 000	299,3 s	0,28 %



Obrázek 2.4. Vliv postupného zjemňování výpočetní sítě ve zjednodušeném 3D CFD modelu ($N \leq 3$) na relativní chyby hmotnostních průtoků jednotlivými trubkami svazku z obrázku 2.3 vůči datům z detailních CFD výpočtů (adaptováno z [A5])

2.4 Druh CFD řešiče a metoda výpočtu

Při sestavování CFD modelů – bez ohledu na to, zda jde o ty standardní, nebo zjednodušené – je vždy nutné podle typu řešeného problému v prvé řadě rozmyslet, zda použijeme

řešič monolitický („coupled“), nebo segregovaný („segregated“). Tyto se liší ve způsobu, jakým je přistupováno k řešení jednotlivých rovnic.

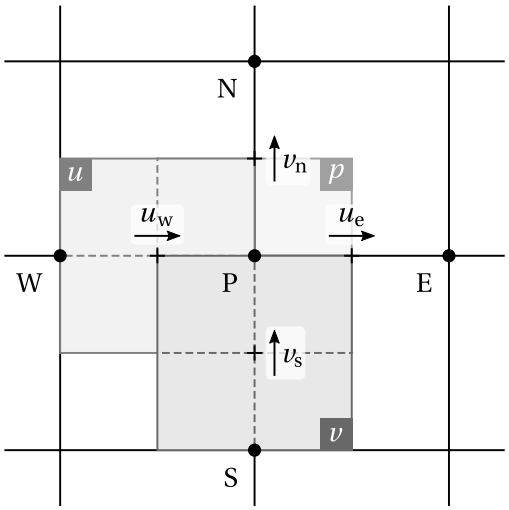
S monolitickým řešičem je z diskretizovaných rovnic pro jednotlivé složky rychlosti a spojitost toku, případně i z rovnice energie či dalších rovnic, sestavena jediná matice, přičemž „řídicí veličinou“ pak je tlak („pressure-based coupled solver“), resp. hustota („density-based coupled solver“). Toto s sebou zcela logicky přináší větší nároky na paměť a výpočetní prostředky obecně. Řešič nadto bývá méně stabilní [7]. Na druhou stranu je ke konvergenci zpravila vyžadováno o něco méně „velkých“ CFD iterací.

Použití segregovaného řešiče (ať už „pressure-based“ nebo „density-based“) naopak vyžaduje rozdělení celé úlohy na několik menších vzájemně závislých problémů, které jsou opět řešeny iteračním způsobem. Důsledkem tohoto rozdělení je sice o něco pomalejší konvergence (ve smyslu potřebného počtu iterací), neboť jsme snížili rychlost vzájemného ovlivňování proměnných, ale výpočet je stabilnější [8]. Proto je také segregovaný řešič v mnoha komerčních CFD softwarech pro širokou škálu běžných průmyslových aplikací stále řešičem výchozím.

S tímto souvisí také paralelizace vlastního výpočtu a skutečnost, že nárůst efektivity výpočtu při užití paralelizace nikdy není lineární. Jelikož jedním z primárních účelů zjednodušených CFD modelů je jejich implementace v optimalizačních algoritmech, je výhodnější spouštět patřičné – relativně malé – úlohy vždy pouze na jednom jádře CPU s využitím méně náročného segregovaného řešiče. Paralelizaci na dostupná jádra CPU je potom vhodné provést až „o úroveň výše“, tedy v rámci samotného optimalizačního procesu, kde již jsou průběhy jednotlivých výpočtů vzájemně nezávislé. Hodnota účelové funkce – například výsledná nerovnoměrnost rozdělení toku do jednotlivých kanálů distribučního systému – je totiž ve zde diskutovaném případě výstupem z iteračního algoritmu, jehož chování nelze pro potřeby optimalizace dost dobře předvídat, a musí tedy nutně jít o přímý optimalizační proces.

Druhým a neméně důležitým rozhodnutím je volba metody výpočtu. Asi nejčastěji bývá použita metoda SIMPLE („Semi-Implicit Method for Pressure-Linked Equations“) [9], případně její různé modifikace poskytující lepší konvergenci – např. SIMPLER („SIMPLE Revised“) [10] či SIMPLEC („SIMPLE Consistent“) [11]. Tyto metody provádí opakovaně vždy jeden predikční krok, který je následován jedním korekčním krokem. Rozšířením zmíněného přístupu o dodatečný (druhý) korekční krok pak vznikla metoda PISO („Pressure Implicit with Splitting of Operators“) [12], která je sice výpočtově náročnější, ale vlivem dvojí korekce dat by měla poskytovat vyšší rychlost konvergence než SIMPLER nebo SIMPLEC.

Metody výše pracují s přesazenou výpočetní sítí (viz obrázek 2.5), tedy se sítí, kde jsou skalární veličiny (tlak, hustota apod.) definovány v centroidech buněk a vektorové veličiny (např. rychlosti proudění) na stěnách buněk, neboť se tak zabrání nefyzikálnímu chování modelů (tzv. „checkerboarding“ [10, kap. 6.2-1]). Existují ovšem i jejich různě upravené verze ([13] apod.) či verze pracující s kolokovanou sítí, které se méně (např. [14]) nebo více (např. [15]) podobají původním algoritmům. Rozsáhlé srovnání zmíněných metod bylo publikováno v článku [16]. Z něj vyplývá, že při malé míře závislosti skalárních

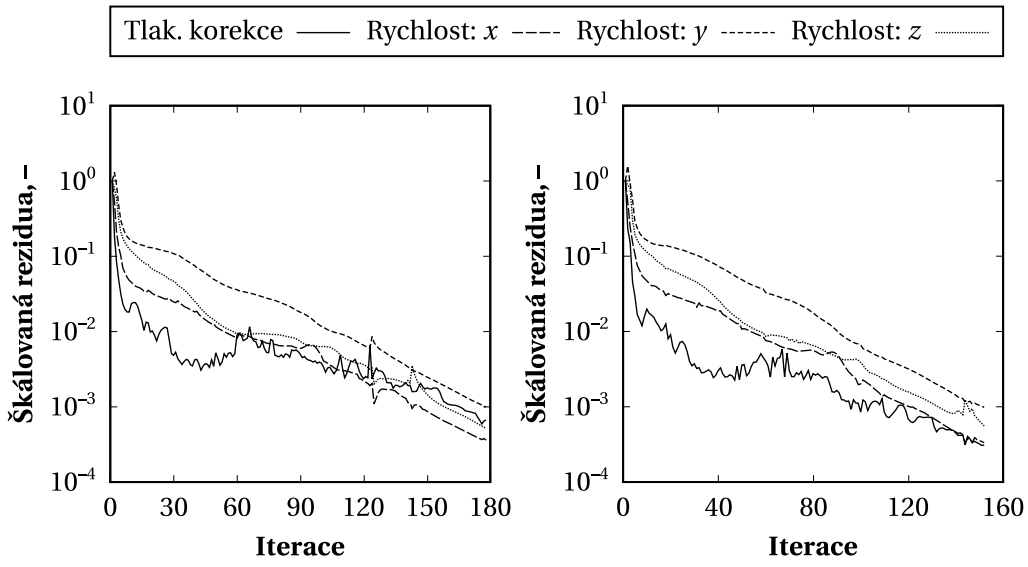


Obrázek 2.5. Dvourozměrná přesazená síť; značení uzlů výpočetní sítě odpovídá klasické konvenci, tj. „P“ pro aktuálně řešený uzel a „W“, „E“, „S“ a „N“ (resp. „w“, „e“, „s“ a „n“) pro západní, východní, jižní a severní uzel (stěnu kontrolního objemu). Vybrané kontrolní objemy pro tlakovou korekci v bodě „P“ a rychlosti u_w a v_s jsou pak vyznačeny písmeny p, u a v.

veličin na charakteru proudění je lepší implementovat metodu PISO. Jestliže naopak mají skalární veličiny výrazný vliv v rovnicích pro přenos hybnosti, je výhodnější aplikovat metodu SIMPLER nebo SIMPLEC, neboť metoda PISO v takovém případě vyžaduje u tranzientních úloh pro konvergenci relativně krátký časový krok. Dle studie [16] přitom obecně nelze říci, která ze tří zmíněných metod je celkově nejlepší, což potvrzují i mnohé novější zdroje (např. monografie [17]).

Na základě testů provedených autorem této práce lze konstatovat, že na relativně hrubé kvádrové výpočetní síti použité ve zjednodušených CFD modelech má metoda SIMPLEC z pohledu rychlosti konvergence o něco lepší vlastnosti než metoda SIMPLER. Metoda SIMPLER je nadto u kvádrové sítě dokonce méně robustní než původní metoda SIMPLE, neboť její aplikace v mnoha případech vedla k divergenci, byť metody SIMPLE a SIMPLEC poskytly zkonvergované řešení při jinak stejném nastavení bez nejmenších potíží. Kromě toho platí, že v mnoha průmyslových úlohách je nevyhnutelně nutné provádět tranzientní simulace, což by se mohlo s metodou PISO ukázat jako problematické. V softwaru diskutovaném dále v kapitole 2.10 byla proto implementována jako výchozí právě metoda SIMPLEC.

Srovnání průběhů škálovaných reziduí získaných pomocí metod SIMPLE a SIMPLEC na jinak identické úloze s mřížkou obsahující ~65 tis. buněk je na obrázku 2.6. Jak lze vidět, hladkost konvergence byla u obou metod zhruba srovnatelná, avšak průměrný čas nutný k dosažení zkonvergovaného řešení byl v tomto případě s metodou SIMPLEC o necelých 15 % kratší. Stejně tak byl menší i potřebný počet iterací CFD řešiče (152 namísto 178).



(a) SIMPLE, $t_{\text{avg}} = 138{,}06$ s (b) SIMPLEC, $t_{\text{avg}} = 119{,}17$ s

Obrázek 2.6. Průběhy škálovaných reziduí získané užitím metod SIMPLE a SIMPLEC v jinak identicky nastavené ustálené úloze pracující s výpočetní síti o velikosti ~65 tis. buněk. Uvedené průměrné výpočetní časy, t_{avg}, potřebné k dosažení zkonvergovaného řešení byly získány pomocí testovacího kódu používaného ve studii [A6].

2.5 Diskretizace

Obecnou transportní rovnici pro skalární veličinu ϕ (teplotu, koncentraci apod.) lze zapsat ve tvaru

$$\underbrace{\frac{\partial(\rho\phi)}{\partial t}}_{\substack{\text{tranzientní}\\ \text{člen}}} + \underbrace{\operatorname{div}(\rho\mathbf{u}\phi)}_{\substack{\text{konvektivní}\\ \text{člen}}} - \underbrace{\operatorname{div}(\Gamma \operatorname{grad}\phi)}_{\substack{\text{difuzní}\\ \text{člen}}} = \underbrace{S(\phi)}_{\substack{\text{zdrojový}\\ \text{člen}}}, \tag{2.7}$$

kde ρ značí hustotu proudicí tekutiny, t čas, $\mathbf{u} = (u, v, w)$ vektor rychlosti proudění a Γ difuzní koeficient. Pro potřeby řešení této rovnice na jednotlivých kontrolních objemech je však nutné ji převést do diskretizované podoby,

$$a_{\text{P}}\phi_{\text{P}} = \sum_{\text{nb}} a_{\text{nb}}\phi_{\text{nb}} + C, \tag{2.8}$$

kde a_{P} je koeficient pro aktuálně řešený uzel „P“ výpočetní sítě (viz obrázek 2.5), ϕ_{P} hodnota veličiny ϕ v tomto uzlu, a_{nb} a ϕ_{nb} koeficienty a hodnoty veličiny ϕ pro uzly sou-

sedící s uzlem „P“ a C konstanta zahrnující konstantní část diskretizovaného zdrojového členu, případně i další konstantní přírůstky pocházející z diskretizací jiných členů.

Použitý způsob diskretizace přitom zásadním způsobem ovlivňuje stabilitu výpočtu a konvergenci. Co se tím rozumí? Odhlédneme-li od chyb ve vstupních datech pro stanovení počátečních a okrajových podmínek, jsou nepřesnosti do výpočtu vnášeny dvěma způsoby, a sice diskretizačními a zaokrouhlovacími chybami. Diskretizační chyby vznikají např. v důsledku užití aproximace Taylorovým polynomem s velmi malým počtem členů. Zaokrouhlovací chyby jsou naopak zapříčiněny faktem, že počítače pracují s reprezentací čísel s pohyblivou desetinnou čárkou pomocí mantisy omezené bitové délky. Je také potřeba si uvědomit, že během řešení se jednotlivé chyby kombinují, přičemž nejde o prostý součet. Může se tedy snadno stát, že kupříkladu v důsledku nárůstu zakrouhlovací chyby výrazně vzroste i diskretizační chyba. Stabilitou výpočtu (resp. použité numerické metody) potom rozumíme, že numerické chyby, které při řešení diskretizovaných rovnic vznikají, nenarůstají, zatímco konvergence označuje v kontextu diskretizačních schémat situaci, kdy se numerické řešení při zjemňování výpočetní sítě postupně blíží řešení analytickému.

Vzhledem k použití relativně hrubé kvádrové sítě budou mít výsledné diskrektizační chyby nezanedbatelný vliv na vlastní průběh výpočtu. Toto je zcela zřejmě nutné zohlednit při volbě diskretizačních schémat a některých dalších parametrů CFD modelu.

2.5.1 Obecné požadavky na diskretizační schémata

Prvním krokem při diskretizaci rovnice 2.7 je její převod do integrálního tvaru a integrace přes kontrolní objem. Ne každé diskretizační schéma je však vhodné pro každý člen takto získané nové rovnice. Například centrální diferencování vede za určitých podmínek k fyzikálně zcela nerealistickým řešením (detailní zdůvodnění je možné nalézt v téměř libovolné knize věnující se problematice metody konečných objemů, resp. výpočtového modelování proudění tekutin – viz například [17, s. 137–141]). Na diskretizační schémata proto klademe následující tři základní požadavky:

- konzervativnost („conservativeness“) – schéma musí zachovávat hodnoty stěnových toků mezi sousedními kontrolními objemy (tj. to, co skrze sdílenou stěnu „odteče“ z kontrolního objemu V_1 do kontrolního objemu V_2, musí být stejné jako to, co do V_2 „přiteče“ z V_1);
- ohraničenost („boundedness“) – schéma musí zajišťovat, že výsledné řešení bude fyzikálně přijatelné (tj. že nikdy nepřekročí hranice, které jsou předem dány modelovaným fyzikálním jevem a okrajovými podmínkami);
- transportivita („transportiveness“) – schéma musí zohledňovat „směrový“ charakter konvektivního přenosu (tj. skutečnost, že při výraznější konvekci závisí ϕ mnohem více na hodnotě proti proudu než po proudu, zatímco při zanedbatelné konvekci se ϕ šíří do všech směrů přibližně stejně).

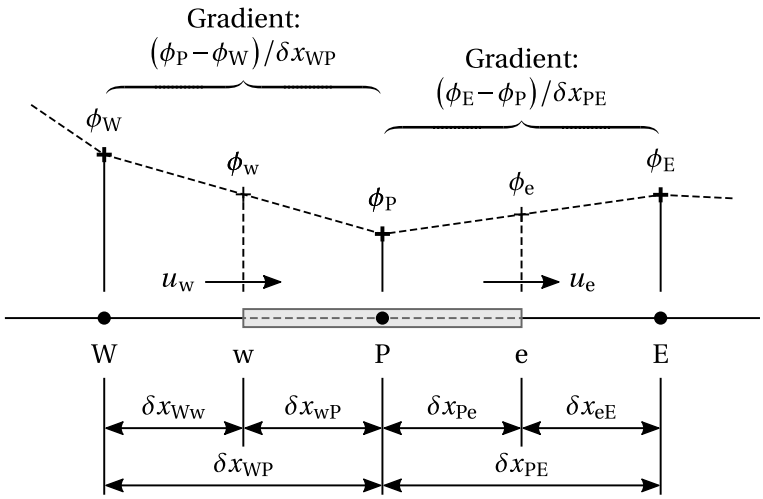
2.5.2 Konvektivní a difuzní člen

Uvažujme pro jednoduchost část diskretizované jednorozměrné výpočetní domény z obrázku 2.7. Lze ukázat, že u výše zmíněného centrálního diferencování,

$$\begin{aligned} \phi_\mathrm{w} &= \frac{\phi_\mathrm{W}\delta x_\mathrm{wP} + \phi_\mathrm{P}\delta x_\mathrm{Ww}}{\delta x_\mathrm{WP}}, \quad \phi_\mathrm{e} = \frac{\phi_\mathrm{P}\delta x_\mathrm{eE} + \phi_\mathrm{E}\delta x_\mathrm{Pe}}{\delta x_\mathrm{PE}}, \\ \Gamma_\mathrm{w} &= \frac{\Gamma_\mathrm{W}\delta x_\mathrm{wP} + \Gamma_\mathrm{P}\delta x_\mathrm{Ww}}{\delta x_\mathrm{WP}}, \quad \Gamma_\mathrm{e} = \frac{\Gamma_\mathrm{P}\delta x_\mathrm{eE} + \Gamma_\mathrm{E}\delta x_\mathrm{Pe}}{\delta x_\mathrm{PE}}, \end{aligned} \tag{2.9}$$

tedy u intuitivního způsobu diskretizace, kdy jsou stěnové hodnoty veličin ovlivněny všemi okolními uzlovými hodnotami stejně, aniž by byl zohledněn směr proudění, je zajištěna konzervativnost, avšak ohraničenost je splněna jen podmínečně a transportivita vůbec. Ačkoliv se tedy toto schéma vyznačuje přesností druhého řádu [18, kap. 6.1] a lze ho s výhodou použít u difuzních členů, k diskretizaci konvektivních členů – nadto pak ve zjednodušených CFD modelech tepelných výměníků a dalších procesních a energetických zařízení, kde je konvekce zpravidla nezanebatelná – se vzhledem ke zmíněným vlastnostem nehodí.

Kromě centrální diference byla tedy pro konvektivní člen navržena i jiná diskretizační schémata, která více či méně úspěšně řeší jeho nedostatky. Jedním z nich je schéma „upwind“, které hodnoty veličiny ϕ v okolních uzlech sítě aproximuje po částech konstantní funkcí zohledňující směr proudění. Takové diskretizační schéma je tudíž méně



Obrázek 2.7. Část diskretizované jednorozměrné výpočetní domény se zvýrazněným kontrolním objemem; značení uzlů výpočetní sítě odpovídá klasické konvenci, tj. „W“ (resp. „w“) pro západní uzel (stěnu kontrolního objemu), „P“ pro aktuálně řešený uzel a „E“ (resp. „e“) pro východní uzel (stěnu kontrolního objemu)

přesné, nicméně je splněn požadavek na transportivitu. Při použití značení z obrázku 2.8 pak pro kladný směr proudění ($u_w, u_e > 0$) lze psát

$$\phi_w = \phi_W, \quad \phi_e = \phi_P, \tag{2.10}$$

zatímco v opačném případě ($u_w, u_e < 0$) platí

$$\phi_w = \phi_P, \quad \phi_e = \phi_E. \tag{2.11}$$

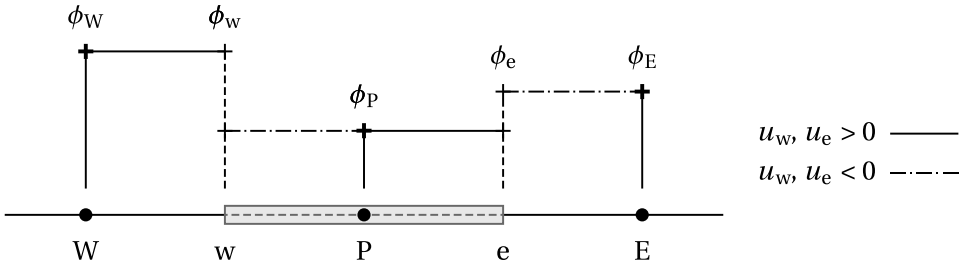
Podobně jako dříve lze snadno ukázat, že schéma „upwind“ splňuje všechny tři výše zmíněné podmínky a jeho aplikaci v uvažovaných zjednodušených 3D CFD modelech nic nebrání. Co se týče přesnosti, schéma vychází ze zpětné diference a vyznačuje se tudíž přesností prvního řádu. Nižší přesnost však nemusí nutně být na závadu, neboť s méně kvalitními výpočetními sítěmi je použití přesnějších schémat a metod mnohdy provázeno numerickými obtížemi (jak bude diskutováno dále v textu).

Kombinací centrálního diferencování a schématu „upwind“ je hybridní schéma [19]. Je-li intenzita konvekce nízká, přesněji při Pécletově čísle

$$\mathrm{Pe} = \frac{F}{D} = \frac{\rho u}{\Gamma/\delta x} \in (-2;\ 2)\,, \tag{2.12}$$

kde F značí konvektivní hmotnostní tok („convective mass flux“), D difuzní vodivost („diffusion conductance“) a δx odpovídající rozměr kontrolního objemu, je použito centrální diferencování. Při výraznější konvekci, kdy by aplikace centrálního diferencování byla problematická, se pak využije schéma „upwind“. Toto se přitom v počítačovém kódu samozřejmě neimplementuje podmíněným příkazem, ale vhodným přepisem odpovídajících výrazů pomocí funkce „max $\{\cdot\}$“[†].

[†]Podmíněným příkazům by se ostatně měl programátor co možná nejvíce vyhýbat, neboť zvyšují komplexitu kódu, což pak je častou příčinou výskytu chyb [20]. Přestože tedy funkce „max $\{\cdot\}$“ bývá interně implementována právě podmíněným příkazem (byť někdy schovaným v ternárním operátoru ?:), zmíněným přepisem se zvýší efektivita kódu a podmínka se přesune do části kódu, která není ovlivnitelná programátorem a lze ji *a priori* považovat za korektní.



Obrázek 2.8. Aproximace stěnových hodnot veličiny ϕ pomocí schématu „upwind“

Jelikož centrální diferencování i schéma „upwind“ jsou konzervativní, je zcela zřejmě konzervativní i schéma hybridní. Ze stejného důvodu – a s ohledem na užití centrální diference či schématu „upwind“ podle aktuální situace u toho kterého kontrolního objemu – jsou zajištěny také ohraničenost a transportivita. Můžeme tudíž konstatovat, že i hybridní schéma, které by mělo oproti schématu „upwind“ přinést částečné zvýšení přesnosti, lze u zjednodušených 3D CFD modelů uvažovat jako jeden z možných způsobů diskretizace.

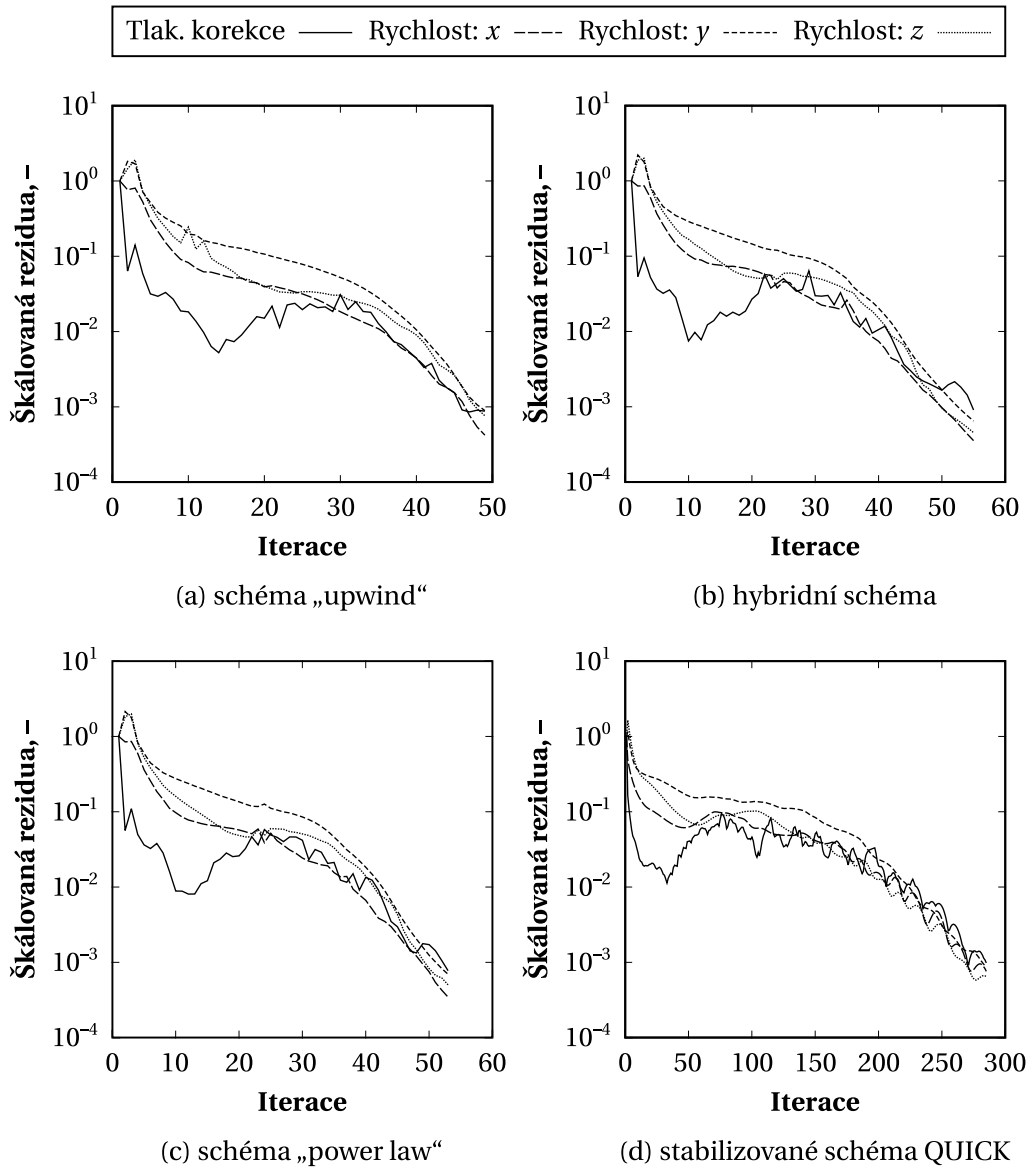
Posledním diskretizačním schématem prvního řádu, které zde bude zmíněno, je schéma „power law“ [10, kap. 5.2-6]. Toto schéma při vyšší intenzitě konvekce ($|\mathrm{Pe}| > 10$) zanedbává vliv difuze a mělo by proto vést k ještě přesnějším výsledkům než hybridní schéma. Podobně jako u hybridního schématu není počítačová implementace provedena podmíněným příkazem, ale pomocí funkce „max $\{\cdot\}$“ a polynomu závislého na Pécletově čísle. Co se týče podmínek z kapitoly 2.5.1, tyto jsou zde kompletně splněny.

Pokud tedy nebudeme uvažovat centrální diferencování, je otázkou, které z výše uvedených schémat – „upwind“, hybridního a „power law“ – je pro potřeby diskretizace konvektivního členu (a případně i difuzního členu, pokud pro něj nepoužijeme centrální diferenci) ve zjednodušených 3D CFD modelech nejlepší. Dále bychom také měli zvážit, zda by se nedosáhlo znatelného zlepšení situace implementací schématu vyššího řádu, například stabilizovaného schématu QUICK („Stabilized Quadratic Upstream Interpolation for Convective Kinetics“) [21], některého schématu z rodiny TVD („Total Variation Diminishing“) [22] apod.

Za tímto účelem bylo otestováno právě stabilizované schéma QUICK, jehož implementace do softwaru z kapitoly 2.10 nevyžadovala výraznější úpravy stávajícího kódu. Ukázalo se však, že aplikace zmíněného schématu vede ke zcela neakceptovatelnému nárůstu výpočetního času, resp. v kombinaci s relativně hrubou kvádrovou výpočetní sítí někdy dokonce k divergenci. Typické rozložení potřebných počtů iterací CFD řešiče a výpočetních časů získané pomocí jedné z testových úloh je uvedeno v tabulce 2.2. Odpovídající průběhy škálovaných reziduí jsou pak znázorněny na obrázku 2.9. Byť by se

Tabulka 2.2. Průměrné výpočetní časy nutné k získání zkonvergovaného řešení typické ustálené úlohy při diskretizaci konvektivního členu pomocí různých schémat. Ostatní nastavení zjednodušeného modelu (včetně výpočetní sítě o velikosti ~7 tis. buněk) bylo vždy stejné. Výsledky byly získány pomocí testovacího kódu ze studie [A6], přičemž průměrná relativní chyba značí průměrnou velikost procentuálních chyb predikovaných průtoků v jednotlivých kanálech distribučního systému vůči hodnotám získaným detailním 3D CFD výpočtem.

Diskretizační schéma	Počet iterací	Výpočetní čas	Průměrná rel. chyba
„upwind“	49	1,672 s	1,43 %
hybridní	55	1,984 s	1,54 %
„power law“	53	1,915 s	1,49 %
stabilizované QUICK	285	8,208 s	1,12 %



(a) schéma „upwind“

(b) hybridní schéma

(c) schéma „power law“

(d) stabilizované schéma QUICK

Obrázek 2.9. Průběhy škálovaných reziduí získané při diskretizaci konvektivního členu pomocí schémat z tabulky 2.2. Ostatní nastavení zjednodušeného modelu (včetně výpočetní sítě o velikosti ~7 tis. buněk) bylo vždy stejné.

tedy problém s divergencí možná vyřešil implementací jiného schématu vyššího řádu (neboť o schématu QUICK je známo, že způsobuje nefyzikální oscilace řešení a může tak způsobit značné potíže [23]), zcela jistě by se výrazněji nesnížila výpočtová náročnost. Uvážíme-li zamýšlené použití diskutovaných modelů a skutečnost, že od nich ani ne-

očekáváme příliš přesná data, vychází jako nejvhodnější schéma „upwind“, případně „power law“.

2.5.3 Zdrojový člen

Standardně je zdrojový člen na pravé straně rovnice 2.7 zintegrované přes kontrolní objem V diskretizován pomocí

$$\int_V S(\phi)\,\mathrm{d}V = \overline{S}\Delta V = S_\mathrm{c} + S_\mathrm{p}\phi_\mathrm{P}, \tag{2.13}$$

kde $\overline{S}$ značí měrnou střední hodnotu $S(\phi)$ v kontrolním objemu V, ΔV jeho objem, S_c konstantní část linearizovaného tvaru a S_p odpovídající koeficient hodnoty veličiny ϕ v bodě „P“ výpočetní sítě (ϕ_P). Pokud bychom se tedy vrátili k obecné diskretizované transportní rovnici 2.8 a předpokládali, že konstanta C obsahuje pouze přírůstek pocházející ze zdrojového členu (tj. že $C = S_\mathrm{c}$), získali bychom

$$a_\mathrm{P}\phi_\mathrm{P} = \sum_\mathrm{nb} a_\mathrm{nb}\phi_\mathrm{nb} + S_\mathrm{c}, \quad \text{kde} \quad a_\mathrm{P} = \sum_\mathrm{nb} a_\mathrm{nb} - S_\mathrm{p}. \tag{2.14}$$

Vzhledem k jednoduchosti této diskretizace a faktu, že se běžně používá i v detailních CFD modelech, není nejmenší důvod diskretizovat zdrojový člen ve zjednodušených modelech jinak.

2.5.4 Tranzientní člen

Volba diskretizačního schématu pro tranzientní člen je z velké části ovlivněna kvalitou výpočetní sítě. Například o standardním explicitním (dopředném) Eulerově schématu je známa jeho náchylnost k nestabilitě, přičemž maximální přípustná délka časového kroku je závislá na druhé mocnině prostorového kroku (tj. toho kterého rozměru buňky výpočetní sítě) [17, kap. 8.2.1]. Je tedy ihned zřejmé, že implementace takového schématu v kombinaci s ne příliš kvalitní kvádrovou výpočetní sítí může snadno snížit robustnost výsledného modelu. Podobné pak je schéma Cranka a Nicolsonové [24], tedy aritmetický průměr explicitní a implicitní diskretizace. Toto schéma se sice vyznačuje stabilitou, avšak maximální délka časového kroku je opět značně omezená výpočetní sítí (byť do menší míry než u explicitního schématu).

Zmíněný problém se však netýká implicitního (zpětného) Eulerova schématu, které je stabilní. Lze totiž volit i výrazně delší časový krok, než jaký by odpovídal limitní hodnotě pro explicitní schéma. Volbu je ovšem nutné provést s rozmyslem, neboť patřičná hodnota musí zohledňovat skutečný stav neustáleného proudění, tedy časové měřítko, ve kterém v proudění probíhají změny. I s výrazně delším časovým krokem nicméně obvykle lze získat řešení, byť to může být na úkor přesnosti dat.

Komerční CFD softwary typicky nabízí osvědčená implicitní diskretizační schémata prvního či druhého řádu, případně schéma Cranka a Nicolsonové. Implicitní schéma

prvního řádu přitom bývá výchozí – a v běžných úlohách týkajících se tepelných výměníků apod. zcela dostačující – možností. Podobně jako u zdrojového členu tedy ani zde není důvod implementovat cokoliv jiného než implicitní schéma prvního řádu. Tranzientní člen z rovnice 2.7 potom po integraci přejde do tvaru

$$\int_{t-\Delta t}^{t} \int_V \frac{\partial(\rho\phi)}{\partial t} \mathrm{d}V \mathrm{d}t = \frac{(\rho\phi)|_t - (\rho\phi)|_{t-\Delta t}}{\Delta t} \Delta V, \tag{2.15}$$

kde Δt značí délku časového kroku. Toto se při užití standardní konvence (veličiny z předchozího časového kroku s horním indexem „0“, veličiny z aktuálního časového kroku bez speciálního označení) projeví v rovnici 2.14 po několika úpravách zohledňujících rovnici spojitosti toku [10, s. 96–99] dodatečným členem $a_\mathrm{P}^0 \phi_\mathrm{P}^0$:

$$a_\mathrm{P}\phi_\mathrm{P} = \sum_\mathrm{nb} a_\mathrm{nb}\phi_\mathrm{nb} + a_\mathrm{P}^0\phi_\mathrm{P}^0 + S_\mathrm{c}. \tag{2.16}$$

Koeficient a_P zde musí být oproti dřívější verzi rovnice 2.14 také upraven,

$$a_\mathrm{P} = \sum_\mathrm{nb} a_\mathrm{nb} + a_\mathrm{P}^0 - S_\mathrm{p}, \tag{2.17}$$

přičemž hodnota a_P^0 je počítána pomocí vztahu

$$a_\mathrm{P}^0 = \rho_\mathrm{P}^0 \frac{\Delta V}{\Delta t}. \tag{2.18}$$

Na závěr ještě poznamenejme, že existují také různá adaptivní schémata, která se snaží vhodně kombinovat vyšší přesnost dopředné diskretizace se stabilitou zpětné diskretizace. Jako příklad lze uvést schéma popsané v článku [25], kde je míra implicitnosti řízena na základě vlastností buněk sítě, nebo akcelerované schéma od de Souzy a kol. [26]. Obecně však platí, že taková schémata vyžadují kvalitní výpočetní síť a jejich implementace ve zjednodušených 3D CFD modelech by tudíž byla problematická.

2.6 Modelování turbulence

Jednorovnicový Spalartův-Allmarasův model [27] ani jeho libovolná modifikace (viz detailní popis této rodiny modelů např. na webu NASA Langley Research Center [28]) nejsou příliš vhodné v případě hrubých mřížek a simulací proudění v doménách komplexních tvarů, kde je obtížné definovat délkové měřítko. Plnohodnotná simulace turbulence by tedy vyžadovala implementaci některého z běžně užívaných modelů (k–ϵ, k–ω apod.) a znamenala by nutnost řešit přinejmenším dvě další soustavy rovnic, což by potažmo vedlo ke znatelnému zpomalení výpočtů. V prvotní verzi simulačního softwaru z kapitoly 2.10 proto bylo přistoupeno pouze ke škálování viskozity pomocí empirického vztahu popsaného v článku [A5]. Tento byl odvozen na základě dat z detailních CFD výpočtů jak

systémů pro distribuci do prostředí s konstantním tlakem [29], tak kompletních distribučních systémů s různými uspořádáními toku („U“, „Z“ a „T“) a trubkového svazku (různé počty řad a trubek v řadách, různá uspořádání trubek) [30]. Model tedy pracuje s „virtuální“ viskozitou $\hat{\mu} = k_\mu \mu$, kde koeficient k_μ je závislý na lokálním Reynoldsově čísle, jemnosti sítě a částečně také geometrii distribučního systému.

Jak bylo ukázáno na obrázku 2.4 v kapitole 2.3.4, tento postup vede k dostatečně přesným výsledkům (z pohledu přibližného modelování v optimalizačních algoritmech či jiných aplikacích odpovídajícího charakteru). Vztah pro koeficient k_μ by u výrazně odlišných zařízení musel zřejmě být upraven, nicméně toto může být předmětem dalšího zkoumání, ukáže-li se, že je potřeba se tím zabývat například v souvislosti s řešením projektů smluvního výzkumu pro zadavatele z aplikační sféry. Co se týče výpočtové náročnosti, v kapitole 2.7 bude ukázáno, že při uvedeném způsobu modelování turbulence lze s vhodnými maticovými řešiči a metodami předpodmínění získávat zkonvergovaná řešení ustálených úloh ve velmi krátkých časech – běžně za jednotky až desítky vteřin v závislosti na počtu buněk ve výpočetní síti.

Výše zmíněný způsob zahrnutí turbulence přitom nijak neovlivní schopnost dopočítávat a vizualizovat vorticitu,

$$\boldsymbol{\omega} = \operatorname{rot} \mathbf{u} = \nabla \times (u, v, w) = \left(\frac{\partial w}{\partial y} - \frac{\partial v}{\partial z}, \frac{\partial u}{\partial z} - \frac{\partial w}{\partial x}, \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \right), \tag{2.19}$$

jejíž velikost je velmi dobrým ukazatelem lokální náchylnosti k zanášení. Příklad aplikace tohoto postupu při predikci zanášení trubkového prostoru tepelného výměníku v jednotce pro termické zneškodňování procesních odplynů a při následné analýze možných nápravných opatření lze najít v článku [A2].

2.7 Maticové řešiče a metody předpodmínění

Navzdory tomu, že se zjednodušené CFD modely velmi podobají modelům standardním, u nich není možné k řešení soustav rovnic přistupovat identicky. Užití kvádrové – a potažmo relativně hrubé – výpočetní sítě má totiž nezanedbatelný negativní vliv na numerické chování a mnohem častěji zde dochází k potížím s konvergencí. Numerické metody pro řešení soustav diskretizovaných rovnic, stejně jako případné metody předpodmínění[†], tudíž musí být voleny s ohledem na specifika zde uvažovaných CFD modelů a nelze se příliš spoléhat na dostupnou literaturu (například monografii týkající se numerických řešičů [31], souhrnný přehled metod předpodmínění [32] či obecné srovnání vybraných metod předpodmínění [33]). Autorem této práce byly proto vypracovány dvě rozsáhlé numerické studie [A6, A7] porovnávající rychlost výpočtu a konvergenci při použití různých maticových řešičů spolu s různými metodami předpodmínění.

[†] Předpodmíněním („preconditioning“) se rozumí, že namísto původního lineárního systému $\mathbf{Ax} = \mathbf{b}$ je řešen systém $\mathbf{M}^{-1}\mathbf{Ax} = \mathbf{M}^{-1}\mathbf{b}$ (příp. $\mathbf{AM}^{-1}(\mathbf{Mx}) = \mathbf{b}$) s maticí $\mathbf{M}$ vhodně zvolenou tak, aby se nový systém – zjednodušeně řečeno – vyznačoval lepší řešitelností.

Veškeré testovací výpočty byly ve zmíněných studiích prováděny pomocí softwaru popsaného dále v kapitole 2.10, do kterého byly přidány potřebné funkcionality, např.:

- práce s předem definovanými sadami testových úloh namísto spouštění úloh jednotlivě,
- automatické opakované provádění výpočtů k získání dostatečného počtu pozorovaných hodnot,
- statistické zpracování výsledných datových souborů atd.

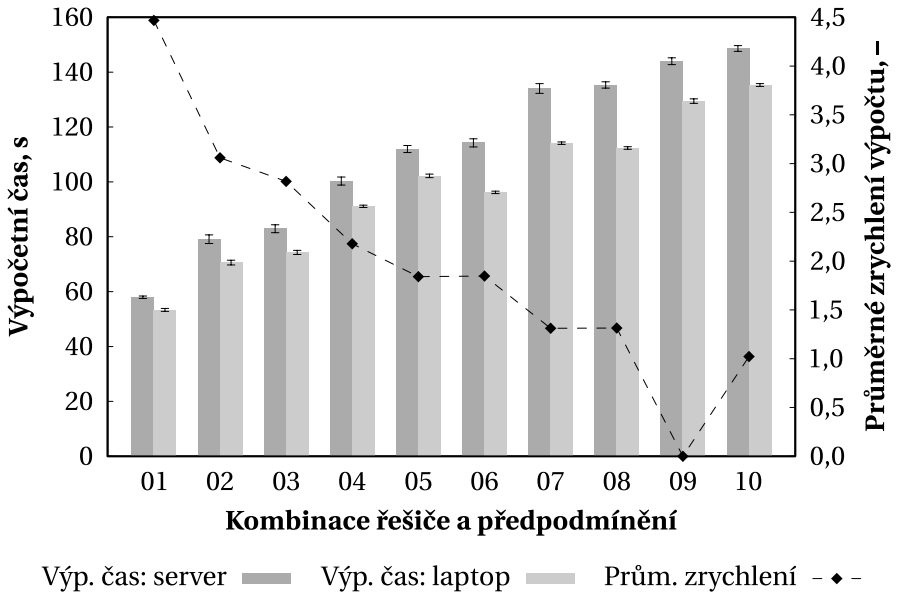
Odpovídající software byl napsán v jazyce Java, který obsahuje Just-in-Time (JIT) kompilátor provádějící optimalizace strojového kódu dle potřeby i za běhu aplikace. Bylo tedy nutné zajistit, aby činnost kompilátoru neovlivňovala měřené hodnoty. Toho bylo dosahováno zařazením dostatečně rozsáhlé „zahřívací" fáze, během které proběhly veškeré dodatečné kompilace a optimalizace kódu, před každou „měřicí" fázi (detaily týkající se uvedeného procesu lze najít ve studii [A7]). Všechny testovací výpočty pak byly vždy opakovány až 80krát (dle velikosti výpočetní sítě), z čehož až 30 opakování představovalo zmíněnou „zahřívací" fázi. Ze sad hodnot získaných během „měřicích" fází nakonec byly dopočteny střední hodnoty a směrodatné odchylky. Celkem bylo ve studiích [A6, A7] otestováno 51 828 různých kombinací výpočetních sítí a numerických metod.

Vhodnost běžně používaných maticových řešičů k aplikaci ve zjednodušených CFD modelech byla zkoumána primárně ve studii [A7]. Kromě nepředpodmíněných řešičů zde byly uvažovány i verze využívající obvyklé metody předpodmínění. Studie [A6] se pak zaměřovala výhradně na použitelnost symetrické neúplné relaxace („Symmetric Successive Over-Relaxation", SSOR) [34] v jiném než Gaussově-Seidelově tvaru, tj. s relaxačními parametry $\omega_F, \omega_R \neq 1{,}0$.

V souladu s kapitolou 2.6 byla turbulence ve všech úlohách modelována umělým škálováním viskozity, nikoliv explicitně pomocí některého ze standardních modelů. Důvodem je skutečnost, že dle dosud provedených testů není tímto zjednodušením nijak zásadně ovlivněna přesnost dat (ve smyslu přibližného modelování), avšak výsledné výpočetní časy jsou výrazně kratší. Jsou ovšem plánovány dodatečné testy, které by měly tento předběžný závěr pomocí širší sady různých testovacích distribučních systémů potvrdit, resp. vyvrátit (v takovém případě by pak nezbylo než implementovat do existujícího 3D CFD softwaru z kapitoly 2.10 i některý z plnohodnotných modelů turbulence).

2.7.1 Simulace proudění bez přenosu energie

Obrázek 2.10 uvádí deset nejlepších kombinací maticových řešičů a metod předpodmínění pro případ, kdy simulace zahrnovaly pouze proudění, nikoliv však přenos energie. Prezentované výpočetní časy odpovídají dosažení zkonvergovaného řešení ustálené úlohy na reprezentativní výpočetní síti s ~25 tis. buňkami. Předpodmínění typu „Jacobi" [35], které bylo použito u kombinace č. 10 ze zmíněného obrázku, odpovídá sestrojení patřičné matice pouze z diagonály matice řešeného lineárního systému.



Kombinace:

01 – p: CG:ILU, m: BiCGstab(3):ILU	06 – p: CG:SSOR, m: BiCGstab(2):SSOR
02 – p: CG:SSOR, m: BiCGstab(3):SSOR	07 – p: CG:ILU, m: BiCGstab(1):ILU
03 – p: CG:ILU, m: BiCGstab(2):ILU	08 – p: CG:ILU, m: BiCGstab(1):—
04 – p: CG:—, m: BiCGstab(3):SSOR	09 – p: BiCGstab(1):—, m: BiCGstab(6):—
05 – p: CG:—, m: BiCGstab(3):ILU	10 – p: CG:Jacobi, m: BiCGstab(1):Jacobi

Obrázek 2.10. Výpočetní časy odpovídající deseti nejlepším kombinacím maticových řešičů a metod předpodmínění pro simulace zahrnující pouze proudění na výpočetní síti s ~25 tis. buňkami; „p“ značí tlakovou korekci a „m“ rovnice pro jednotlivé složky rychlosti. Výpočty byly prováděny na dvou strojích s výrazně odlišnými výkony (server: Intel Xeon E5 2698 v4, 128 GB RAM; laptop: Intel Core i5-6300U, 16 GB RAM) za účelem zjištění, zda druh CPU významně ovlivňuje výpočty prováděné pouze na jednom jádře. Průměrné zrychlení bylo počítáno pomocí vztahu $t_B/t_P - 1$, kde t_P značí výpočetní čas při užití dané kombinace řešičů a metod předpodmínění, zatímco t_B čas odpovídající stejným řešičům, avšak nepředpodmíněným (adaptováno z [A7]).

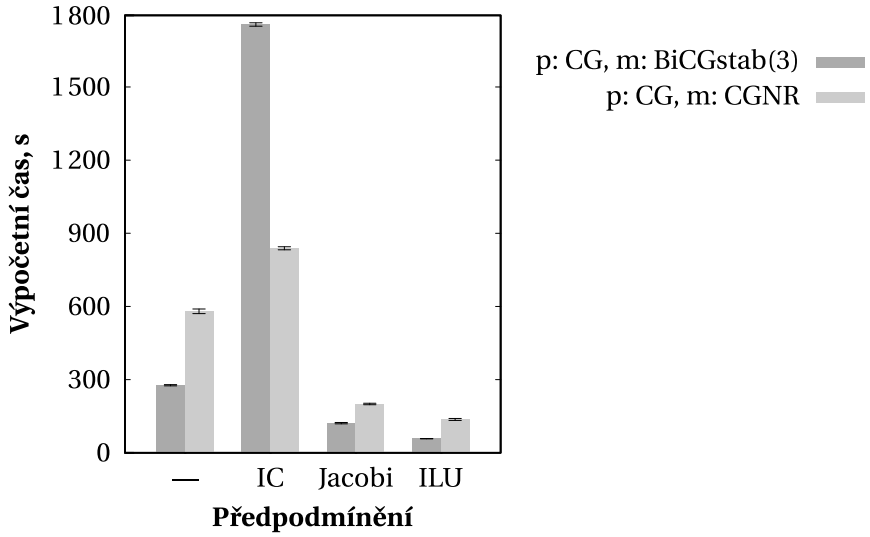
Z grafu vyplývá, že u relativně hrubých kvádrových sítí dokáže největší rychlost výpočtu poskytnout metoda sdružených gradientů („Conjugate Gradient Method“, CG) [36] předpodmíněná pomocí neúplného LU rozkladu se zaplněním odpovídajícím původní matici[†] („Incomplete Lower-Upper Factorization with Zero-Level Fill-In“, ILU(0); pro jednoduchost dále označováno pouze jako ILU) [39] v případě řešení tlakové korekce a stejně předpodmíněná stabilizovaná metoda bi-sdružených gradientů („Bi-Conjugate Gradient Method Stabilized with Minimization of Residuals over l-Dimensional Subspa-

[†]Neúplné rozklady se zaplněním řádu $n > 0$, tj. ILU(n) [37], ILUT(n) [31, kap. 10.4] a IC(n) [38], nebyly zatím testovány, nicméně mohou představovat další způsob, jak zlepšit numerické chování modelu.

ces“, BiCGstab(l)) [40] s $l = 3$ při řešení složek rychlosti proudění. Druhou nejrychleji konvergující kombinací pak tvoří stejné maticové řešiče předpodmíněné pomocí SSOR v Gaussově-Seidelově tvaru (tedy s oběma relaxačními faktory rovnými jedné; jinak relaxované verze SSOR byly uvažovány až ve studii [A6]). Z obrázku je dále patrné, že aplikací předpodmínění lze dosáhnout až 4,5násobného zrychlení výpočtu. Výsledky získané pro hrubší síť obsahující řádově méně buněk pak byly podobné, přičemž nejvyšší míra zrychlení dosahovala hodnoty přibližně 4,0.

Co se týče například hojně využívané zobecněné metody minimálních reziduí („Generalized Minimal Residual Method“, GMRES) [41], neúplného LU rozkladu s tolerancí a zaplněním odpovídajícím původní matici („Dual-Threshold Incomplete LU Factorization with Zero-Level Fill-In“, ILUT(0); dále označováno pouze jako ILUT) [42] či neúplného Choleského rozkladu se zaplněním odpovídajícím původní matici („Incomplete Cholesky Factorization with Zero-Level Fill-In“, IC(0); dále jen IC) [43], tyto se ukázaly být nepřípustně pomalými, resp. mnohdy dokonce výrazně náchylnějšími k divergenci. Z ostatních numerických metod zmíněných ve studii [A7] má smysl zde uvést snad jen metodu sdružených gradientů aplikovanou na systém normálních rovnic s minimalizací normy rezidua („Conjugate Gradient Method on the Normal Residuals“, CGNR) [31, kap. 8.3.1], která sice může být použita k řešení rovnic pro složky rychlosti proudění, avšak – jak je vidět z obrázku 2.11 – obecně taktéž vede k relativně pomalé konvergenci.

Ačkoliv se z obrázku 2.10 může zdát, že rozdíly mezi použitými procesory jsou podstatné, není tomu tak. Průměrný relativní rozdíl mezi výpočetními časy pozorovanými na serveru a na laptopu byl totiž přibližně 0,1 %. Lze tedy konstatovat, že z pohledu



Obrázek 2.11. Výpočetní časy získané pomocí dvou kombinací identicky předpodmíněných numerických řešičů pro simulace zahrnující pouze proudění na výpočetní síti odpovídající obrázku 2.10; „p“ značí tlakovou korekci a „m“ rovnice pro jednotlivé složky rychlosti

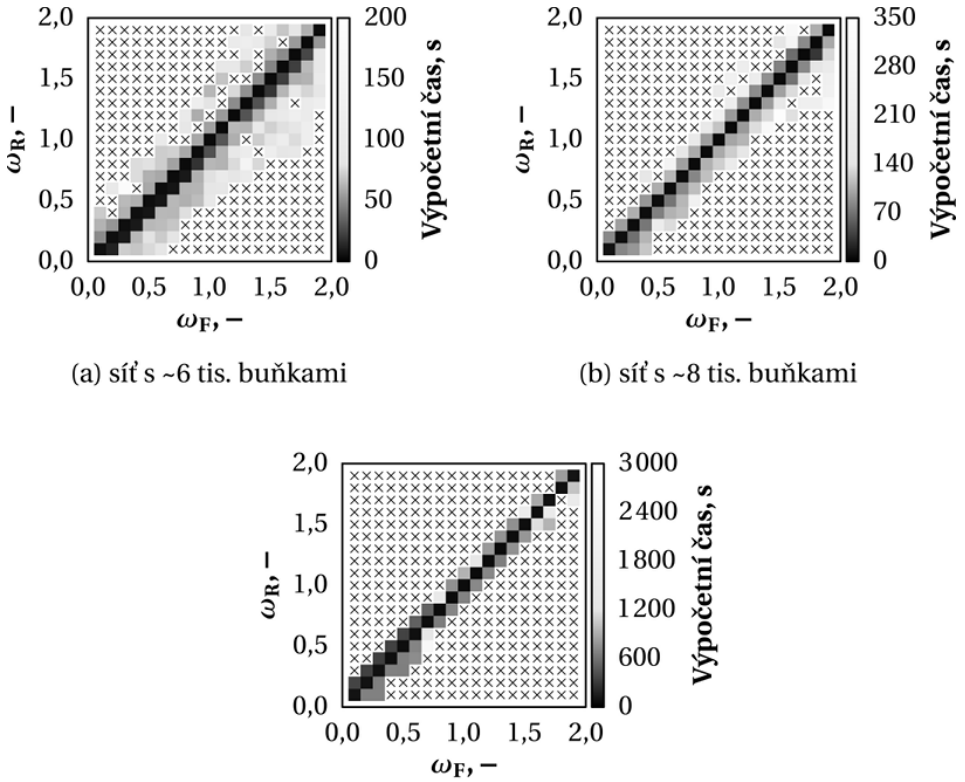
jednojádrových výpočtů (jejichž volba bude zdůvodněna dále v kapitole 2.9) není mezi moderními procesory významný rozdíl, byť by tyto byly určeny pro hardwarově diametrálně odlišné aplikace.

S ohledem na výsledky studie [A7] prezentované výše byly v následné studii [A6] uvažovány pouze vybrané metody řešení soustav lineárních rovnic a předpodmínění. V případě tlakové korekce byla vždy použita metoda sdružených gradientů, zatímco rovnice pro jednotlivé složky rychlosti proudění byly vždy řešeny pomocí stabilizované metody bi-sdružených gradientů s parametrem $l \leq 3$. Na tomto místě je také vhodné podotknout, že oproti studii [A7] se zde pracovalo s jinými testovacími výpočetními sítěmi a nadto byl kód testovací javové aplikace mezi jednotlivými studiemi vylepšen. Výpočetní časy uváděné pro jednotlivé studie tedy nelze přímo srovnávat. Toto však nevadí, neboť veškeré nově uvažované výpočetní sítě byly kromě symetrické neúplné relaxace testovány i za použití neúplného LU rozkladu. Bylo tudíž možné zjistit, zda – a případně do jaké míry – je aplikace symetrické neúplné relaxace s různými hodnotami relaxačních parametrů ω_F, ω_R výhodnější.

Během testovacích výpočtů byly postupně užívány různé hodnoty relaxačních parametrů ω_F, $\omega_R = 0{,}1, 0{,}2, \ldots, 1{,}8, 1{,}9$. Čtvercová okolí slibných kombinací (ω_F, ω_R) poté byla dále vyhodnocována s krokem 0,02, tj. užitím hodnot $\omega - 0{,}04$, $\omega - 0{,}02$, ω, $\omega + 0{,}02$ a $\omega + 0{,}04$. Ukazatelem vhodnosti té které kombinace numerických metod byl opět výpočetní čas nutný k dosažení zkonvergovaného řešení ustálené úlohy.

Podle dřívějších testů se kombinace metody sdružených gradientů a symetrické neúplné relaxace zdála vhodná i pro řešení tlakové korekce. Nejprve tedy byly provedeny testovací výpočty, při nichž byla tlaková korekce řešena právě tímto způsobem, zatímco u rovnic pro jednotlivé složky rychlosti proudění byly řešiče předpodmíněny pomocí neúplného LU rozkladu. Ukázalo se však, že tento postup je při hodnotách ω_F, $\omega_R \neq 1{,}0$ zcela nevyhovující (zvláště pak u rozsáhlejších sítí), neboť většina výpočtů buď skončila divergencí, nebo trvala nepřípustně dlouhou dobu. Zmíněné chování je znázorněno pro tři různě velké výpočetní sítě na obrázku 2.12 (detailní specifikaci parametrů těchto sítí, nastavení CFD úloh a iteračních a dalších limitů lze nalézt ve studii [A6]). Je tedy zřejmé, že v případě tlakové korekce má smysl použít diskutovanou kombinaci numerického řešiče a předpodmínění snad jen v Gaussově-Seidelově tvaru.

V následných testovacích výpočtech zahrnujících symetrickou neúplnou relaxaci s ω_F, $\omega_R \neq 1{,}0$ proto bylo při řešení tlakové korekce užíváno výhradně předpodmínění pomocí neúplného LU rozkladu. Co se týče rovnic pro jednotlivé složky rychlosti proudění, bylo zjištěno, že s rostoucí hodnotou parametru l klesají nejen výpočetní časy, ale i stabilita výpočtu. Jinak řečeno, volbou vyšší hodnoty parametru l ve stabilizované metodě bi-sdružených gradientů sice dosáhneme rychlejšího výpočtu, ale za cenu vyšší pravděpodobnosti divergence atp. – viz obrázek 2.13. Z tohoto obrázku je také zřejmé, že při $l = 3$ se nejlepší kombinace relaxačních parametrů nacházely v okolí $(1{,}0; 1{,}0)$ a hodnoty $\omega_F \notin [0{,}5; 1{,}5]$ byly veskrze nevhodné. Přesto se však zmíněná metoda v kombinaci s předpodmíněním pomocí symetrické neúplné relaxace osvědčila bez ohledu



(a) síť s ~6 tis. buňkami

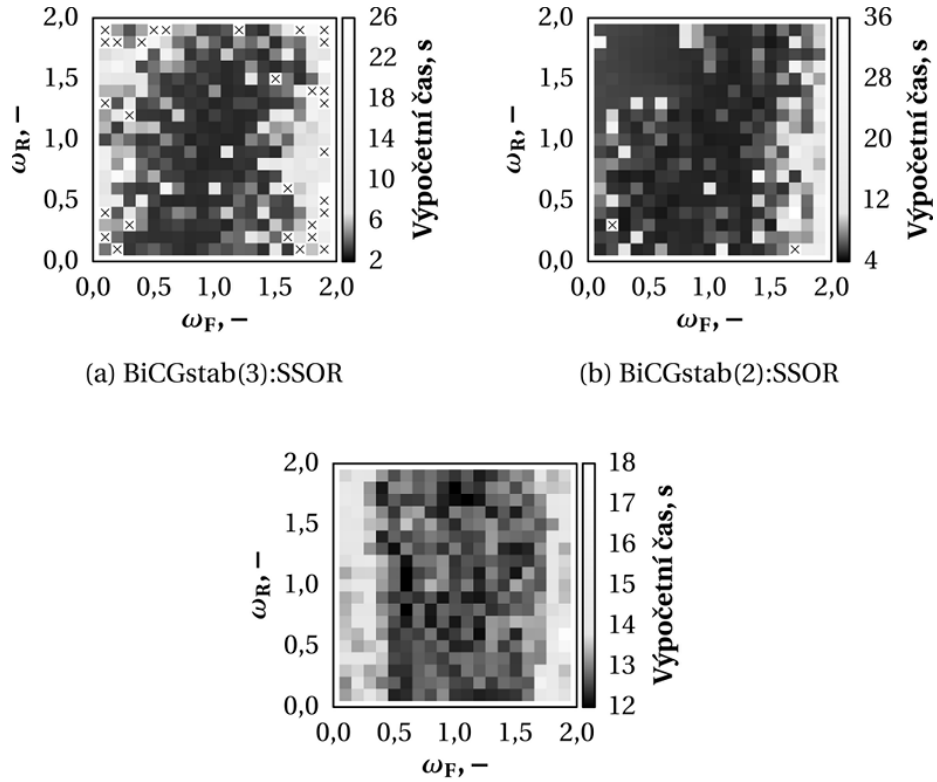
(b) síť s ~8 tis. buňkami

(c) síť s ~25 tis. buňkami

Obrázek 2.12. Průměrné výpočetní časy pro různé kombinace relaxačních parametrů ω_F a ω_R získané pomocí tří různě velkých výpočetních sítí; tlaková korekce byla řešena pomocí CG:SSOR a složky rychlosti proudění pomocí BiCGstab(3):ILU. Diagonální křížky značí kombinace relaxačních parametrů, se kterými nebyl proces řešení úspěšný (v důsledku divergence, příp. vzácněji vlivem překročení časového či iteračního limitu). Stupnice byly upraveny za účelem snazší identifikace nejlepších kombinací (ω_F, ω_R). Dále je nutné podotknout, že vzhledem k potřebným rozsahům stupnic jsou jejich dolní meze rovny nule, i když jsou vykreslovaná data vždy výrazně nenulová (adaptováno z [A6]).

na velikost sítě a lze říci, že při rozumné volbě relaxačních parametrů ω_F, ω_R by nemělo docházet k výrazným numerickým potížím.

Důkladnějším prozkoumáním čtvercových okolí slibných kombinací (ω_F, ω_R) byl získán např. graf uvedený na obrázku 2.14. Pro pozdější snadnou volbu relaxačních parametrů v softwaru z kapitoly 2.10 byly pomocí vážené metody nejmenších čtverců zjištěny průběhy závislostí $\omega_R = \alpha\omega_F + \beta$. Potřebné váhy byly přitom počítány vztahem $W_i = (t_{\min}/t_i)^4$, kde $t_{\min}$ značí minimální výpočetní čas pozorovaný s určitou kombinací výpočetní sítě a numerických metod a t_i výpočetní čas zjištěný u patřičné ($i$té) kombinace relaxačních parametrů (ω_F, ω_R). Tím došlo k adekvátnímu potlačení vlivu kombinací relaxačních parametrů, s nimiž byly výpočetní časy dále od času minimálního.



(a) BiCGstab(3):SSOR

(b) BiCGstab(2):SSOR

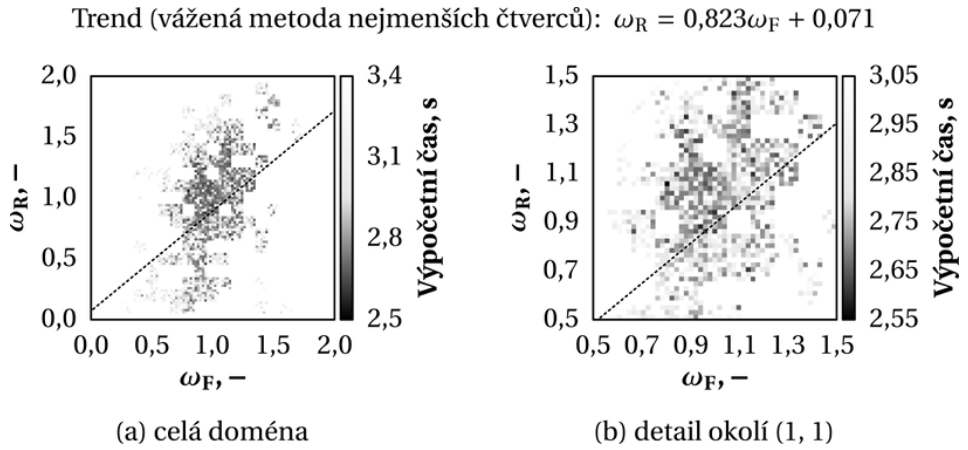
(c) BiCGstab(1):SSOR

Obrázek 2.13. Průměrné výpočetní časy pro síť obsahující ~6 tis. buněk a různé kombinace numerických řešičů a metod předpodmínění. Tlaková korekce byla vždy řešena pomocí CG:ILU, rovnice pro jednotlivé složky rychlosti proudění pak pomocí BiCGstab(l):SSOR s $l = 1,\ 2,$ resp. 3. Diagonální křížky značí kombinace relaxačních parametrů, se kterými nebyl proces řešení úspěšný. Stupnice byly upraveny za účelem snazší identifikace nejlepších kombinací (ω_F, ω_R) (adaptováno z [A6]).

Z výsledků získaných pomocí všech testovaných výpočetních sítí byl nakonec identickým způsobem nalezen celkový trend

$$\omega_R = 0{,}861\,\omega_F + 0{,}056. \tag{2.20}$$

Vzhledem ke skutečnosti, že nejrychleji konvergující výpočty pracovaly s $\omega_F \approx 0{,}9$, by při uvážení rovnice 2.20 měl i zpětný chod symetrické neúplné relaxace využívat relaxační parametr menší než 1,0 (tedy že je u obou chodů vhodné takto zlepšovat konvergenci na úkor rychlosti výpočtu). Obecně také lze říci, že u simulací nezahrnujících přenos energie byla konvergence s nejlepší kombinací (ω_F, ω_R) v průměru o ~13 % pomalejší než při předpodmínění neúplným LU rozkladem.



(a) celá doména (b) detail okolí (1, 1)

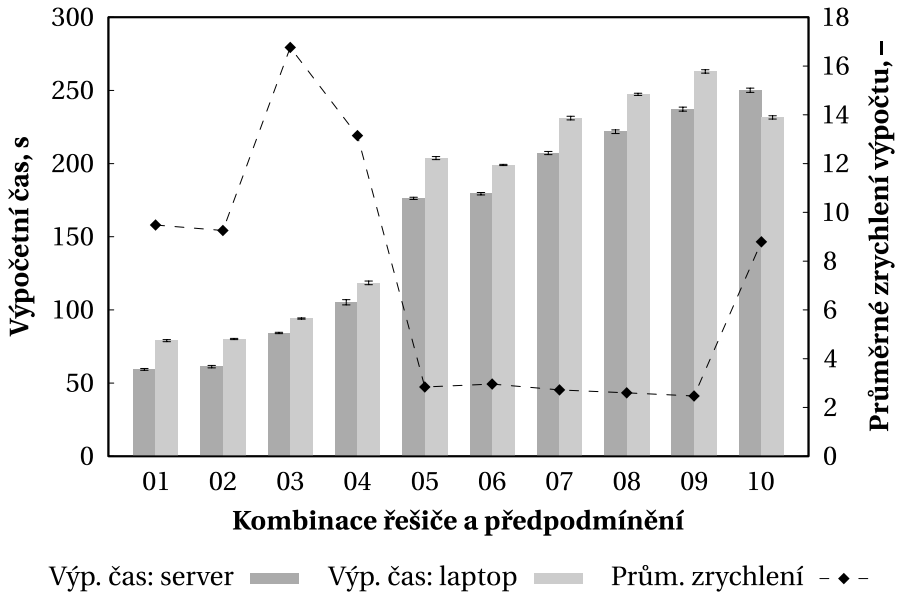
Obrázek 2.14. Průměrné výpočetní časy pro síť odpovídající obrázku 2.13 a okolí slibných kombinací relaxačních parametrů (ω_F, ω_R). Tlaková korekce byla vždy řešena pomocí CG:ILU, rovnice pro jednotlivé složky rychlosti proudění pak pomocí BiCGstab(3):SSOR. Stupnice vč. jejich hraničních hodnot byly výrazně upraveny za účelem snazší identifikace nejlepších kombinací relaxačních parametrů (adaptováno z [A6]).

2.7.2 Simulace proudění včetně přenosu energie

V případě simulací zahrnujících i přenos energie byly výsledky z pohledu numerických metod použitých pro řešení tlakové korekce a složek rychlosti proudění podobné. U řešiče použitého pro energii ovšem lze vidět značně rozdílné kombinace metod (viz obrázek 2.15). Je zřejmé, že všech deset nejlepších kombinací používalo pro řešení tlakové korekce metodu sdružených gradientů předpodmíněnou pomocí neúplného LU rozkladu. Pro složky rychlosti byla vždy nejlepší stabilizovaná metoda bi-sdružených gradientů s $l = 2$, resp. $l = 3$; opět předpodmíněná výhradně pomocí neúplného LU rozkladu. Co se týče rovnice energie, kromě stabilizované metody bi-sdružených gradientů s různými hodnotami parametru l lze vidět i metodu sdružených gradientů aplikovanou na systém normálních rovnic s minimalizací normy rezidua. Nejkratších výpočetních časů však zde bylo dosaženo užitím stabilizované metody bi-sdružených gradientů s $l = 2$ předpodmíněné pomocí symetrické neúplné relaxace v Gaussově-Seidelově tvaru, příp. neúplného LU rozkladu. Odpovídající zrychlení výpočtu pak bylo přibližně 9,5násobné, resp. u třetí nejlepší kombinace dokonce 16,8násobné.

U hrubší sítě obsahující řádově méně buněk byly nejvhodnější kombinace z pohledu řešení tlakové korekce a složek rychlosti proudění v podstatě identické, ovšem pro řešení rovnice energie byla použita stejně předpodmíněná stabilizovaná metoda bi-sdružených gradientů výhradně s $l = 2$, resp. $l = 1$. Pozorované zrychlení výpočtu přitom bylo 3,6–6,9násobné.

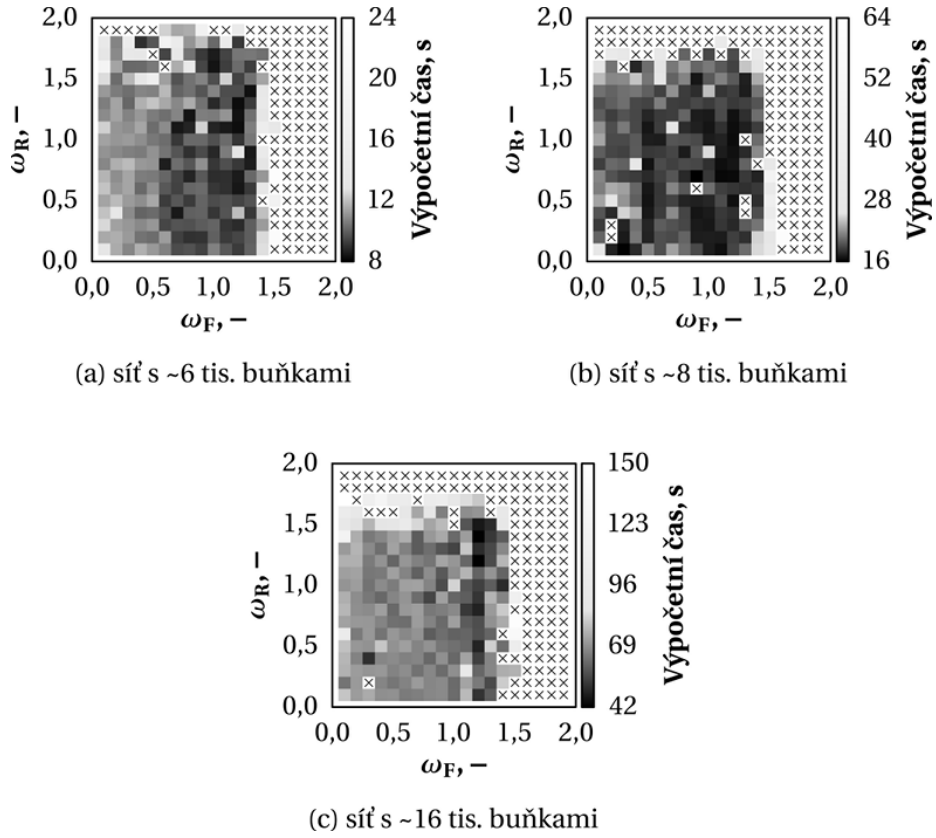
Následně byla opět testována vhodnost symetrické neúplná relaxace, avšak zatím pouze v případě řešení rovnice energie (ostatní řešiče využívaly podobně jako dříve



Kombinace:
01 – p: CG:ILU, m: BiCGstab(2):ILU, e: BiCGstab(2):SSOR
02 – p: CG:ILU, m: BiCGstab(2):ILU, e: BiCGstab(2):ILU
03 – p: CG:ILU, m: BiCGstab(3):ILU, e: BiCGstab(1):SSOR
04 – p: CG:ILU, m: BiCGstab(3):ILU, e: BiCGstab(1):Jacobi
05 – p: CG:ILU, m: BiCGstab(3):ILU, e: BiCGstab(2):—
06 – p: CG:ILU, m: BiCGstab(3):ILU, e: BiCGstab(3):—
07 – p: CG:ILU, m: BiCGstab(3):ILU, e: BiCGstab(6):—
08 – p: CG:ILU, m: BiCGstab(3):ILU, e: BiCGstab(7):—
09 – p: CG:ILU, m: BiCGstab(3):ILU, e: BiCGstab(8):—
10 – p: CG:ILU, m: BiCGstab(3):ILU, e: CGNR:Jacobi

Obrázek 2.15. Výpočetní časy odpovídající deseti nejlepším kombinacím maticových řešičů a metod předpodmínění pro simulace zahrnující proudění i přenos energie na výpočetní síti s ~25 tis. buňkami; „p“ značí tlakovou korekci, „m“ rovnice pro jednotlivé složky rychlosti a „e“ energii. Stejně jako v případě proudění bez přenosu energie byly výpočty prováděny na dvou strojích s výrazně odlišnými výkony. Průměrné zrychlení bylo počítáno identicky s obrázkem 2.10 (adaptováno z [A7]).

předpodmínění neúplným LU rozkladem; pro detaily týkající se nastavení CFD úloh atd. je čtenář opět odkázán na studii [A6]). Bylo zjištěno, že nejlepší je řešit rovnice pro jednotlivé složky rychlosti proudění i rovnici energie pomocí stabilizované metody bi-sdružených gradientů, a to s $l = 3$ v případě složek rychlosti a $l = 1$ u energie (viz obrázek 2.16). Tato kombinace byla – nejspíš vlivem vyšší stability BiCGstab(1) – relativně robustní. Druhou nejlepší možností pak bylo řešit oba typy rovnic pomocí stejné metody, avšak vždy s parametrem $l = 2$. Uvedený postup se nicméně vyznačoval výrazně větším



(a) síť s ~6 tis. buňkami

(b) síť s ~8 tis. buňkami

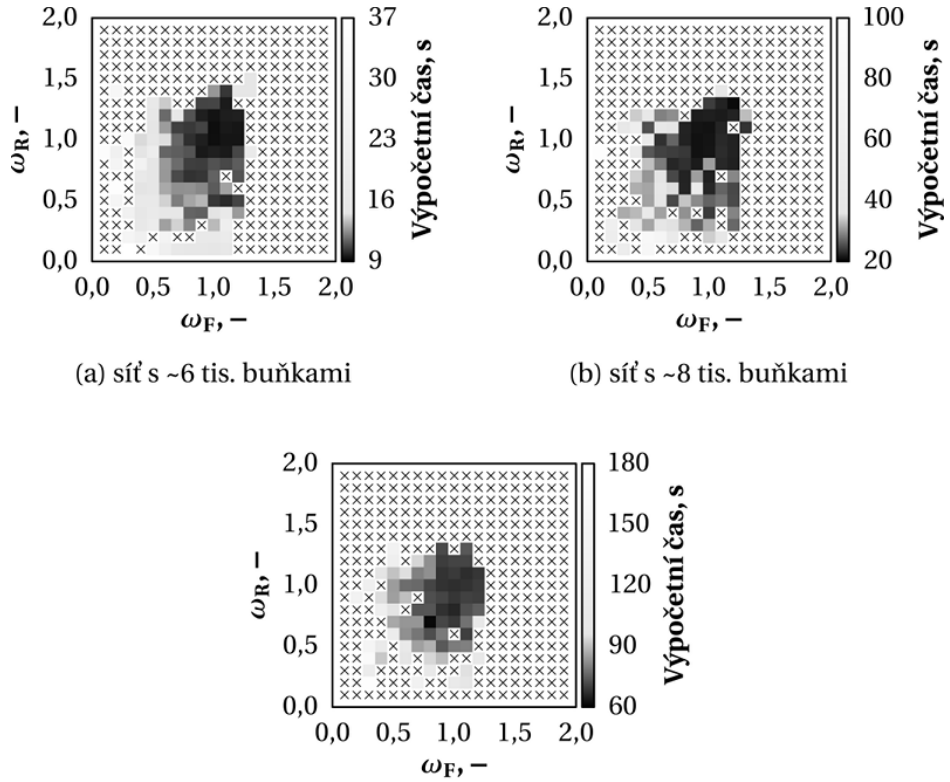
(c) síť s ~16 tis. buňkami

Obrázek 2.16. Průměrné výpočetní časy pro různé kombinace relaxačních parametrů ω_F a ω_R získané pomocí tří různě velkých výpočetních sítí; tlaková korekce byla řešena pomocí CG:ILU, složky rychlosti proudění pomocí BiCGstab(3):ILU a rovnice energie pomocí BiCGstab(1):SSOR. Diagonální křížky značí kombinace relaxačních parametrů, se kterými nebyl proces řešení úspěšný (v důsledku divergence, příp. vzácněji vlivem překročení časového či iteračního limitu). Stupnice byly upraveny za účelem snazší identifikace nejlepších kombinací (ω_F, ω_R) (adaptováno z [A6]).

výskytem numerických potíží a vedl k průměrně o ~38 % delším výpočetním časům. Použitelné hodnoty relaxačních parametrů byly navíc pouze v relativně malém okolí $(1,\ 1)$, jak je zřejmé z obrázku 2.17. Dále lze říci, že příčinou neúspěšných řešení byla v menší míře divergence a naopak mnohem častěji pomalá konvergence.

Stejně jako v případě simulace pouhého proudění byla i zde detailně vyhodnocena okolí slibných kombinací relaxačních parametrů (ω_F, ω_R) a následně byly zjištěny trendy pro jednotlivé sítě a celkový trend. Pro nejlepší kombinaci numerických metod (p: CG, m: BiCGstab(3), e: BiCGstab(1)) byl tento

$$\omega_R = 0{,}866\omega_F + 0{,}042. \tag{2.21}$$



(a) síť s ~6 tis. buňkami

(b) síť s ~8 tis. buňkami

(c) síť s ~16 tis. buňkami

Obrázek 2.17. Průměrné výpočetní časy pro různé kombinace relaxačních parametrů ω_F a ω_R získané pomocí tří různě velkých výpočetních sítí; tlaková korekce byla řešena pomocí CG:ILU, složky rychlosti proudění pomocí BiCGstab(2):ILU a rovnice energie pomocí BiCGstab(2):SSOR. Diagonální křížky značí kombinace relaxačních parametrů, se kterými nebyl proces řešení úspěšný (v důsledku překročení časového limitu, příp. vzácněji vlivem divergence či překročení iteračního limitu). Stupnice byly upraveny za účelem snazší identifikace nejlepších kombinací (ω_F, ω_R) (adaptováno z [A6]).

U druhé nejlepší kombinace (p: CG, m + e: BiCGstab(2)) pak byl nalezen trend

$$\omega_R = 0{,}972\omega_F + 0{,}005. \tag{2.22}$$

Jelikož nejkratší výpočetní časy byly získány s $\omega_F \approx 1{,}2$ (u nejlepší kombinace), resp. $\omega_F \approx 1{,}1$, lze říci, že s ohledem na trendy z rovnic 2.21 a 2.22 by i zpětný chod symetrické neúplné relaxace měl v obou případech pracovat s $\omega_F > 1{,}0$ (tedy že můžeme oba chody mírně akcelerovat, aniž by to mělo negativní vliv na konvergenci).

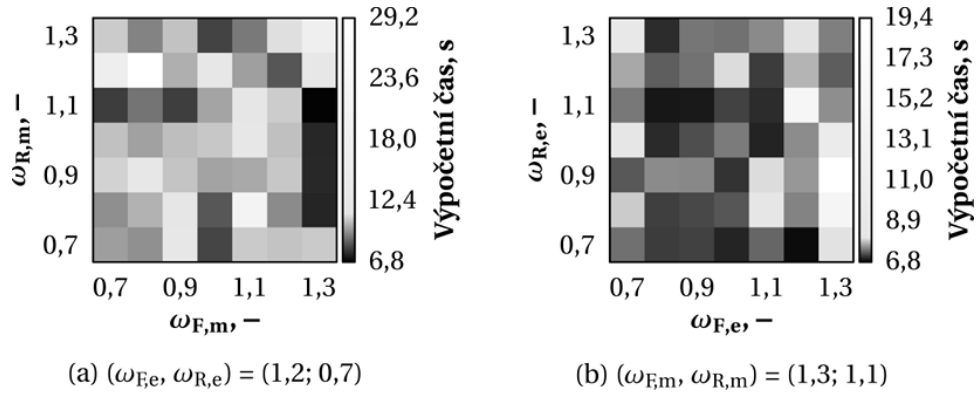
Co se týče výpočetních časů odpovídajících rovnici energie předpodmíněné symetrickou neúplnou relaxací v porovnání s předpodmíněním pomocí neúplného LU rozkladu,

tyto byly při uvažování nejlepší nalezené kombinace relaxačních parametrů ω_F, ω_R v průměru o ~31 % delší.

Poslední sada testů zahrnovala případy, kdy byla symetrická neúplná relaxace použita při řešení jak rovnic složek rychlosti proudění, tak energie (tlaková korekce však byla z dříve uvedeného důvodu stále řešena pomocí metody sdružených gradientů předpodmíněné pomocí neúplného LU rozkladu). Vzhledem k nutnosti volit čtyři relaxační parametry namísto dvou ovšem byla testována pouze omezená doména odpovídající $\omega_F, \omega_R \in [0{,}7;\ 1{,}3]$, kde se dal očekávat výskyt nejlepších kombinací relaxačních parametrů. Tím došlo k výraznému snížení počtu vyhodnocovaných případů – z jednotek milionů na desetitisíce –, přičemž stále byla získána relativně vypovídající data.

Případné nezískání řešení bylo způsobeno spíše pomalou konvergencí než divergencí a frekvence takových numerických potíží byla opět výrazně vyšší při použití kombinace numerických metod zahrnující stabilizovanou metodu bi-sdružených gradientů s parametrem $l = 2$. Vizualizace výsledků je zde obtížnější, neboť by ideálně byly potřeba čtyřrozměrné grafy. Jednou z možností, jak toto omezení obejít, je vykreslení výpočetních časů pro jednu z dvojic relaxačních parametrů s tím, že druhá dvojice je předem pevně zvolena. Příklady patřičných grafů jsou znázorněny na obrázku 2.18.

Při uvážení faktu, že dvojice relaxačních parametrů jsou relativně nezávislé, bylo možné odpovídající trendy hledat odděleně. Pro nejlepší kombinaci numerických metod



(a) $(\omega_{F,e}, \omega_{R,e}) = (1{,}2;\ 0{,}7)$ (b) $(\omega_{F,m}, \omega_{R,m}) = (1{,}3;\ 1{,}1)$

Obrázek 2.18. Dvourozměrné grafy průměrných výpočetních časů pro síť s ~6 tis. buňkami a různé kombinace relaxačních parametrů, kde byla pevně zvolena (a) nejlepší kombinace parametrů nalezená při řešení rovnice energie, $(\omega_{F,e}, \omega_{R,e}) = (1{,}2;\ 0{,}7)$, resp. (b) nejlepší kombinace parametrů nalezená při řešení jednotlivých složek rychlosti proudění, $(\omega_{F,m}, \omega_{R,m}) = (1{,}3;\ 1{,}1)$. Tlaková korekce byla řešena pomocí CG:ILU, složky rychlosti proudění pomocí BiCGstab(3):SSOR a rovnice energie pomocí BiCGstab(1):SSOR. Stupnice byly upraveny za účelem snazší identifikace nejlepších kombinací (ω_F, ω_R) (adaptováno z [A6]).

(p: CG, m: BiCGstab(3), e: BiCGstab(1)) byly nalezeny závislosti

$$\begin{aligned} \omega_{\mathrm{R,m}} &= 0{,}936\omega_{\mathrm{F,m}} + 0{,}013 \quad \text{a} \\ \omega_{\mathrm{R,e}} &= 0{,}949\omega_{\mathrm{F,e}} + 0{,}011, \end{aligned} \tag{2.23}$$

kde dolní indexy „m“ a „e“ značí složky rychlosti proudění, resp. energii. U druhé nejlepší kombinace (p: CG, m + e BiCGstab(2)) pak šlo o

$$\begin{aligned} \omega_{\mathrm{R,m}} &= 0{,}965\omega_{\mathrm{F,m}} + 0{,}011 \quad \text{a} \\ \omega_{\mathrm{R,e}} &= 0{,}990\omega_{\mathrm{F,e}} + 0{,}010. \end{aligned} \tag{2.24}$$

Z rovnic výše vylývá, že u složek rychlosti proudění je možné oba chody symetrické neúplné relaxace mírně akcelerovat ($\omega_{\mathrm{F,m}} \in [1{,}1;\ 1{,}3]$, $\omega_{\mathrm{R,m}} \leq \omega_{\mathrm{F,m}}$), a to bez ohledu na diskutované numerické metody. V případě rovnice energie pak lze oba chody akcelerovat pouze při užití BiCGstab(1) ($\omega_{\mathrm{F,e}} \leq 1{,}2$, $\omega_{\mathrm{R,e}} \leq \omega_{\mathrm{F,e}}$), zatímco s BiCGstab(2) by dopředný chod měl být prováděn bez akcelerace a mírně akcelerovat lze až zpětný chod ($\omega_{\mathrm{F,e}} \approx 1{,}0$, $\omega_{\mathrm{R,e}} \geq \omega_{\mathrm{F,e}}$).

Výpočetní časy dosažené při předpodmínění obou zmíněných typů rovnic pomocí symetrické neúplné relaxace byly v porovnání s předpodmíněním výhradně pomocí neúplného LU rozkladu průměrně přinejmenším o ~35 % delší. Nadto je vzhledem k získaným datům nutné vzít v potaz skutečnost, že druhá nejlepší kombinace numerických metod zahrnující BiCGstab(2) se sama o sobě u řešených úloh oproti té první vyznačovala výrazně delšími výpočetními časy, a to až o ~67 %.

2.7.3 Doporučené použití symetrické neúplné relaxace

Na základě experimentů provedených v rámci studie [A6] lze konstatovat, že ve zjednodušených CFD modelech je možné použít symetrickou neúplnou relaxaci vcelku úspěšně při řešení rovnic jednotlivých složek rychlosti proudění i energie. Podmínkami však jsou aplikace spolu se stabilizovanou metodou bi-sdružených gradientů s $l \leq 3$ a vhodná volba relaxačních parametrů, přičemž platí, že užitím $\omega_{\mathrm{F}}, \omega_{\mathrm{R}} \neq 1{,}0$ lze dosáhnout výraznějšího zkrácení výpočetního času než prostou relaxací v Gaussově-Seidelově tvaru (tedy s $\omega_{\mathrm{F}} = \omega_{\mathrm{R}} = 1{,}0$). Naopak v případě řešení tlakové korekce, kde bylo nejlepších výsledků dosaženo pomocí klasické metody sdružených gradientů, se použití zmíněného předpodmínění ukázalo při $\omega_{\mathrm{F}} = \omega_{\mathrm{R}} = 1{,}0$ jako méně vhodné, resp. s $\omega_{\mathrm{F}}, \omega_{\mathrm{R}} \neq 1{,}0$ jako zcela nevhodné.

Velkou výhodou symetrické neúplné relaxace je skutečnost, že matici, která je používána během vlastního předpodmíněného iteračního výpočtu, lze získat bez ohledu na vlastnosti původní matice reprezentující řešený lineární systém. V porovnání s efektivním neúplným LU rozkladem, kde toto rozhodně neplatí, však vede symetrická neúplná relaxace k průměrnémů nárůstu výpočetních časů alespoň o ~23 %. Jako snadno automatizovatelný kompromis tedy lze doporučit implementaci obou metod s tím, že jako výchozí bude použit neúplný LU rozklad, zatímco „záložní řešení“ v podobě symetrické

neúplné relaxace začne numerický řešit používat až v případě výskytu divergence či jiných numerických potíží.

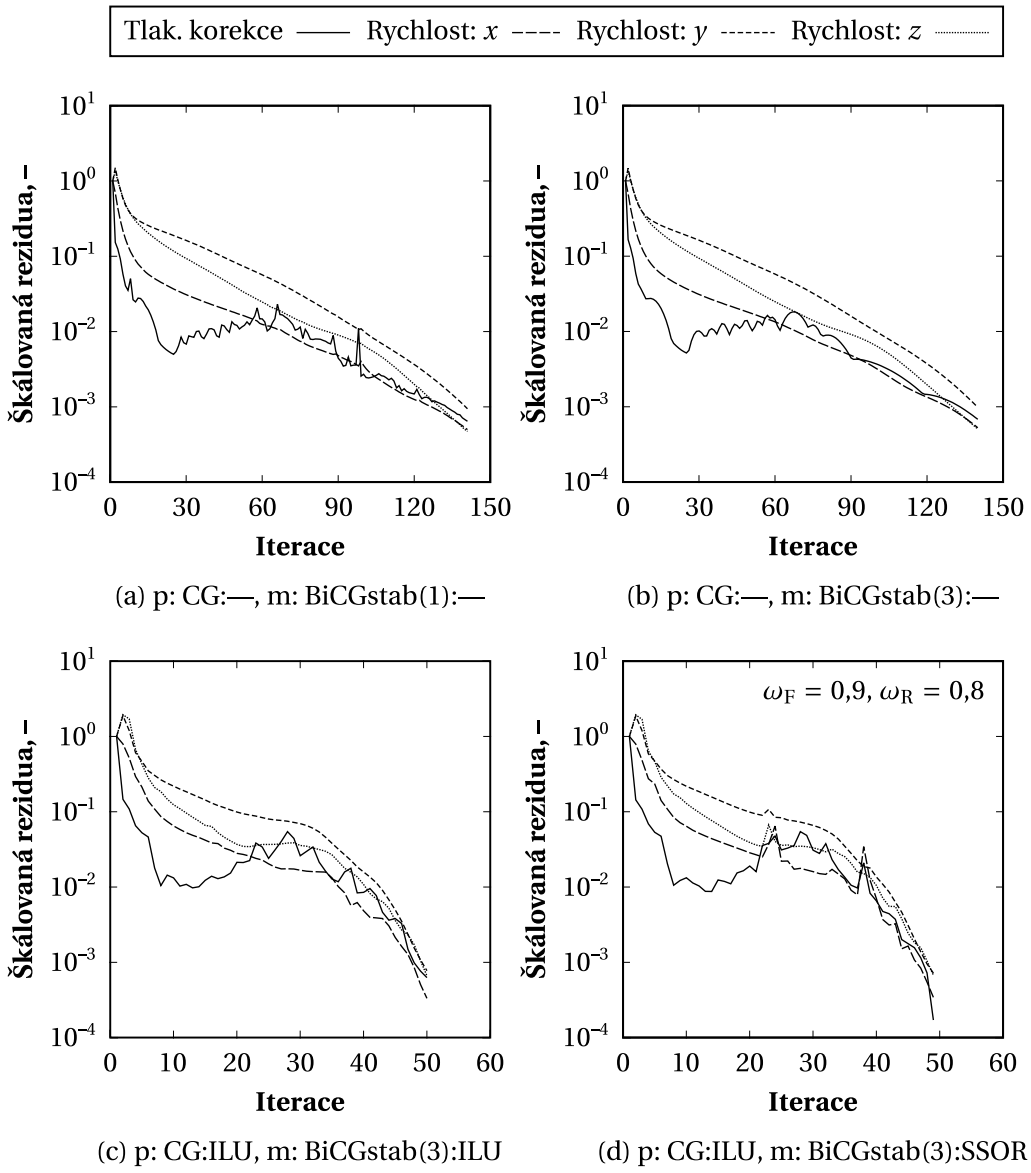
2.7.4 Hladkost konvergence

Uvážíme-li popsané použití různých metod předpodmínění, nabízí se otázka, jaký bude toto mít vliv na konvergenci. S ohledem na data obdržená během testovacích výpočtů se obecně dá říci, že pokud určitá kombinace předpodmíněných numerických metod poskytla zkonvergované řešení, odpovídající výpočetní čas byl kratší než v případě stejné kombinace bez předpodmínění, nicméně průběhy jednotlivých škálovaných reziduí nebyly nutně hladší. Stejně tak platí, že u předpodmíněných metod docházelo k častějšímu výskytu numerických potíží v podobě divergence.

Nejprve se podrobněji podívejme na kombinaci metody sdružených gradientů (pro tlakovou korekci) a stabilizované metody bi-sdružených gradientů s $l \leq 3$ (pro jednotlivé složky rychlosti proudění). Bez použití předpodmínění jsou v takovém případě průběhy škálovaných reziduí u jednotlivých složek rychlosti zpravidla hladké, přičemž hladkost průběhu škálovaného rezidua u tlakové korekce roste s hodnotou parametru l (viz obrázky 2.19a a 2.19b). Na rychlost konvergence (ve smyslu potřebného počtu „velkých“ iterací CFD řešiče) však toto v zásadě nemá vliv. Na obrázcích 2.19c a 2.19d pak je vidět, že při použití neúplného LU rozkladu je konvergence marginálně hladší než při použití symetrické neúplné relaxace a – jak bude zřejmé z obrázku 2.23a na straně 45 – výrazně hladší než u předpodmínění typu Jacobi.

V případě, že zjednodušený CFD model zahrnuje i přenos energie (taktéž řešený pomocí stabilizované metody bi-sdružených gradientů), je situace podobná. Průběhy škálovaných reziduí u jednotlivých složek rychlosti jsou vesměs opět hladké, zatímco hladkost průběhu škálovaného rezidua rovnice energie roste s rostoucím l (viz obrázky 2.20a a 2.20b). Ani zde přitom volba hodnoty l příliš neovlivní počet „velkých“ iterací CFD řešiče nutných k dosažení zkonvergovaného řešení. Obrázky 2.20c až 2.20f naopak ukazují, že pokud aplikujeme předpodmínění pomocí neúplného LU rozkladu, resp. symetrické neúplné relaxace, má již volba parametru l nezanedbatelný vliv i na potřebný počet iterací. Dále je také zřejmé, že v simulacích s přenosem energie poskytuje marginálně hladší konvergenci symetrická neúplná relaxace, byť za cenu možného většího počtu iterací CFD řešiče.

Pro srovnání uveďme grafy škálovaných reziduí získané pomocí jiných – nepředpodmíněných – metod, které se k použití ve zjednodušených CFD výpočtech příliš nehodí, ačkoliv se obecně mohou vyznačovat hladší konvergencí. Konkrétně jde o zobecněnou metodu minimálních reziduí, metodu sdružených gradientů aplikovanou na systém normálních rovnic s minimalizací normy rezidua a metodu kvazi-minimálních reziduí („Quasi-Minimal Residual Method“, QMR) [44]. Průběhy získané pomocí zobecněné metody minimálních reziduí (obrázek 2.21a) jsou dle očekávání hladké, nicméně průměrný výpočetní čas, který byl potřeba k získání zkonvergovaného řešení ustálené úlohy (176,97 s) je vysoce nad hodnotami pozorovanými u dříve testovaných kombinací numerických metod zahrnujících metodu sdružených gradientů pro řešení tlakové ko-



(a) p: CG:—, m: BiCGstab(1):—

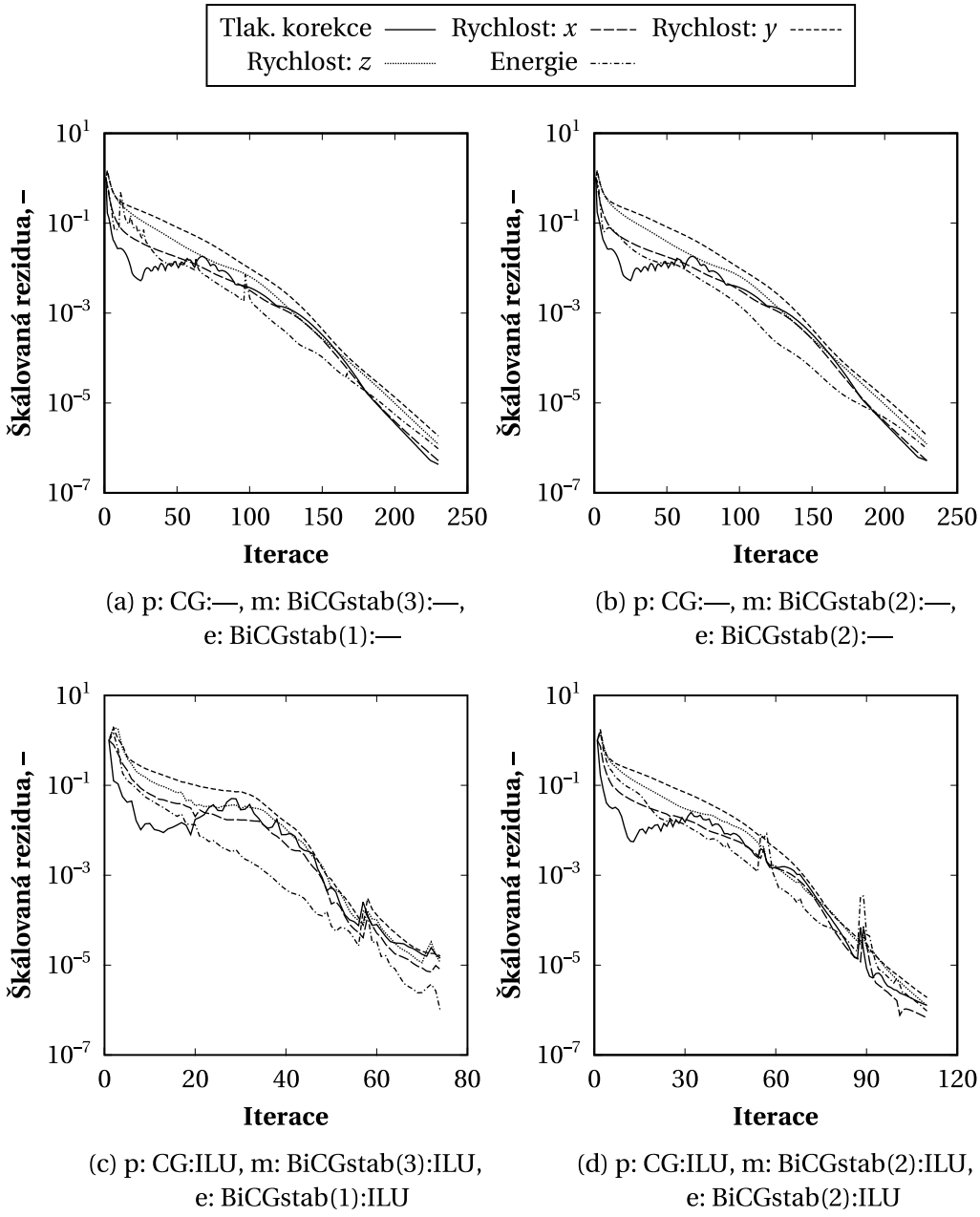
(b) p: CG:—, m: BiCGstab(3):—

(c) p: CG:ILU, m: BiCGstab(3):ILU

(d) p: CG:ILU, m: BiCGstab(3):SSOR

Obrázek 2.19. Průběhy škálovaných reziduí získané užitím různých kombinací numerických metod a způsobů předpodmínění na výpočetní síti s ~6 tis. buňkami; „p“ značí tlakovou korekci a „m“ rovnice pro jednotlivé složky rychlosti. Výpočty byly prováděny pomocí testovacího kódu používaného ve studii [A6].

rekce a stabilizovanou metodu bi-sdružených gradientů pro jednotlivé složky rychlosti proudění (od 2,56 s výše). Metoda sdružených gradientů aplikovaná na systém normálních rovnic s minimalizací normy rezidua (obrázek 2.21b) je vzhledem k potřebnému výpočetnímu času na první pohled zcela nepřijatelná. U metody kvazi-minimálních



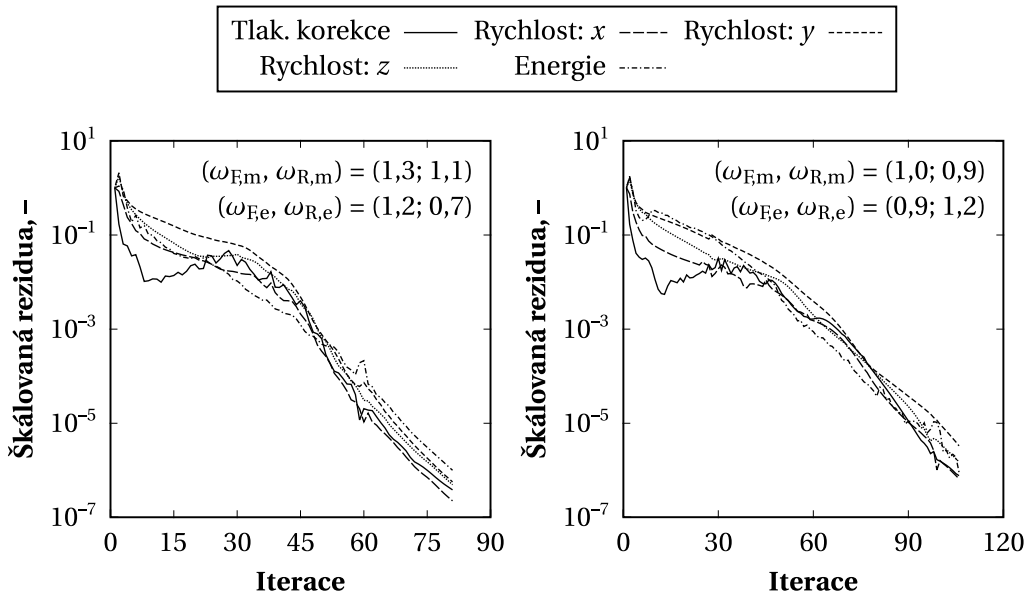
(a) p: CG:—, m: BiCGstab(3):—, e: BiCGstab(1):—

(b) p: CG:—, m: BiCGstab(2):—, e: BiCGstab(2):—

(c) p: CG:ILU, m: BiCGstab(3):ILU, e: BiCGstab(1):ILU

(d) p: CG:ILU, m: BiCGstab(2):ILU, e: BiCGstab(2):ILU

Obrázek 2.20. Průběhy škálovaných reziduí získané užitím různých kombinací numerických metod a způsobů předpodmínění na výpočetní síti odpovídající obrázku 2.19; „p“ značí tlakovou korekci, „m“ rovnice pro jednotlivé složky rychlosti a „e“ energii. Výpočty byly prováděny pomocí testovacího kódu používaného ve studii [A6].



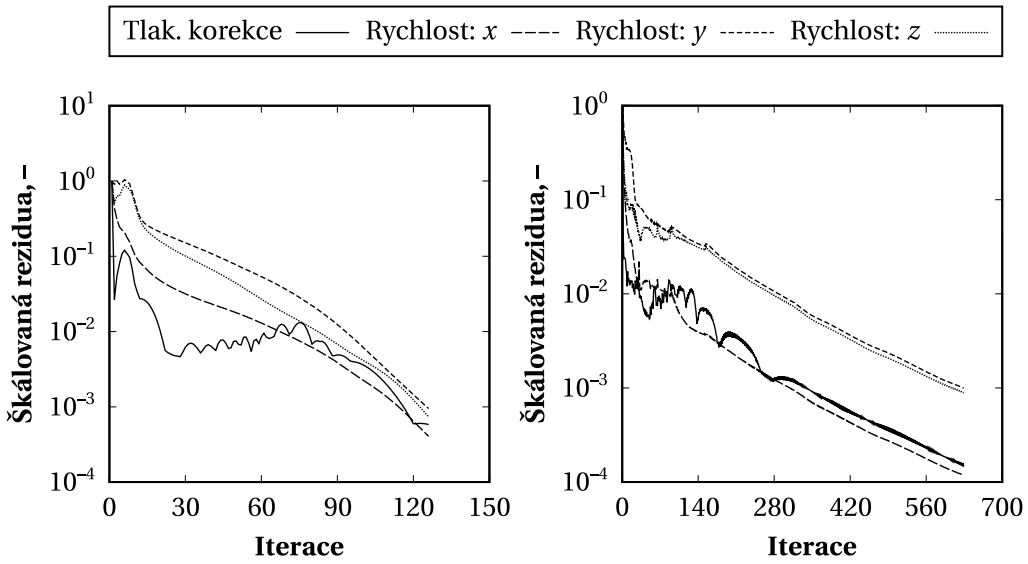
(e) p: CG:ILU, m: BiCGstab(3):SSOR, e: BiCGstab(1):SSOR

(f) p: CG:ILU, m: BiCGstab(2):SSOR, e: BiCGstab(2):SSOR

Obrázek 2.20 (pokračování). Průběhy škálovaných reziduí získané užitím různých kombinací numerických metod a způsobů předpodmínění na výpočetní síti odpovídající obrázku 2.19; „p“ značí tlakovou korekci, „m“ rovnice pro jednotlivé složky rychlosti a „e“ energii. Výpočty byly prováděny pomocí testovacího kódu používaného ve studii [A6].

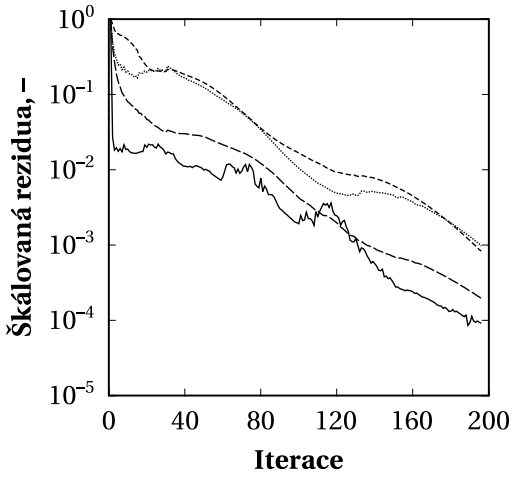
reziduí (obrázek 2.21c) pak byl výpočetní čas relativně blízký hodnotám pozorovaným s CG/BiCGstab(l). Průběhy škálovaných reziduí sice nejsou zcela hladké, nicméně vcelku stabilně klesají a i u větších výpočetních sítí tedy nejspíš nebude potřeba výrazné snížení relaxačního faktoru v metodě SIMPLEC (resp. relaxačních faktorů v metodě SIMPLE). Mohlo by se tedy zdát, že zmíněná metoda bude dobře použitelná. Pokud se však podíváme na průběhy škálovaných reziduí získané s různě předpodmíněnými verzemi této metody (viz obrázek 2.22), je zřejmé, že tomu tak není. Nepředpodmíněná verze QMR se totiž i se stávající relativně malou výpočetní sítí chovala zdaleka nejlépe a nelze tedy příliš očekávat, že u větších výpočetních sítí tomu bude jinak. Kromě toho je zde nutné podotknout, že při aplikaci neúplného LU rozkladu a neúplného Choleského rozkladu byl výpočet velmi náchylný k divergenci a k získání zkonvergovaného řešení bylo nutné výrazně snížit relaxační faktor použitý v metodě SIMPLEC (na dvě třetiny až polovinu běžné hodnoty).

Obrázek 2.23 pak uvádí průběhy škálovaných reziduí pro osvědčenou kombinaci metody sdružených gradientů a stabilizované metody bi-sdružených gradientů s $l = 3$, které byly předpodmíněny pomocí metody „Jacobi“, resp. neúplného Choleského rozkladu. Z průběhů je vidět, že v porovnání s dříve diskutovaným neúplným LU rozkladem



(a) GMRES, $t_{\text{avg}} = 176{,}97$ s

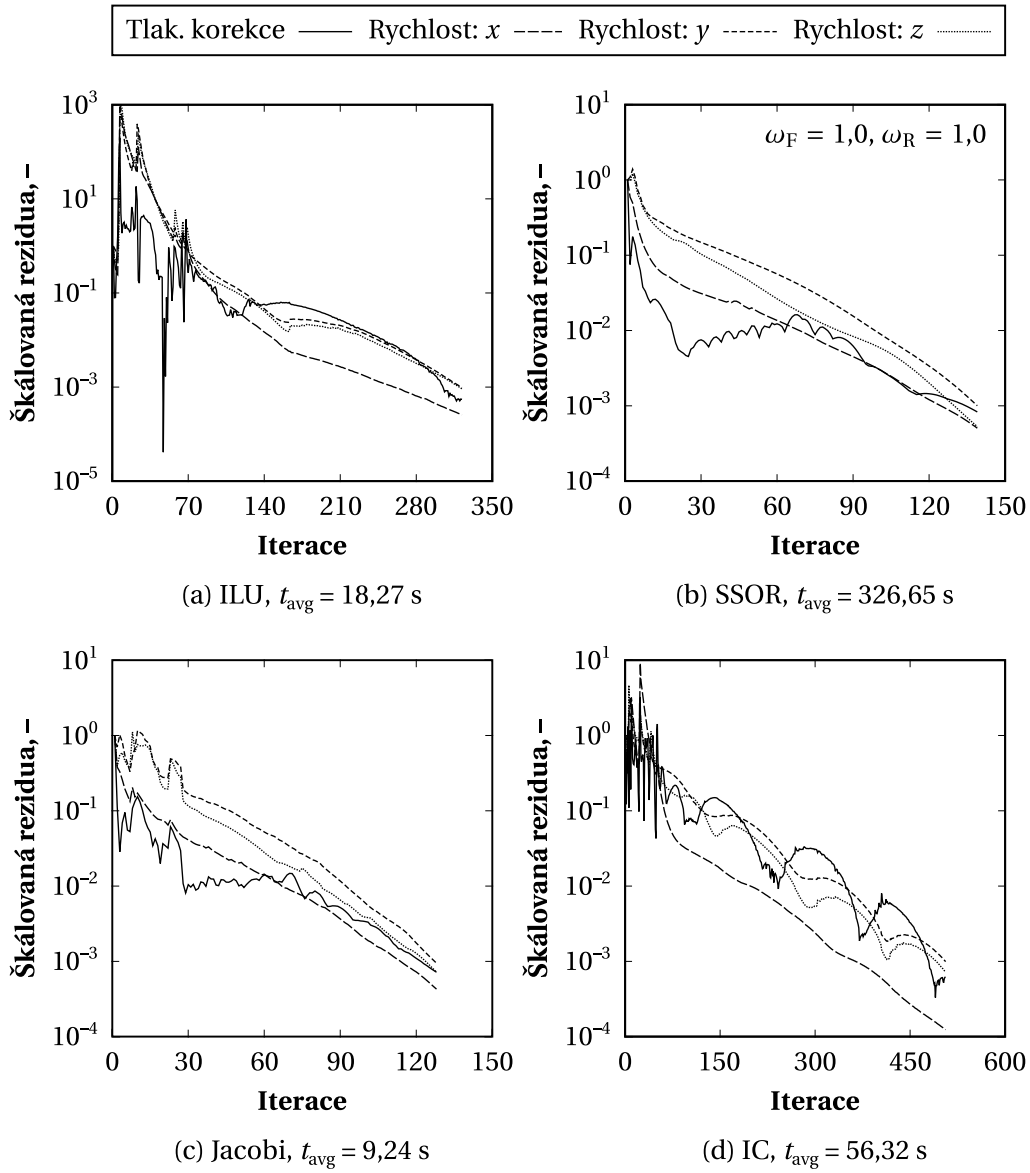
(b) CGNR, $t_{\text{avg}} = 1\,233{,}89$ s



(c) QMR, $t_{\text{avg}} = 5{,}34$ s

Obrázek 2.21. Průběhy škálovaných reziduí získané při řešení tlakové korekce i rovnic pro jednotlivé složky rychlosti pomocí nepředpodmíněných metod GMRES, CGNR a QMR; t_{avg} značí průměrný výpočetní čas nutný k získání zkonvergovaného řešení ustálené úlohy. Veškeré výpočty byly prováděny se stejným nastavením CFD úlohy jako u obrázku 2.19.

či symetrickou neúplnou relaxací se zde použité způsoby předpodmínění vyznačují horší konvergencí. Pro orientační porovnání s výše uvedenými výsledky jsou opět zmíněny průměrné výpočetní časy nutné k získání zkonvergovaného řešení ústálené úlohy. I v tomto případě bylo při použití neúplného Choleského rozkladu nutné snížit relaxační faktor



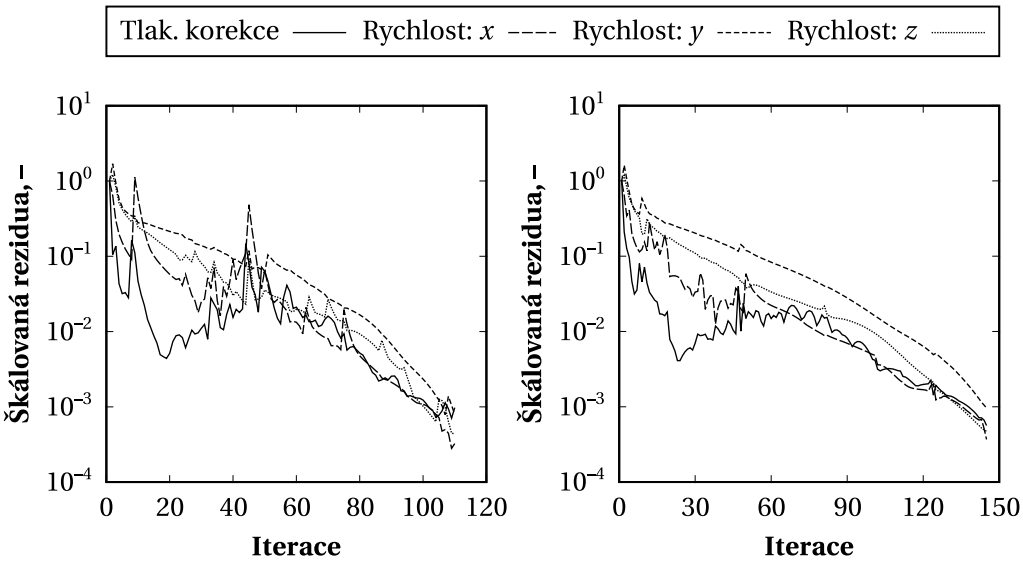
(a) ILU, t_{avg} = 18,27 s

(b) SSOR, t_{avg} = 326,65 s

(c) Jacobi, t_{avg} = 9,24 s

(d) IC, t_{avg} = 56,32 s

Obrázek 2.22. Průběhy škálovaných reziduí získané při řešení tlakové korekce i rovnic pro jednotlivé složky rychlosti pomocí QMR, přičemž řešiče byly předpodmíněny různými (avšak pro všechny typy rovnic vždy stejnými) způsoby; t_{avg} značí průměrný výpočetní čas nutný k získání zkonvergovaného řešení ustálené úlohy. Odhlédneme-li od případného snížení relaxačního faktoru v metodě SIMPLEC, byly veškeré výpočty prováděny se stejným nastavením CFD úlohy jako u obrázku 2.19.



(a) Jacobi, $t_{\text{avg}} = 4{,}25$ s (b) IC, $t_{\text{avg}} = 26{,}39$ s

Obrázek 2.23. Průběhy škálovaných reziduí získané při řešení tlakové korekce pomocí CG a rovnic pro jednotlivé složky rychlosti pomocí BiCGstab(3), přičemž řešiče byly předpodmíněny různými (avšak pro všechny typy rovnic vždy stejnými) způsoby; t_{avg} značí průměrný výpočetní čas nutný k získání zkonvergovaného řešení ustálené úlohy. Odhlédneme-li od případného snížení relaxačního faktoru v metodě SIMPLEC, byly veškeré výpočty prováděny se stejným nastavením CFD úlohy jako u obrázku 2.19.

použitý v metodě SIMPLEC zhruba na polovinu běžné hodnoty, jelikož jinak výpočty ihned končily divergencí. Tyto skutečnosti jednoznačně potvrzují sníženou vhodnost, resp. úplnou nevhodnost zmíněných způsobů předpodmímnění.

Co se týče předpodmínění pomocí neúplného LU rozkladu s tolerancí, tato metoda vedla i na relativně malé testovací výpočetní síti k téměř okamžité divergenci bez ohledu na případné změny relaxačního faktoru v metodě SIMPLEC. Odtud plyne, že neúplný LU rozklad s tolerancí je pro potřeby zjednodušených CFD výpočtů zcela nevhodný, neboť u větších sítí lze zpravidla očekávat výrazně větší numerické potíže.

2.8 Pořadí proměnných

Vhodným způsobem indexování proměnných („ordering“) lze dosáhnout příhodnějšího rozložení prvků v řešených maticích. Toto pak má vliv nejen na chování numerického řešiče (viz například studie [45] týkající se metody sdružených gradientů nebo [46] disku-

tující zobecněnou metodu minimálních reziduí), ale také na volbu předpodmínění [38] a případnou paralelní implementaci† CFD řešiče [48].

Opomeneme-li paralelizaci, kterou, jak bude vysvětleno v kapitole 2.9, nemá u zjednodušených CFD modelů příliš smysl uvažovat, je v komerčních CFD softwarech často využíváno např. indexování podle Cuthillové a McKeeho [49], resp. jeho obrácená [50] či bloková [51] varianta. Alternativní způsoby indexování však zatím v aplikaci popisované dále (kap. 2.10) nebyly z časových důvodů implementovány a výsledky veškerých provedených testovacích výpočtů tudíž odpovídají přirozenému pořadí proměnných. Jelikož ale může být vhodné indexování cestou ke zvýšení robustnosti modelu, zvláště pak při předpodmínění pomocí některého z neúplných rozkladů [31, s. 322–323], je plánováno přidat patřičné funkcionality do již existujícího automatického generátoru výpočetní sítě. Následně pak bude zkoumáno, který konkrétní způsob indexování by byl v kombinaci se zjednodušenou kvádrovou výpočetní sítí nejvhodnější.

2.9 Aspekty počítačové implementace

Zjednodušené CFD modely se od těch standardních do určité míry liší a při jejich počítačové implementaci je proto nutné zvážit primárně tyto faktory:

- provádění výpočtů na jednom, resp. více jádrech CPU,
- inteligentní inicializace řešení,
- inteligentní řízení limitů v maticových řešičích,
- způsob práce s daty v paměti a
- použití specializovaných knihoven pro lineární algebru.

V následujícím textu bude každý z faktorů výše rozebrán podrobněji.

2.9.1 Jednojádrový vs. vícejádrový výpočet

Provádění potřebných maticových výpočtů na více jádrech procesoru je v případě standardních CFD modelů zcela běžné a těžko si představíme komerční CFD software, který by toto nedělal. U zjednodušených CFD modelů je však zcela namístě se zamyslet, zda takový přístup nebude spíše kontraproduktivní.

K tomu, aby vícejádrový výpočet mohl fungovat, je zpravidla potřeba nejen rozšířit stávající výpočetní síť o nové buňky, které posléze realizují propojení jednotlivých výpočetních domén, ale hlavně na úrovni operačního systému vytvořit nové procesy‡ a zajistit

†U paralelní implementace je nutné zvážit celou škálu souvisejících faktorů – například to, zda jsou data modelu ve sdíleném adresním prostoru nebo v distribuovaném adresním prostoru (a jaký způsob komunikace mezi výpočetními uzly je v takovém případě použit)[47] a podobně.

‡Jeden proces (tzv. „master“) vždy celý paralelní výpočet řídí, zbylé procesy („slave“) jsou výkonné.

co možná nejefektivnější komunikaci mezi nimi. Toto je však spojeno s nezanedbatelným zdržením v důsledku latence a propustnosti použitého komunikačního rozhraní („Message Passing Interface“, MPI) a vlastní komunikační vrstvy (např. sdílený adresní prostor v RAM, síťový standard InfiniBand [52] atp.) a skutečnosti, že iterační výpočet musí z podstaty věci probíhat do určité míry synchronně. Nárůst výpočetního výkonu s počtem využitých jader CPU proto není ani zdaleka lineární.

Při výpočtu pouze na jednom jádře CPU naopak není potřeba jakkoliv rozšiřovat výpočetní síť, vytvářet a spravovat dodatečné procesy, předávat data či čekat na dokončení dílčích výpočtů na jiných jádrech CPU. Vezmeme-li v úvahu relativné malé počty buněk ve zjednodušených výpočetních sítích, lze v kombinaci s dnes běžně dostupnými kapacitami velmi rychlých Level 1–3 vyrovnávacích pamětí v moderních procesorech mnohdy celou zjednodušenou CFD úlohu spočítat na jednom jádře rychleji. Než abychom tedy dosáhli méně významného zrychlení výpočtu paralelizací maticových operací, je při vyhodnocování větší sady úloh, optimalizaci apod. (tedy v situacích, pro které jsou zjednodušené CFD úlohy určeny) mnohem výhodnější paralelizovat provádění kompletních CFD výpočtů v rámci sady či náležitě přizpůsobeného optimalizačního procesu.

2.9.2 Inteligentní inicializace řešení

Ve standardních CFD modelech se typicky používá jeden z následujících způsobů nastavení počátečních hodnot proměnných:

- inicializace dle okrajových podmínek,
- hybridní inicializace, příp.
- inicializace nulami.

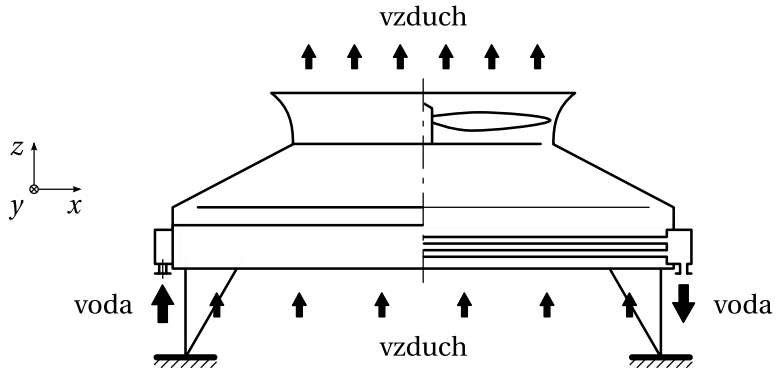
Začněme poslední možností, která je sice nejjednodušší, nicméně obvykle také nejméně vhodná. Inicializací nulami se totiž rozumí, že hodnoty dotčených veličin (rychlostí proudění u, v a w, tlakových korekcí atd.) jsou na začátku výpočtu ve všech buňkách sítě prostě nulové. Takový postup však u zjednodušených modelů mnohdy povede bez výraznější relaxace k divergenci.

Při inicializaci dle okrajových podmínek se naopak v celé doméně nastaví hodnoty veličin dle parametrů na vstupu a výstupu. V obvyklých úlohách proudění tekutin v procesních a energetických zařízeních to znamená, že tlak je nastaven podle výstupní zóny (příp. zón; „pressure outlet“), zatímco rychlosti proudění jsou odhadnuty pomocí informací ze vstupní zóny (příp. zón; „velocity inlet“ u nestlačitelného proudění, resp. „mass flow inlet“ v případě stlačitelného proudění).

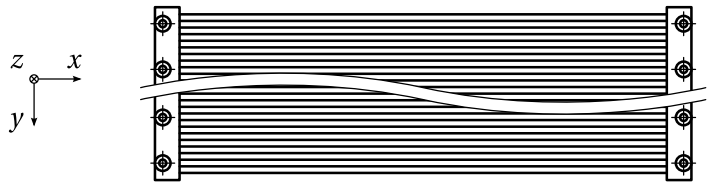
Poslední možnost, tedy hybridní inicializace, je v podstatě inicializací dvojí, neboť CFD řešič nejprve provede obvyklou inicializaci (např. dle okrajových podmínek), kterou pak na značně zhrubené síti následují jednotky až desítky iterací standardního výpočtu. Výsledky jsou posléze extrapolovány zpět na původní síť, čímž lze oproti zbylým dvěma postupům získat výrazně lepší počáteční odhad.

Zjednodušené CFD modely jsou náchylnější k výskytu numerických potíží, ale navzdory tomu by měly být co možná nejvíce autonomní. Z popisu výše proto vyplývá, že užití inicializace nulami je nevhodné, zatímco hybridní inicializace je vzhledem k již relativně hrubé výpočetní síti *de facto* identická se samotným zjednodušeným CFD výpočtem. To však neznamená, že jedinou zbylou možností je inicializace dle okrajových podmínek. Zatímco standardní CFD modely jsou koncipovány jako co možná nejobecnější simulační nástroje, u zjednodušených CFD modelů lze naopak předpokládat, že o modelovaných zařízeních či distribučních systémech v nich máme k dispozici nemálo informací. V konečném důsledku tudíž lze vcelku snadno a s minimem programátorského úsilí inicializovat řešení inteligentněji.

Jako příklad si představme distribuční systém vzduchového chladiče z obrázku 2.24. Hrdla pro přívod a odvod chlazené vody jsou k distributoru a kolektoru připojena zespod. Pokud bychom tedy použili inicializaci dle okrajových podmínek, byl by počáteční odhad tlaku sice přijatelný, avšak složky rychlosti proudění by byly takřka v celé výpočetní doméně nesprávné. V obrázku totiž vidíme, že pro vstupní a výstupní hrdla platí $u = v = 0$ a $w \neq 0$, zatímco v distributoru a kolektoru jsou všechny tři složky rychlosti proudění



(a) vzduchový chladič



(b) pohled zespodu na trubkový svazek s prostřídaným uspořádáním trubek

Obrázek 2.24. Vzduchový chladič se svazkem s prostřídaným uspořádáním trubek a hrdly připojenými ke spodním plochám distributoru a kolektoru

obecně nenulové a v trubkovém svazku je $u \neq 0$, $v \approx 0$ a $w \approx 0$. Mnohem lepší by proto bylo udělat třeba následující:

- v trubkách svazku dopočíst průměrnou rychlost proudění ve směru osy x a tuto zde použít jako počáteční odhad u spolu s $v = w = 0$;
- v hlavních kanálech dopočíst průměrnou rychlost proudění celým průřezem ve směru osy x a tuto použít jako počáteční odhad u, analogicky dopočíst průměrnou rychlost proudění ve směru osy z a její kladnou hodnotu použít v distributoru a zápornou hodnotu v kolektoru jako odhad w, zatímco odhad zbylé složky rychlosti proudění, v, by zde zůstal nulový.

S takovou „inteligentní“ inicializací, která neodpovídá striktně okrajovým podmínkám, byť je z jejich hodnot částečně odvozena, by bylo dosaženo lepší konvergence a řešení úlohy by tak potažmo bylo získáno dříve. Podobně by pak šlo postupovat i u jiných geometrií či uspořádání distribučních systémů a zařízení. Jedinou potenciální nevýhodou zde je nutnost triviálním způsobem zasáhnout do zdrojového kódu simulační aplikace, nicméně implementace zjednodušených CFD modelů se bez takových zásahů obvykle stejně neobejde.

2.9.3 Inteligentní řízení interních limitů v maticových řešičích

Interní maticové řešiče typicky pracují se čtyřmi interními limity. Konkrétně jde o maximální povolený počet iterací, $i_{\max}$, a relativní, ϵ_{r}, absolutní, ϵ_{a}, a divergenční, ϵ_{d}, toleranci pro škálované reziduum, r. Pomocí rezidua r_1 získaného v první iteraci pak v každé další iteraci $i \leq i_{\max}$ kontrolujeme, zda bylo dosaženo konvergence, tj. zda platí

$$\begin{aligned} &r_i < \max\{r_1\epsilon_{\mathrm{r}}, \epsilon_{\mathrm{a}}\} \quad \text{a zároveň} \\ &r_i < r_1\epsilon_{\mathrm{d}}. \end{aligned} \tag{2.25}$$

Běžně je přitom numerickým řešičem vraceno řešení – v kontrastu s výjimkou či jiným způsobem indikace chyby – pouze při splnění obou uvedených podmínek. Kromě toho je vhodné pomocí rezidua kontrolovat, zda není aktuální řešení degenerováno natolik, že se z něj stalo „nečíslo“ („Not a Number“, `NaN`). Vzhledem k definici `NaN` to lze provést následovně:

```
if (r != r) {
  // r == NaN
  ...
}
```

Tento způsob kontroly je navíc v mnoha jazycích (např. ve standardním C či C++) jediným možným. Některé moderní jazyky (kupříkladu Java či C/C++ od revize ISO/IEC 9899:1999 [53]) sice mají pro zmíněný účel k dispozici speciální metody, ovšem tyto jsou interně implementovány identickým postupem.

Obvyklou praxí je nastavit interní limity maticových řešičů na začátku CFD výpočtu a dále je neměnit. Iterační limit bývá $i_{\max} = 10\,000$, zatímco hodnoty ostatních limitů je potřeba nastavit s ohledem na řešenou úlohu a použitý datový typ proměnných. V dnešní době již při běžných výpočtech nebývá nutné uchylovat se za účelem úspory paměti k užití datového typu float (v některých jazycích označován jako single). Používá se tedy převážně datový typ double poskytující vyšší přesnost, u kterého se hodnoty interních limitů obvykle pohybují kolem

$$\begin{aligned} \epsilon_{\mathrm{r}} &= 10^{-5}, \\ \epsilon_{\mathrm{a}} &= 10^{-50}, \\ \epsilon_{\mathrm{d}} &= 10^{5}. \end{aligned} \tag{2.26}$$

Nic nám však nebrání hodnoty limitů v maticových řešičích měnit podle toho, jak se postupně snižují rezidua CFD řešiče. Ačkoliv jsou tedy ve vyvinutém softwaru popsaném dále v kapitole 2.10 používány konstantní absolutní a divergenční tolerance, iterační limit a relativní tolerance jsou aktualizovány před každým jednotlivým maticovým výpočtem.

Iterační limit je měněn pouze v závislosti na aktuální „velké“ iteraci CFD řešiče, I. Na začátku CFD výpočtu, kdy jsou jednotlivá skalární pole daleko od konečného řešení a vyšší počet iterací maticového řešiče může proto být spíše kontraproduktivní (nemá smysl řešit příliš přesně zadání s nepřesnými vstupy), je tento limit nižší. Jeho hodnota potom v průběhu prvních 20 „velkých“ iterací postupně roste až na zmíněných 10 000:

$$i_{\max} = \min\left\{500I, 10\,000\right\}. \tag{2.27}$$

Analogicky je postupně snižována hodnota relativní tolerance, která závisí na uživatelem zadané relativní toleranci pro tu kterou veličinu ϕ, $\epsilon_{\mathrm{r},\phi}$, a hodnotě odpovídajícího neškálovaného rezidua před započetím procesu řešení lineárního systému, R_ϕ:

$$\epsilon_{\mathrm{r}} = \begin{cases} 5\hat{\epsilon}\,(11 - I), & I \leq 10 \\ 3\hat{\epsilon}/(I - 10), & I \in [11; 25] \\ \hat{\epsilon}/5, & I > 25 \end{cases} \qquad \text{kde} \quad \hat{\epsilon} = \min\left\{\epsilon_{\mathrm{r},\phi}, R_\phi\right\} \tag{2.28}$$

Tím je zajištěno, aby nedocházelo k nadměrně přesnému řešení soustav lineárních rovnic, což kromě zjevného prodloužení výpočetního času může vést i k numerickým potížím. Dále je vhodné upravit kód maticového řešiče tak, aby při překročení interního iteračního limitu nevracel chybu, jak bývá obvyklé, ale aktuální vektor řešení. Je totiž výrazně lepší pokračovat v CFD řešiči s mezivýsledkem, byť nižší kvality, než aby celý proces řešení havaroval.

U některých maticových řešičů – typicky těch, které jsou odvozeny z metody bi-sdružených gradientů – může nastat tzv. rozpad („solver breakdown“) [54], tedy situace, kdy by provedení dodatečných iterací vedlo pouze k další degeneraci řešení. Jsou sice k dispozici metody, které se snaží rozpadům předcházet (např. metoda kvazi-minimálních reziduí

či různě ošetřené verze metody bi-sdružených gradientů [55]), nicméně ani u nich nelze výskyt rozpadu zcela vyloučit. V takovém případě je doporučeno zmíněné čtyři interní limity rozšířit o limit pátý, ϵ_b, nutný k detekci patřičné situace. Vybrané interní hodnoty jsou pak v průběhu maticového výpočtu porovnávány namísto s „exaktní" nulou právě s ϵ_b. Jsou-li menší, značí to hrozící rozpad řešiče a je tedy lepší iterační proces zastavit. V softwaru z kapitoly 2.10 se za tímto účelem používá limit $\epsilon_b = 10^{-30}$ a iterační proces matricového řešiče je v případě jeho dosažení opět ukončen vrácením aktuálního řešení namísto chyby.

2.9.4 Práce s daty v paměti

Způsob práce s daty v paměti není u zjednodušených CFD modelů vlivem výrazně menších výpočetních sítí natolik kritický, nicméně se stále vyplatí nepřistupovat k této problematice bez rozmyslu. Není totiž ani tak podstatné, kolik která datová struktura zabere v RAM místa, jako to, co bude rychlejší z pohledu přístupu.

Jednoznačně zde platí, že všechny skalární proměnné a další pomocné objekty by měly být pokud možno uloženy ve statických polích (statických ve smyslu „fixní velikosti", nikoliv ve smyslu klíčového slova static). V javové syntaxi by tedy taková proměnná byla deklarována následovně:

```
double[] arr = new double[size];
```

Potřebujeme-li z nějakého důvodu použít datovou strukturu s předem neznámým počtem prvků, v mnoha moderních jazycích se přímo nabízí užití kolekcí. Typickými aplikacemi jsou algoritmus vytvářející vlastní výpočetní síť nebo práce se speciálními zónami (stěny či buňky ve vstupní a výstupní oblasti apod.), kde je nutné co nejrychleji mezi sebou převádět pomocné či lokální zónové indexy a globální indexy objektů. Musíme ale mít na paměti, že v každém programovacím jazyce existuje mnoho různých typů kolekcí ne všechny z nich podporují všechny běžné operace (přidání prvku, odebrání prvku, vrácení prvku dle indexu, hledání indexu dle prvku atd.).

Kolekce kromě toho mohou pracovat odlišnými způsoby. Pokud bychom se opět měli držet Javy, jinak fungují různé implementace seznamu (List), fronty (Queue), mapy (Map) či množiny (Set). Některé kolekce nadto nemusí poskytovat tzv. „thread-safe" implementaci, což může být potenciální přičinou potíží i v případě, že bychom simulační aplikaci cílili na výpočet na jednom jádře CPU (nelze zaměňovat s během v jednom vlákně!). Vždy je proto potřeba nejprve zvážit, jaké konkrétní operace a vlastnosti budou u té které kolekce vyžadovány (příp. ideálně také v jakém zastoupení) a až podle toho zvolit vhodný typ kolekce.

V implementacích zjednodušených CFD modelů pravděpodobně většina případů vyžadujících užití kolekce – na rozdíl od statických polí – zahrnuje jednorázové (byť třeba postupné) naplnění dané „referenční" kolekce prvky a její pozdější prosté procházení iterátorem, vracení prvků dle indexů, či zjišťování, zda ten který prvek je v kolekci obsažen. Na základě různých benchmarků javových kolekcí (např. testu [56], který je sice starší, ale dle zkoušek provedených autorem této práce stále platným i v novějších verzích

běhového prostředí Javy) se k danému účelu nejlépe hodí ArrayList. Tento se u zmíněných operací vyznačuje komplexitou $O(1)$ u přidání prvku na konec kolekce (add()), vrácením prvku dle indexu (get()) a iterování skrze kolekci (next()), resp. $O(n)$ v případě zjištění, zda je určitý prvek v kolekci obsažen (contains()).

Pokud bychom potřebovali frontu, ať už FIFO („first-in, first-out“) nebo LIFO („last-in, first-out“), nebo nám pro patřičný účel stačila namísto seznamu fronta, je nejvhodnější použít ArrayDeque. Na takové kolekci nelze ze zjevných důvodů provádět některé operace běžně dostupné u seznamů (např. vrácení prvku dle indexu) a samotná datová struktura vyžaduje oproti ArrayListu více místa v RAM vlivem jiné politiky změn velikosti při přidávání a odebírání prvků. Toto je však vykoupeno vyšší celkovou rychlostí přístupů v případě, že je nutné přidávat prvky nejen na konec kolekce, ale i na její začátek (ArrayList: $O(n)$, ArrayDeque: $O(1)$).

Ve dříve zmíněných speciálních případech, kdy musíme převádět mezi pomocnými či lokálními zónovými indexy a globálními indexy objektů, pak je v Javě nejvhodnějším typem kolekce HashMap. Jde o datovou strukturu uchovávající páry (klíč, hodnota), která se vyznačuje komplexitou vyhledání prvku (get()) $O(1)$. Velkou výhodou zde je skutečnost, že indexy objektů ve výpočetní síti jsou standardně unikátní celá čísla a mohou proto přímo sloužit jako klíče nejen pro identifikaci jednotlivých prvků v mapě, ale také v hashovací tabulce, kterou HashMap interně využívá.

Zaměřme se nyní na data, která musí být z důvodu prováděných operací (tj. typové kompatibility s volanými metodami) uložena jako matice či vektory. Je zřejmé, že vektory se složkami rychlosti proudění, tlaky atd. má smysl deklarovat jako husté. Pokud bychom pro maticové výpočty používali například knihovnu Parallel Colt [57] (kterou využívá software z kapitoly 2.10) a datový typ double, mohla by deklarace patřičného vektoru vypadat takto[†]:

```
DoubleMatrix1D vec = new DenseDoubleMatrix1D(size);
```

Naopak dvourozměrné matice reprezentující soustavy linearizovaných rovnic je žádoucí deklarovat jako řídké, neboť v nich je mnoho prvků striktně nulových a ušetří se tak nezanedbatelné množství paměti. Syntaxe je zde analogická, například

```
DoubleMatrix2D mat = new SparseRCDoubleMatrix2D(rows, cols);
```

Existuje mnoho formátů ukládání řídkých dat (viz např. [58, s. 57–60]), z nichž nejběžnější jsou různé slovníkové či seznamové způsoby a komprimace po řádcích nebo po sloupcích. Komprimované formáty přitom ukládají nenulové prvky matice do jednorozměrného statického pole a jejich poloha v matici pak je určena dvěma jednorozměrnými indexovými poli. V dříve zmíněné knihovně Parallel Colt uvedeným formátům odpovídají datové typy SparseDoubleMatrix2D, resp. SparseCCDoubleMatrix2D (komprimace po sloupcích) a SparseRCDoubleMatrix2D (komprimace po řádcích). Dále knihovna nabízí i typy

[†]Vzhledem k pozdějšímu volání metody solve() pro řešení soustavy rovnic, která je univerzální a nerozlišuje mezi hustými a řídkými maticemi a vektory, a faktu, že DoubleMatrix1D je abstraktní třídou, je nutné deklaraci uvést tímto „typově smíšeným“ způsobem.

SparseCCMDoubleMatrix2D a SparseRCMDoubleMatrix2D, které jsou také komprimované, ale ukládají jednotlivé sloupce či řádky jako řídké vektory (SparseDoubleMatrix1D).

Za účelem zjištění, který z uvedených způsobů je pro potřeby zjednodušeného CFD modelování při použití knihovny Parallel Colt nejvýhodnější[†], byl proveden jednoduchý benchmark. Z reálného výpočtu byla převzata matice systému linearizovaných rovnic o hodnosti ~14 000 spolu s vektorem pravé strany a bylo zkoumáno, nakolik se způsob uložení matice projeví ve spotřebě RAM a jak dlouho bude za jinak identických podmínek trvat jedno vyřešení patřičného systému. Výsledky jsou shrnuty v tabulce 2.3 a vyplývá z nich, že pro ukládání matic je nejlepší používat datový typ SparseRCDoubleMatrix2D, a to jak z pohledu spotřeby paměti, tak i z pohledu výpočetního času. Na tomto místě je vhodné upozornit na výrazně delší výpočetní časy změřené při použití datových typů SparseCCMDoubleMatrix2D a SparseRCMDoubleMatrix2D (tedy na skutečnost, že u odpovídajících hodnot v tabulce opravdu nechybí desetinné čárky).

S problematikou výše souvisí také způsob volání metod pro ukládání hodnot do matic a vektorů na specifickou pozici, resp. jejich čtení, k čemuž typicky dochází při vyplňování koeficientů jednotlivých linearizovaných rovnic. Toto je běžně prováděno cyklem, iterováním skrze kolekci indexů a podobně, čili máme dopředu jistotu, že indexy řádku a sloupce nepřekročí rozsah matice či vektoru. V takovém případě nemá smysl používat výchozí metody, které vždy kontrolují přijatelnost indexů, neboť by šlo o zcela zbytečnou operaci prodlužující výpočet. Jsou-li tedy v použité knihovně pro lineární algebru k dispozici méně „bezpečné“, avšak rychlejší metody pro ukládání hodnot na a jejich čtení ze zadaných pozic, je nanejvýš vhodné je používat. V kontextu knihovny Parallel Colt pak jde o náhradu metod get() a set() metodami getQuick() a setQuick().

2.9.5 Specializované knihovny pro lineární algebru

Volba vhodné knihovny s rutinami pro provádění maticových operací závisí primárně na použitém programovacím jazyku a je jedním z nejdůležitějších faktorů ovlivňujících

[†] V jiných knihovnách s jinými implementacemi patřičných algoritmů může samozřejmě být situace odlišná.

Tabulka 2.3. Vliv datového typu matice soustavy o hodnosti ~14 000 na průměrnou spotřebu RAM a průměrný čas nutný k jednomu vyřešení testovacího lineárního systému pomocí BiCGstab(2)

Datový typ	Spotřeba RAM	Průměrný výpočetní čas
SparseDoubleMatrix2D	2 570 kB	10,83 ms
SparseCCDoubleMatrix2D	1 267 kB	2,666 ms
SparseRCDoubleMatrix2D	1 199 kB	2,609 ms
SparseCCMDoubleMatrix2D	3 006 kB	3 783 ms
SparseRCMDoubleMatrix2D	3 871 kB	5 702 ms

rychlost výpočtu. Mnohé dostupné knihovny přitom nejsou ničím jiným než wrappery poskytujícími API[†] pro přístup k funcionalitám jiných (nízkoúrovňových) knihoven pro lineární algebru napsaných např. ve Fortranu – obvykle Basic Linear Algebra Subprograms (BLAS) [59] či Linear Algebra PACKage (LAPACK) [60] (resp. jejich různé implementace), Intel Math Kernel Library (MKL) [61], AMD Optimizing CPU Libraries (AOCL) [62] apod. Tyto nízkoúrovňové knihovny mohou být poskytovány pod některou z open-source licencí (GPL, BSD, ...), nebo může jít o proprietární kódy s různými distribučními modely. Dále je vhodné vzít v potaz, že knihovny distribuované výrobci hardwaru jsou optimalizovány přímo pro jejich čipy a mohou tak být v některých situacích vhodnější než knihovny nezávislé na architektuře a platformě.

V případě Javy, která je mezi vývojáři softwaru velmi oblíbená, je k dispozici relativně široká škála wrapperů či knihoven obsahujících implementace numerických rutin přímo v tomto jazyce – již zmíněný Parallel Colt, oj! Algorithms [63], Efficient Java Matrix Library (EJML) [64], Matrix Toolkits Java (MTJ) [65], Universal Java Matrix Package (UJMP) [66] a další. Jednotlivé balíky se liší v prvé řadě dostupnými funkcionalitami, což je souhrnně popsáno například v Arndtově přehledu javových knihoven pro operace s hustými a řídkými maticemi [67]. Druhým a neméně důležitým rozdílem pak je výpočetní efektivita jednotlivých operací v závislosti na hodnosti matice. Obecně zde platí, že při práci s maticemi větších hodností by měly pokud možno být preferovány knihovny volající numerické rutiny v nativním kódu (tj. zkompilované a optimalizované pro použitý hardware).

Z mnoha dostupných testů porovnávajících různé javové knihovny pro lineární algebru lze uvést například Abelesův obsáhlý benchmark [68], ve kterém nejsou výsledky uvedeny agregovaně, ale naopak jsou rozděleny podle typu prováděné operace. Uživatel si pak tudíž může knihovnu vybrat na míru podle toho, které konkrétní operace bude jeho aplikace využívat (tj. na efektivitu ostatních operací, byť by třeba byla nižší, nemusí brát velký zřetel) a zda je ochoten volat nativní kód (a potažmo se smířit s tím, že pro fungování jeho aplikace bude dost možná nutné nainstalovat dodatečný software třetí strany).

2.10 Vyvinutý 3D CFD software

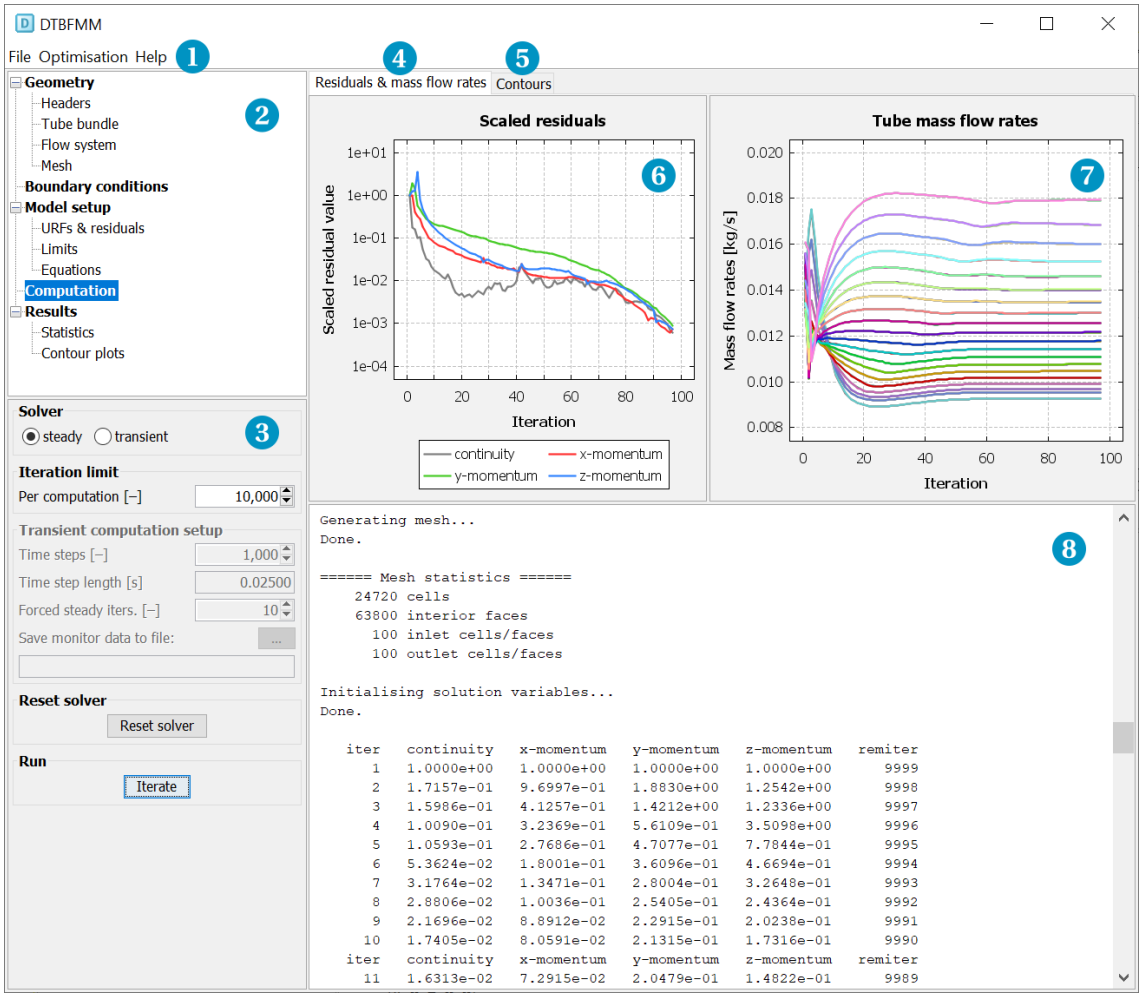
V Javě vyvinutý 3D CFD software Dense Tube Bundle Flow Modeller (DTBFMM[‡]) [A8] byl poprvé představen v publikaci [A5]. Od té doby do něj byla implementována mnohá vylepšení, z nichž ta nejdůležitější, která se týkala primárně zvýšení efektivity výpočtů, byla popsána v článcích [A7] (volba maticových řešičů a metod předpodmínění obecně) a [A6] (výhody a doporučení ohledně předpodmínění pomocí symetrické neúplné rela-

[†] „Application Programming Interface“, rozhraní pro programování aplikací

[‡] Poslední „M“ ve zkratce má historický původ v „Matrix version“. Dřívější verze aplikace totiž používaly algebraické modely proudění (nikoliv zjednodušené CFD) a výpočetní síť byla procházena postupně po jednotlivých prvcích. Je zřejmé, že tento přístup byl neefektivní a v mnoha případech s sebou nesl nezanedbatelné obtíže při provádění korekcí hodnot veličin.

xace). Kromě toho ale bylo v softwaru oproti prvotní verzi provedeno také velké množství dalších úprav – byly přidány optimalizačních funkcionality, došlo k různým drobným úpravám uživatelského rozhraní, byly zefektivněny některé části výkonného kódu apod. Snímek hlavního okna aktuální verze aplikace je na obrázku 2.25.

Výpočetní síť je pro zadanou geometrii distribučního systému (rozměry distributoru a kolektoru, uspořádání a rozměry trubkového svazku atd.) vytvářena přímo softwarem,



Obrázek 2.25. Uživatelské rozhraní vyvinuté javové aplikace DTBFMM; 1 … hlavní menu, 2 … strom vlastností CFD modelu, 3 … panel vlastností dostupných pro vybranou položku ve stromu, 4 … záložka s informacemi o průběhu výpočtu, 5 … záložka obsahující interaktivní vrstevnicové grafy různých veličin, 6 … graf škálovaných reziduí, 7 … graf průtoků jednotlivými trubkami distribučního systému (legenda je nyní z prostorových důvodů skryta), 8 … textové pole s informacemi o průběhu výpočtu, stavovými informacemi atd.; zobrazené údaje týkající se výpočtu odpovídají distribučnímu systému z obrázku 2.3 na straně 13 a celkovému hmotnostnímu průtoku vody ve výši 0,5 kg s^{-1}

a to v závislosti na požadované jemnosti sítě a růstovém faktoru. Vlastní výpočet je však s ohledem na jeho charakter samozřejmě na síti nezávislý. Pokud by tedy byla síť dodána v potřebném formátu jinak (resp. byl adekvátně rozšířen automatický generátor sítě), nijak by to neovlivnilo způsob použití aplikace. Podporovány jsou ustálené i tranzientní výpočty, při kterých lze časově závislá data týkající se průtoků ukládat na disk pro případné pozdější zpracování. Pro zaručení efektivity maticových výpočtů jsou veškeré takové operace v aplikaci prováděny pomocí knihovny Parallel Colt.

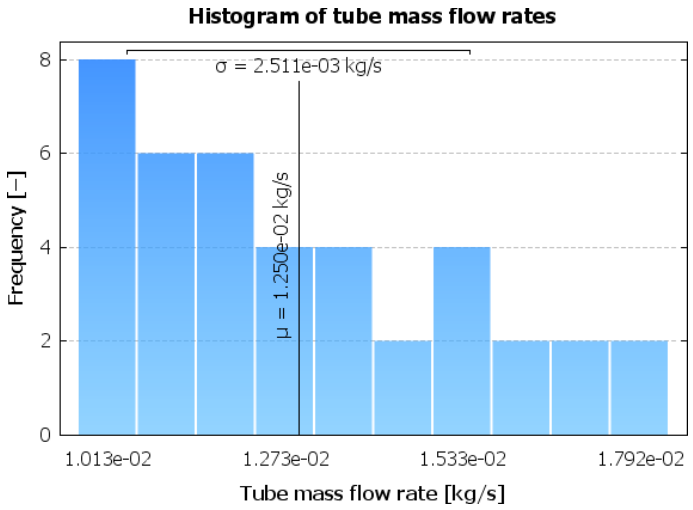
Co se týče okrajových podmínek, k dispozici jsou obě kombinace vstupů a výstupů standardně používané pro modelování proudění v procesních a energetických zařízeních. Vstupní zóny tedy mohou být buď s předem daným celkovým hmotnostním průtokem („mass flow inlet"), nebo se zadanou rychlostí proudění („velocity inlet"), zatímco výstupy jsou vždy se zadaným tlakem („pressure outlet"). Na stěnách trubek ve svazku lze nastavit nenulový tepelný tok za účelem simulace ohřevu či chlazení tekutiny. V tuto chvíli obsahuje databáze tekutin v aplikaci pouze vodu a vzduch, nicméně z programátorského hlediska by nebylo nijak obtížné přidat dle potřeby i další tekutiny.

Implementovaný CFD řešič je segregovaný, přičemž řídicí proměnnou je tlak. K dispozici jsou metody SIMPLE a SIMPLEC. Diskretizační schéma použité pro rovnice tlakové korekce je vždy prvního řádu. U rovnic pro jednotlivé složky rychlosti a u rovnice energie může uživatel volit mezi schématem „upwind" prvního řádu, centrální diferencí (která je sice využitelná jen omezeně, ale její přidání představovalo pouhé jednotky řádků kódu navíc), hybridním schématem, schématem „power law" a stabilizovaným schématem QUICK druhého řádu. Tranzientní formulace je implicitní, prvního řádu. V souladu s kapitolou 2.6 zatím není turbulence modelována žádným ze standardních způsobů (např. k–ϵ), ale pouze zjednodušeně škálováním molekulární viskozity pomocí empiricky zjištěného vztahu v závislosti na Reynoldsově čísle a dalších parametrech.

Průběh každého výpočtu je doprovázen standardním výpisem informací o aktuálním stavu škálovaných reziduí, resp. dle potřeby i dalších informací. Kromě toho jsou výpočty vždy zakončeny (je-li k dispozici alespoň nějaké řešení, byť třeba zatím nezkonvergované) analýzou stavu distribuce tekutiny. Pro každý kanál distribučního systému je vypsána patřičná hodnota průtoku a pro celý systém je vypočtena relativní směrodatná odchylka od rovnoměrného rozdělení dle rovnice zmíněné například v autorově disertační práci [A3]. Nakonec je v textovém poli zobrazen histogram průtoků s vyznačenou střední hodnotou a směrodatnou odchylkou, jehož ukázka je na obrázku 2.26.

Pro potřeby zpracovávání výsledků výpočtů jsou k dispozici funkcionality pro časové průměrování průtoků jednotlivými trubkami (neboť tyto téměř vždy kolísají) a vizualizaci dat. Průměrovat lze buď od uživatelem zadaného výpočetního času, nebo může software vhodný počáteční čas odhadnout automaticky na základě zadané maximální přípustné procentuální odchylky od střední hodnoty[†]. Vizualizace dat je možná skrze in-

[†]Nalezený čas pak je nejmenší takový, při kterém leží veškeré následující průběhy průtoků ve všech trubkách distribučního systému vždy v pásech určených aktuálními středními hodnotami a zadanou maximální přípustnou procentuální odchylkou.

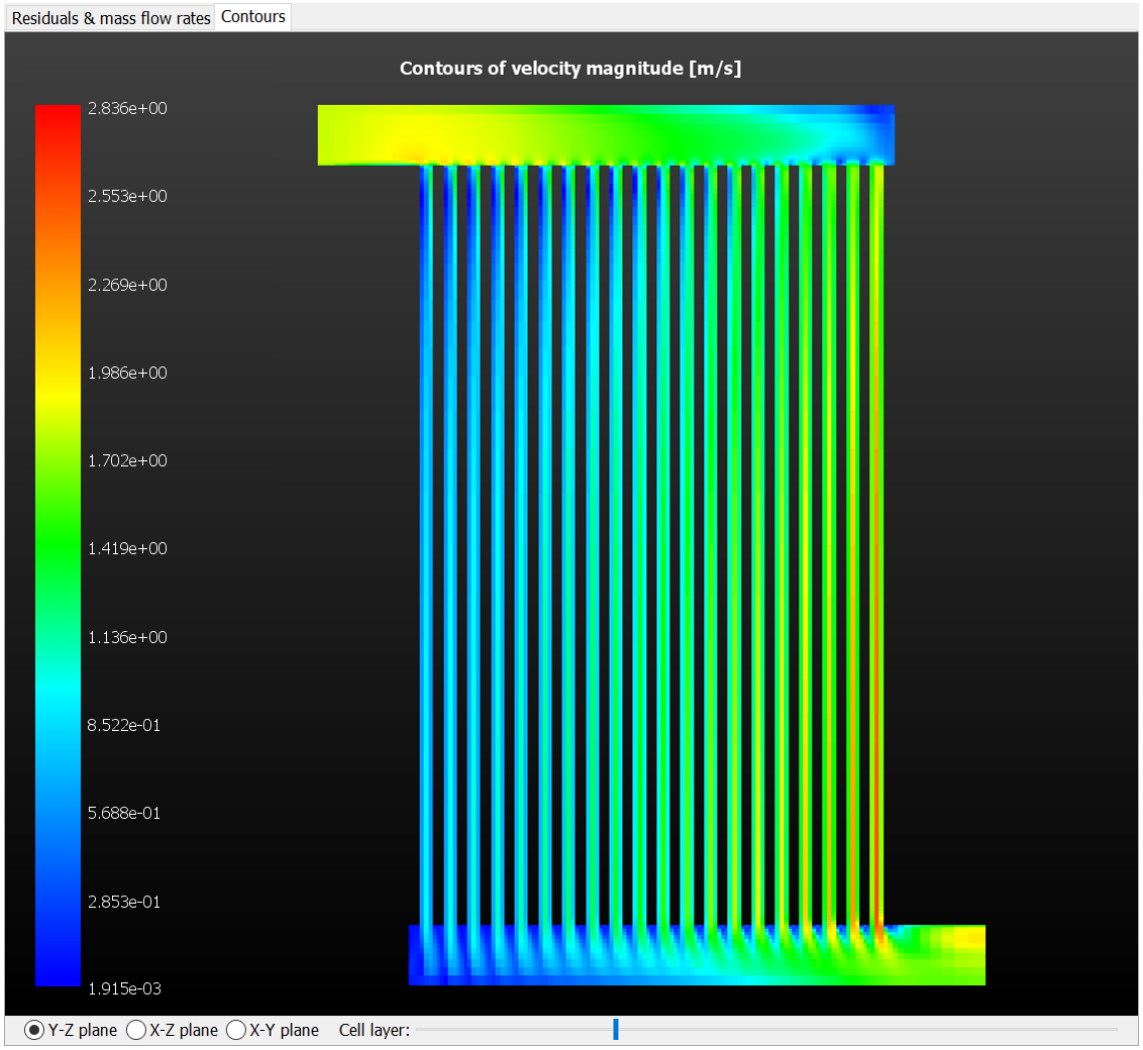


Obrázek 2.26. Histogram průtoků s vyznačenou střední hodnotou, μ, a směrodatnou odchylkou, σ; data odpovídají distribučnímu systému z obrázku 2.3 na straně 13 a celkovému hmotnostnímu průtoku vody ve výši 0,5 kg s^{-1}

teraktivní vrstevnicové grafy, a to ve třech hlavních rovinách (y–z, x–z a x–y). Zobrazovat lze následující veličiny:

- složky rychlosti proudění ve směrech os x, y a z,
- velikost rychlosti,
- velikost vorticity,
- statický tlak,
- teplotu,
- hustotu,
- dynamickou viskozitu,
- virtuální dynamickou viskozitu (tj. „uměle“ škálovanou molekulární viskozitu používanou pro přibližné modelování turbulence),
- měrnou tepelnou kapacitu při konstantním tlaku a
- tepelnou vodivost.

Rozsahy barevných stupnic lze zadat ručně, příp. mohou být softwarem nastaveny automaticky dle aktuálně vykreslovaných dat. Kromě toho lze uživatelsky měnit druh pozadí grafů (bílé, světlý přechod, tmavý přechod) a typ vrstevnicového grafu, tedy zda mají být použity spojité barevné stupnice, nebo diskrétní stupnice obsahující pouze velmi omezený počet barev. Ukázkový interaktivní graf je na obrázku 2.27.



Obrázek 2.27. Interaktivní vrstevnicový graf velikosti rychlosti; na ovládacím panelu ve spodní části grafu je zvolena vizualizace dat v rovině y–z z vrstvy buněk ležící přibližně ve třetině tloušťky celé domény ve směru kolmém na tuto rovinu; data odpovídají distribučnímu systému z obrázku 2.3 na straně 13 a celkovému hmotnostnímu průtoku vody ve výši 3,0 kg s^{-1}

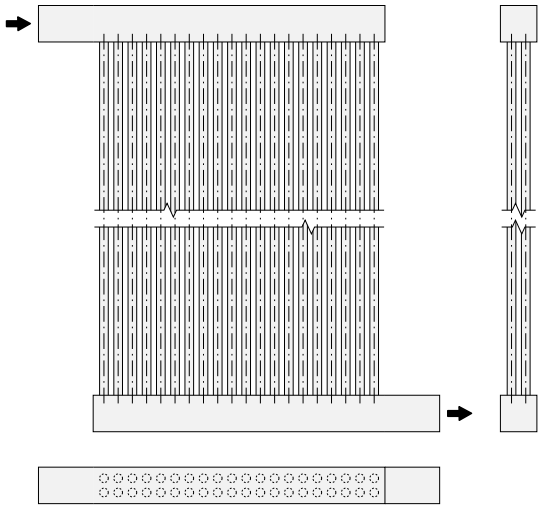
Další vestavěnou funcionalitou je nástroj pro optimalizaci rozměrů distributoru a kolektoru. Pro zadané přípustné rozsahy hlavních rozměrů je optimální geometrie nalezena pomocí ustálených výpočtů a akcelerované Hookeovy-Jeevesovy metody [A9] (v případě, že optimalizační doména je vícerozměrná), resp. metody zlatého řezu [69] (u jednorozměrné domény). Účelová funkce může minimalizovat buď relativní směrodatnou odchylku od rovnoměrného rozdělení, nebo celkovou tlakovou ztrátu v systému. Záznam o průběhu optimalizačního procesu může být ukládán na disk. Na konci procesu pak

je zobrazeno shrnutí vyhodnocených geometrií seřazených od nejlepší po nejhorší podle zvoleného optimalizačního kritéria.

Z pohledu přesnosti predikce rozdělení toku je popsaný software přijatelný. V provedených testech se relativní chyba vůči datům získaným detailními CFD výpočty pomocí komerčního softwaru ANSYS Fluent pohybovala u nejhrubších možných sítí (tj. takových, kde příčný průřez každé trubky obsahoval vždy pouze jedinou buňku) do 5 %. S rostoucí jemností výpočetní sítě se pak predikce postupně zpřesňovala, jak bylo ukázáno na obrázku 2.4 na straně 15. Prozatimní nevýhodou však je skutečnost, že predikované tlakové ztráty jsou vůči hodnotám získaným z detailních CFD modelů typicky o cca 40–80 % vyšší. I zde sice platí, že s rostoucí jemností výpočetní sítě se chyba snižuje, ovšem výrazně pomaleji než v případě predikce rozdělení toku.

2.11 Srovnání CFD softwaru s komerčními aplikacemi: uživatelský pohled

Uvažujme nyní pro srovnání situaci, kdy je potřeba analyzovat určitý distribuční systém; kupříkladu ten z obrázku 2.28. Jeho analýza pomocí vyvinutého 3D CFD softwaru a některé – libovolné – komerční CFD aplikace (např. ANSYS Fluent) by zcela zřejmě byla odlišně náročná a poskytla by výsledky odlišné přesnosti. Zběžné srovnání obou přístupů je uvedeno v tabulce 2.4. Z ní je patrné, že navzdory nižší přesnosti dat a prozatimním omezením vyvinutého softwaru dává jeho užití pro předběžné analýzy smysl, neboť s minimálním úsilím ze strany uživatele získáme rychle vše potřebné pro tepelně-hydraulické



Obrázek 2.28. Uvažovaný distribuční systém; rozměry distributoru a kolektoru jsou $40 \times 40 \times 320$ mm (Š × V × D), trubky mají vnitřní průměr 10 mm a délku 2 000 mm

Tabulka 2.4. Srovnání vyvinutého softwaru a běžně užívaných komerčních aplikací

Kategorie	Vyvinutý software	Komerční aplikace
Příprava modelu	kompletní nastavení modelu bylo provedeno během jedné minuty, neboť stačilo zadat pouze nejnutnější údaje (charakteristické rozměry distribučního systému, požadovanou jemnost sítě a okrajové podmínky)	příprava geometrie, tvorba sítě a nastavení modelu vyžadovaly značné znalosti a bylo potřeba specifikovat mnohem větší množství parametrů; celková doba přípravy modelu byla v řádu hodin
Vlastní výpočet	řešení ustálené úlohy bylo získáno za 97 vteřin (síť: ~28 tis. buněk)	paralelní řešení ustálené úlohy na 8 jádrech CPU trvalo zhruba dvě hodiny (síť: ~1,3 mil. buněk)
Analýza dat o distribuci toku	analýza údajů proběhla na konci výpočtu automaticky a výsledky byly zobrazeny uživateli	analýzu dat o distribuci bylo nutné provést manuálně pomocí údajů z monitorovacích ploch
Výhody	intuitivnost softwaru, rychlost nastavení modelu, krátké výpočetní časy, přítomnost nástroje pro tvarovou optimalizaci hlavních kanálů	flexibilita, lepší kontrola nad vlastním výpočtem (neboť uživatel musí vše detailně nastavit sám), dostupnost široké škály metod a řešičů
Nevýhody	nižší přesnost[†]; nelze aplikovat na geometrie nekompatibilní s kvádrovou sítí, resp. takové, pro které dosud nebyl vytvořen automatický generátor sítě	výrazně větší výpočtová náročnost a potažmo delší výpočetní časy, nutnost kvalifikované obsluhy

posouzení patřičného systému. Velkou výhodou zmíněného softwaru také je automatizovatelnost výpočtů a vestavěný nástroj pro tvarovou optimalizaci hlavních kanálů.

2.12 Budoucí zaměření výzkumu

V souvislosti s přesností dat získaných pomocí zjednodušených CFD modelů, která z podstaty věci nemůže být stejná jako u modelů detailních, se nabízí otázka, do jaké míry je „na vině“ hrubá kvádrová mřížka, resp. modelování turbulence zjednodušeně škálováním molekulární viskozity. Je sice pravdou, že určitý podíl na nepřesnosti může mít užití diskretizačních schémat prvního řádu, nicméně jejich vliv nejspíš nebude natolik výrazný. Aplikace stabilizovaného diskretizačního schématu QUICK druhého řádu totiž v provedených testech buď nepřinesla oproti schématům „upwind“ či „power law“ (která jsou prvního řádu) pozorovatelné zlepšení a naopak měla za následek výrazně delší výpočetní časy, nebo dokonce byla v mnoha případech zdrojem numerických potíží.

[†]V případě diskutovaného modelu byla relativní chyba hmotnostních průtoků trubkami oproti detailnímu výpočtu v komerční aplikaci max. ~1,7 %.

V budoucnu se proto výzkum v oblasti zjednodušeného CFD modelování bude zaměřovat hlavně na zhodnocení, zda má smysl implementovat některý standardní model turbulence. Toto by samozřejmě vedlo ke znatelnému zpomalení výpočtů, avšak mohlo by se to ukázat jako vhodný prostředek ke zpřesnění výpočtů. Zjednodušený 3D CFD model včetně turbulence (formou k–ϵ) byl již vytvořen v rámci diplomové práce [70] a nyní tedy zbývá provést vhodné srovnávací výpočty. Na základě takto získaných dat a odpovídajících výpočetních časů pak bude možné posoudit, nakolik je implementace standardního modelu turbulence vhodná, resp. zda ho do softwaru z předešlé kapitoly přidat například jen jako uživatelsky aktivovatelnou funkcionalitu.

Další oblastí, kde je prostor ke zlepšení – konkrétně ke zkrácení výpočetních časů a částečně i ke snížení míry výskytu numerických potíží – je způsob indexování proměnných. Namísto aktuálně používaného přirozeného pořadí by bylo vhodné implementovat i další typy indexování, což ovšem bude vyžadovat nejen úpravy existujícího automatického generátoru výpočetní sítě, ale také provedení testovacích výpočtů nezanedbatelného rozsahu. Dopředu totiž nelze říci, který ze způsobů indexování povede u zjednodušených kvádrových výpočetních sítí k nejlepším výsledkům.

V neposlední řadě je potřeba zmínit zjednodušené 3D CFD modelování pomocí běžně dostupných CFD softwarů. Uživatel tak sice nemá úplnou kontrolu nad celým výpočtem (neboť ne všechny aspekty modelu pak lze uživatelsky měnit či řídit), ale na druhou stranu také není vývoj zmíněných modelů natolik časově a programátorsky náročný. Ideálním kandidátem z řad dostupných CFD softwarů je zde OpenFOAM [71], který je distribuován jako open source. Modely – včetně výpočetních sítí – se v něm kompletně definují pomocí sad textových konfiguračních souborů a výpočty (případné optimalizační úlohy nevyjímaje) by tedy bylo možné snadno zautomatizovat například naprogramováním parametrického generátoru takových souborů. Z programátorského hlediska by přitom šlo o výrazně jednodušší úlohu než psaní celého CFD kódu od základu. V tomto ohledu už proto jsou prováděny určité kroky v rámci aktuálně vypsaných bakalářských a diplomových prací.

Kapitola

3

Modelování založené na principu FEA

Z předchozí kapitoly je zřejmé, že ačkoliv je zjednodušené 3D CFD modelování značně univerzální, může být u větších zařízení vyžadujících rozsáhlejší výpočetní sítě i přes různá implementovaná zjednodušení náročnější z pohledu výpočetního času. Pokud však taková zařízení sama o sobě nejsou příliš komplexní – například jde-li o velké distribuční systémy v kotlích na odpadní teplo, které se obvykle skládají pouze z menšího počtu řad trubek – mohly by dostatečně dobře posloužit i výrazně jednodušší modely. Z tohoto důvodu byl započat vývoj a testování modelu proudění založeného na analýze metodou konečných prvků („Finite Element Analysis“, FEA).

V případě zmíněného způsobu modelování se lze inspirovat kupříkladu u metod pro návrh sítí pro distribuci vody, které vzhledem k rozsáhlosti úloh vyžadují právě maticovou implementaci. Na rozdíl od aplikací v procesním či energetickém průmyslu však zde zpravidla bývá cílem najít takovou strukturu potrubního systému, která by pouze na základě kapacit jednotlivých hran zajistila splnění lokálních požadavků na dodávaná množství, případně se zjišťují místa úniku vody, simuluje se šíření znečišťujících látek a podobně. Jen vzácněji (třeba v práci [72]) se lze setkat s modely, které alespoň částečně zohledňují tlakové ztráty a další změny tlaku související s prouděním, resp. vliv těchto faktorů na výsledné rozdělení toku.

Cílem aktuálně prováděného výzkumu je rozšířit uvedenou strategii na vybraná procesní a energetická zařízení obsahující strukturou relativně nekomplikované trubkové svazky (již zmíněné kotle na odpadní teplo apod.) a získat tak nástroj, kterému by z po-

hledu nutného výpočetního času mohlo zjednodušené 3D CFD modelování konkurovat jen s obtížemi. Od modelu se výhledově očekává schopnost predikovat nejen charakter proudění pracovních látek, ale hlavně přenos tepla a odpovídající mechanické namáhání svazku vlivem nerovnoměrné distribuce tekutiny v kombinaci s nerovnoměrným teplotním polem v mezitrubkovém prostoru. V kontextu kapitoly 1.1 by tedy mělo jít o komplexní řešení poskytující veškeré základní informace o analyzovaném zařízení, které jsou podstatné z pohledu tepelně-hydraulického návrhu a předcházení provozním potížím.

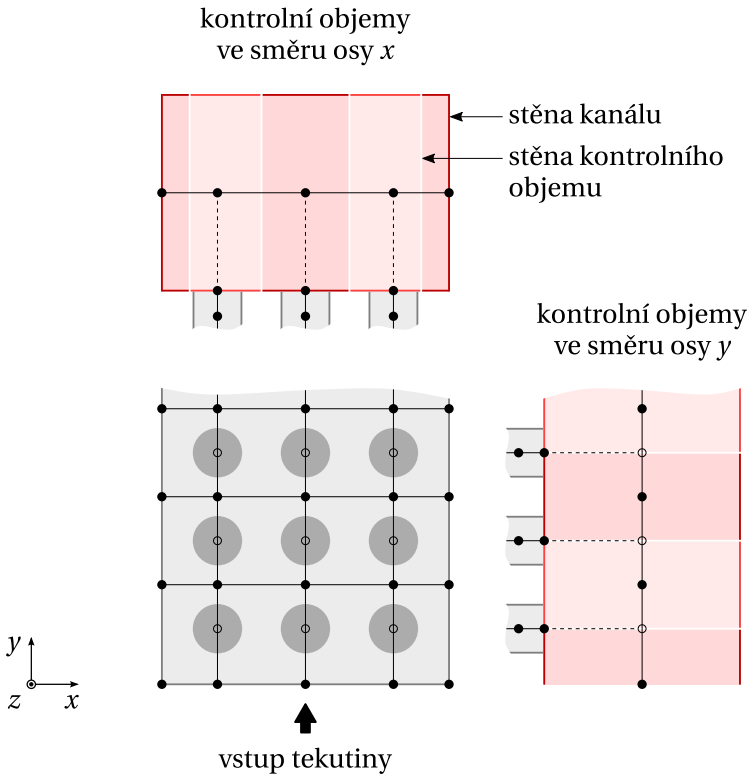
3.1 Hlavní výhody a nevýhody

Implementace algebraických modelů využívajících kvazi-1D výpočetní sítě (například toho z článku [A10], který je určen pro simulace proudění v jednodušších trubkových svazcích, či modelu popsaného v publikaci [A11], který lze použít k analýzám radiálních katalyzátorů) s sebou nepřináší žádné větší numerické komplikace. Jakmile však je nutné řešit distribuční systém, který pomocí kvazi-1D sítě dostatečně dobře popsat nelze, a musíme tedy pracovat s výpočetní sítí vyšší dimenze (typicky s kvazi-3D sítí z obrázku 3.1, kde jsou hlavní distribuční kanály reprezentovány dvourozměrnou sítí a jednotlivé paralelní větve jednorozměrnými sítěmi), vyvstane otázka, jak v iteračním výpočtu provádět korekce veličin. U algebraických modelů je totiž síť procházena prvek po prvku a odpovídající nelineární rovnice jsou na těchto prvcích řešeny postupně, nikoliv jako vzájemně provázaná soustava linearizovaných rovnic. V článku [A4] bylo sice ukázáno, že nalezení vhodného způsobu provádění korekcí není nemožné, ale zároveň rozhodně nejde o přímočarou operaci. Nadto platí, že způsob, který u jedné geometrie vede ke zkonvergovanému řešení, nemusí vůbec být vhodný u geometrie jiné.

Maticový způsob výpočtu analogický tomu z článku [72] (respektive z velké části maticový způsob, jak bude popsáno dále v textu) uvedené omezení eliminuje. Pokud tedy najdeme jeden – libovolný – funkční způsob provádění korekcí, lze očekávat, že tento bude použitelný i při modelování jakéhokoliv jiného distribučního systému. Nutno ovšem poznamenat, že nalezení vhodného korekčního algoritmu není zcela triviální, a to hlavně u prvků výpočetní sítě, které reprezentují komplexnější struktury. Za nevýhodu diskutovaného přístupu může také být považováno obtížné korektní zahrnutí vlivu turbulence. Toto by nám však nemuselo příliš vadit, neboť bychom zde mohli – stejně jako u zjednodušených 3D CFD modelů popsaných v kapitole 2 – turbulenci modelovat jen přibližně prostým škálováním viskozity.

3.2 Matematický model

Odpovídající matematický model byl detailně popsán v článku [A13]. Využívá se v něm analogie Hookeova zákona, který je běžně aplikován v pružnostně-pevnostních modelech založených na metodě konečných prvků. Zde se však opíráme o skutečnost, že při



Obrázek 3.1. Část typické kvazi-3D výpočetní sítě distributoru a připojeného svazku trubek; plné body představují běžné uzly sítě, prázdné body virtuální uzly uvažované pouze interně ve speciálních typech prvků sítě (adaptováno z [A12])

znalosti tlakového gradientu lze hmotnostní tok určitým kanálem (hranou výpočetní sítě) zjednodušeně získat pomocí součinu „poddajnosti“ tohoto kanálu a patřičného rozdílu v tlacích mezi koncovými body. Pro libovolnou orientovanou hranu výpočetní sítě, jež spojuje uzly i a j, tedy musí platit

$$\dot{\mathbf{m}}_{ij} = \mathbf{K}_{ij}\mathbf{p}_{ij}, \tag{3.1}$$

kde matice poddajnosti, vektor tlaků v uzlech a vektor součtů hmotnostních toků obecně nabývají tvarů

$$\mathbf{K}_{ij} = \begin{pmatrix} k_{ii} & k_{ij} \\ k_{ji} & k_{jj} \end{pmatrix}, \quad \mathbf{p}_{ij} = \begin{pmatrix} p_i \\ p_j \end{pmatrix} \quad \text{a} \quad \dot{\mathbf{m}}_{ij} = \begin{pmatrix} \dot{m}_i \\ \dot{m}_j \end{pmatrix}.$$

Zároveň je nutné, aby alespoň jeden z uzlů výpočetní sítě byl uzlem vstupním a alespoň jeden z uzlů byl uzlem výstupním, čili aby alespoň v těchto dvou uzlech byly součty

hmotnostních toků nenulové. V jiných než vstupních a výstupních uzlech pak zcela zřejmě musí platit $\dot{m}_i = 0$.

Na začátku iteračního výpočtu jsou z okrajových podmínek získány počáteční odhady hmotnostních toků, tlaků v uzlech a poddajností prvků výpočetní sítě (tyto lze získat na základě hydraulického odporu toho kterého kanálu, tj. pomocí Darcyho-Weisbachovy rovnice [73, s. 340]). Pak je proveden predikční krok, ve kterém je pomocí matice sestavené podle rovnice 3.1 pro veškeré prvky získán nový odhad tlakového pole. Je potřeba zdůraznit, že toto pole zatím nijak nezohledňuje nelinearity. V následném korekčním kroku se proto ze získaných tlakových rozdílů pro jednotlivé prvky dopočtou nové odhady hmotnostních toků, avšak již se zohledněním vlivu nelinearit. Korekční krok přitom není prováděn maticově, ale na každém jednotlivém prvku sítě zvlášť, neboť jde o nezávislý iterační výpočet. Nakonec je v případě vícero vstupních či výstupních uzlů aktualizován vektor pravé strany a jsou upraveny odhady poddajností. Článkem [72] je doporučováno škálovat poddajnosti prvků poměrem odpovídajících hmotnostních toků po korekčním kroku, $\dot{m}_{\mathrm{corr}}$, a před ním, $\dot{m}_{\mathrm{pre}}$, tj. v následující iteraci $(I+1)$ pracovat s $k_{I+1} = k_I \left(\dot{m}_{\mathrm{corr}}/\dot{m}_{\mathrm{pre}}\right)$. Testováním ovšem bylo zjištěno, že toto má za následek horší konvergenci a výpočet je pak nutné výrazně relaxovat. Mnohem výhodnější tedy je počítat nové hodnoty poddjaností pomocí

$$k_{I+1} = k_I \sqrt{\frac{\dot{m}_{\mathrm{corr}}}{\dot{m}_{\mathrm{pre}}}}, \tag{3.2}$$

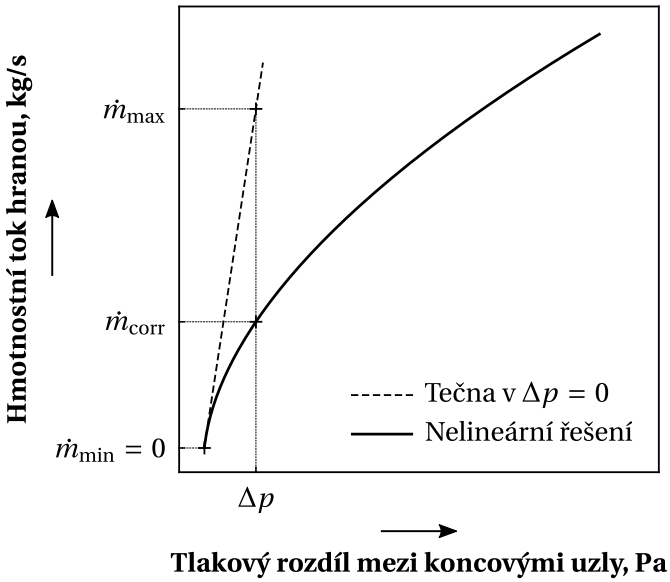
což sice způsobí částečný nárůst výpočetního času, ale značně se zlepší konvergence. Nadto pak ani není nutné do výpočtu zapracovávat jakoukoliv relaxaci[†].

3.2.1 Způsob zahrnutí nelinearit

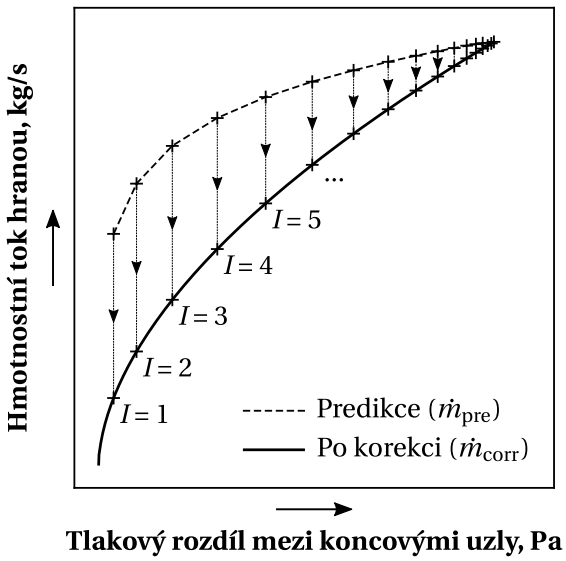
Kromě tlakové ztráty třením jsou nelinearity do modelu vnášeny také místními odpory, závislostí fyzikálních vlastností tekutiny na teplotě a tlaku či v důsledku dělení a slučování proudů. U třecí ztráty je přitom situace relativně jednoduchá. Třecí součinitel vystupující v Darcyho-Weisbachově rovnici totiž nemusíme vůbec počítat iteračním řešením implicitní Colebrookovy-Whiteovy rovnice [74], ale můžeme ho snadno odhadnout libovolnou vhodnou – například Churchillovou [75] – aproximací. Jak ukazuje obrázek 3.2, pro známý odhad tlakového rozdílu Δp mezi krajními uzly prvku z predikčního kroku najdeme v korekčním kroku například pomocí Taylorova rozvoje prvního řádu (tj. tečny) a metody bisekce s mezemi $\dot{m}_{\mathrm{min}} = 0$ a $\dot{m}_{\mathrm{max}}$ korigovaný odhad, $\dot{m}_{\mathrm{corr}}$, odpovídající nelineárnímu řešení. I kdyby však bylo nutné zahrnout více druhů nelinearit, patřičné nelineární řešení by stále bylo monotónní a bylo by tudíž možné použít identický postup. Typická historie konvergence hmotnostního průtoku hranou je znázorněna na obrázku 3.3.

Poněkud složitější je situace v místech, kde se proud tekutiny dělí do většího počtu menších proudů, nebo kde se naopak více menších proudů slučuje do jednoho většího

[†] Použití druhé odmocniny v rovnici 3.2 totiž je samo o sobě jistou formou relaxace.



Obrázek 3.2. Odhad mezí pro metodu bisekce použitou v korekčním kroku pro hranu, u níž je tlakový rozdíl mezi koncovými uzly roven Δp (adaptováno z [A13])



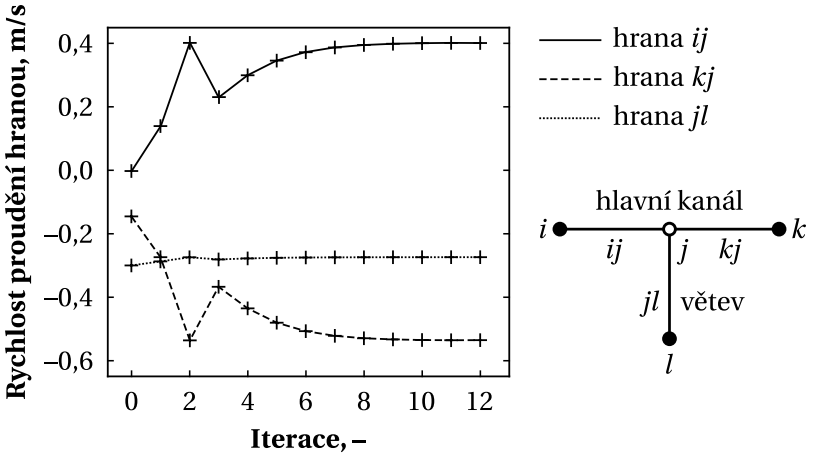
Obrázek 3.3. Typická historie konvergence hmotnostního průtoku hranou získaná při škálování poddajnosti pomocí rovnice 3.2; *I* značí „velké“ iterace řešiče (adaptováno z [A13])

(tedy například v blízkém okolí trubkovnic). Aktuální verze modelu v takovém případě užívá ve výpočetní síti speciální prvek složený ze tří nebo více hran, v němž dvě hrany reprezentují hlavní kanál a zbylá hrana (resp. zbylé hrany) připojené větve. V korekčním kroku se pak předpokládá, že odtékající či přitékající tekutina je mezi všemi těmito větvemi rozdělena poměrně podle tlakových rozdílů mezi společným uzlem a koncovými uzly větví. Vzhledem ke způsobu implementace modelu jsou hrany představující hlavní kanál orientovány směrem do společného uzlu a hrany představující větve směrem ven z tohoto uzlu. Tlakové ztráty třením v jednotlivých hranách jsou počítány úplně stejně jako u běžných hran, zatímco změna tlaku vlivem místního odporu (vtok do, resp. výtok z větve) je získána s využitím koeficientů z literatury [76]. Změna statického tlaku, ke které dochází v důsledku dělení či slučování proudů, je odhadnuta na základě empirických vztahů z článku [77].

Uvažujme nyní prvek z obrázku 3.4, který je složený z hran ij, kj a jl. Za předpokladu identických ploch příčných průřezů hran reprezentujících hlavní kanál ($A_{ij} = A_{kj}$) pak pro poměr ploch $R_A = A_{jl}/A_{ij} = A_{jl}/A_{kj}$ a rychlosti proudění v hranách, v_{ij}, v_{kj} a v_{jl}, platí

$$R_v = \frac{|v_{jl}|}{\max\left\{|v_{ij}|,\ |v_{kj}|\right\}} \in \left[0;\ \frac{2}{R_A}\right]. \tag{3.3}$$

Rychlosti proudění jsou přitom vzhledem k platnosti zákona zachování hmoty vzájemně závislé, čili finální hodnotu R_v zjistíme opět metodou bisekce, jejíž počáteční meze zvolíme stejně jako v rovnici výše. Pro prvotní odhad $R_v = 1/R_A$ a aktuální hodnotu v_{jl} zjištěnou například pomocí rovnice pro místní ztrátu a tlaků p_i a p_l známých z predikčního kroku jsme tudíž schopni užitím rovnice 3.3 najít také odhady zbylých rychlostí.



Obrázek 3.4. Typická historie konvergence rychlostí proudění hranami ve složeném prvku s koncovými uzly i, j a k získaná v korekčním kroku; v tomto případě dochází ke slučování proudů, přičemž tok hlavním kanálem je ve směru od uzlu i k uzlu k (adaptováno z [A13])

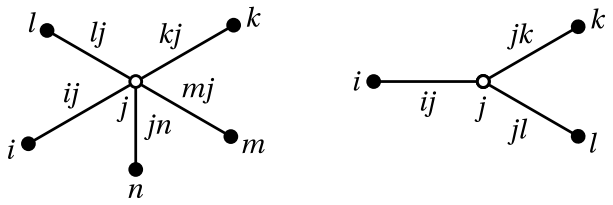
Nyní pomocí nelineárních závislostí, které musí být splněny, zkontrolujeme, zda jsou výsledné tlaky v hraničních uzlech dostatečně blízké predikovaným tlakům p_i, p_j a p_k. Není-li tomu tak, adekvátně se změní meze pro metodu bisekce a celý proces se opakuje. Typická historie konvergence rychlostí proudění hranami složeného prvku získaná tímto postupem je taktéž ukázána na obrázku 3.4.

Otázkou však je, nakolik by výše popsaný přístup byl použitelný například v situaci, kdy je k hlavnímu kanálu připojeno větší množství řad trubek. Stávající předpoklad poměrného rozdělení tekutiny do všech řad v každém místě podél hlavního kanálu totiž vůbec nemusí být realistický [3] a – jak bude ukázáno dále v kapitole 3.2.3 – nemusí vést k dostatečně přesným výsledkům. Výzkumné aktivity jsou nyní proto zaměřeny na rozšíření modelu o dva nové prvky znázorněné na obrázku 3.5. První z nich je složený z pěti hran („dvojitý“ T-kus, obrázek 3.5a) a bude sloužit k modelování širších hlavních kanálů s vyšším počtem řad připojených větví. Druhý prvek potom je složený ze tří hran (Y-kus, obrázek 3.5b) a je zamýšlen k modelování rozdělení jednoho kanálu do dvou separátních větví, neboť toto také bývá ve svazcích mnohdy potřeba, jak je zřejmé z obrázku 3.6.

3.2.2 Okrajové podmínky

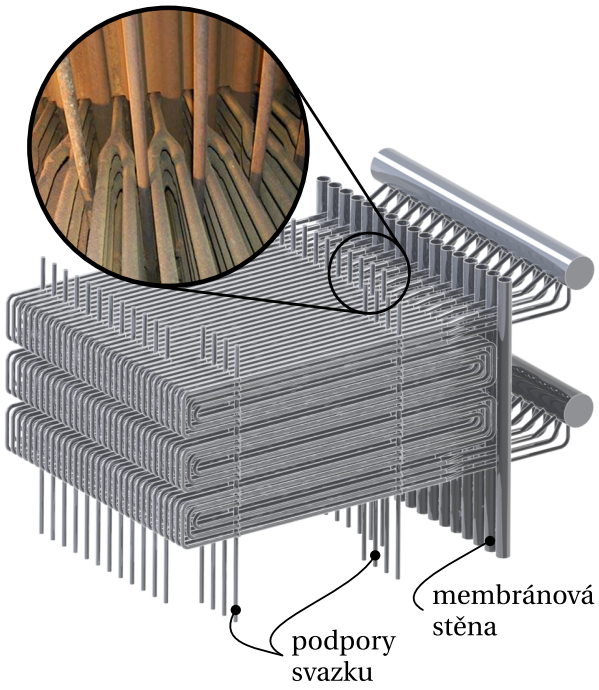
Podobně jako u CFD modelů je i při modelování založeném na principu FEA potřeba specifikovat referenční tlak a množství tekutiny proudící systémem. Není zde přitom podstatné, jestli tlak nastavíme u výstupních uzlů a hmotnostní toky u uzlů vstupních, nebo hodnoty přiřadíme obráceným způsobem či dokonce smíšeně. V každém krajním uzlu však lze zadat pouze jednu z těchto dvou veličin.

Popisovaný model zatím pracuje s výpočetní sítí sestávající ze vzájemně propojených jednorozměrných podsítí. Kanály reprezentované kontrolními objemy pro jednotlivé hrany jsou proto uzavřené (tj. celé jejich pláště jsou tvořeny „fyzickými“ stěnami) a musí pro ně být specifikovány drsnosti povrchu. Příčný průřez každého kanálu je uživatelsky definovatelný. Po přidání prvku z obrázku 3.5a, který umožní pracovat i s běžnými dvourozměrnými podsítěmi, pak bude situace podobná; jediný rozdíl bude spočívat



(a) „dvojitý“ T-kus (b) Y-kus

Obrázek 3.5. Nové typy prvků výpočetní sítě, pro něž jsou v rámci aktuálního výzkumu vytvářeny odpovídající algoritmy korekčního kroku



Obrázek 3.6. Trubkový svazek přehříváku páry s detailním pohledem na způsob rozdvojování trubek (převzato z [78])

v tom, že kontrolní objemy v dvourozměrné části sítě budou striktně čtyřbokými hranoly a z jejich obvodových stěn budou přinejmenším dvě stěnami „fyzickými“.

Co se týče výměny tepla, tuto model v současné době nezahrnuje, nicméně patřičné rozšíření o ohřev či chlazení tekutiny v trubkovém svazku by nemělo být příliš komplikované. V uvedeném ohledu už dokonce bylo provedeno několik základních kroků.

3.2.3 Předběžné srovnání s detailními CFD modely

Za účelem předběžného odhadu přesnosti vyvíjeného modelu byly provedeny simulace proudění v několika různých distribučních systémech podobných těm ze vzduchových chladičů, pro které byla k dispozici data z detailních CFD simulací. Jejich parametry jsou uvedeny v tabulce 3.1.

Proudění distribučními systémy bylo uvažováno jako adiabatické. Pracovní látkou byla voda o teplotě 300 K, jejíž fyzikální vlastnosti byly získávány pomocí knihovny IAPWS[†] [80]. Jelikož model v predikčním kroku předpokládá laminární proudění, dalo se očekávat, že pro nižší hodnoty Reynoldsova čísla bude přesnost předpovědi rozdě-

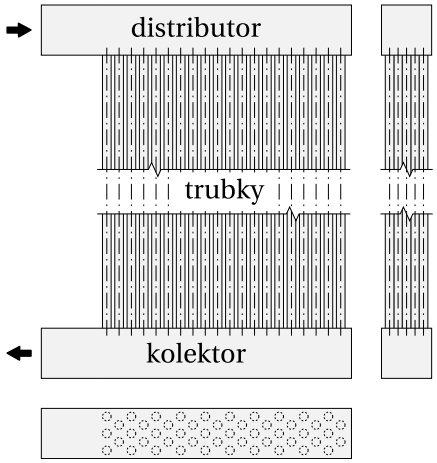
[†] Vyvinutý software ovšem podporuje kromě IAPWS také práci s knihovnou CoolProp [79], jejíž databáze pokrývá celkem 122 běžně používaných procesních médií. Nutnost simulovat proudění jiné látky než vody by tudíž v naprosté většině případů znamenala změnu pouze několika málo řádků kódu.

Tabulka 3.1. Geometrické parametry testovaných distribučních systémů; uvažována byla uspořádání toku „U“ i „Z“, vnitřní průměry trubek ve svazcích byly vždy rovny 10 mm, jejich délky pak 2 m

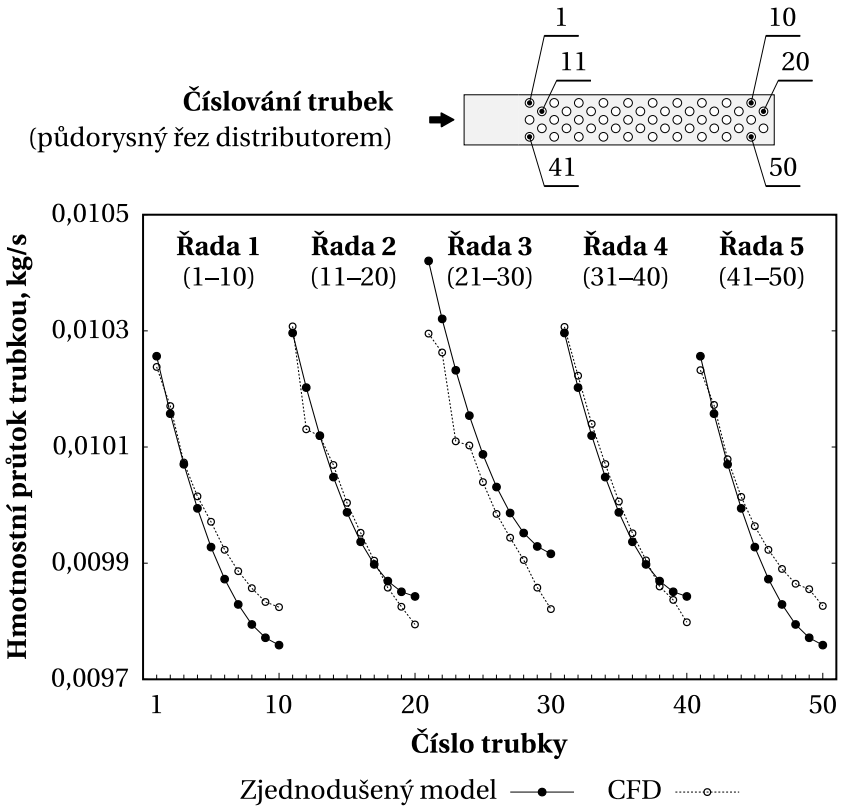
Označení	Hlavní kanály (Š × V × D)	Svazek
A	40 × 40 × 320 mm	2 řady po 20 trubkách, 90°
B	40 × 40 × 280 mm	3 řady po 10 trubkách, 60°
C	55 × 55 × 320 mm	3 řady po 20 trubkách, 90°
D	55 × 55 × 280 mm	5 řad po 10 trubkách, 60°
E	65 × 70 × 235 mm	5 řad po 10 trubkách, 45°
F	70 × 70 × 320 mm	4 řady po 20 trubkách, 90°

lení toku do jednotlivých trubek svazku přijatelná. Toto také bylo provedenými testy potvrzeno – například pro distribuční systém D z tabulky 3.1 s uspořádáním toku „U“ (viz také obrázek 3.7) jsou průtoky jednotlivými trubkami zjištěné stávajícím modelem a detailní CFD simulací porovnány na obrázku 3.8. V následujících testech proto byly voleny spíše vyšší celkové hmotnostní průtoky, aby se ověřilo, nakolik je model – aktuálně bez jakéhokoliv zahrnutí vlivu turbulence – použitelný i v případě výrazně turbulentního proudění, které je v procesních zařízeních obvykle preferováno z důvodu následných vyšších hodnot součinitele přestupu tepla.

Obrázek 3.9 ukazuje relativní chyby průtoků trubkami distribučních systémů A–F z tabulky 3.1 vůči datům z detailních CFD výpočtů v případě silně turbulentního proudění. Celkové hmotnostní průtoky, uvedené v legendě obrázku, zde totiž byly voleny tak, aby průměrná Reynoldsova čísla v trubkách byla přinejmenším ~24 000. Ačkoliv je vidět,



Obrázek 3.7. Schéma distribučního systému D z tabulky 3.1 s uspořádáním toku „U“

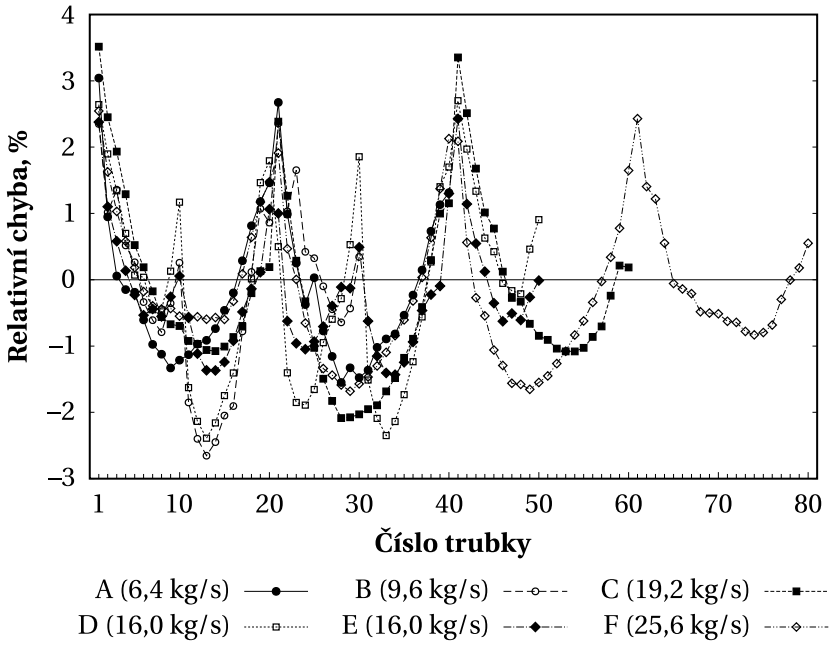


Obrázek 3.8. Srovnání hmotnostních průtoků jednotlivými trubkami distribučního systému z obrázku 3.7, které byly zjištěny zjednodušeným modelem a detailní CFD simulací při průměrné hodnotě Reynoldsova čísla v trubkách $\mathrm{Re_t} \approx 1\,500$. Maximální velikost relativní chyby vůči datům z CFD simulace, která byla zjišťována pro každou jednotlivou trubku vztahem $\epsilon_\mathrm{R} = 100\,(\dot{m}/\dot{m}_\mathrm{CFD} - 1)$, je přibližně 1,2 %.

že i při výrazné turbulenci a vyšším počtu řad trubek jsou velikosti relativních chyb pod 4 %, z obrázku 3.10 je zřejmé, že predikci by stále šlo podstatně zlepšit. Opomeneme-li tedy dočasný a značně nerealistický předpoklad jednorozměrnosti hlavních kanálů, pak jednou z možných úprav, která by jistě přinesla nezanedbatelné zpřesnění modelu, je právě zahrnutí vlivu turbulence – například již zmíněným škálováním viskozity, které bylo s úspěchem použito u zjednodušeného 3D CFD modelování.

3.3 Vyvinutý software

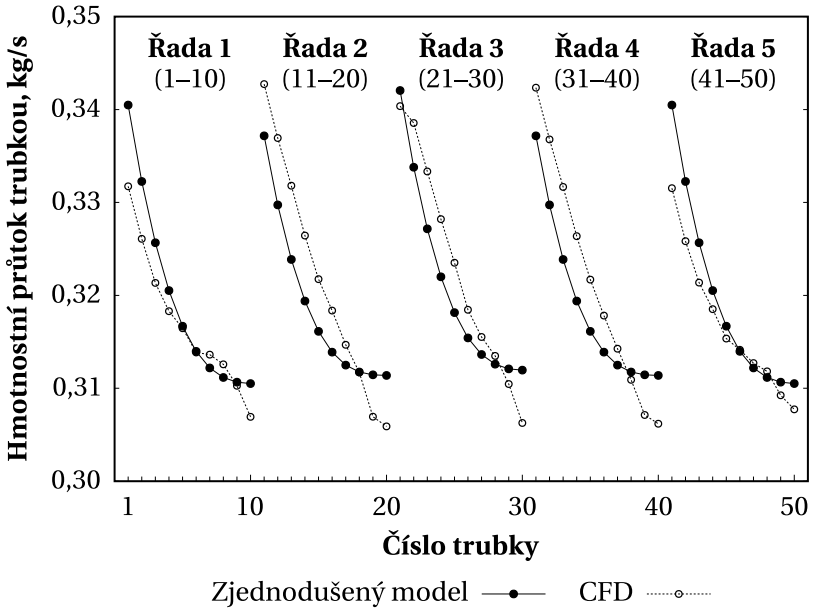
Simulační software nutný k testování modelu využívá funkcionality balíku NumPy [81] pro numerické výpočty v jazyce Python a je stále v rané fázi vývoje. Zatím proto není dostupné ani žádné grafické uživatelské rozhraní. Vizualizace dat je však i přes absenci



Obrázek 3.9. Relativní chyby predikovaných hmotnostních průtoků jednotlivými trubkami distribučních systémů z tabulky 3.1 s uspořádáním toku „U“ vůči datům získaným detailními CFD simulacemi. Celkové hmotnostní průtoky pracovní látky jednotlivými systémy jsou uvedeny v legendě a odpovídají průměrným hodnotám Reynoldsova čísla v trubkách $\mathrm{Re}_t \approx 24\,000$ u systému A, resp. $\mathrm{Re}_t \approx 48\,000$ u ostatních systémů. Číslování trubek je analogické obrázku 3.8 (adaptováno z [A13]).

takového rozhraní snadná. Jakékoliv dvourozměrné grafy lze totiž jednoduše zobrazit (a případně také exportovat do souboru pro další použití) přímo ve vývojovém prostředí The Scientific Python Development Environment (Spyder) [82]. Vizualizaci hodnot vázaných na 3D strukturu modelu je pak možné provést relativně přímočarým exportem dat skrze knihovnu The Visualization Toolkit (VTK) [83] a jejich následným načtením do aplikace Kitware ParaView [84]. Jak navíc můžeme ukázat kupříkladu na horkovodním kotli na odpadní teplo schematicky znázorněném na obrázku 3.11, který je součástí existujícího zařízení pro kombinovanou výrobu tepla a elektrické energie, zmíněná aplikace umožňuje zobrazování dat zároveň z vícero zdrojů. Snadno tedy lze získat třeba obrázek 3.12 s hodnotami týkajícími se nejen modelovaných trubkových svazků, ale současně také s daty odpovídajícími mezitrubkovému prostoru, která byla spočtena pomocí jiného softwaru.

Výpočtové časy pozorované při vyhodnocování testovacích úloh se i při stávající implementaci modelu pohybovaly nejvýše v řádu desítek vteřin. Z pohledu budoucího použití v optimalizačních algoritmech je ale samozřejmě žádoucí tyto časy co možná nejvíce zkrátit. Pokoušet se o to paralelizací vlastního maticového výpočtu (predikčního kroku)

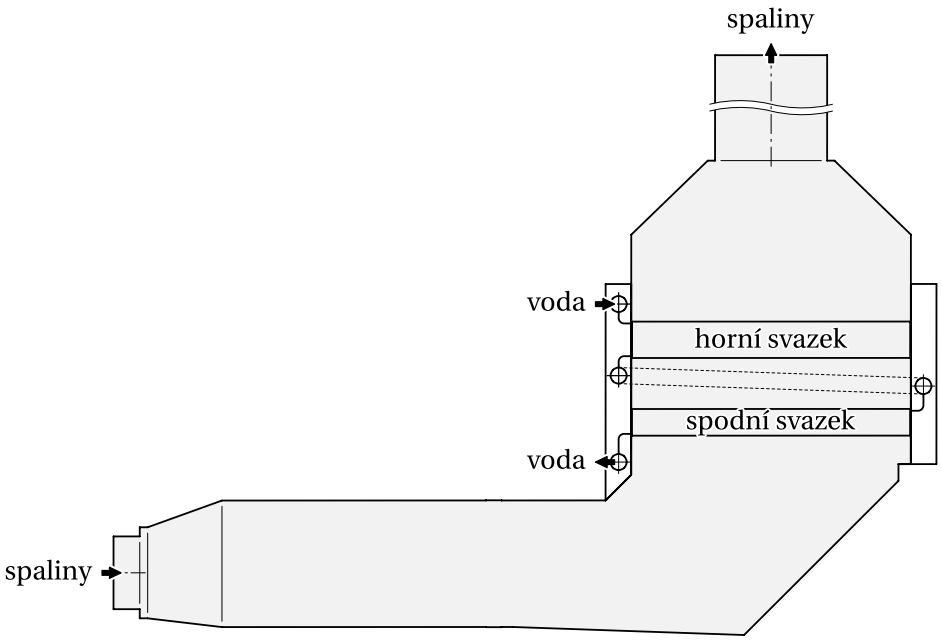


Obrázek 3.10. Srovnání hmotnostních průtoků jednotlivými trubkami distribučního systému z obrázku 3.7, které byly zjištěny zjednodušeným modelem a detailní CFD simulací při průměrné hodnotě Reynoldsova čísla v trubkách $\mathrm{Re}_t \approx 48\,000$; číslování trubek odpovídá obrázku 3.8

ovšem nedává příliš smysl, protože hodnosti matic reprezentujících systémy lineárních rovnic jsou při uvažovaném způsobu zjednodušení výpočetní sítě malé. Naopak u korekčního kroku, který je řešen nezávisle pro každý jednotlivý prvek sítě, už paralelizace smysl dává a nic jí nebrání. Pravděpodobně nejlepší strategií je zde asynchronní dávkové zpracování, tedy řešení korekčních úloh pomocí skupiny asynchronně pracujících výkonných procesů, z nichž každý běží na jiném jádře procesoru. V existující implementaci v jazyce Python k tomu byla využita vestavěná knihovna multiprocessing (nutno však podotknout, že toto vylepšení bylo do kódu zapracováno až po vydání článku [A13]). Vhodnou alternativou by nicméně mohla být také knihovna ray [85], která dle testů zmíněných v článku [86] poskytuje ve srovnání s knihovnou multiprocessing dodatečné 5–25násobné zkrácení výpočetních časů (v závislosti na charakteru paralelizované úlohy).

Režie v podobě procesorového času nutného pro konstrukci, provoz a destrukci[†] výkonných procesů a komunikaci s řídicím procesem rozhodně není zanedbatelná. Vždy je proto lepší na základě dostupného počtu jader odhadnout vhodný maximální počet korekčních úloh odesílaných výkonným procesům v jedné dávce namísto ponechání způsobu rozložení paralelně běžících úloh na použité paralelizační knihovně (aktuálně využitý přístup). Takový odhad přitom lze vyprodukovat vcelku snadno. Jakmile by pak

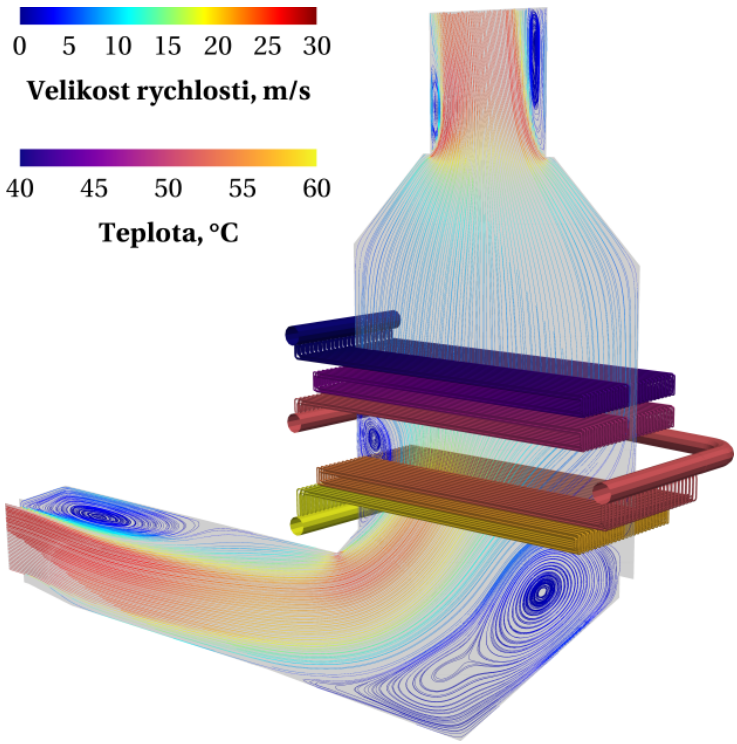
[†]V případě, že není použit programovací jazyk s automatickou správou paměti („garbage collection“).



Obrázek 3.11. Schéma horkovodního kotle na odpadní teplo s vyznačenou polohou modelovaných trubkových svazků; kolektor horního svazku je spojen s distributorem dolního svazku pomocí externího potrubí, které je zakresleno čárkovanou čarou

při výpočtu některý výkonný proces dokončil zpracovávání aktuální dávky korekčních úloh, požádal by řídicí proces o dávku novou, resp. by došlo k jeho ukončení a případné destrukci, pokud by již nebyly dostupné žádné úlohy ke zpracování.

Dalším vylepšením stávajícího počítačového kódu, které však ještě z časových důvodů implementováno nebylo, je použití stromové reprezentace výpočetní sítě namísto standardní kolekce objektů. Pokud bychom docela rozumně a realisticky požadovali, aby jakákoliv souvislá část sítě byla funkčně (ve smyslu rozhraní v objektově orientovaném kódu) identická, měla by uvedená modifikace dvě zásadní výhody. První výhoda by spočívala v možnosti rekurzivního procházení sítě při řešení korekčních úloh. Toto by zcela zřejmě mělo vliv na rychlost dohledávání sousedních elementů a jejich vlastností, resp. také na případnou paralelizaci výpočtu. Druhým a potenciálně významnějším důsledkem stromové reprezentace by pak byla skutečnost, že by řešič pracoval s jakoukoliv souvislou částí sítě – bez ohledu na její velikost či strukturu – úplně stejně jako s jediným prvkem. Vyvinutou aplikaci by tudíž šlo propojit s jinými výpočetními nástroji a zahrnout tak i komplikovanější části geometrie, jejichž modelování stávajícím kódem by nemuselo být dostatečně přesné či proveditelné. K tomu by stačilo vytvořit prvek implementující stejné kódové rozhraní, pouze s tím rozdílem, že vlastní propojení odpovídajících koncových uzlů by nyní bylo realizováno externí výpočetní aplikací (ANSYS Fluent atp.).



Obrázek 3.12. Vizualizace proudnic ve spalinovodu horkovodního kotle na odpadní teplo získaných 2D výpočtem v aplikaci ANSYS Fluent spolu s teplotami média v trubkovém svazku, které byly zjištěny prvotní (a dosud značně zjednodušenou) verzí teplotního modelu. Data odpovídající trubkovému svazku byla exportována pomocí knihovny VTK, zatímco data týkající se rychlostního pole byla ve vizualizační aplikaci Kitware ParaView načtena přímo z nativních datových souborů ANSYS Fluent.

3.4 Budoucí zaměření výzkumu

Jak bylo poznamenáno v předešlém textu, vývoj modelu založeného na metodě konečných prvků je stále v rané fázi. Dosud tudíž není zcela zřejmé, nakolik bude takový model použitelný v praxi. Následný výzkum v této oblasti bude proto zaměřen primárně na rozšíření modelu o nové prvky výpočetní sítě z obrázku 3.5 a na dokončení již započaté úpravy iteračního řešiče tak, aby zohledňoval i přenos tepla. Kromě toho budou testovány možnosti přibližného a zároveň výpočtově „levného“ zahrnutí vlivu turbulence. Na základě výsledků obdržených se zjednodušenými 3D CFD modely z kapitoly 2 lze očekávat, že by mohlo opět dostačovat škálování viskozity pomocí vhodného empirického vztahu, nicméně bez řádného otestování není možné v tomto ohledu činit žádné konkrétnější závěry.

Ukáže-li se patřičný způsob zjednodušeného modelování opravdu po úpravách a testech zmíněných výše jako vhodný, bude pozornost zaměřena na rozšíření modelu o me-

chanické namáhání svazku. Další kroky se pak budou týkat vlastní počítačové implementace. Nejprve bude provedena změna způsobu reprezentace výpočetní sítě do stromového tvaru a související zvýšení efektivity paralelizace korekčního kroku. Poté bude přidána možnost propojení s jinými výpočetními kódy – ať už formou API, nebo vytvořením rutin schopných (i) exportovat data do a importovat data ze souborů ve vhodných datových formátech a (ii) spouštět odpovídající externí výpočetní aplikace. Nakonec bude zkoumán vliv případných mikrooptimalizací kódu za účelem dodatečného zkrácení výpočetních časů.

Kapitola

4

Numerická disipace v CFD modelech

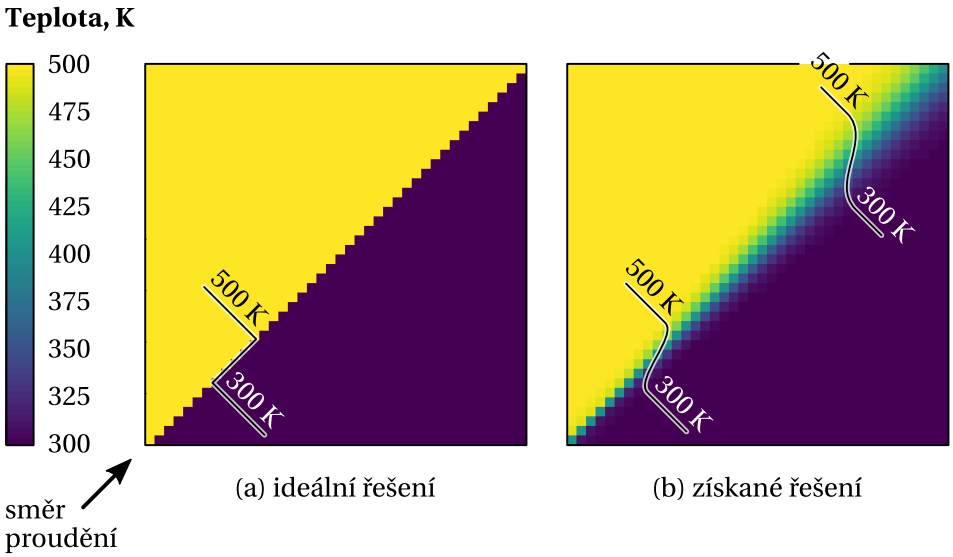
Při modelování proudění tekutin se vlivem obrovské výpočtové náročnosti přímé numerické simulace („Direct Numerical Simulation“, DNS) [87] prakticky využívají převážně simulace velkých vírů („Large Eddy Simulation“, LES) [88], různé adaptivní metody – např. „Detached Eddy Simulation“ (DES) [89] nebo „Scale-Adaptive Simulation“ (SAS) [90] –, resp. modelování pomocí časově průměrovaných Navierových-Stokesových rovnic („(Unsteady) Reynolds-Averaged Navier-Stokes equations“, (U)RANS). U posledního zmíněného přístupu, který je přesný nejméně, jsou hlavním zdrojem chyb použité modely turbulence [91]. (U)RANS modely ale přesto v inženýrské praxi zcela převažují, neboť jejich nároky na výpočetní hardware jsou relativně malé.

Numerická disipace ve zmíněných modelech má původ v diskretizaci použitých rovnic a nezanedbatelnou měrou ovlivňuje přesnost výsledných dat. Chyba takto vnesená do výpočtu je přitom zpravidla významná bez ohledu na to, jak vysokého řádu – tj. jaké formální přesnosti – jsou použité numerické metody. Odhad míry numerické disipace je ovšem zcela logicky nejvíce potřeba u komerčně používaných CFD modelů pracujících s metodami nízkých řádů. Je také nasnadě, že výzkum nových diskretizačních schémat minimalizujících numerickou disipaci je v oblasti CFD výpočtů jedním z aktuálních témat [92].

Hlavním projevem numerické disipace je tzv. numerická viskozita („numerical viscosity“), která je důsledkem faktu, že se uvnitř kontrolních objemů uvažují konstantní hodnoty veličin, tedy že řešení získané CFD modelem je po částech konstantní. Vý-

znamný vliv zde ale má i to, zda je výpočetní síť „zarovnána“ s vlastním prouděním. Patřičnou situaci lze snadno ilustrovat například následujícím experimentem: uvažujme dvourozměrnou doménu popsanou pravoúhlou sítí (40×40 buněk), ve které dochází k laminárnímu proudění ve směru rovnoběžném s diagonálou (z pohledu „zarovnání“ sítě jde o nejhorší možný případ). Uvažujme dále, že na vstupu je v okrajové podmínce v místě diagonály nespojitost v určité skalární proměnné – například teplotě. Pokud bychom nyní (byť poněkud nefyzikálně) položili odpovídající difuzní koeficient rovný nule, mělo by i řešení úlohy obsahovat zmíněnou nespojitost, jak ukazuje obrázek 4.1a. Ve skutečnosti však získáme řešení podobné tomu z obrázku 4.1b, kde je na první pohled patrné, že nespojitost byla nahrazena hladkým přechodem, který se ve směru proudění postupně rozšiřuje. I kdybychom v okolí rozhraní použili jemnější síť, problém by nezmizel; pouze by byl přechodový pás díky menším buňkám užší.

Výzkum metod pro odhad chyb vzniklých v důsledku diskretizace rozhodně není ničím novým, jak naznačuje článek Marchiho a da Silvy [93] již z roku 2002. Často (viz třeba publikace [94]) je jako modelový případ zkoumán Taylorův-Greenův vír, pro který existuje exaktní analytické řešení, případně jiné víry, u nichž jsou k dispozici potřebné údaje (jako příklad lze uvést studii [95]). Snadno ale můžeme nalézt mnoho dalších studií zaměřujících se na numerickou disipaci v LES [96] a implicitních LES (ILES) [97] modelech, ILES využívajících adaptivní filtrování [98], různě zjednodušených formách DNS [99] nebo dokonce u zcela obecných parciálních diferenciálních rovnic [100]. Kromě toho se čtenář může setkat i s články zabývajícími se příbuznými tématy; kupříkladu odhadem zaokrouhlovacích chyb [101], aplikací Richardsonovy extrapolace pro odhad řešení nezá-



(a) ideální řešení (b) získané řešení

Obrázek 4.1. Vliv numerické viskozity na řešení úlohy, v níž byl použit nulový difuzní koeficient

vislého na výpočetní síti [102], metodami určení nutného prostorového rozlišení sítí v LES modelech [103], explicitním filtrováním v těchto modelech pro predikci skalárních veličin bez ovlivnění sítí a řádem diskretizačního schématu [104], konstrukcí sítě zohledňující skutečný charakter proudění za účelem snížení výsledné chyby [105] a podobně.

Značnou nevýhodou mnoha způsobů odhadu míry numerické disipace nicméně je, že vyžadují přístup do vlastního CFD řešiče, což zcela zřejmě významně omezuje jejich praktickou aplikaci. Naproti tomu metoda popsaná ve výše zmíněném článku Schrannera a kol. [94] (resp. v publikaci [106], která poskytuje některé dodatečné informace ohledně implementace patřičného postupu) považuje CFD řešič za „černou skříňku“, odkud pouze přebírá vybrané interně počítané údaje. Vyhodnocování míry numerické disipace je pak prováděno až na základě těchto údajů pomocí dvou hlavních ukazatelů – rezidua diskretizované integrální rovnice pro kinetickou energii, ϵ^{n}, a numerické kinematické viskozity, ν^{n}. Postup odvození užitých vztahů může čtenář snadno najít v citovaných článcích ([94, 106]) a nemá proto smysl uvádět ho v této práci.

Motivací pro implementaci zde popisované metody přitom je primárně snaha odhadnout chybu v datech z CFD výpočtů, případně – což by bylo ještě lepší – se pokusit najít závislost mezi indikátory míry numerické disipace (nebo i některých k tomu potřebných pomocných údajů) a přesným řešením. V publikaci [94] jsou sice výsledky obdržené diskutovaným postupem v případě Taylorova-Greenova víru porovnány s přesnými údaji získanými užitím spektrálních metod, přesto by však bylo vhodné provést srovnání také s daty pro v procesní praxi běžnější situace – například vířivé proudění v trubkách [107].

Reziduum diskretizované integrální rovnice pro kinetickou energii je pro určitý kontrolní objem V – buňku výpočetní sítě – počítáno vztahem

$$\epsilon_V^{\mathrm{n}} = -\left(\frac{\Delta E_V^{\mathrm{kin}}}{\Delta t}\right) - F_V^{\mathrm{kin}} - F_V^{\mathrm{ac}} + F_V^{\mathrm{vis}} + W_V^{\mathrm{p}} - \epsilon_V^{\mathrm{vis}}. \tag{4.1}$$

První člen na pravé straně rovnice 4.1, který představuje časovou změnu kinetické energie, je zde zjišťován tříbodovou diferenční formulí druhého řádu (detaily lze dohledat v článku [106]). Odpovídající hodnoty kinetické energie jsou pak v jednotlivých po sobě jdoucích časových krocích počítány pomocí

$$E_V^{\mathrm{kin}} = \left[\rho e^{\mathrm{kin}}\right]_V \Delta V, \tag{4.2}$$

v čemž zápis $[\cdot]_V$ znamená „v kontrolním objemu V“, ρ značí hustotu pracovní látky, e^{kin} měrnou kinetickou energii danou vztahem

$$e^{\mathrm{kin}} = \frac{1}{2}\sum_{i=1}^{3} u_i^2, \tag{4.3}$$

u_i $i$tou složku vektoru rychlosti proudění $\mathbf{u} = (u_1, u_2, u_3)$ a ΔV objem daného kontrolního objemu (buňky výpočetní sítě). Celkový tok kinetické energie („kinetic energy flux“)

skrze stěny A tvořící hranici ∂V kontrolního objemu V lze vyjádřit ve tvaru

$$F_V^{\text{kin}} = \sum_{A \in \partial V} \left[\sum_{i=1}^{3} n_i \left(\rho e^{\text{kin}} u_i \right) \right]_A \Delta A, \tag{4.4}$$

kde zápis $[\cdot]_A$ analogicky znamená „na stěně A“, n_i je $i$tá složka normálového vektoru $\mathbf{n} = (n_1, n_2, n_3)$ stěny A a ΔA plocha této stěny. Přítomnost složek normálového vektoru zde nepředstavuje žádný problém, neboť CFD řešiče musí vyhodnocovat stěnové toky a informace o velikostech a orientacích stěn proto bývají běžně k dispozici. Například v softwaru ANSYS Fluent je definován tzv. vektor plochy stěny („face area vector“, makro `F_AREA`), který směrem a orientací odpovídá normálovému vektoru dané stěny a velikostí potom vlastní ploše stěny. Podobně se počítají i celkový akustický tok („acoustic flux“),

$$F_V^{\text{ac}} = \sum_{A \in \partial V} \left[\sum_{i=1}^{3} n_i \left(p u_i \right) \right]_A \Delta A, \tag{4.5}$$

kde p značí tlak, a celkový viskózní tok („viscous flux“),

$$F_V^{\text{vis}} = \sum_{A \in \partial V} \left[\sum_{i=1}^{3} n_i \sum_{j=1}^{3} \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \delta_{ij} \frac{2}{3} \sum_{k=1}^{3} \frac{\partial u_k}{\partial x_k} \right) u_j \right]_A \Delta A, \tag{4.6}$$

kde μ představuje efektivní dynamickou viskozitu a δ_{ij} Kroneckerovo delta. Práce tlakových sil („pressure work“) je vyjádřena vztahem

$$W_V^{\text{p}} = \left[p \sum_{i=1}^{3} \frac{\partial u_i}{\partial x_i} \right]_V \Delta V. \tag{4.7}$$

Posledním členem v rovnici 4.1 pak je viskózní disipace („viscous dissipation“),

$$\epsilon_V^{\text{vis}} = [\nu]_V \, \epsilon_V, \tag{4.8}$$

ve které ν značí efektivní kinematickou viskozitu a ϵ_V disipační funkci,

$$\epsilon_V = \left[\rho \left(\sum_{i=1}^{3} \sum_{j=1}^{3} \left(\left(\frac{\partial u_i}{\partial x_j} \right)^2 + \frac{\partial u_i}{\partial x_j} \frac{\partial u_j}{\partial x_i} - \delta_{ij} \frac{2}{3} \frac{\partial u_i}{\partial x_j} \sum_{k=1}^{3} \frac{\partial u_k}{\partial x_k} \right) \right) \right]_V \Delta V. \tag{4.9}$$

Druhý ukazatel – numerická kinematická viskozita – je podílem hodnoty rezidua diskretizované integrální rovnice pro kinetickou energii (z rovnice 4.1) a hodnoty disipační funkce (z rovnice 4.9):

$$\nu_V^{\text{n}} = \frac{\epsilon_V^{\text{n}}}{\epsilon_V}. \tag{4.10}$$

Samozřejmě zde nejde o skutečnou hodnotu kinematické viskozity, ale o míru numerické difuze, tj. opět chybu vnesenou do výpočtu v důsledku užití některých druhů diskretizačních schémat.

Celkové hodnoty veličin výše za určitou subdoménu D se pak zjistí prostým součtem hodnot ze všech odpovídajících kontrolních objemů V:

$$\begin{aligned} \epsilon_D^{\mathrm{n}} &= \sum_{V \in D} \epsilon_V^{\mathrm{n}}, \quad \text{resp.} \\ \nu_D^{\mathrm{n}} &= \sum_{V \in D} \nu_V^{\mathrm{n}}. \end{aligned} \tag{4.11}$$

Výpočet obou ukazatelů míry numerické disipace popsaných výše, tedy rezidua diskretizované integrální rovnice pro kinetickou energii a numerické kinematické viskozity, byl implementován v jazyce C formou uživatelsky definovaných funkcí („User-Defined Functions“, UDF). Tyto lze snadno použít při výpočtech v aplikaci ANSYS Fluent, k čemuž není potřeba žádná znalost programování – bohatě dostačuje uživatelská znalost zmíněné aplikace.

4.1 Struktura kódu UDF

Kód nutný k získání odhadu míry numerické disipace je napsán tak, aby mohl být použit v jednojádrových i vícejádrových výpočtech, a skládá se z několika oddělených funkcí. První funkce je spouštěna automaticky při načtení zkompilované knihovny a slouží k rezervování potřebného počtu uživatelských proměnných a inicializaci vybraných interních datových struktur, které nezávisí na výpočetní síti a jsou proto deklarovány jako proměnné nativních datových typů jazyka C. Druhá funkce je výkonná a provádí vlastní výpočet údajů, na základě kterých lze posuzovat míru numerické disipace, resp. některé další související operace. Kromě toho pak kód obsahuje také funkce pro

- uživatelskou aktivaci či deaktivaci vybraných funkcionalit,
- výpis aktuálních významů jednotlivých indexů uživatelských proměnných (jelikož, jak bude vysvětleno dále, se některé indexy musí v zájmu efektivity kódu v průběhu výpočtu měnit),
- nové započetí procesu hodnocení míry numerické disipace (tj. zahození veškerých aktuálních dílčích či souhrnných hodnot a údajů uložených v pomocných datových strukturách) a
- výpis informací o jednotkách všech počítaných hodnot (neboť jejich uvádění ve standardních výpisech by vedlo k výraznému snížení přehlednosti).

Veškeré tyto funkce jsou podrobněji popsány v následujících kapitolách.

4.1.1 Funkce on_loading

Funkce on_loading je – jak její název napovídá – spuštěna automaticky při načtení zkompilované knihovny. Jejím primárním účelem je rezervování potřebného počtu uživatelských proměnných („User-Defined Memory“, UDM), které budou následně využívány při vyhodnocování numerické disipace. Nutno však podotknout, že tato operace, tedy zavolání funkce Reserve_User_Memory_Vars(num), někdy při automatickém provedení selže, na což upozorňuje i oficiální příručka aplikace ANSYS Fluent [108, kap. 3.2.12.7]. Vždy je proto kódem kontrolováno, zda bylo rezervování proměnných úspěšné, a případně je v konzole aplikace zobrazeno varování spolu s informací, kolik proměnných musí uživatel před pokračováním manuálně rezervovat, např.:

```
> BEWARE: (1) YOU NEED TO HAVE RESERVED 28 UDM(s) BEFORE
  ↪ USING libudf.
          (2) UDM OFFSET IS FIXED AT 0!
```

Ukládání většiny proměnných je potřeba provádět v relaci k současnému, minulému a někdy i předminulému časovému kroku. Jelikož by ale užívání fixních indexů proměnných vyžadovalo na konci každého časového kroku kopírování obrovských množství dat, je namísto toho u každé dotčené proměnné pracováno se „základním“ indexem a variabilními (průběžně „rotovanými“) indexovými posuny odpovídajícími těmto třem časovým krokům. Funkce on_loading tedy obsahuje inicializaci základních indexů, přičemž výchozí indexové posuny jsou nastaveny rovnou při deklaraci patřičných tří statických proměnných typu int. Kromě toho jsou inicializovány také indexy proměnných pro ukládání průměrů hodnot počítaných v každém časovém kroku, nicméně tyto již nejsou vázány na konkrétní časový krok a není u nich tudíž nutné je dále korigovat.

Nakonec jsou do konzoly vypsány informace ohledně aktuálního nastavení průměrování hodnot, detailního výpisu do konzoly a zápisu reziduí diskretizované integrální rovnice pro kinetickou energii a numerických kinematických viskozit ze všech jednotlivých buněk na disk. Tato nastavení jsou uživatelsky měnitelná funkcemi toggle_averaging, toggle_detailed_printout a toggle_celldata_output, jak je podrobněji posáno v kapitolách 4.1.3 až 4.1.5.

4.1.2 Funkce numerical_dissipation

Funkce numerical_dissipation tvoří stěžejní část kódu UDF, neboť provádí vlastní výpočet dílčích a souhrnných hodnot týkajících se odhadu míry numerické disipace. Získaná data jsou touto funkcí také ukládána do uživatelských proměnných pro potřeby vizualizace či dalšího zpracování, souhrnné statistiky za celou doménu jsou zapisovány na disk a jsou dopočítávány průměrné hodnoty (je-li to uživatelem požadováno).

Hlavní část kódu, tj. zjišťování všech dílčích hodnot pro jednotlivé buňky sítě, je prováděna pouze na výpočetních uzlech („serial node“, resp. „non-host nodes“ v případě paralelního výpočtu). Pokud již jsou k dispozici údaje z alespoň tří po sobě jdoucích časových kroků, dopočítávají se pro jednotlivé buňky i reziduum diskretizované integrální

rovnice pro kinetickou energii a numerická kinematická viskozita. Jinak je do dotčených proměnných uložena hodnota NaN. Následně jsou získány průměry všech „buňkových“ hodnot, je-li toto požadováno uživatelem. Vlastní průměrování je přitom prováděno inkrementálně, což nevyžaduje jinak zcela zbytečné uchovávání kompletní historie všech počítaných údajů. To znamená, že namísto výpočtu aritmetického průměru ze sady hodnot $\{a_1, a_2, \ldots, a_N\}$ pomocí obvyklého vztahu

$$\overline{a}_N = \frac{1}{N} \sum_{i=1}^{N} a_i \tag{4.12}$$

je v prvním časovém kroku po aktivaci průměrování použit triviální vztah $\overline{a}_1 = a_1$, zatímco v každém dalším časovém kroku je průměr počítán pomocí

$$\overline{a}_N = \frac{(N-1)\,\overline{a}_{N-1} + a_N}{N} = \overline{a}_{N-1} + \frac{a_N - \overline{a}_{N-1}}{N}. \tag{4.13}$$

V kódu je pak implementován upravený výraz na konci rovnice 4.13 výše, neboť se tak v porovnání s prvním uvedeným – intuitivním – zápisem sníží zaokrouhlovací chyba.

Poslední operací prováděnou na výpočetních uzlech je předání veškerých dat nutných ke zjištění souhrnných hodnot za celou doménu na hlavní uzel pro finální zpracování a zápis na disk.

Naopak na hlavním uzlu („serial node“, resp. „host node“) je nejprve proveden příjem dat z výpočetních uzlů. Pomocí nich je pak dopočteno souhrnné reziduum diskretizované integrální rovnice pro kinetickou energii za celou doménu a odpovídající souhrnná hodnota numerické kinematické viskozity. Tyto dvě statistiky včetně veškerých dalších souhrnných hodnot za celou doménu jsou pro stávající časový krok zapsány na disk. V závislosti na uživatelském nastavení jsou případně do jiného souboru uložena také rezidua z jednotlivých buněk (resp. jejich průměry, je-li aktivní průměrování). Nakonec jsou do konzoly aplikace ANSYS Fluent vypsány zmíněné dvě nejdůležitější statistiky za předchozí[†] časový krok (případně i ostatní souhrnné údaje – opět dle aktuálního nastavení), což může vypadat kupříkladu takto:

```
> OVERALL DISSIPATION VALUES FOR THE PREVIOUS TIME STEP (flow
  ↪  time: 0.42 s):
    EPSn =  4.95007e-01
    NUn  =  2.80716e-03
```

[†]Vzhledem k výpočtu časové změny kinetické energie v rovnici 4.1 pomocí tříbodové diferenční formule je hodnota ϵ^{n} dostupná nejvýše pro předchozí časový krok. Hodnotu ν^{n} jsme tudíž schopni spočítat také nejvýše pro předchozí časový krok. Zbylé hodnoty (F^{kin} apod.) sice známe i pro stávající časový krok, nicméně v zájmu konzistence vypisovaných údajů jsou i tyto uváděny pro krok předchozí.

4.1.3 Funkce toggle_averaging

Hodnoty, ať už dílčí pro tu kterou buňku sítě nebo souhrnné za celou doménu, počítané pro konkrétní časový krok nemusí být vzhledem k možným fluktuacím příliš vypovídající. V takovém případě je lepší sledovat jejich průměry. Tyto ovšem ve výchozím nastavení počítány nejsou a je proto nutné jejich výpočet aktivovat pomocí na vyžádání spustitelné („on demand“) funkce toggle_averaging.

Uvedená funkce je spouštěna pouze na hlavním uzlu, neboť právě tam dochází k případnému výpočtu průměrů. Aktivace či deaktivace (v závislosti na stavu před zavoláním funkce) patřičné funkcionality je indikována výpisem do konzoly, který může vypadat například takto:

```
> Computing of average values has been enabled.
```

Po aktivaci průměrování jsou do konzoly během výpočtu vypisovány kromě rezidua diskretizované integrální rovnice pro kinetickou energii a numerické kinematické viskozity (jsou-li již k dispozici) také jejich průměrné hodnoty, a to v následujícím formátu:

```
> OVERALL DISSIPATION VALUES FOR THE PREVIOUS TIME STEP (flow
  ↪  time: 0.42 s):
    EPSn =  4.95007e-01 (Avg(EPSn) =  4.89059e-01)
    NUn  =  2.80716e-03 (Avg(NUn)  =  2.83756e-03)
```

Veškeré „buňkové“ průměrné hodnoty jsou pro případnou analýzu či vizualizaci přístupné skrze rezervované uživatelské proměnné. Zjištění indexu proměnné, která obsahuje požadovanou průměrnou hodnotu, lze provést na vyžádání spustitelnou funkcí list_udm_indices popsanou v kapitole 4.1.6.

4.1.4 Funkce toggle_detailed_printout

Standardně jsou v každém časovém kroku do konzoly vypsány pouze dvě souhrnné hodnoty (jsou-li již k dispozici), a sice reziduum diskretizované integrální rovnice pro kinetickou energii a numerická kinematická viskozita. Tyto, jak bylo poznamenáno výše, odpovídají předchozímu časovému kroku $t - \Delta t$. Kromě nich ale mohou být pro uživatele někdy zajímavé i další spočtené hodnoty pro tentýž časový krok, jejichž výpis však je ve výchozím nastavení zakázán.

Aktivaci výpisu dodatečných hodnot lze provést na vyžádání spustitelnou funkcí toggle_detailed_printout. Tato je opět spouštěna pouze na hlavním uzlu, jelikož z něj je prováděn výpis do konzoly. Aktivace či deaktivace (v závislosti na stavu před zavoláním funkce) pak je potvrzena podobně jako u funkce toggle_averaging, např.:

```
> Detailed printout has been enabled.
```

Je-li tedy vypisování dodatečných hodnot povoleno, získá uživatel na konci časového kroku namísto obvyklé stručné informace výrazně podrobnější zprávu, která uvádí i všechny ostatní veličiny vystupující v rovnicích 4.1 a 4.10:

```
> OVERALL DISSIPATION VALUES FOR THE PREVIOUS TIME STEP (flow
  ↪ time: 0.42 s):
    EPSn =  4.95007e-01
    NUn  =  2.80716e-03
> Overall intermediate dissipation values:
    Ekin     =  2.32786e-03
    dEkin/dt =  4.87339e-03
    Fkin     = -8.15009e-01
    Fac      =  3.16270e-01
    Fvis     =  4.29578e-04
    Wp       =  2.21763e-03
    epsvis   =  1.50579e-03
    eps      =  1.76337e+02
```

Slovem „overall“ se zde rozumí, že jde o součty patřičných hodnot ze všech buněk v celé výpočetní doméně (v kontrastu s hodnotami pro jednotlivé buňky, které jsou přístupné skrze rezervované uživatelské proměnné).

Pro úplnost ještě dodejme, že uvádění jednotek hodnot ve výpisech výše by zcela zřejmě snižovalo jejich přehlednost. Lze si je proto nechat zobrazit na vyžádání spustitelnou funkcí list_units, jak je popsáno v kapitole 4.1.7.

4.1.5 Funkce toggle_celldata_output

Funkce numerical_dissipation může pro potřeby případného dalšího zpracování v každém časovém kroku ukládat na disk rezidua diskretizované integrální rovnice pro kinetickou energii a numerické kinematické viskozity (příp. jejich průměrné hodnoty, je-li aktivováno průměrování) ze všech buněk v doméně. Výpočetní sítě CFD modelů ovšem běžně bývají značně rozsáhlé a častý zápis větších objemů dat by tudíž mohl vést k výraznému zpomalení výpočtu. Zmíněná funkcionalita je proto ve výchozím stavu deaktivována, nicméně ji lze aktivovat (resp. dle potřeby zase deaktivovat) pomocí na vyžádání spustitelné funkce toggle_celldata_output. Stejně jako ostatní podobné funkce je i tato spouštěna pouze na hlavním uzlu. Povolení či zakázání zápisu na disk (v závislosti na stavu před zavoláním funkce) je pak jako obvykle potvrzeno výpisem do konzoly, např.:

```
> Writing of cell data to disk has been disabled.
```

4.1.6 Funkce list_udm_indices

Indexy všech uživatelských proměnných, u kterých je to možné, zůstávají v průběhu výpočtu neměnné. Část dílčích hodnot ovšem musí být k dispozici ze dvou či tří po sobě

jdoucích časových kroků a v zájmu zachování výpočetní efektivity je proto vhodné namísto kopírování dat mezi jednotlivými uživatelskými proměnnými pouze průběžně „rotovat“ odpovídající indexy.

Pokud by tedy uživatel chtěl analyzovat či vizualizovat některou veličinu, musí vědět, který index uživatelské proměnné u ní aktuálně odpovídá kterému časovému kroku. K tomu slouží na vyžádání spustitelná funkce list_udm_indices, která z hlavního uzlu do konzoly aplikace ANSYS Fluent vypíše všechny používané proměnné včetně jejich významů a specifikace, ke kterému časovému kroku se váží. Výpis pak může vypadat například takto:

```
> UDM INDICES (BEWARE: MOST OF THESE CHANGE WITH EACH
  ↪  TIME STEP!):
    User-Defined Memory  0 ... EPSn @ t-dt
    User-Defined Memory  1 ... NUn @ t-dt
    User-Defined Memory  2 ... dEkin/dt @ t-dt
    User-Defined Memory  5 ... Ekin @ t-2dt
    User-Defined Memory  3 ... Ekin @ t-dt
    User-Defined Memory  4 ... Ekin @ t
    User-Defined Memory  7 ... Fkin @ t-dt
    User-Defined Memory  6 ... Fkin @ t
    User-Defined Memory  9 ... Fac @ t-dt
    User-Defined Memory  8 ... Fac @ t
    User-Defined Memory 11 ... Fvis @ t-dt
    User-Defined Memory 10 ... Fvis @ t
    User-Defined Memory 13 ... Wp @ t-dt
    User-Defined Memory 12 ... Wp @ t
    User-Defined Memory 15 ... epsvis @ t-dt
    User-Defined Memory 14 ... epsvis @ t
    User-Defined Memory 17 ... eps @ t-dt
    User-Defined Memory 16 ... eps @ t
    User-Defined Memory 18 ... Avg(EPSn) @ t-dt
    User-Defined Memory 19 ... Avg(NUn) @ t-dt
    User-Defined Memory 20 ... Avg(dEkin/dt) @ t-dt
    User-Defined Memory 21 ... Avg(Ekin) @ t-dt
    User-Defined Memory 22 ... Avg(Fkin) @ t-dt
    User-Defined Memory 23 ... Avg(Fac) @ t-dt
    User-Defined Memory 24 ... Avg(Fvis) @ t-dt
    User-Defined Memory 25 ... Avg(Wp) @ t-dt
    User-Defined Memory 26 ... Avg(epsvis) @ t-dt
    User-Defined Memory 27 ... Avg(eps) @ t-dt
```

Z výpisu je patrné, že reziduum diskretizované integrální rovnice pro kinetickou energii (zde označeno EPSn) a numerická kinematická viskozita (NUn), odpovídající předchozímu časovému kroku (t−dt), jsou vždy uloženy v první a druhé uživatelské

proměnné s indexy 0 a 1[†]. Stejně tak se nemění ani indexy časové změny kinetické energie (dEkin/dt) či dosavadních průměrů všech počítaných hodnot (Avg(·)), které jsou ukládány pouze pro jediný časový krok.

4.1.7 Funkce list_units

Účelem na vyžádání spustitelné funkce list_units je vypsat jednotky všech počítaných veličin, přičemž vlastní výpis je prováděn pouze z hlavního uzlu. Výstup funkce do konzoly tedy vypadá následovně:

```
> UNITS:
    EPSn     ... kg m^2 s^-3
    NUn      ... m^2 s^-1
    Ekin     ... kg m^2 s^-2
    dEkin/dt ... kg m^2 s^-3
    Fkin     ... kg m^2 s^-3
    Fac      ... kg m^2 s^-3
    Fvis     ... kg m^2 s^-3
    Wp       ... kg m^2 s^-3
    epsvis   ... kg m^2 s^-3
    eps      ... kg s^-2
```

4.1.8 Funkce reset_data

V určitých situacích může být potřeba zahodit veškeré dosud spočtené dílčí hodnoty, jejich průměry a podobně. Toto lze provést na vyžádání spustitelnou funkcí reset_data, která na výpočetních uzlech nastaví hodnoty všech

- interních součtů odpovídajících celé výpočetní doméně, které jsou uloženy v polích typu real (tzn. float nebo double podle toho, s jakou přesností právě ANSYS Fluent pracuje), na nulu,
- souhrnných průměrů typu real odpovídajících celé výpočetní doméně na NaN a
- rezervovaných uživatelských proměnných v každé buňce výpočetní sítě na NaN.

Důvodem pro nastavování vybraných proměnných na NaN namísto nuly je skutečnost, že hodnoty NaN jsou při vizualizaci ignorovány. Tímto způsobem je pak *de facto* indikováno, že patřičné hodnoty nejsou dosud k dispozici. Pokud se totiž uživatel pokusí některou takovou proměnnou vizualizovat, získá prázdný graf, nikoliv graf nulových hodnot, který by mohl být mylně považován za indikátor nízké míry numerické disipace.

[†]Přesněji v proměnných s indexy udm_offset + 0 a udm_offset + 1, kde základní indexový posun udm_offset je zjištěn z návratové hodnoty funkce provádějící rezervování uživatelských proměnných:

```
udm_offset = Reserve_User_Memory_Vars(num);
```

Provedení výše popsané operace je nakonec každým výpočetním uzlem potvrzeno výpisem do konzoly. Při paralelním výpočtu například na čtyřech jádrech procesoru by tudíž výpis vypadal takto:

```
> [Node 0] Intermediate data have been re-set.
> [Node 1] Intermediate data have been re-set.
> [Node 2] Intermediate data have been re-set.
> [Node 3] Intermediate data have been re-set.
```

4.2 Použití UDF

Nejjednodušším způsobem, jak u CFD modelu v aplikaci ANSYS Fluent načíst a aktivovat funkcionality z výše popsaného kódu, je učinit tak pomocí žurnálu[†]. Tento může obsahovat například následující příkazy:

```
/define/user-defined/compiled-functions/compile libudf yes y
  ↪ numdis.c
␣
␣
/define/user-defined/compiled-functions/load libudf
/define/user-defined/function-hooks/execute-at-end
  ↪ numerical_dissipation::libudf
␣
```

Za pozornost zde stojí prázdné řádky[‡], které jsou vyznačeny symbolem „␣“. Jelikož kód UDF může být rozdělen do vícero souborů a může také vyžadovat vlastní hlavičkové soubory, ptá se funkce […]/compile postupně na jejich jednotlivá jména. Zadání každého z nich je pak potvrzeno klávesou Enter, přičemž úplnost seznamu souborů se zdrojovým kódem, resp. hlavičkových souborů, je indikována zadáním prázdného vstupu. Úplně prvním řádkem žurnálu tedy aplikaci ANSYS Fluent poskytneme veškeré informace ohledně názvu výsledné knihovny a zdrojových souborů, načež dalším řádkem zdánlivě bez příkazů potvrdíme, že kromě dříve uvedeného souboru numdis.c není uživatelský zdrojový kód v žádném jiném souboru. Následující řádek, opět zdánlivě bez příkazů, potvrzuje, že nezadáváme žádný vlastní hlavičkový soubor a lze tedy začít s kompilací kódu. Pak už je pouze načtena nově zkompilovaná knihovna libudf a je aktivováno automatické spouštění v ní obsažené funkce numerical_dissipation na konci každého časového kroku. I zde však je možné zadat více než jednu funkci a je proto potřeba potvrdit kompletnost seznamu funkcí zadáním prázdného vstupu (tedy prázdným řádkem na konci žurnálu).

[†] Žurnál („journal“) je soubor obsahující příkazy textového rozhraní („Text User Interface“, TUI) aplikace ANSYS Fluent a zpravidla má příponu jou. Nejde tedy o nic jiného než o analogii shellových skriptů (.sh) známých z unixových operačních systémů nebo dávkových souborů (.bat) z operačních systémů DOS či Microsoft Windows.

[‡] Důsledně vzato jde o řádky obsahující pouze bílé znaky.

Funkce on_loading, která je automaticky spuštěna při načtení knihovny, provádí rezervování uživatelských proměnných. Jak ovšem bylo uvedeno v kapitole 4.1.1, úspěšnost této operace nelze při volání z kódu UDF garantovat. Pokud se tedy po otevření žurnálu výše objeví v konzoli patřičné varování ohledně nutnosti rezervovat uživatelské proměnné ručně, lze toto provést příkazem

```
/define/user-defined/user-defined-memory 28
```

Číslo „28“ zde značí potřebný počet proměnných k rezervování, který je vždy specifikován v odpovídajícím varování v konzoli aplikace ANSYS Fluent.

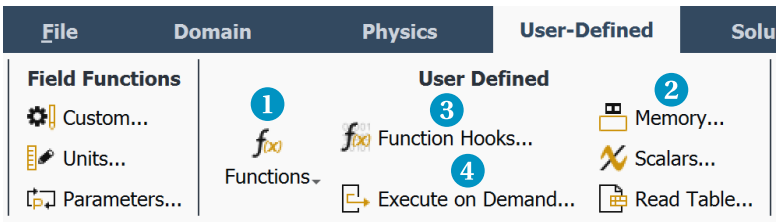
Alternativně lze kompilaci UDF, její načtení, případné rezervování uživatelských proměnných a aktivaci automatického spouštění funkce numerical_dissipation na konci každého časového kroku provést v grafickém uživatelském rozhraní. Veškerá potřebná dialogová okna jsou dostupná skrze prvky v oddílu User Defined na záložce User-Defined pásu karet (viz obrázek 4.2), konkrétně:

- Functions → Compiled...: kompilace zdrojového kódu a načtení vytvořené knihovny (příp. Functions → Manage... pro samotné načtení dříve zkompilované knihovny; tlačítko 1 na obrázku 4.2),
- Memory...: rezervování uživatelských proměnných (2) a
- Function Hooks...: aktivace automatického spouštění funkce numerical_dissipation na konci časového kroku (3).

K funkcím spustitelným na vyžádání pak lze přistupovat skrze tlačítko Execute on Demand... (tlačítko 4 na obrázku 4.2), resp. přes textové rozhraní. Pokud bychom takto chtěli zavolat například funkci toggle_averaging z aktuálně načtené knihovny libudf, provedli bychom to příkazem

```
/define/user-defined/execute-on-demand toggle_averaging::libudf
```

Po načtení knihovny a aktivování automatického volání funkce numerical_dissipation na konci každého časového kroku lze standardním způsobem spustit iterační výpočet. Informace týkající se průběžně počítaných hodnot se budou objevovat v konzoli a zároveň



Obrázek 4.2. Část pásu karet aplikace ANSYS Fluent s prvky pro správu uživatelsky definovaných funkcí

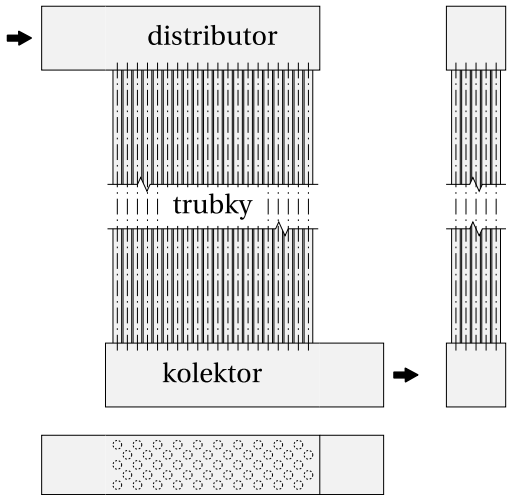
budou ukládány na disk pro potřeby případného dalšího zpracování. Zapisováno přitom bude do souboru numdis-data.txt a – při aktivním ukládání dat z jednotlivých buněk výpočetní sítě – také do souboru numdis-celldata.txt.

Soubor numdis-data.txt obsahuje pro každý časový krok patřičný výpočetní čas, délku časového kroku, veškeré souhrnné hodnoty vystupující v rovnicích 4.1 a 4.10 a průměrné souhrnné hodnoty rezidua diskretizované integrální rovnice pro kinetickou energii a numerické kinematické viskozity. Jestliže dosud některá hodnota nebyla spočtena – například proto, že ještě není dostupná historie údajů ze tří po sobě jdoucích časových kroků –, je do souboru namísto ní uvedeno nan (tj. textová reprezentace NaN v jazyce C). Dále je vhodné poznamenat, že pokud již soubor numdis-data.txt na disku existuje, není přepsán, ale na jeho konec je přidána hlavička nového záznamu a za ni jsou postupně zapisována data z jednotlivých časových kroků.

Naopak soubor numdis-celldata.txt, je-li jeho ukládání vyžádáno uživatelem, je v každém časovém kroku přepsán novými hodnotami reziduí diskretizované integrální rovnice pro kinetickou energii a numerické kinematické viskozity pro každou jednotlivou buňku (resp. jejich průměrnými hodnotami, je-li aktivováno průměrování).

4.3 Vizualizace dat

Za účelem ukázky vizualizace počítaných hodnot byl sestaven CFD model jednoduchého distribučního systému z obrázku 4.3, jehož geometrické parametry jsou uvedeny v tabulce 4.1. Tabulka 4.2 pak obsahuje vlastní nastavení CFD modelu. Jelikož mezní vrstva byla modelována pomocí stěnových funkcí (konkrétně pomocí Non-Equilibrium Wall Functions [109, kap. 4.17.4]), lze odpovídající histogram hodnot y^+ stěnových buněk nalézt



Obrázek 4.3. Distribuční systém analyzovaný za účelem ukázky vizualizace dat

Tabulka 4.1. Geometrické parametry distribučního systému z obrázku 4.3

Parametr	Hodnota
Uspořádání toku	„Z“
Trubkovnice	šířka 65 mm, délka 235 mm
Vstupní/výstupní oblast	délka 70 mm
Výška hlavních kanálů	70 mm
Trubkový svazek	5 řad po 10 trubkách, prostřídané uspořádání (45°), podélná rozteč 22,06 mm, příčná rozteč 11,03 mm
Trubky	vnitřní průměr 10 mm, délka 2 000 mm

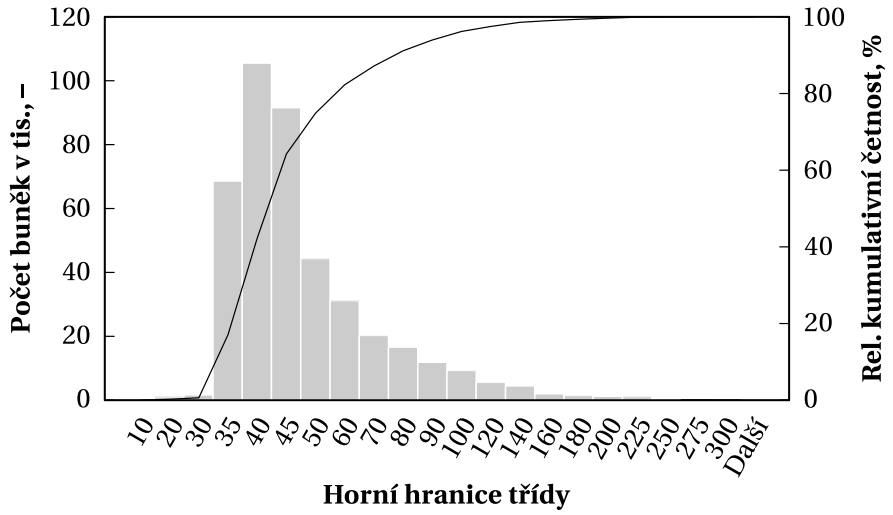
Tabulka 4.2. Nastavení CFD modelu

Parametr	Hodnota
Řešič	pressure-based
Metoda výpočtu	SIMPLE
Aktivní rovnice	proudění, turbulence, energie
Proudění	neustálené, implicitní formulace 2. řádu, časový krok 0,0125 s
Model turbulence	Realizable k–ϵ, Non-Equilibrium Wall Functions
Diskretizace	2. řádu (tlak), „upwind“ 2. řádu (vše ostatní)
Stěny	adiabatické, absolutní drsnost 0,15 mm
Vstup	mass flow inlet; voda, 300 K, 16 kg s^{-1}
Výstup	pressure outlet; 101 325 Pa
Výpočetní síť	~2,32 mil. hexaedrálních buněk, histogram y^+ – viz obrázek 4.4

na obrázku 4.4. Z něj je zřejmé, že podmínka platnosti použitých funkcí byla splněna, neboť drtivá většina hodnot y^+ leží v požadovaném rozsahu $[30,\ 300]$.

Jak bylo naznačeno v kapitole 4.1.6, veškeré počítané „buňkové“ hodnoty jsou ukládány do uživatelských proměnných, pomocí kterých je lze snadno vizualizovat. Tyto proměnné jsou zapisovány i do nativních datových souborů aplikace ANSYS Fluent a vizualizaci je tedy možné provádět i později, či dokonce v jiných aplikacích (např. pomocí ANSYS CFD-Post [110], dříve zmíněného open-source softwaru Kitware ParaView a podobně).

Obrázek 4.5 ukazuje hodnoty rezidua diskretizované integrální rovnice pro kinetickou energii v rovinných řezech nad trubkovnicemi hlavních kanálů distribučního systému z tabulky 4.1. Pro maximální zvýraznění rozložení hodnot rezidua na jednotlivých řezech je každý z nich opatřen vlastní stupnicí. Z obrázku je zřejmé, že ideální hodnoty $\epsilon^n = 0$ nabývá reziduum jen v některých místech; jinde je naopak kladné či záporné. Nejvýraz-



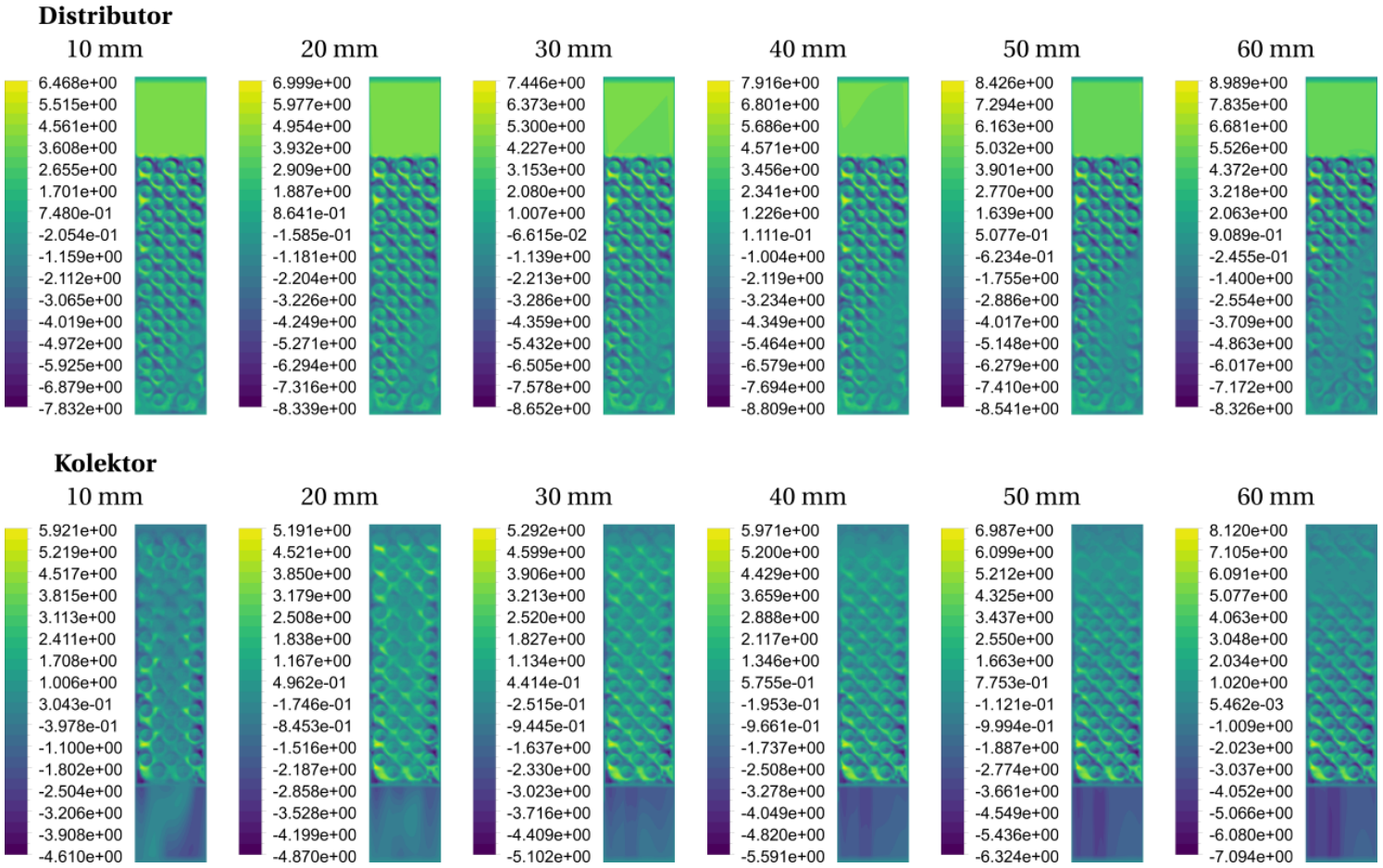
Obrázek 4.4. Histogram hodnot y^+ stěnových buněk v CFD modelu z tabulky 4.2

nější odchylky od nuly jsou přitom znatelné v místech mezi průměty jednotlivých trubek, kde byla evidentně kvalita sítě nižší. Co se týče absolutních velikostí hodnot rezidua, je nutné zde pamatovat na skutečnost, že proudění v distribučním systému bylo silně turbulentní s průměrnou hodnotou Reynoldsova čísla v trubkách přibližně 40 000. Dále také lze v distributoru – zvláště pak ve výše položených řezech – spatřit „rozmazaná" místa, která souvisí se značnou recirkulací tekutiny v okolí uzavřeného konce kanálu (viz proudnice na obrázku 4.6).

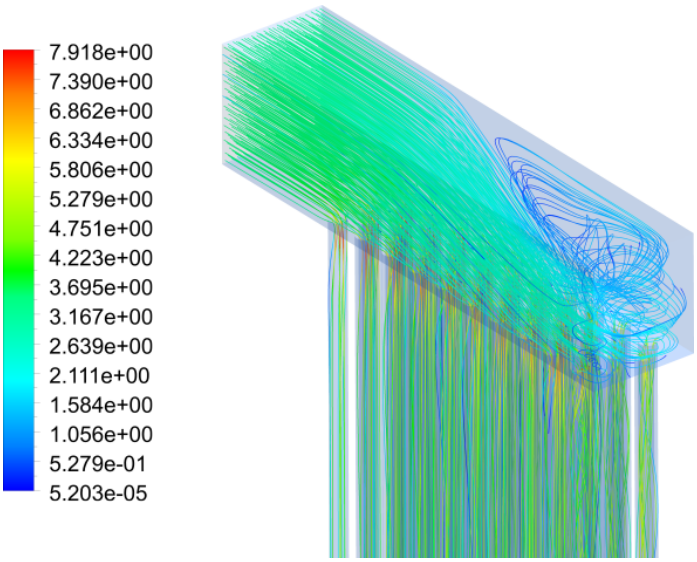
Rozložení hodnot numerické kinematické viskozity, ν^{n}, tedy míry numerické difuze v důsledku skutečnosti, že buňky nejsou „zarovnány" s lokálním směrem proudění, je znázorněno na obrázku 4.7. I zde byla stupnice vygenerována pro každý řez zvlášť (v tomto případě jsou dokonce rozdíly mezi rozsahy hodnot u jednotlivých řezů výrazně větší než u obrázku 4.5). Lze si všimnout, že největší míra numerické difuze je v distributoru poblíž vstupní plochy, zatímco v kolektoru už jsou hodnoty ν^{n} podstatně nižší bez ohledu na vzdálenost rovinného řezu od trubkovnice.

Kromě výše zmíněných dvou nejdůležitějších hodnot ale samozřejmě lze identickým způsobem vykreslovat i grafy dílčích[†] či průměrných[‡] hodnot (byly-li na uživatelovu žádost počítány). Jako příklad zmiňme viskózní disipaci, ϵ^{vis}, jejíž rozložení v rovinných řezech nad trubkovnicemi jsou na obrázku 4.8. V distributoru je zde – stejně jako na obrázku 4.5 – patrná oblast u uzavřeného konce s výraznou recirkulací tekutiny. Hodnoty viskózní disipace v rovinných řezech skrze jednotlivé řady trubek ve svazku jsou potom ukázány na obrázku 4.9. Oblast recirkulace tekutiny u uzavřeného konce distributoru

[†] Indexy uživatelských proměnných: 2–17 (resp. udm_offset + 2 až udm_offset + 17; viz kap. 4.1.6)
[‡] Indexy uživatelských proměnných: 18–27 (resp. udm_offset + 18 až udm_offset + 27; viz kap. 4.1.6)



Obrázek 4.5. Reziduum diskretizované integrální rovnice pro kinetickou energii (ϵ^n, kg m^2 s^{-3}) na rovinných řezech nad trubkovnicemi hlavních kanálů distribučního systému z obrázku 4.3. Vstup do distributoru je vždy v horní části grafu, výstup z kolektoru pak v dolní části. Čísla nad jednotlivými vrstevnicovými grafy značí kolmou vzdálenost patřičného řezu od trubkovnice.



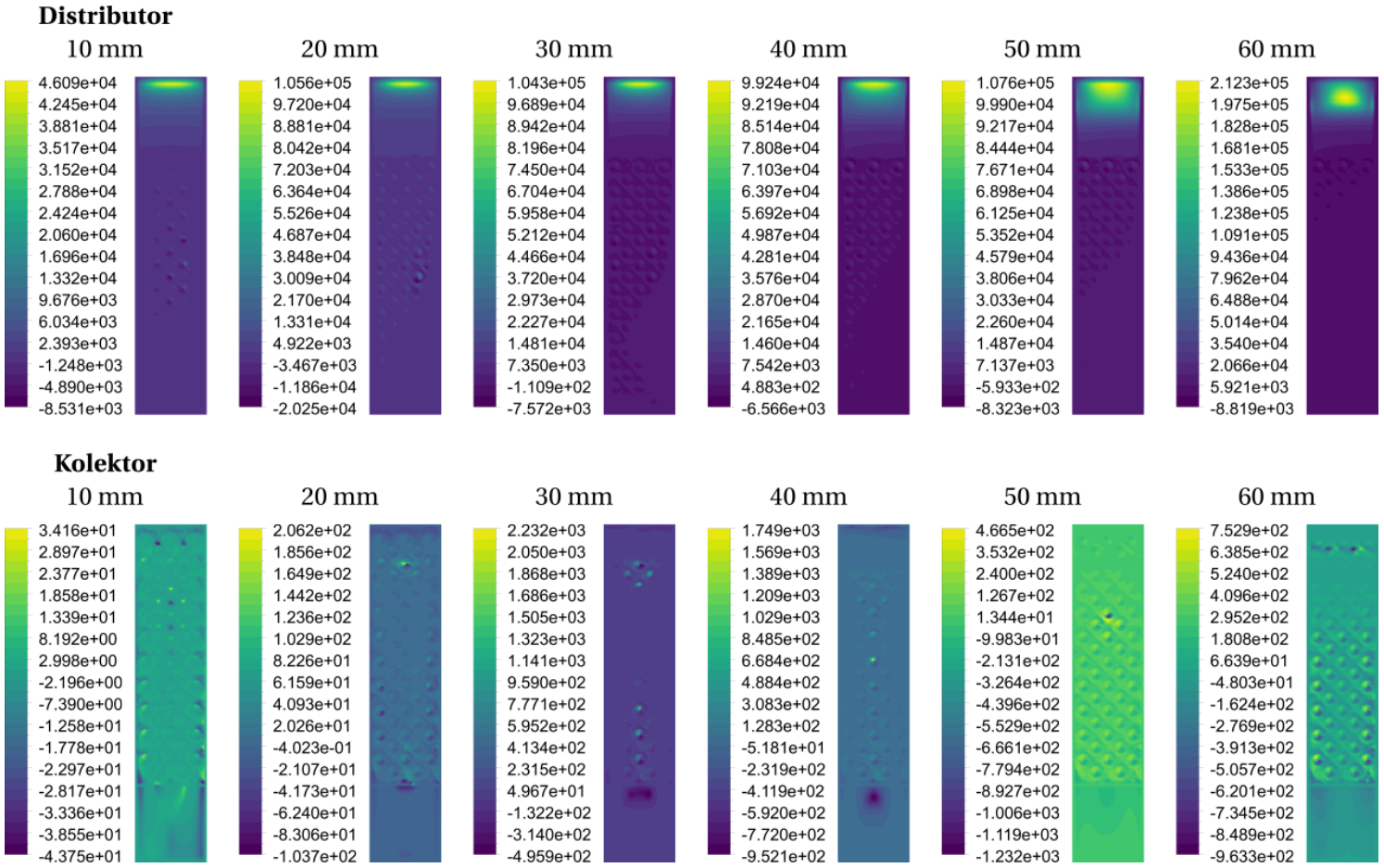
Obrázek 4.6. Proudnice v distributoru obarvené pomocí velikosti rychlosti (m s^{-1}); v okolí uzavřeného konce kanálu je zřetelně patrná oblast se značnou recirkulací tekutiny

je zde vzhledem k nutnému většímu rozsahu škály zřejmá méně, ale naopak jsou dobře viditelné oblasti s vyššími hodnotami ϵ^{vis} v kolektoru.

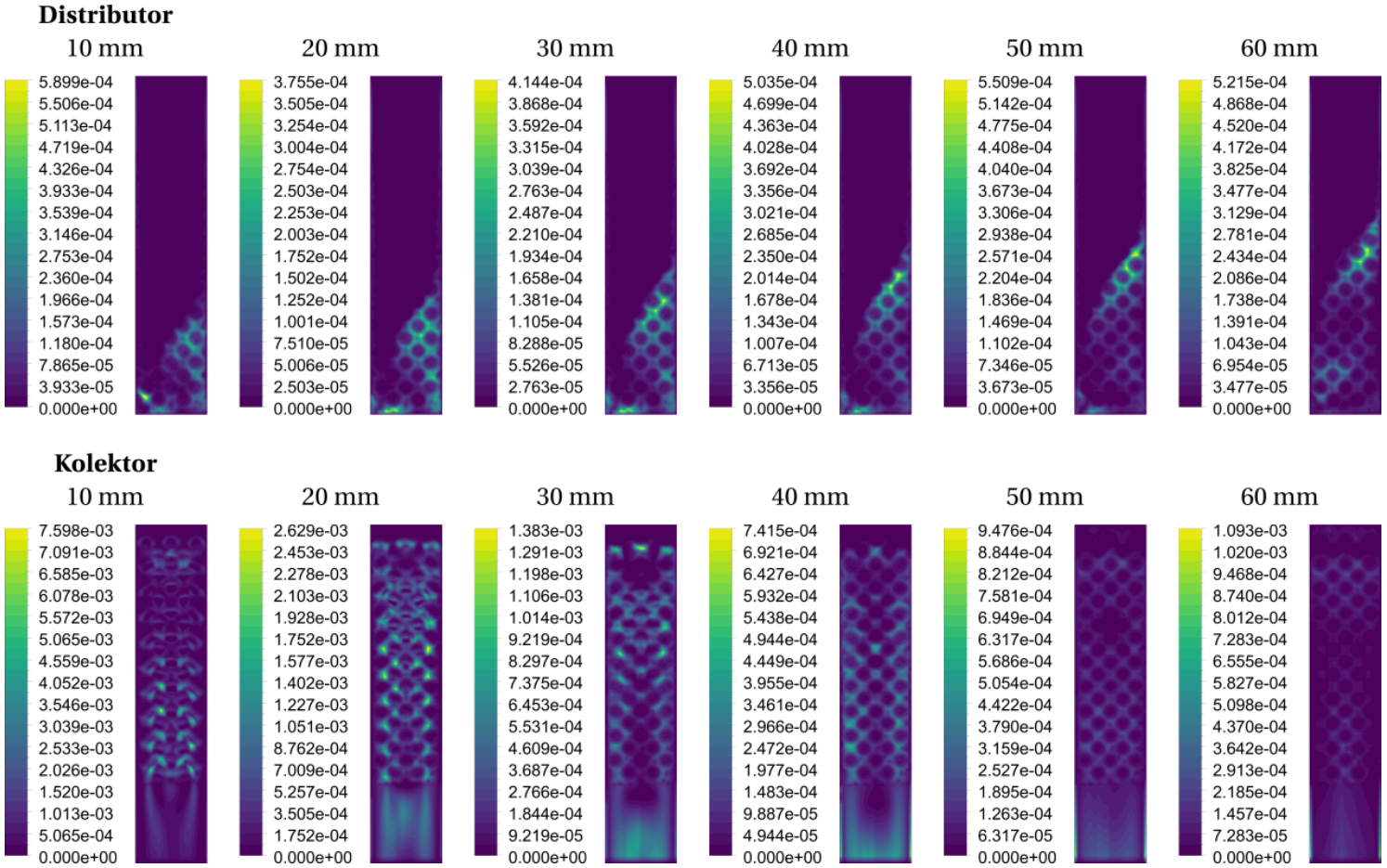
4.4 Budoucí zaměření výzkumu

Vyvinutý kód je plně funkční a nyní tudíž zbývá provést srovnání pomocí něj obdržených údajů s experimentálními daty pro alespoň některé v procesní praxi běžně se vyskytující typy proudění. K tomu bude nutné sestavit například detailní CFD model odpovídající fyzikálnímu experimentu diskutovanému v článku [107], ve kterém byly u vířivého proudění v trubce měřeny vybrané veličiny pomocí laserové Dopplerovské anemometrie („Laser Doppler Anemometry“, LDA).

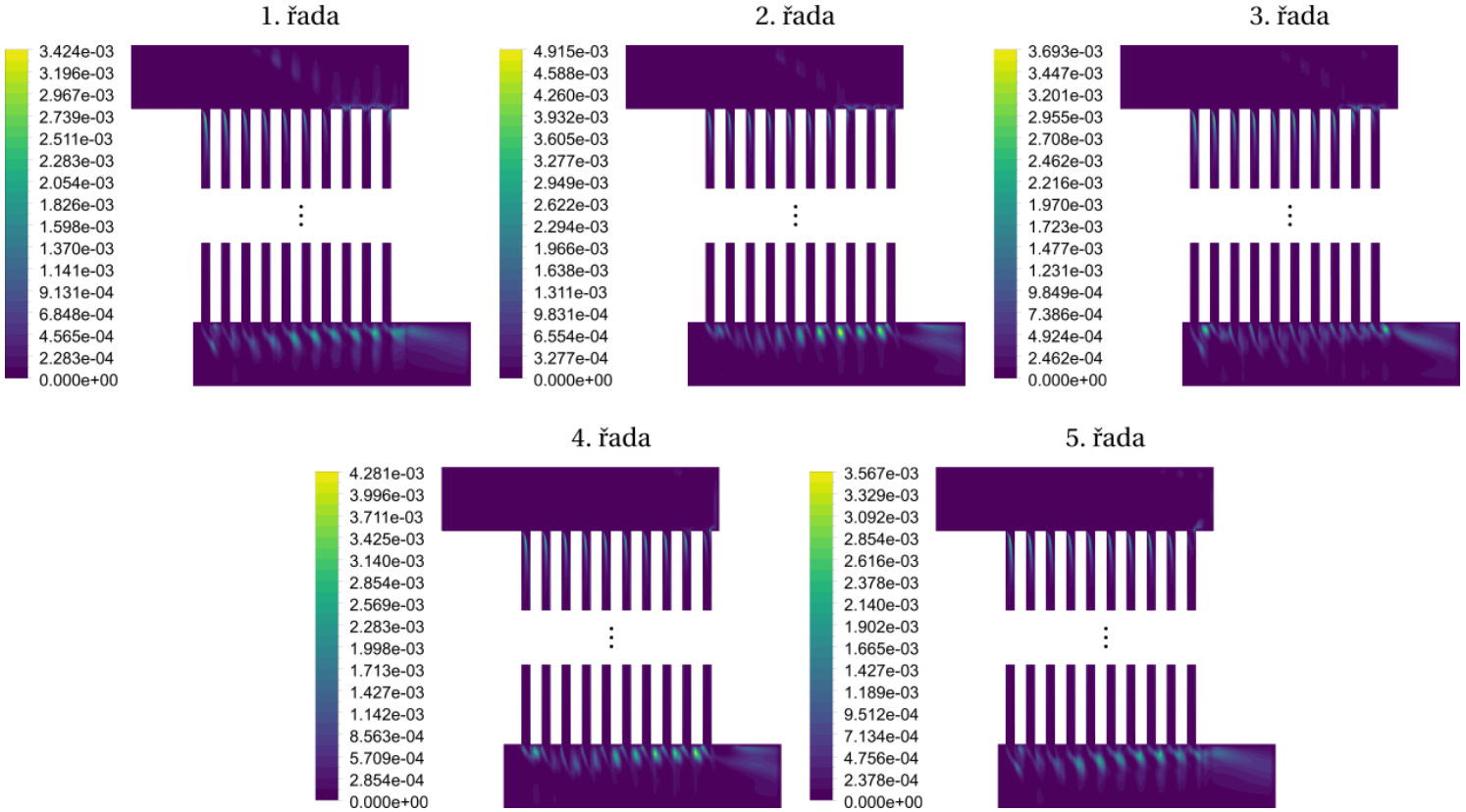
Je zřejmé, že pro získání kvalitních dat o veličinách měřených v tomto experimentu bude s ohledem na požadovanou přesnost nutně muset být použit vhodný model turbulence – nejspíše některý z LES modelů, příp. SAS. Analogicky tomu bude muset být přizpůsobena také jemnost výpočetní sítě. Lze tedy realisticky očekávat, že patřičný vícejádrový výpočet poběží na clusteru řádově týdny. Přitom však bude možné extrahovat údaje o zájmových veličinách včetně míry numerické disipace, která by teoreticky měla být úměrná rozdílu mezi fyzikálním a numerickým experimentem. Toto pak umožní hodnocení použitelnosti zde popisovaného přístupu.



Obrázek 4.7. Numerická kinematická viskozita (ν^n, $m^2\,s^{-1}$) na rovinných řezech z obrázku 4.5



Obrázek 4.8. Viskózní disipace (ϵ^{vis}, kg m^2 s^{-3}) na rovinných řezech z obrázku 4.5



Obrázek 4.9. Viskózní disipace (ϵ^{vis}, kg m^2 s^{-3}) na rovinných řezech skrze jednotlivé řady trubek v distribučním systému z obrázku 4.3. Nahoře je vždy zobrazen distributor, dole pak kolektor.

Kapitola

5

Závěr

Předložená habilitační práce se zabývá problematikou výpočtového modelování proudění tekutin v procesních a energetických zařízeních, a to jak z pohledu dvou specifických zjednodušených způsobů tvorby patřičných modelů, tak z pohledu implementace odhadu míry numerické disipace v klasických CFD simulacích. Nejprve byla pozornost věnována zjednodušenému CFD modelování, které je vhodné zejména pro použití v prvotní fázi návrhu zařízení či k implementaci v optimalizačních algoritmech. Uplatnění by však snadno mohlo nalézt i při řešení provozních potíží („troubleshooting“), případně v oblasti virtuálního navrhování („virtual prototyping“). V práci byla popsána hlavní zjednodušení zajišťující výsledky přijatelné přesnosti a představeny doposud získané výsledky. Zároveň byly nastíněny směry, kterými by se mohl ubírat další výzkum potenciálně vedoucí k dodatečnému zpřesnění modelů, zkrácení výpočetních časů či zvýšení uživatelské přívětivosti výsledných simulačních nástrojů a snížení nároků kladených na uživatele ve věci znalosti programování. Již nyní však zjednodušené CFD modely dokáží vracet zkonvergovaná řešení ustálených úloh v řádu jednotek vteřin až minut a lze tedy říci, že odpovídající způsob modelování může být i přes nižší přesnost dat vhodným nástrojem pro přibližné analýzy proudění v procesních či energetických aparátech menších až středních velikostí. Jejich použití přitom může vést k nezanedbatelnému zkrácení návrhového procesu zařízení a ke snížení pravděpodobnosti pozdějšího výskytu vybraných provozních potíží. V zásadě jediným požadavkem, který musí být pro praktické nasazení patřičných modelů splněn, je dostatečná kompatibilita geometrie distribučního systému v zařízení s výpočetní sítí složenou pouze z kvádrových buněk. Co se týče vyvinutého CFD softwaru, jeho aktuálně největším omezením je existence automatického generátoru výpočetních sítě pouze pro určité typy distribučních systémů, ovšem

doplnění patřičných algoritmů i pro jiná potřebná geometrická provedení by nemělo představovat výraznější překážku.

Dále bylo v práci představeno modelování založené na metodě konečných prvků, jehož zamýšlenými aplikacemi jsou přibližné analýzy a řešení provozních potíží u rozsáhlých, avšak strukturou relativně jednoduchých trubkových svazků například v kotlích na odpadní teplo. Primárním cílem zde je nejen simulace proudění tekutin, ale hlavně simulace přenosu tepla a odhad výsledného mechanického namáhání takového svazku, ke kterému dochází v důsledku nerovnoměrné distribuce tekutiny a nerovnoměrného tepelného zatížení teplosměnných ploch. Potřebný matematický aparát je stále v rané fázi vývoje a zatím zahrnuje pouze proudění, nicméně již byla započata i implementace přenosu tepla. Informace ohledně použitelnosti modelu jsou dosud jen omezené, ovšem na základě dat získaných pomocí stávající verze vyvíjeného simulačního softwaru se tento způsob modelování zdá být perspektivní. Stejně jako u zjednodušeného CFD modelování byla také zde naznačena možná budoucí vylepšení matematického aparátu a vhodné směry dalšího výzkumu. Kromě toho byla vzhledem ke stádiu vývoje modelu zdůrazněna i nutnost provést jeho řádnou validaci. Pokud se patřičný způsob provádění přibližných analýz opravdu osvědčí, bude i v případě výše zmíněných typů zařízení možné s minimem uživatelského úsilí do určité míry předcházet některým provozním potížím (praskání trubek apod.), resp. zvýšit spolehlivost těchto zařízení a prodloužit jejich životnost.

Nakonec byla diskutována implementace metody *a posteriori* odhadu chyb v klasických CFD modelech. Popisovaný algoritmus považuje CFD řešič za „černou skříňku“ a míru numerické disipace odhaduje pouze na základě údajů, které jsou řešičem běžně poskytovány například pro potřeby vizualizace dat. Kompletní zdrojový kód je k dispozici ve formě sady uživatelsky definovaných funkcí využitelných při jednojádrových i vícejádrových simulacích v aplikaci ANSYS Fluent. Nyní zbývá ověřit, nakolik takto získané výsledky odpovídají realitě, resp. nalézt závislost mezi jednotlivými spočtenými hodnotami a přesným řešením. Toto je plánováno provést pomocí dat z literatury a detailních CFD simulací pracujících s vhodnými modely turbulence. Je přitom nutné poukázat na skutečnost, že v současné praxi jsou CFD modely používané při návrzích či analýzách procesních a energetických zařízení (typicky URANS) zcela běžně považovány za přesné, byť jejich přesnost mnohdy značně utrpí právě v důsledku nedostatečně kvalitní výpočetní sítě, nevhodných diskretizačních schémat atd. Pokud se tedy potvrdí použitelnost zmíněného přístupu, bude vyvinutá sada uživatelsky definovaných funkcí velmi cenným nástrojem pro hodnocení kvality tranzientních CFD modelů a přesnosti takto získávaných dat, resp. pro odhad řešení nezávislého na výpočetní síti.

Reference

Publikace autora citované v této práci

[A1] Tomáš Pačíska, Vojtěch Turek, Zdeněk Jegla & Bohuslav Kilkovský. Suitability of some commonly available software for unconventional condenser analysis. *Applied Thermal Engineering* **70**(2): 1195–1201, 2014. doi: j.applthermaleng.2014.04.061.

[A2] Vojtěch Turek, Petr Bělohradský & Zdeněk Jegla. Geometry optimization of a gas preheater inlet region – A case study. *Chemical Engineering Transactions* **29**: 1339–1344, 2012. doi: 10.3303/CET1229224.

[A3] Vojtěch Turek. *New Elements of Heat Transfer Efficiency Improvement in Systems and Units*. Disertační práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Brno, 2012. Vedoucí disertační práce: doc. Ing. Zdeněk Jegla, Ph.D.

[A4] Vojtěch Turek, Dominika Fialová, Zdeněk Jegla & Bohuslav Kilkovský. Efficient 2D model of flow distribution in dense tube bundles. *Chemical Engineering Transactions* **45**: 1177–1182, 2015. doi: 10.3303/CET1545197.

[A5] Vojtěch Turek, Dominika Fialová & Zdeněk Jegla. Efficient flow modelling in equipment containing porous elements. *Chemical Engineering Transactions* **52**: 487–492, 2016. doi: 10.3303/CET1652082.

[A6] Vojtěch Turek. Improving performance of simplified computational fluid dynamics models via symmetric successive overrelaxation. *Energies* **12**(12): article id 2438, 2019. doi: 10.3390/en12122438.

[A7] Vojtěch Turek. On improving computational efficiency of simplified fluid flow models. *Chemical Engineering Transactions* **70**: 1447–1452, 2018. doi: 10.3303/CET1870242.

[A8] Vojtěch Turek. *Dense Tube Bundle Flow Modeller Quick Start Guide, Version 1.1*. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav procesního inženýrství, Brno, 2018.

[A9] Vojtěch Turek. *Dokumentace vyvinuté výpočtové metody pro analýzu systémů distribuce toku a jejich tvarovou optimalizaci.* Tech. zpr. TE02000236/DV069. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav procesního inženýrství, 2018.

[A10] Vojtěch Turek, Jiří Hájek, Zdeněk Jegla & Petr Stehlík. Optimum design of distribution systems in heat exchangers. *Asia-Pacific Journal of Chemical Engineering* **6**(5): 750–759, 2011. DOI: 10.1002/apj.516.

[A11] Vojtěch Turek, Ladislav Bébar & Zdeněk Jegla. Simplified pressure drop and flow distribution modelling in radial catalytic converters. *Chemical Engineering Transactions* **39**: 853–858, 2014. DOI: 10.3303/CET1439143.

[A12] Vojtěch Turek. *Výpočtový model pro analýzu distribuce toku ve výměníku tepla.* Tech. zpr. TE02000236/DV037. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav procesního inženýrství, 2016.

[A13] Tomáš Létal, Vojtěch Turek & Dominika Fialová. Nonlinear finite element analysis-based flow distribution model for engineering practice. *Chemical Engineering Transactions* **76**: 157–162, 2019. DOI: 10.3303/CET1976027.

Ostatní citované zdroje

[1] Heat Transfer Research, Inc. *HTRI Xchanger Suite User's Guide, Version 8.0.* Heat Transfer Research, Inc., Navasota, TX, USA, 2019.

[2] Aspen Technology, Inc. *Aspen Exchanger Design & Rating, Version 11.0.* Aspen Technology, Inc., Bedford, MA, USA, 2019.

[3] Martin Chýlek. *Tok látek v nestandardních procesních a energetických zařízeních.* Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Brno, 2018. Vedoucí diplomové práce: Ing. Vojtěch Turek, Ph.D.

[4] Oracle, Inc. *What is Java and why do I need it?* [online]. 2019. Dostupné z https://java.com/en/download/faq/whatis_java.xml [cit. 2019-07-02].

[5] André Bakker. *Computational Fluid Dynamics Lectures* [online]. 2008. Dostupné z http://www.bakker.org/dartmouth06/engs150/ [cit. 2019-07-01].

[6] ANSYS, Inc. *ANSYS Fluent User's Guide, Release 19.5.* ANSYS, Inc., Canonsburgh, PA, USA, 2019.

[7] Michele Benzi, Gene H. Golub & Jörg Liesen. Numerical solution of saddle point problems. *Acta Numerica* **14**: 1–137, 2005. DOI: 10.1017/S0962492904000212.

[8] Panayot S. Vassilevski. *Multilevel Block Factorization Preconditioners: Matrix-based Analysis and Algorithms for Solving Finite Element Equations.* Springer, London, UK, 2008.

[9] Suhas V. Patankar & Dudley B. Spalding. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *International Journal of Heat and Mass Transfer* **15**(10): 1787–1806, 1972. doi: 10.1016/0017-9310(72)90054-3.

[10] Suhas V. Patankar. *Numerical Heat Transfer and Fluid Flow*. Taylor & Francis, New York, NY, USA, 1980.

[11] Jeffrey P. van Doormaal & George D. Raithby. Enhancements of the SIMPLE method for predicting incompressible fluid flows. *Numerical Heat Transfer* **7**(2): 147–163, 1984. doi: 10.1080/01495728408961817.

[12] Raad I. Issa. Solution of the implicitly discretised fluid flow equations by operator-splitting. *Journal of Computational Physics* **62**(1): 40–65, 1986. doi: 10.1016/0021-9991(86)90099-9.

[13] Xun-Liang Liu, Wen-Quan Tao & Ya-Ling He. A simple method for improving the SIMPLER algorithm for numerical simulations of incompressible fluid flow and heat transfer problems. *Engineering Computations* **22**(8): 921–939, 2005. doi: 10.1108/02644400510626488.

[14] Mohammed M. Rahman & Timo Siikonen. An improved SIMPLE method on a collocated grid. *Numerical Heat Transfer, Part B: Fundamentals* **38**(2): 177–201, 2000. doi: 10.1080/104077900750034661.

[15] Dmitry K. Kolmogorov, Wen Z. Shen, Niels N. Sørensen & Jens N. Sørensen. Fully consistent SIMPLE-like algorithms on collocated grids. *Numerical Heat Transfer, Part B: Fundamentals* **67**(2): 101–123, 2015. doi: 10.1080/10407790.2014.949583.

[16] Dong S. Jang, Rajiv Jetli & Sumanta Acharya. Comparison of the PISO, SIMPLER, and SIMPLEC algorithms for the treatment of the pressure-velocity coupling in steady flow problems. *Numerical Heat Transfer* **10**(3): 209–228, 1986. doi: 10.1080/10407788608913517.

[17] Henk K. Versteeg & Weeratunge Malalasekera. *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*. 2. vyd. Pearson Education Ltd., Harlow, UK, 2007.

[18] John H. Mathews & Kurtis D. Fink. *Numerical Methods Using MATLAB*. 4. vyd. Prentice Hall, Inc., Upper Saddle River, NJ, USA, 2004.

[19] Dudley B. Spalding. A novel finite difference formulation for differential expressions involving both first and second derivatives. *International Journal for Numerical Methods in Engineering* **4**(4): 551–559, 1972. doi: 10.1002/nme.1620040409.

[20] Martin Fowler. *Refactoring: Improving the Design of Existing Code*. 2. vyd. Addison-Wesley Professional, Boston, MA, USA, 2018.

[21] TOSHIYUKI HAYASE, JOSEPH A. HUMPHREY & RALPH GREIF. A consistently formulated QUICK scheme for fast and stable convergence using finite-volume iterative calculation procedures. *Journal of Computational Physics* **98**(1): 108–118, 1992. DOI: 10.1016/0021-9991(92)90177-Z.

[22] AMI HARTEN. High resolution schemes for hyperbolic conservation laws. *Journal of Computational Physics* **49**(3): 357–393, 1983. DOI: 10.1016/0021-9991(83)90136-5.

[23] BRIAN P. LEONARD. Simple high-accuracy resolution program for convective modelling of discontinuities. *International Journal for Numerical Methods in Fluids* **8**(10): 1291–1318, 1988. DOI: 10.1002/fld.1650081013.

[24] JOHN CRANK & PHYLLIS NICOLSON. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. *Mathematical Proceedings of the Cambridge Philosophical Society* **43**(1): 50–67, 1947. DOI: 10.1017/S0305004100023197.

[25] WLODZIMIERZ W. TWORZYDLO, JOHN T. ODEN & EARL A. THORNTON. Adaptive implicit/explicit finite element method for compressible viscous flows. *Computer Methods in Applied Mechanics and Engineering* **95**(3): 397–440, 1992. DOI: 10.1016/0045-7825(92)90195-P.

[26] DENIS A. DE SOUZA, MARCOS A. MARTINS & ALVARO L. COUTINHO. Edge-based adaptive implicit/explicit finite element procedures for three-dimensional transport problems. *Communications in Numerical Methods in Engineering* **21**(10): 545–552, 2005. DOI: 10.1002/cnm.767.

[27] PHILIPPE R. SPALART & STEVEN R. ALLMARAS. A one-equation turbulence model for aerodynamic flows. *Recherche Aerospatiale* **1**: 5–21, 1994.

[28] CHRISTOPHER RUMSEY. *The Spalart-Allmaras Turbulence Model* [online]. 2019. Dostupné z https://turbmodels.larc.nasa.gov/spalart.html [cit. 2019-09-25].

[29] DOMINIKA FIALOVÁ. *Analýza distribuce toku v systémech s hustými svazky trubek.* Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Brno, 2015. Vedoucí bakalářské práce: Ing. Vojtěch Turek, Ph.D.

[30] DOMINIKA FIALOVÁ. *Distribuce toku v zařízeních s hustými svazky trubek.* Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Brno, 2017. Vedoucí diplomové práce: Ing. Vojtěch Turek, Ph.D.

[31] YOUSEF SAAD. *Iterative Methods for Sparse Linear Systems.* 2. vyd. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, USA, 2003.

[32] ARE M. BRUASET. *A Survey of Preconditioned Iterative Methods.* CRC Press, Boca Raton, FL, USA, 1995.

[33] Sangback Ma. Comparisons of the ILU(0), Point-SSOR, and SPAI preconditioners on the CRAY-T3E for nonsymmetric sparse linear systems arising from PDEs on structured grids. *The International Journal of High Performance Computing Applications* **14**(1): 39–48, 2000. DOI: 10.1177/109434200001400103.

[34] David M. Young. On the accelerated SSOR method for solving large linear systems. *Advances in Mathematics* **23**(3): 215–271, 1977. DOI: 10.1016/S0001-8708(77)80029-7.

[35] Hendrik A. van der Vorst. *Iterative Krylov Methods for Large Linear Systems*. Cambridge University Press, Cambridge, UK, 2003.

[36] Magnus R. Hestenes & Eduard Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards* **49**(6): 409–436, 1952. DOI: 10.6028/jres.049.044.

[37] Andrew Chapman, Yousef Saad & Larry Wigton. High-order ILU preconditioners for CFD problems. *International Journal for Numerical Methods in Fluids* **33**(6): 767–788, 2000. DOI: 10.1002/1097-0363(20000730)33:6<767::AID-FLD28>3.0.CO;2-C.

[38] Jennifer Scott & Miroslav Tůma. The importance of structure in incomplete factorization preconditioners. *BIT Numerical Mathematics* **51**(2): 385–404, 2011. DOI: 10.1007/s10543-010-0299-8.

[39] Jacobus A. Meijerink & Hendrik A. van der Vorst. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Mathematics of Computation* **31**(137): 148–162, 1977. DOI: 10.1090/S0025-5718-1977-0438681-4.

[40] Gerard L. Sleijpen & Diederik R. Fokkema. BiCGstab(l) for linear equations involving unsymmetric matrices with complex spectrum. *Electronic Transactions on Numerical Analysis* **1**: 11–32, 1993.

[41] Yousef Saad & Martin H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing* **7**(3): 856–869, 1986. DOI: 10.1137/0907058.

[42] Yousef Saad. ILUT: A dual threshold incomplete LU factorization. *Numerical Linear Algebra with Applications* **1**(4): 387–402, 1994. DOI: 10.1002/nla.1680010405.

[43] Thomas A. Manteuffel. An incomplete factorization technique for positive definite linear systems. *Mathematics of Computation* **34**(150): 473–497, 1980. DOI: 10.1090/S0025-5718-1980-0559197-0.

[44] Roland W. Freund & Noël M. Nachtigal. QMR: A quasi-minimal residual method for non-Hermitian linear systems. *Numerische Mathematik* **60**(1): 315–339, 1991. DOI: 10.1007/BF01385726.

[45] Eduardo F. D'Azevedo, Peter A. Forsyth & Wei-Pai Tang. Ordering methods for preconditioned conjugate gradient methods applied to unstructured grid problems. *SIAM Journal on Matrix Analysis and Applications* **13**(3): 944–961, 1992. doi: 10.1137/0613057. (Cit. 26. 09. 2019).

[46] Laura C. Dutto. The effect of ordering on preconditioned GMRES algorithm, for solving the compressible Navier-Stokes equations. *International Journal for Numerical Methods in Engineering* **36**(3): 457–497, 1993. doi: 10.1002/nme.1620360307.

[47] Frederic T. Chong & Anant Agarwal. Shared memory versus message passing for iterative solution of sparse, irregular problems. *Parallel Processing Letters* **9**(1): 159–170, 1999. doi: 10.1142/S0129626499000177.

[48] Shun Doi & Takumi Washio. Ordering strategies and related techniques to overcome the trade-off between parallelism and convergence in incomplete factorizations. *Parallel Computing* **25**(13): 1995–2014, 1999. doi: 10.1016/S0167-8191(99)00064-2.

[49] Elizabeth H. Cuthill & James McKee. Reducing the bandwidth of sparse symmetric matrices. In: *Proceedings of the 24th National Conference of the Association for Computing Machinery, New York, NY, USA, 26 – 28 August 1969*, 157–172. doi: 10.1145/800195.805928.

[50] John A. George. *Computer Implementation of the Finite-Element Method*. Disertační práce. Stanford University, Department of Computer Science, Stanford, CA, USA, 1971. Vedoucí disertační práce: Prof. George E. Forsythe.

[51] Iain S. Duff & Gérard A. Meurant. The effect of ordering on preconditioned conjugate gradients. *BIT Numerical Mathematics* **29**(4): 635–657, 1989. doi: 10.1007/BF01932738.

[52] Dhabaleswar K. Panda & Sayantan Sur. Infiniband. In: *Encyclopedia of Parallel Computing*. Ed. David Padua. Springer, Boston, MA, USA, 2011, 927–935. doi: 10.1007/978-0-387-09766-4_21.

[53] ISO/IEC. *ISO/IEC 9899:1999: Programming Languages – C*. International Organization for Standardization, Geneva, Switzerland, 1999.

[54] Peter R. Graves-Morris. The breakdowns of BiCGStab. *Numerical Algorithms* **29**(1–3): 97–105, 2002. doi: 10.1023/A:1014864007293.

[55] Zhi-Hao Cao. Avoiding breakdown in variants of the BI-CGSTAB algorithm. *Linear Algebra and its Applications* **263**: 113–132, 1997. doi: 10.1016/S0024-3795(96)00525-3.

[56] Leo Lewis. *Java collection performance* [online]. 2011. Dostupné z https://dzone.com/articles/java-collection-performance [cit. 2019-08-28].

[57] Piotr Wendykier & James G. Nagy. Parallel Colt: A high-performance Java library for scientific computing and image processing. *ACM Transactions on Mathematical Software* **37**(3): 31:1–31:22, 2010. doi: 10.1145/1824801.1824809.

[58] Richard Barrett, Michael W. Berry, Tony F. Chan, James Demmel, June Donato, Jack J. Dongarra, Victor Eijkhout, Roldan Pozo, Charles Romine & Henk Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. 2. vyd. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, USA, 1994.

[59] Susan Blackford, James Demmel, Jack J. Dongarra, Iain Duff, Sven Hammarling, Gwendolyn Henry, Michael A. Heroux, Linda Kaufman, Andrew Lumsdaine, Antoine P. Petitet, Roldan Pozo, Karin Remington & Clint Whaley. An updated set of Basic Linear Algebra Subprograms (BLAS). *ACM Transactions on Mathematical Software* **28**(2): 135–151, 2002. doi: 10.1145/567806.567807.

[60] Edward Anderson, Zhaojun Bai, Christian H. Bischof, Susan Blackford, James Demmel, Jack J. Dongarra, Jeremy J. Du Croz, Anne Greenbaum, Sven Hammarling, Alan M. McKenney & Danny C. Sorensen. *LAPACK Users' Guide*. 3. vyd. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, USA, 1999. doi: 10.1137/1.9780898719604.

[61] Intel. *Intel Math Kernel Library 2019, Update 4*. Intel, Inc., Santa Clara, CA, USA, 2019.

[62] AMD. *AMD Optimizing CPU Libraries (AOCL) Version 2.0*. AMD, Inc., Santa Clara, CA, USA, 2019.

[63] Optimatika. *oj! Algorithms Version 47.1.1*. Optimatika, Stockholm, Sweden, 2019.

[64] Peter Abeles. *Efficient Java Matrix Library* [online]. 2019. Dostupné z https://ejml.org/ [cit. 2019-08-29].

[65] Samuel Halliday. *Matrix Toolkits Java* [online]. 2019. Dostupné z https://github.com/fommil/matrix-toolkits-java/ [cit. 2019-08-29].

[66] Holger Arndt, Markus Bundschus, Andreas Nägele, Rand Huso & Frode Carlsen. *Universal Java Matrix Package* [online]. 2019. Dostupné z https://ujmp.org/ [cit. 2019-08-29].

[67] Holger Arndt. *Dense and Sparse Matrix Libraries for Java: An Overview* [online]. 2019. Dostupné z https://java-matrix.org/ [cit. 2019-08-29].

[68] Peter Abeles. *Java Matrix Benchmark* [online]. 2019. Dostupné z https://lessthanoptimal.github.io/Java-Matrix-Benchmark/ [cit. 2019-08-29].

[69] Jeannette Kiefer. Sequential minimax search for a maximum. *Proceedings of the American Mathematical Society* **4**(3): 502–506, 1953. doi: 10.2307/2032161.

[70] Miroslav Rebej. *Simplified Flow Distribution Modelling*. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Brno, 2019. Vedoucí diplomové práce: Ing. Vojtěch Turek, Ph.D.

[71] Henry G. Weller, Gavin Tabor, Hrvoje Jasak & Christer Fureby. A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computers in Physics* **12**(6): 620–631, 1998. doi: 10.1063/1.168744.

[72] Oleg I. Dudar & Elena S. Dudar. Application of 1 D finite element method in combination with laminar solution method for pipe network analysis. *IOP Conference Series: Materials Science and Engineering* **262**: article id 012085, 2017. doi: 10.1088/1757-899X/262/1/012085.

[73] Frank M. White. *Fluid Mechanics*. 4. vyd. McGraw-Hill, Inc., New York, NY, USA, 1998.

[74] Cyril F. Colebrook & Cedric M. White. Experiments with fluid friction in roughened pipes. *Proceedings of the Royal Society of London. Series A – Mathematical and Physical Sciences* **161**(906): 367–381, 1937. doi: 10.1098/rspa.1937.0150.

[75] Stuart W. Churchill. Friction-factor equation spans all fluid-flow regimes. *Chemical Engineering* **84**(24): 91–92, 1977.

[76] Isaak E. Idelchik. *Handbook of Hydraulic Resistance*. 4. vyd. Begell House, Inc., Redding, CT, USA, 2008.

[77] Bernard J. Bailey. Fluid flow in perforated pipes. *Journal of Mechanical Engineering Science* **17**(6): 338–347, 1975. doi: 10.1243/JMES_JOUR_1975_017_048_02.

[78] Zdeněk Jegla & Dominika Fialová. Development of heat and fluid flow distribution modelling system for analysing multiple-distributed designs of process and power equipment. *Chemical Engineering Transactions* **70**: 1471–1476, 2018. doi: 10.3303/CET1870246.

[79] Ian H. Bell, Jorrit Wronski, Sylvain Quoilin & Vincent Lemort. Pure and pseudo-pure fluid thermophysical property evaluation and the open-source thermophysical property library CoolProp. *Industrial & Engineering Chemistry Research* **53**(6): 2498–2508, 2014. doi: 10.1021/ie4033999.

[80] Juan J. Romera. *IAPWS: A Python Implementation of the IAPWS Standard* [online]. 2020. Dostupné z https://github.com/jjgomera/iapws/ [cit. 2020-01-02].

[81] Travis E. Oliphant. *A Guide to NumPy*. Trelgol Publishing, Spanish Fork, UT, USA, 2006.

[82] The Spyder Website Contributors. *The Scientific Python Development Environment* [online]. 2018. Dostupné z https://www.spyder-ide.org/ [cit. 2019-10-07].

[83] Will Schroeder, Ken Martin & Bill Lorensen. *The Visualization Toolkit*. 4. vyd. Kitware, Inc., Clifton Park, NY, USA, 2006.

[84] Utkarsh Ayachit. *The ParaView Guide: A Parallel Visualization Application*. Kitware, Inc., Clifton Park, NY, USA, 2015.

[85] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I. Jordan & Ion Stoica. Ray: A distributed framework for emerging AI applications. In: *Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI '18), Carlsbad, CA, USA, 8 – 10 October 2018*, 561–577.

[86] Robert Nishihara. *10x Faster Parallel Python without Python Multiprocessing* [online]. 2019. Dostupné z `https://towardsdatascience.com/10x-faster-parallel-python-without-python-multiprocessing-e5017c93cce1/` [cit. 2019-10-16].

[87] Steven A. Orszag. Analytical theories of turbulence. *Journal of Fluid Mechanics* **41**(2): 363–386, 1970. doi: 10.1017/S0022112070000642.

[88] Joseph Smagorinsky. General circulation experiments with the primitive equations: I. The basic experiment. *Monthly Weather Review* **91**(3): 99–164, 1963. doi: 10.1175/1520-0493(1963)091<0099:GCEWTP>2.3.CO;2.

[89] Michael Strelets. Detached eddy simulation of massively separated flows. In: *Proceedings of the 39th Aerospace Sciences Meeting and Exhibit, Reno, NV, USA, 8 – 11 January 2001,* article id 2001-0879. doi: 10.2514/6.2001-879.

[90] Florian R. Menter & Yury Egorov. The Scale-Adaptive Simulation method for unsteady turbulent flow predictions. Part 1: Theory and model description. *Flow, Turbulence and Combustion* **85**(1): 113–138, 2010. doi: 10.1007/s10494-010-9264-5.

[91] Heng Xiao, Jinlong Wu, Jianxun Wang, Rui Sun & Christopher J. Roy. Quantifying and reducing model-form uncertainties in Reynolds-averaged Navier-Stokes simulations: A data-driven, physics-informed Bayesian approach. *Journal of Computational Physics* **324**: 115–136, 2016. doi: 10.1016/j.jcp.2016.07.038.

[92] Yujie Zhu & Xiangyu Hu. Free-stream preserving linear-upwind and WENO schemes on curvilinear grids. *Journal of Computational Physics* **399**: article id 108907, 2019. doi: 10.1016/j.jcp.2019.108907.

[93] Carlos H. Marchi & António F. da Silva. Unidimensional numerical solution error estimation for convergent apparent order. *Numerical Heat Transfer, Part B: Fundamentals* **42**(2): 167–188, 2002. doi: 10.1080/10407790190053888.

[94] Felix S. Schranner, Julian A. Domaradzki, Stefan Hickel & Nikolaus A. Adams. Assessing the numerical dissipation rate and viscosity in numerical simulations of fluid flows. *Computers & Fluids* **114**: 84–97, 2015. doi: 10.1016/j.compfluid.2015.02.011.

[95] Brandon Williams. Estimating grid-induced errors in unsteady CFD solutions using a discrete error transport equation. In: *Proceedings of the 48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition, Orlando, FL, USA, 4 – 7 January 2010,* article id 1361.

[96] NEK SHARAN, GEORGIOS MATHEOU & PAUL E. DIMOTAKIS. Mixing, scalar boundedness, and numerical dissipation in large-eddy simulations. *Journal of Computational Physics* **369**: 148–172, 2018. DOI: 10.1016/j.jcp.2018.05.005.

[97] MOUTASSEM EL RAFEI, LÁSZLÓ KÖNÖZSY & ZEESHAN RANA. Investigation of numerical dissipation in classical and implicit large eddy simulations. *Aerospace* **4**(4): article ID 59, 2017. DOI: 10.3390/aerospace4040059.

[98] GUANGRUI SUN & JULIAN A. DOMARADZKI. Implicit LES using adaptive filtering. *Journal of Computational Physics* **359**: 380–408, 2018. DOI: 10.1016/j.jcp.2018.01.009.

[99] ED M. KOMEN, LEONARDO H. CAMILO, AFAQUE SHAMS, BERNARD J. GEURTS & BARRY KOREN. A quantification method for numerical dissipation in quasi-DNS and under-resolved DNS, and effects of numerical dissipation in quasi-DNS and under-resolved DNS of turbulent channel flows. *Journal of Computational Physics* **345**: 565–595, 2017. DOI: 10.1016/j.jcp.2017.05.030.

[100] GIACOMO CASTIGLIONI, GUANGRUI SUN & JULIAN A. DOMARADZKI. On the estimation of artificial dissipation and dispersion errors in a generic partial differential equation. *Journal of Computational Physics* **397**: article ID 108843, 2019. DOI: 10.1016/j.jcp.2019.07.041.

[101] ALEXANDROS SYRAKOS & APOSTOLOS GOULAS. Estimate of the truncation error of finite volume discretization of the Navier-Stokes equations on colocated grids. *International Journal for Numerical Methods in Fluids* **50**(1): 103–130, 2006. DOI: 10.1002/fld.1038.

[102] BANTWAL R. BALIGA & IURII LOKHMANETS. Generalized Richardson extrapolation procedures for estimating grid-independent numerical solutions. *International Journal of Numerical Methods for Heat & Fluid Flow* **26**(3/4): 1121–1144, 2016. DOI: 10.1108/HFF-10-2015-0445.

[103] LARS DAVIDSON. Large Eddy Simulations: How to evaluate resolution. *International Journal of Heat and Fluid Flow* **30**(5): 1016–1025, 2009. DOI: 10.1016/j.ijheatfluidflow.2009.06.006.

[104] SENTHILKUMARAN RADHAKRISHNAN & JOSETTE BELLAN. Explicitly-filtered LES for the grid-spacing-independent and discretization-order-independent prediction of a conserved scalar. *Computers & Fluids* **111**: 137–149, 2015. DOI: 10.1016/j.compfluid.2015.01.003.

[105] MATTHEW J. HARRIS. *Flow Feature Aligned Mesh Generation and Adaptation.* Disertační práce. University of Sheffield, Department of Mechanical Engineering, Sheffield, UK, 2013. Vedoucí disertační práce: Prof. Ning Qin.

[106] GIACOMO CASTIGLIONI & JULIAN A. DOMARADZKI. A numerical dissipation rate and viscosity in flow simulations with realistic geometry using low-order compressible Navier-Stokes solvers. *Computers & Fluids* **119**: 37–46, 2015. DOI: 10.1016/j.compfluid.2015.07.004.

[107] Mira Pashtrapanska, Jovan Jovanović, Hermann Lienhart & Franz Durst. Turbulence measurements in a swirling pipe flow. *Experiments in Fluids* **41**(5): 813–827, 2006. doi: 10.1007/s00348-006-0206-x.

[108] ANSYS, Inc. *ANSYS Fluent Customization Manual, Release 19.5*. ANSYS, Inc., Canonsburgh, PA, USA, 2019.

[109] ANSYS, Inc. *ANSYS Fluent Theory Guide, Release 19.5*. ANSYS, Inc., Canonsburgh, PA, USA, 2019.

[110] ANSYS, Inc. *ANSYS CFD-Post User's Guide, Release 19.5*. ANSYS, Inc., Canonsburgh, PA, USA, 2019.

Přílohy: výběr z publikovaných prací

Příloha 1 Vojtěch Turek. Improving performance of simplified computational fluid dynamics models via symmetric successive overrelaxation. *Energies* **12**(12): article id 2438, 2019. doi: 10.3390/en12122438.

Příloha 2 Tomáš Létal, Vojtěch Turek & Dominika Fialová. Nonlinear finite element analysis-based flow distribution model for engineering practice. *Chemical Engineering Transactions* **76**: 157–162, 2019. doi: 10.3303/CET1976027.

Příloha 3 Vojtěch Turek. On improving computational efficiency of simplified fluid flow models. *Chemical Engineering Transactions* **70**: 1447–1452, 2018. doi: 10.3303/CET1870242.

Příloha 4 Vojtěch Turek, Dominika Fialová & Zdeněk Jegla. Efficient flow modelling in equipment containing porous elements. *Chemical Engineering Transactions* **52**: 487–492, 2016. doi: 10.3303/CET1652082.

Příloha 5 Vojtěch Turek, Ladislav Bébar & Zdeněk Jegla. Simplified pressure drop and flow distribution modelling in radial catalytic converters. *Chemical Engineering Transactions* **39**: 853–858, 2014. doi: 10.3303/CET1439143.

Příloha 6 Vojtěch Turek, Jiří Hájek, Zdeněk Jegla & Petr Stehlík. Optimum design of distribution systems in heat exchangers. *Asia-Pacific Journal of Chemical Engineering* **6**(5): 750–759, 2011. doi: 10.1002/apj.516.

Příloha 1

Vojtěch Turek. Improving performance of simplified computational fluid dynamics models via symmetric successive overrelaxation. *Energies* **12**(12): article ID 2438, 2019. DOI: 10.3390/en12122438.

energies

MDPI

Article

Improving Performance of Simplified Computational Fluid Dynamics Models via Symmetric Successive Overrelaxation

Vojtěch Turek

Sustainable Process Integration Laboratory–SPIL, NETME Centre, Faculty of Mechanical Engineering, Brno University of Technology–VUT Brno, Technická 2, 616 00 Brno, Czech Republic; turek@fme.vutbr.cz

Received: 23 May 2019; Accepted: 21 June 2019; Published: 25 June 2019

Abstract: The ability to model fluid flow and heat transfer in process equipment (e.g., shell-and-tube heat exchangers) is often critical. What is more, many different geometric variants may need to be evaluated during the design process. Although this can be done using detailed computational fluid dynamics (CFD) models, the time needed to evaluate a single variant can easily reach tens of hours on powerful computing hardware. Simplified CFD models providing solutions in much shorter time frames may, therefore, be employed instead. Still, even these models can prove to be too slow or not robust enough when used in optimization algorithms. Effort is thus devoted to further improving their performance by applying the symmetric successive overrelaxation (SSOR) preconditioning technique in which, in contrast to, e.g., incomplete lower–upper factorization (ILU), the respective preconditioning matrix can always be constructed. Because the efficacy of SSOR is influenced by the selection of forward and backward relaxation factors, whose direct calculation is prohibitively expensive, their combinations are experimentally investigated using several representative meshes. Performance is then compared in terms of the single-core computational time needed to reach a converged steady-state solution, and recommendations are made regarding relaxation factor combinations generally suitable for the discussed purpose. It is shown that SSOR can be used as a suitable fallback preconditioner for the fast-performing, but numerically sensitive, incomplete lower–upper factorization.

Keywords: computational fluid dynamics; symmetric successive overrelaxation; preconditioning; performance

1. Introduction

In engineering practice, it is often the case that process equipment is designed according to various rules of thumb. No optimization is generally done and, at best, a single computational fluid dynamics (CFD) simulation is carried out to verify that the design meets the key requirements of the future operator of the apparatus. This means that suboptimal designs or solutions, potentially leading to operating problems, are not uncommon.

One of the ways to remedy the situation is to use simplified CFD models. In spite of them not being as accurate as the standard CFD models, it has been shown [1] that they can provide useful quantitative information. What is more, these models feature significantly shorter computational times and their application in optimization algorithms is therefore much less cumbersome. To obtain solutions even faster, however, the numerical methods used to solve the underlying linear systems of equations can also be preconditioned. This means that instead of solving the original linear system

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{1}$$

where **A** is the coefficient matrix, **x** the solution vector, and **b** the right-hand side vector, one considers the system

$$\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{b}. \tag{2}$$

here, **M** denotes the preconditioning matrix such that $\mathbf{M}^{-1}\mathbf{A}$ has a smaller condition number than **A** and, therefore, linear system (2) features better convergence. It should also be noted that $\mathbf{M}^{-1}\mathbf{A}$ often is not formed explicitly but, instead, $\mathbf{M}\mathbf{u} = \mathbf{v}$ is solved for various auxiliary vectors **u** and **v** within the numerical solution method itself.

There are many different preconditioning techniques (i.e., ways to choose **M**) available, and the selection of the best one for a particular purpose depends mainly on the type of equation that is being solved and the employed ordering of the variables. However, the most commonly used techniques are likely various flavors of incomplete lower–upper factorization (ILU) [2] (these were numerically investigated by Chapman et al. [3]) and symmetric successive overrelaxation (SSOR) [4]. Although SSOR was originally intended for symmetric matrices, it was shown [5] to also work when the matrices are not symmetric. Assuming the splitting $\mathbf{A} = \mathbf{D} + \mathbf{L} + \mathbf{U}$, where **D** is the diagonal of **A** and **L** and **U** its strictly lower and upper triangular parts, respectively, the SSOR preconditioning technique is applied within the numerical solution method as two SOR sweeps using different values of ω. The iteration process for auxiliary vectors **u**, **v** (depend on the actual solution method used) can then be written as

$$\begin{aligned} \mathbf{u}^{(k+1/2)} &= (\mathbf{D} + \omega_{\mathrm{F}}\mathbf{L})^{-1}[(1-\omega_{\mathrm{F}})\mathbf{D} - \omega_{\mathrm{F}}\mathbf{U}]\mathbf{u}^{(k)} + \omega_{\mathrm{F}}(\mathbf{D} + \omega_{\mathrm{F}}\mathbf{L})^{-1}\mathbf{v} \text{ (forward sweep)} \\ \mathbf{u}^{(k+1)} &= (\mathbf{D} + \omega_{\mathrm{R}}\mathbf{U})^{-1}[(1-\omega_{\mathrm{R}})\mathbf{D} - \omega_{\mathrm{R}}\mathbf{L}]\mathbf{u}^{(k+1/2)} + \omega_{\mathrm{R}}(\mathbf{D} + \omega_{\mathrm{R}}\mathbf{U})^{-1}\mathbf{v}, \text{ (backward sweep)} \end{aligned} \tag{3}$$

where ω_{F} and ω_{R} denote the corresponding forward and backward relaxation factors. In other words, direct application of the inverse of the preconditioning matrix, $\mathbf{M}^{-1}$, is replaced by preconditioned fixed point iteration.

The advantages of SSOR are evidenced by the existence of a multitude of papers discussing improved versions of this technique or its extensions to various specific applications. Bai [6] studied SSOR-like preconditioners for non-Hermitian positive definite linear systems, for which the respective matrix was either Hermitian-dominant or skew–Hermitian-dominant. The paper also discussed the results of numerical implementations, showing that Krylov subspace iteration methods, when accelerated using SSOR-like preconditioners, are efficient solvers for classes of non-Hermitian positive definite linear systems. A "shifted" version of SSOR for non-Hermitian positive definite linear systems with a dominant Hermitian part was proposed by Tan [7]. Zhang [8], on the other hand, introduced an SSOR-like preconditioner for saddle point problems with a dominant skew-Hermitian part. A class of hybrid preconditioning methods for accelerated solution of saddle point problems was discussed by Wang [9], while Chen et al. [10] proposed a version of SSOR suitable for preconditioning of large dense complex linear systems arising from three-dimensional electromagnetic scattering. Wu and Li [11] introduced a modified SSOR technique for the solution of Helmholtz equations.

Preconditioning can also be done block-wise. This was discussed, e.g., by Zhang and Cheng [12] in terms of large sparse saddle point problems, and by Huang and Lu [13], who focused on SSOR block preconditioners applied in image restoration. Because, in fact, preconditioning means obtaining an easily invertible approximation of the original matrix, one can also use SSOR for just this purpose as shown, for example, by Meng et al. [14] in the context of fast recovery of density 3D data from gravity data. Similarly, a massively-parallel GPU implementation of the conjugate gradient method, which uses the approximate inverse matrix derived from SSOR as the preconditioning matrix, was proposed by Helfenstein and Koko [15].

Performance comparisons of SSOR and other, simpler preconditioning techniques were presented, e.g., by Meyer [16], who focused on genomic evaluation and by Sanjuan et al. [17], who used SSOR to accelerate parallel wind field calculations. In the latter paper, the authors also evaluated a new, reordered sparse matrix storage format and showed that this format can markedly shorten computational time. This confirms the earlier findings of Duff and Meurant [18], who investigated the effect of ordering

on the convergence of the conjugate gradient method preconditioned, among others, using SSOR, or DeLong and Ortega [19], who focused on parallel implementations of SOR in terms of natural and multicolor orderings. Chen et al. [20], on the other hand, proposed a novel reordering technique for SSOR approximate inverse preconditioner, used together with a GPU-accelerated conjugate gradient solver, which should maximize the coalescing of global memory accesses.

Many SSOR-like, a priori preconditioned numerical solution methods using different splittings of the coefficient matrix have also been proposed for various types of problems. A three-parameter extension of the SSOR method intended for singular saddle point problems, which commonly arise, e.g., in fluid dynamics, was proposed by Li and Zhang [21]. A differently accelerated generalized three-parameter method for both singular and non-singular problems was introduced by Pan [22]. Similarly, a three-parameter unsymmetric SOR method for such saddle point problems was proposed, for example, by Liang and Zhang [23], who also discussed the choice of optimal values of the parameters. Many different SSOR-like methods are available for augmented systems, as well. Wang and Huang [24] introduced a four-parameter method, Louka and Missirlis [25] introduced a five-parameter extrapolated form of SSOR, and Najafi and Edalatpanah [26] introduced an improved version of the modified SSOR method for large sparse augmented systems, proposed earlier by Darvishi and Hessari [27]. Another improved SSOR method intended for the solution of augmented systems was proposed by Salkuyeh et al. [28]. In the case of complex systems, one can use, for instance, the accelerated method by Huang et al. [29], that is, an accelerated version of the method by Edalatpour et al. [30], in which the solution vector is split into two subvectors and different relaxation factors are used when solving for each of them. Likewise, one can employ the preconditioned variant of the generalized SSOR method by Hezari et al. [31] or the method by Salkuyeh et al. [32], which solves a real system obtained from the original, complex one. Block linear systems can be solved, e.g., using the block-preconditioned SSOR method by Pu and Wang [33].

The majority of the papers mentioned above discuss convergence (or at least semi-convergence) of the proposed methods, and many also include some information on the optimal selection of the relaxation factors. Kushida [34] focused on the estimation of convergence of the original SSOR preconditioner via a condition number, while general discussion related to SSOR-like methods for non-Hermitian positive definite linear systems was published in [35]. Augmented systems were addressed, for example, by Wang and Huang [36]. Similarly, there are papers focusing on convergence and optimal selection of the relaxation factors in the case of methods for block 2×2 linear systems [37], saddle point problems [38], parallel SSOR implementations [39], the Poisson equation [40], etc. In all these cases, however, convergence was investigated via spectral analysis, which is often prohibitively expensive [41]. The present paper, focusing on fast estimation of suitable SSOR relaxation factors in engineering practice, therefore, investigates the convergence experimentally using several different simplified 3D CFD flow models. The suitability of specific combinations of relaxation factors is assessed on the basis of mean computational times needed to reach converged steady-state solutions. The best-performing combinations of relaxation factors are then given together with the obtained relaxation factor trends.

2. Materials and Methods

Three different flow systems, with both the "U" and the "Z" flow arrangements ("U": outlet on the same side of the flow system as the inlet, "Z": outlet on the opposite side), were used to generate test cases. Moreover, in two of these three flow systems, the mesh fineness was also varied (coarser and finer mesh). This yielded ten flow system configurations in total, with simplified, cuboid cell-only meshes of different sizes ranging from ~6000 cells to ~41,000 cells (see Table 1). The meshes were generated automatically by the employed benchmarking software (see further) using the key parameters listed in Table 1. Due to the cuboid nature of the meshes, cell sizes were, in all computational domains, governed primarily by how many cell faces comprised a tube cross section (coarser mesh: 1 face only, finer mesh: 2×2 faces) and by the utilized cell growth factor. Sample meshes are shown in Figure 1.

Table 1. Flow system parameters: W, L, and H denote width, length, and height, respectively, p_r row pitch, and p_t tube pitch.

Parameter	Flow System 1	Flow System 2	Flow System 3
Headers ($W \times L \times H$)	40 × 320 × 40 mm	60 × 340 × 60 mm	100 × 340 × 100 mm
Inlet/outlet region (L)	60 mm	60 mm	60 mm
Tube bundle	2 inline rows, 20 tubes/row, $p_r = p_t =$ 15.6 mm	3 staggered rows (45°), 10 tubes/row, $p_r = 15.6$ mm, $p_t = 2p_r$	5 staggered rows (45°), 10 tubes/row, $p_r = 15.6$ mm, $p_t = 2p_r$
Tubes	⌀10 mm × 500 mm	⌀10 mm × 500 mm	⌀10 mm × 500 mm
Flow arrangement	"U" or "Z"	"U" or "Z"	"U" or "Z"
Mesh	coarser: ~6000 cells, finer: ~25,000 cells	coarser: ~8000 cells, finer: ~41,000 cells	coarser: ~16,000 cells

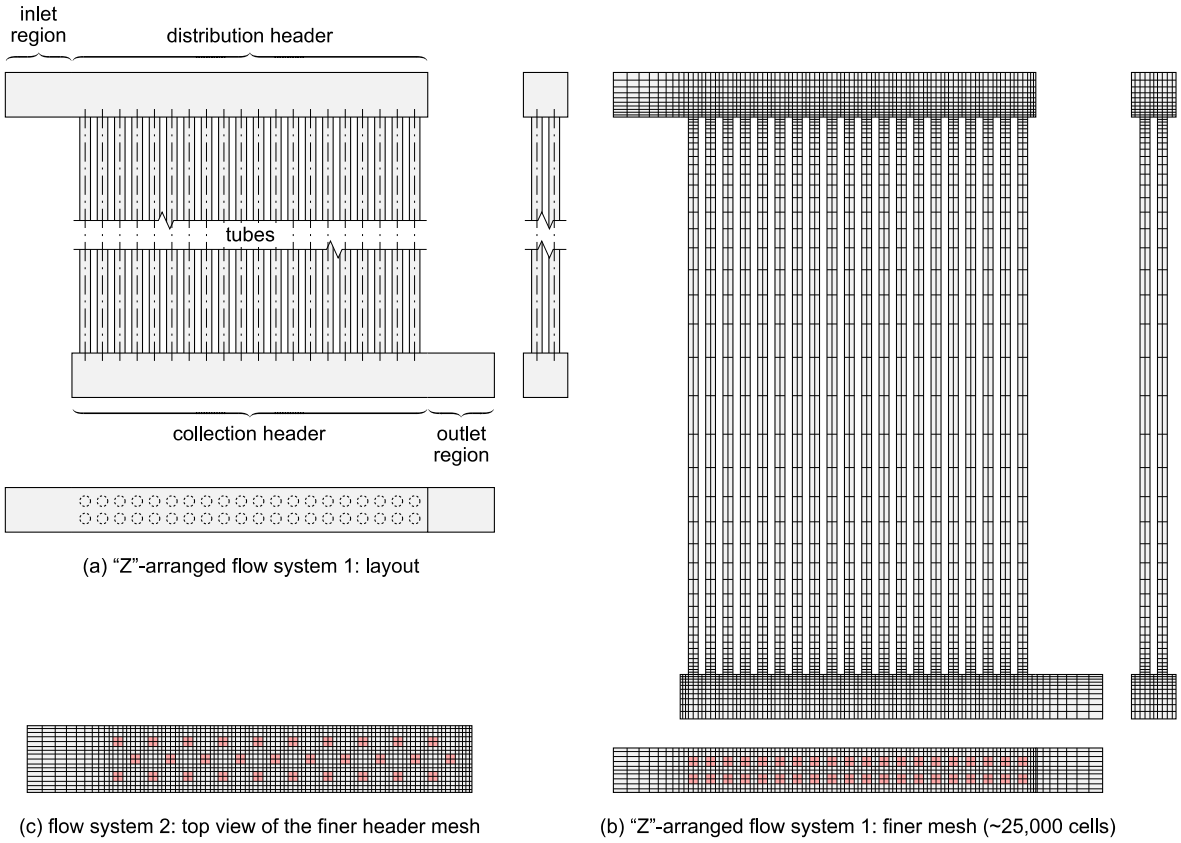


Figure 1. Sample simplified (cuboid cell-only) meshes generated automatically by the benchmarking software using the growth factor of 1.15; clockwise from top left: (**a**) layout of the "Z"-arranged flow system 1 from Table 1; (**b**) the corresponding finer mesh (~25,000 cells, groups of red cells in the top view denote locations of tube cross-sections); (**c**) top view of the finer mesh of the distribution header from flow system 2 (inlet region being on the left); other parts of this mesh, as well as the remaining meshes, were generated analogously.

As for boundary conditions, 0.5 kg/s of water at 300 K was fed into the inlet while pressure at the outlet was set to 101,325 Pa. All walls were adiabatic except for tube walls, where a specific heat flux of 15 kW/m^2 was set when the energy equation was enabled. Steady state simulations were carried out using the same CFD setup as in [42], that is, the SIMPLEC pressure-velocity coupling [43] was employed together with the Power Law discretization scheme [44]. Standard scaled residual limits, i.e., 10^{-3} for continuity and momentum and 10^{-6} for energy, were used. Only the natural ordering of the variables was considered in this study.

The SSOR preconditioning technique was paired with two widely used numerical solution methods, which were shown by an earlier study [42] to perform very well in simplified 3D CFD models. The conjugate gradient (CG) numerical solution method [45] was employed in the pressure correction equation. The momentum and energy equations, on the other hand, were solved using the bi-conjugate gradient stabilized numerical method with the minimization of residuals over L-dimensional subspaces (BiCGstab(L)) [46], with L = 1, 2, or 3. Performance of the ILU preconditioner (which is efficient, but the construction of the respective preconditioning matrix may not always be possible) was taken as the baseline.

The SSOR-preconditioned numerical solution methods were tested with various tuples of ω_F, $\omega_R = 0.1, 0.2, \ldots, 1.8, 1.9$ in successive steps, depending on the obtained results. Promising combinations of ω_F and ω_R were then taken as pivots, and their square neighborhoods were tested further—that is, all combinations of ω_F, ω_R with the respective values being $\omega - 0.04$, $\omega - 0.02$, ω, $\omega + 0.02$, and $\omega + 0.04$ were evaluated except for the original pivot point (ω_F, ω_R). In order to be able to compare SSOR to the baseline (ILU), all the ten meshes were also evaluated using the ILU-preconditioned combinations of solution methods. In total, 50,620 CFD model setups were tested.

The same benchmarking simplified 3D CFD Java software application was used as in [42], and, therefore, the reader is kindly referred to this paper for details (please note that the software is not publicly available). The benchmarking procedure itself was almost the same, as well, with the only difference being that the numbers of warm-up and test runs were smaller for the larger meshes (see Table 2). Such a measure was necessary to keep the times required to complete the respective benchmarks within reasonable bounds. This did not introduce any problems, because with larger cell counts all Java initializations and compilations had been finished within much less warm-up runs, and, therefore, it was not needed to carry out many of them before the timing phase. Mean test-run computational time was then taken as the final performance metric. Unlike in [42], however, only one machine (Intel Xeon E5 2698 v4 CPU, 128 GB RAM) was used instead of two largely disparate ones. The reason for this simplification was that, as shown in the respective paper, single-core computational times proved to be virtually identical, irrespective of whether the machine was a high-performance server or a regular laptop with an ultra-low voltage CPU. Please see the paragraph titled Supplementary Materials on how to obtain the data set containing all the mean computational times together with other relevant information.

Table 2. Numbers of warm-up and test runs and test case limits.

Mesh Size (Cells)	Warm-Up Runs	Test Runs	Iteration Limit	Computational Time Limit
~6000	30	50	1000	1800 s
~8000	30	50	1000	1800 s
~16,000	20	40	1500	2700 s
~25,000	10	30	2000	3600 s
~41,000	10	30	2000	3600 s

Because this study targeted fast computation, two kinds of limits were set in the solution process as detailed in Table 2. The first one concerned the number of CFD solver iterations, while the second one applied to the actual computational time. Any combination of numerical solution methods and preconditioning techniques which exceeded at least one of these two limits was marked as failing to reach a solution. Additionally, since robustness is one of the factors that must be considered when evaluating the suitability of numerical solution methods, no user interventions (e.g., changes to the internal residual limits of the numerical methods, CFD relaxation factors, etc.) were allowed during a solution process.

3. Results

This section is split into four parts. The first part presents the results pertaining to flow-only simulations. The second part discusses flow and energy transport simulations, in which only the energy equation was solved using SSOR-preconditioned numerical methods. The third part, on the other hand, summarizes data obtained via benchmarks, where SSOR was used for both the momentum and the energy equations. The last part then compares the SSOR-related data with results corresponding to the cases where solely the ILU preconditioning technique was used.

3.1. Flow-Only Simulations with SSOR-Preconditioned Momentum Equations

As mentioned above, the pressure correction equation was always solved using CG. This solution method was first coupled with SSOR, while the momentum equations were solved using BiCGstab(3):ILU. Within the several initial benchmarks, however, it became clear that CG:SSOR is wholly unsuitable. No matter the actual relaxation factors ω_F and ω_R, the majority of test cases either failed (mostly due to divergence) or resulted in very long computational times. Those setups which did not fail featured $\omega_F \approx \omega_R$ and, what is more, their amount decreased with an increasing mesh size, as shown in Figure 2. This was most probably a consequence of the fact that SSOR is sensitive to the ordering of variables. Only the ILU preconditioning technique was, therefore, used for the pressure correction equation, from then on.

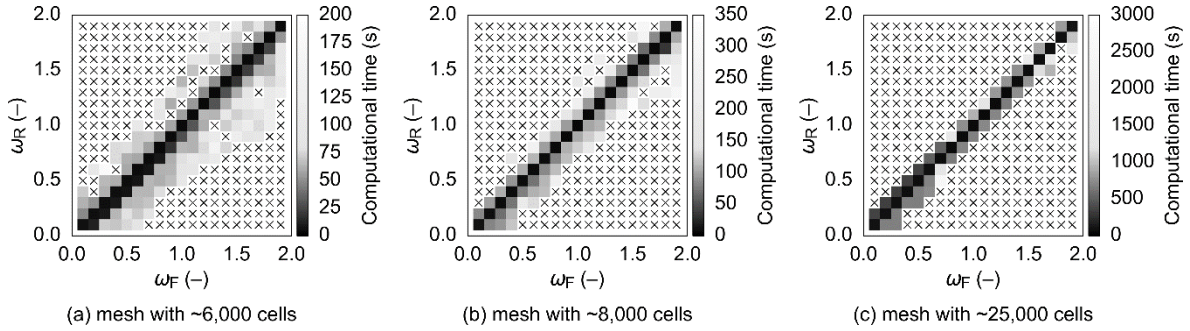


Figure 2. Mean computational times for three different meshes and various combinations of ω_F and ω_R, the pressure correction equation was solved using CG:SSOR and the momentum equations using BiCGstab(3):ILU; diagonal crosses indicate failing combinations of ω_F and ω_R. Please note that the colorbars have been adjusted so that the best (i.e., the most relevant) combinations of ω_F, ω_R are easily identifiable. Further, due to the necessary colorbar ranges, the respective lower limits have been set to zero even though the data start at higher values.

The shortest mean computational times obtained with BiCGstab(2) and BiCGstab(1) instead of BiCGstab(3) were, on average, ~80% and ~260% longer, respectively. Conversely, stability of the solution process was greater with these two methods, and fewer combinations of ω_F and ω_R resulted in failures (again, mostly because of divergence; see Figure 3). It is also evident from the figure that the best-performing combinations were around $\omega_F = \omega_R = 1.0$, while with ω_F below 0.5 or above 1.5 the computational times were much longer, or solution failures occurred. As for the CFD setup involving BiCGstab(3) in particular, mean computational times started at 2.56 s for the smallest mesh and 117.33 s when the largest mesh was used.

The solution behavior mentioned above was also observed for the larger meshes. Only BiCGstab(3):SSOR was, therefore, used to solve the momentum equations in the following flow-only benchmarks because of its superior performance. To generate these, the combinations of ω_F and ω_R obtained for the ten flow systems were sorted by mean computational time and the respective ordered sets were then used to find the most common combinations yielding the most favorable computational times. In other words, the best tuples (ω_F, ω_R) were taken as pivot points whose square neighborhoods were evaluated further.

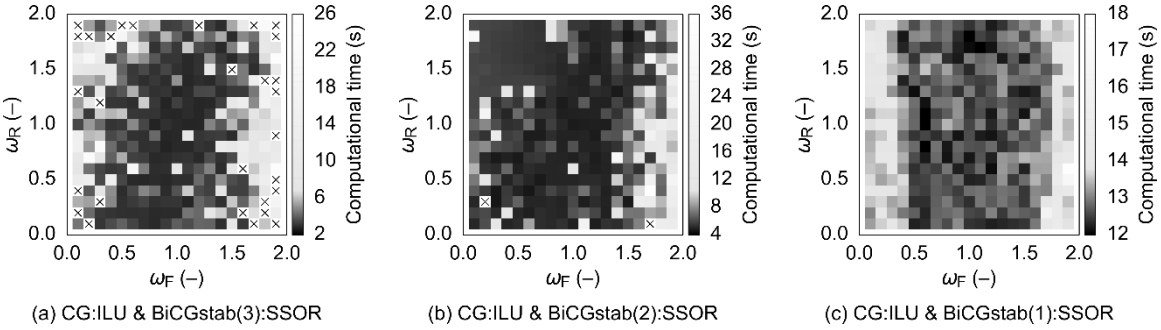


Figure 3. Mean computational times for the smallest mesh and the cases when the pressure correction equation was solved using CG:ILU and the momentum equations using BiCGstab(3):SSOR, BiCGstab(2):SSOR, and BiCGstab(1):SSOR; diagonal crosses indicate failing combinations of ω_F and ω_R. Please note that the colorbars have been adjusted so that the best (i.e., the most relevant) combinations of ω_F, ω_R are easily identifiable.

Example plots resulting from such evaluations are shown in Figure 4. Both pertain to the same mesh—the smallest one in this particular case. The plot on the left (Figure 4a) shows mean computational times for all the pivots and their respective neighborhoods. The plot on the right (Figure 4b) displays a cropped area corresponding to ω_F, $\omega_R \in [0.5, 1.5]$, where the best-performing combinations of relaxation factors are located. The dotted lines in both these plots represent the trend obtained using the standard weighted least squares method. Because the goal was to minimize computational time, the weights for individual combinations (ω_F, ω_R) were calculated as $w_i = (t_{min}/t_i)^4$, where t_{min} denotes the minimum computational time observed with a specific mesh and t_i denotes the mean computational time corresponding to the respective (i-th) combination of relaxation factors evaluated using this mesh. The fourth power of the computational time ratio instead of just the ratio itself was used to adequately limit the influence of combinations that yielded solutions in longer time frames. Weights for combinations leading to solution failures were set to zero.

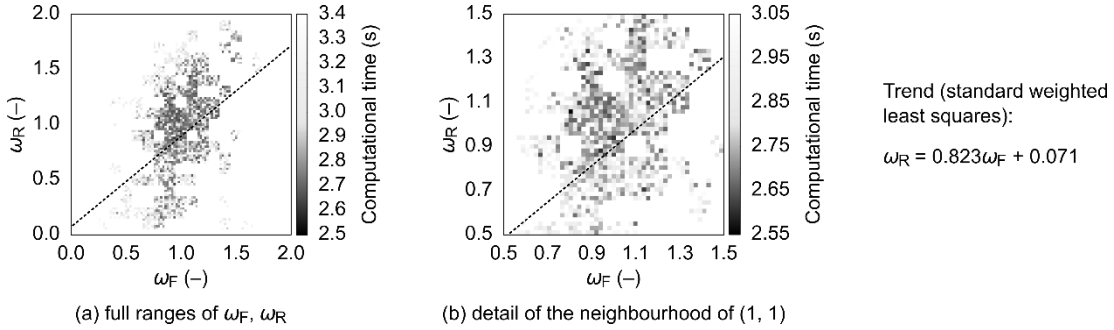


Figure 4. Mean computational times for the neighborhoods of pivot points corresponding to the smallest mesh; the pressure correction equation was solved using CG:ILU and the momentum equations using BiCGstab(3):SSOR; please note that, for the sake of clarity, the time ranges have been severely limited in both plots. Again, the colorbars have been adjusted so that the best (i.e., the most relevant) combinations of ω_F, ω_R are easily identifiable.

Trends for all the mesh sizes as well as the overall relaxation factor trends are listed in Table 3 and shown in Figure 5. Although the R^2 values are relatively low, this is caused by the fact that the data featured many less-relevant points scattered over the (0, 2) × (0, 2) relaxation factor domain, which were assigned small, but still non-zero weights. In any case, the standard errors for the trend coefficients are quite reasonable, and it is obvious that all the trends are very similar. Considering the actual values of the coefficients a and b and the fact that the best combinations of relaxation factors

featured $\omega_F \approx 0.9$, it follows that, when solving the momentum equations in flow-only scenarios, both SSOR sweeps should be slightly underrelaxed to gain the shortest computational time.

Table 3. Relaxation factor trends, $\omega_R = a\omega_F + b$, for individual mesh sizes and the overall relaxation factor trend for the solution of momentum equations using BiCGstab(3):SSOR; R^2 denotes the coefficient of determination and $SE(a)$ and $SE(b)$ the standard error values for the coefficients a, b.

Mesh Size (Cells)	a	b	R^2	$SE(a)$	$SE(b)$
~6000	0.823	0.071	0.479	0.014	0.009
~8000	0.843	0.094	0.457	0.015	0.007
~16,000	0.888	0.011	0.795	0.013	0.003
~25,000	0.798	0.051	0.593	0.019	0.007
~41,000	0.898	0.028	0.754	0.015	0.007
Overall trend	0.861	0.056	0.584	0.007	0.003

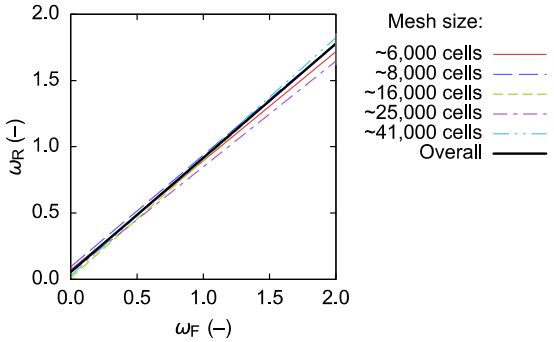


Figure 5. Relaxation factor trends for the solution of momentum equations using BiCGstab(3):SSOR; the overall trend was obtained using the merged data set and identical weights.

3.2. *Flow & Energy Transport Simulations with SSOR-Preconditioned Energy Equation*

Based on the solution behavior observed in the flow-only scenarios, only CG:ILU was used to solve the pressure correction equation in the flow and energy transport simulations. Momentum equations were also preconditioned only with ILU, while SSOR was used just for the energy equation. Various combinations of BiCGstab(L) for the momentum and energy equations were tested first, and the two most suitable combinations were then evaluated in detail using square neighborhoods of promising relaxation factor tuples (i.e., the pivot points).

The best results overall were obtained using BiCGstab(3) and BiCGstab(1) for the momentum equations and the energy equation, respectively (see Figure 6). This setup was relatively robust, most probably because of the better stability and smoothness of convergence resulting from the use of BiCGstab(1). Figure 7 shows the second-best combination, featuring only BiCGstab(2). On average, this setup was ~38% slower, and more combinations of ω_F and ω_R resulted in solution failures. It can also be seen that all the suitable relaxation factor tuples were in a relatively small neighborhood of (1, 1). Furthermore, with the SSOR-preconditioned energy equation, a significantly larger percentage of failures than before was due to slow convergence (that is, the respective iteration limits were exceeded).

The data sets mentioned above were then combined with data sets obtained by evaluating square neighborhoods of the promising tuples of ω_F and ω_R to get the corresponding relaxation factor trends. Again, the standard weighted least squares method was used with the weights being calculated in the same manner as before. Because the best setup and the second-best one (featured in Figures 6 and 7) were, at least in some cases, on par, the trends were calculated for both of them (see Table 4 and Figure 8). It is of note here that all benchmarks involving the largest mesh and BiCGstab(1) had failed. This suggests that the respective solution method simply is too slow when combined with SSOR. In any case, the best results were obtained with ω_F around 1.2 or 1.1 when BiCGstab(1) or BiCGstab(2) were used, respectively. This means that, given the calculated relaxation factor trends, ω_R should also

be a little above 1.0 (i.e., it is best to slightly overrelax both SSOR sweeps when solving the energy equation).

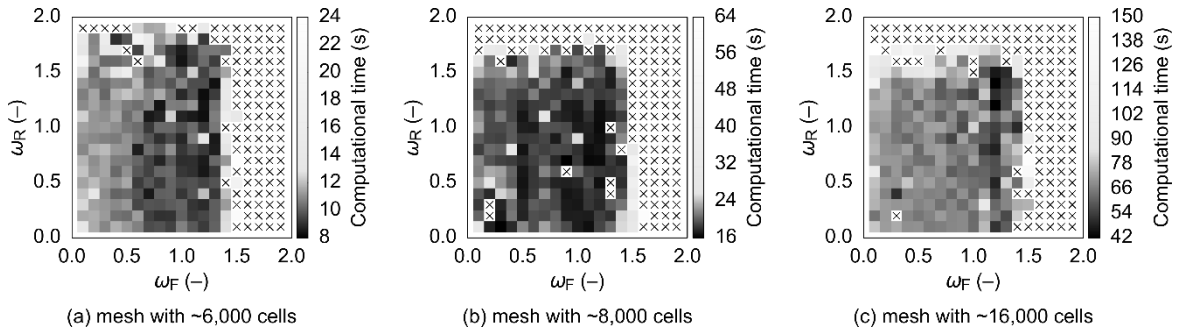


Figure 6. Mean computational times for three different meshes and various combinations of ω_F and ω_R, CG:ILU was used to solve the pressure correction equation, BiCGstab(3):ILU was used for the momentum equations, and BiCGstab(1):SSOR was used for the energy equation; diagonal crosses indicate failing combinations of ω_F and ω_R. Please note that the colorbars have been adjusted so that the best (i.e., the most relevant) combinations of ω_F, ω_R are easily identifiable.

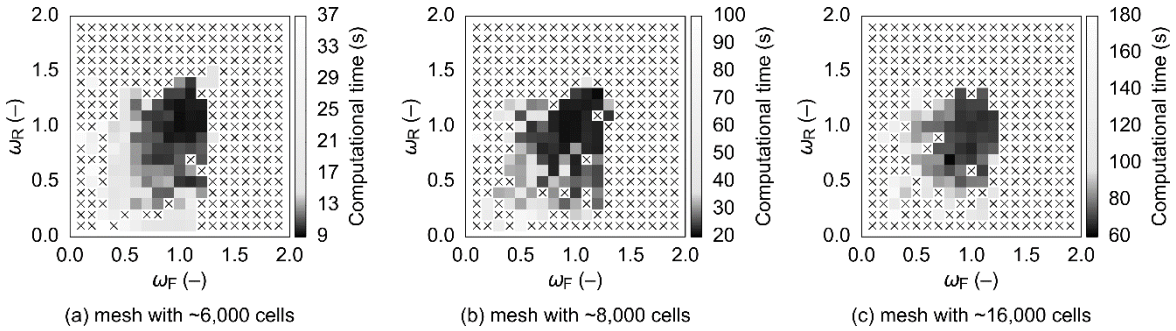


Figure 7. Mean computational times corresponding to the meshes from Figure 6 and the cases when the momentum equations were solved using BiCGstab(2):ILU, and the energy equation was solved using BiCGstab(2):SSOR; diagonal crosses indicate failing combinations of ω_F and ω_R. Please note that the colorbars have been adjusted so that the best (i.e., the most relevant) combinations of ω_F, ω_R are easily identifiable.

Table 4. Relaxation factor trends, $\omega_R = a\omega_F + b$, for individual mesh sizes and the overall relaxation factor trends for the two discussed setups using either BiCGstab(1):SSOR or BiCGstab(2):SSOR to solve the energy equation; R^2 denotes the coefficient of determination, $SE(a)$ and $SE(b)$ denote the standard error values for the coefficients a, b, and "n/a" denotes the fact that all the respective benchmarks had failed, and thus the trend could not be obtained.

Mesh Size (Cells)	BiCGstab(1)					BiCGstab(2)				
	a	***b***	R^2	***SE*(*a*)**	***SE*(*b*)**	***a***	***b***	R^2	***SE*(*a*)**	***SE*(*b*)**
~6000	0.867	0.047	0.555	0.027	0.009	0.960	0.007	0.929	0.008	0.004
~8000	0.762	0.120	0.564	0.023	0.013	0.971	0.005	0.936	0.007	0.003
~16,000	0.717	0.069	0.512	0.037	0.011	0.995	0.003	0.944	0.006	0.002
~25,000	0.905	0.012	0.888	0.009	0.002	0.964	0.012	0.922	0.013	0.006
~41,000	n/a	n/a	n/a	n/a	n/a	0.994	0.001	0.940	0.013	0.001
Overall trend	0.866	0.042	0.676	0.009	0.003	0.972	0.005	0.939	0.004	0.001

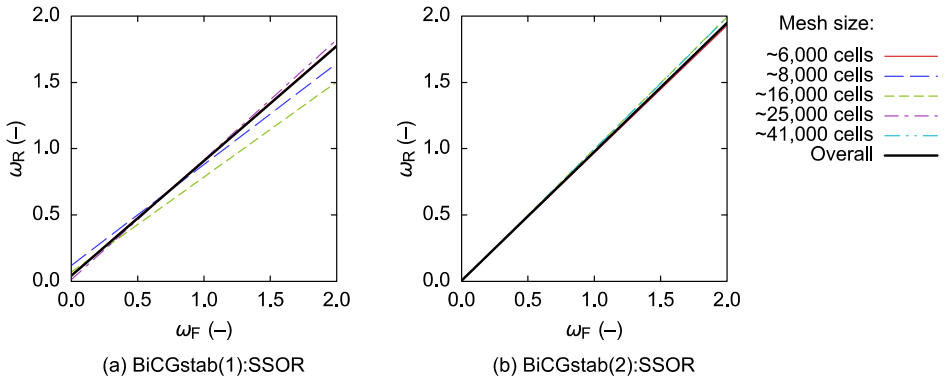


Figure 8. Relaxation factor trends for the two discussed setups using either BiCGstab(1):SSOR or BiCGstab(2):SSOR to solve the energy equation; the overall trends were obtained using the merged data sets and identical weights.

3.3. Flow & Energy Transport Simulations with SSOR-Preconditioned Momentum and Energy Equations

The last set of SSOR benchmarks involved both the momentum and the energy equations being preconditioned using this technique. However, because of a significantly larger relaxation factor domain (four factors had to be chosen instead of two), only ω_F, ω_R = 0.7, 0.8, . . . , 1.3 were evaluated, i.e., only the subdomain where the best-performing relaxation factor quadruples were expected to lie. Additionally, only the "Z"-arranged flow system meshes, and the two setups identified in Section 3.2 as the most promising ones, were considered. Such a reduction led to a decrease in the number of combinations to be evaluated from more than 2.6 million (10 meshes × 2 setups × 130,321 factor quadruples) to ~24 thousand (5 meshes × 2 setups × 2,401 factor quadruples). This still provided enough information to get a general sense of how solution processes would likely behave.

The respective benchmarks generally resulted in computational times and solution failure percentages comparable to those reached when just the energy equation was preconditioned using SSOR. As before, the failures mostly occurred due to slow convergence or—less often—because of divergence. Only with rare combinations of SSOR relaxation factors were the solution processes so slow that the respective time limit was exceeded.

The best-performing combinations of relaxation factors are listed in Table 5. It can be seen that with almost all meshes, the setup involving only BiCGstab(2) resulted in markedly longer computational times. Additionally, it should be noted that there were other combinations of factors providing similar numerical performance, but all of them were clustered around the values mentioned in the table.

Table 5. Combinations of relaxation factors resulting in the shortest mean computational times when both the momentum and the energy equations were preconditioned using SSOR; setup "B3/B1" denotes the case when BiCGstab(3) was used to solve the momentum equations and BiCGstab(1) the energy equation, while setup "B2/B2" corresponds to only BiCGstab(2) being used for both these equation types.

Mesh Size (Cells)	Setup	Momentum		Energy		Mean Computational Time
		ω_F	ω_R	ω_F	ω_R	
~6000	B3/B1	1.3	1.1	1.2	0.7	6.87 s
	B2/B2	1.0	0.9	0.9	1.2	8.34 s
~8000	B3/B1	1.0	1.0	0.8	1.0	14.46 s
	B2/B2	1.3	1.3	1.0	1.1	22.58 s
~16,000	B3/B1	1.3	1.3	1.0	0.7	81.65 s
	B2/B2	0.8	0.7	1.0	1.3	57.96 s
~25,000	B3/B1	1.1	0.7	1.2	1.2	99.16 s
	B2/B2	1.3	1.3	1.0	1.2	134.85 s
~41,000	B3/B1	1.1	0.8	1.2	0.8	325.18 s
	B2/B2	1.0	0.8	1.0	1.1	542.41 s

Visualizing the obtained data in one plot per data set is not possible because that would require four-dimensional plots. One could, however, fix the momentum relaxation factor tuple to, e.g., the values from the best-performing quadruple mentioned in Table 5, and then plot the corresponding two-dimensional energy relaxation factor map (or vice versa). Examples of such plots are shown in Figures 9 and 10.

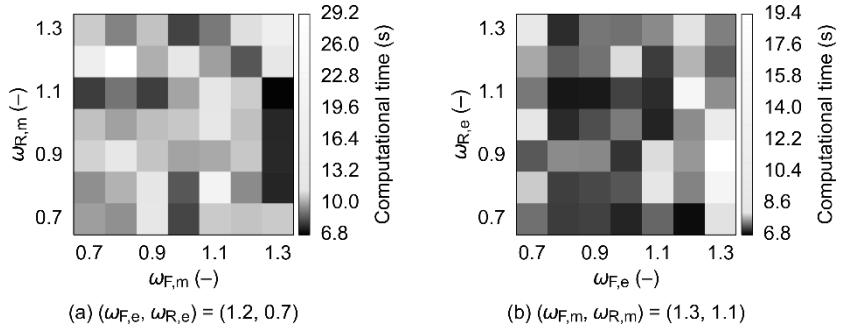


Figure 9. Two-dimensional plots of mean computational times obtained for the smallest "Z"-arranged mesh with (**a**) the energy relaxation factor tuple fixed to (ω_F, ω_R) = (1.2, 0.7) and (**b**) the momentum relaxation factor tuple fixed to (ω_F, ω_R) = (1.3, 1.1); BiCGstab(3):SSOR was used to solve the momentum equations and BiCGstab(1):SSOR the energy equation. Please note that the colorbars have been adjusted so that the best (i.e., the most relevant) combinations of ω_F, ω_R are easily identifiable.

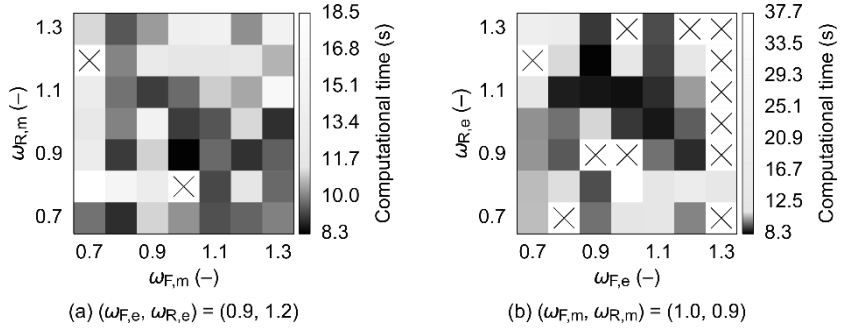


Figure 10. Two-dimensional plots of mean computational times obtained for the smallest "Z"-arranged mesh with (**a**) the energy relaxation factor tuple fixed to (ω_F, ω_R) = (0.9, 1.2) and (**b**) the momentum relaxation factor tuple fixed to (ω_F, ω_R) = (1.0, 0.9); BiCGstab(2):SSOR was used to solve both the momentum equations and the energy equation; diagonal crosses indicate failing combinations of ω_F and ω_R. Please note that the colorbars have been adjusted so that the best (i.e., the most relevant) combinations of ω_F, ω_R are easily identifiable.

Because, here, one must choose two relatively independent relaxation factor tuples, it is best to generate two trends for each combination of numerical solution methods. This can be done by "flattening" the four-dimensional data to two dimensions (while still considering all the data points). In other words, if one sought, e.g., the momentum relaxation factor trend, one would disregard the energy-related part of the relaxation factor quadruple and thus have multiple data points with different weights (calculated just as before) for each momentum relaxation factor tuple. The respective trends would then, again, be calculated via the standard weighted least squares method (see Tables 6 and 7 and Figures 11 and 12). From the results, it follows that for the momentum equations, the SSOR forward sweeps should generally be carried out with ω_F between ca. 1.1 and 1.3, while the backward sweeps should use $\omega_R \leq \omega_F$. As for the energy equation and BiCGstab(1), the forward sweep should, again, be slightly overrelaxed (ω_F up to ca. 1.2) and $\omega_R \leq \omega_F$, while with BiCGstab(2) ω_F should be around 1.0 (i.e., without any forward sweep relaxation) and $\omega_R \geq \omega_F$.

Table 6. Relaxation factor trends, $\omega_R = a\omega_F + b$, for individual mesh sizes and the overall relaxation factor trends for the setup where BiCGstab(3):SSOR was used to solve the momentum equations and BiCGstab(1):SSOR the energy equation; R^2 denotes the coefficient of determination and $SE(a)$ and $SE(b)$ the standard error values for the coefficients a, b.

Mesh Size (Cells)	Momentum					Energy				
	a	***b***	R^2	***SE*(*a*)**	***SE*(*b*)**	***a***	***b***	R^2	***SE*(*a*)**	***SE*(*b*)**
~6000	0.684	0.141	0.636	0.011	0.006	0.798	0.097	0.576	0.014	0.007
~8000	0.876	0.062	0.610	0.014	0.007	0.774	0.101	0.585	0.013	0.006
~16,000	0.957	0.000	0.995	0.001	0.000	0.985	0.000	0.937	0.005	0.000
~25,000	1.034	0.000	0.957	0.004	0.001	0.960	0.002	0.922	0.006	0.001
~41,000	0.931	0.001	0.914	0.006	0.001	0.892	0.001	0.905	0.006	0.001
Overall trend	0.936	0.013	0.893	0.003	0.001	0.949	0.011	0.884	0.003	0.001

Table 7. Relaxation factor trends, $\omega_R = a\omega_F + b$, for individual mesh sizes and the overall relaxation factor trends for the setup where BiCGstab(2):SSOR was used to solve both the momentum equations and the energy equation; R^2 denotes the coefficient of determination and $SE(a)$ and $SE(b)$ the standard error values for the coefficients a, b.

Mesh Size (Cells)	Momentum					Energy				
	a	***b***	R^2	***SE*(*a*)**	***SE*(*b*)**	***a***	***b***	R^2	***SE*(*a*)**	***SE*(*b*)**
~6000	0.864	0.054	0.786	0.009	0.005	0.963	0.031	0.856	0.008	0.004
~8,000	0.957	0.026	0.811	0.009	0.005	0.946	0.030	0.871	0.007	0.004
~16,000	1.007	0.007	0.845	0.009	0.004	0.980	0.016	0.904	0.007	0.003
~25,000	0.965	0.000	0.962	0.004	0.000	1.155	0.000	0.974	0.004	0.000
~41,000	0.993	0.002	0.930	0.006	0.001	1.007	0.001	0.961	0.004	0.001
Overall trend	0.965	0.011	0.882	0.003	0.001	0.990	0.010	0.923	0.003	0.001

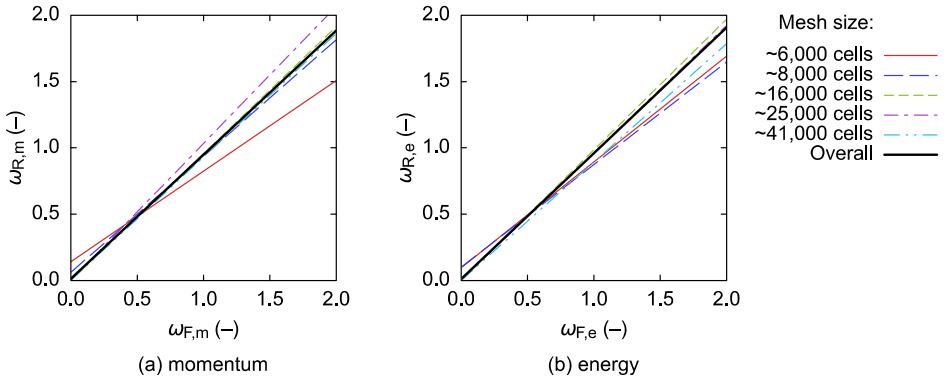


Figure 11. Relaxation factor trends for (**a**) momentum and (**b**) energy obtained with BiCGstab(3):SSOR and BiCGstab(1):SSOR as the solution methods for the momentum equations and the energy equation, respectively; the overall trends were obtained using the merged data sets and identical weights.

3.4. Comparison to ILU-Only Simulations

In order to be able to assess the potential benefit of using SSOR, the meshes listed in Table 1 were also evaluated via the two combinations of numerical solution methods from Table 5, but only with the ILU preconditioning technique being employed. The results, summarized in Table 8 and visually compared in Figure 13, suggest that utilizing SSOR leads to at least a ~23% increase (on average) in computational time.

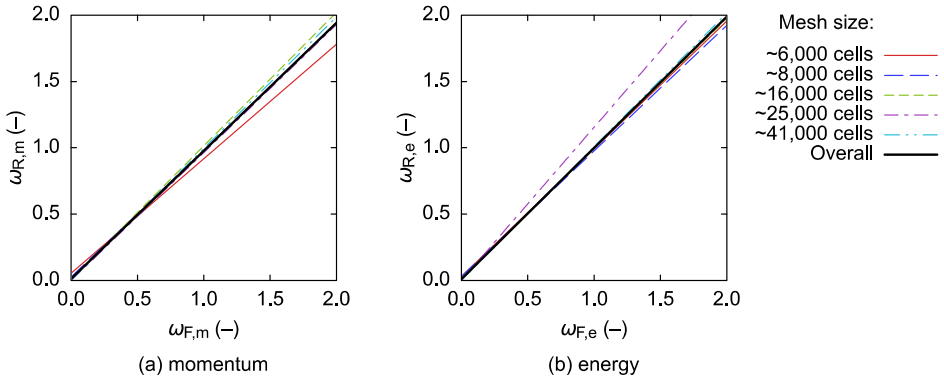


Figure 12. Relaxation factor trends for (**a**) momentum and (**b**) energy obtained with BiCGstab(2):SSOR as the solution method for both the momentum equations and the energy equation; the overall trends were obtained using the merged data sets and identical weights.

Table 8. Comparison of the mean computational times (s) obtained when solely the ILU preconditioning technique was used, and the overall best-case mean computational times reached with the momentum equations and/or the energy equation being preconditioned with SSOR.

Mesh Size (Cells)	ILU		SSOR		
	Momentum	Momentum & Energy	Momentum	Energy	Momentum & Energy
~6000	3.09	7.61	2.56	8.28	6.87
~8000	4.02	9.60	4.59	14.95	14.46
~16,000	12.93	30.92	15.60	42.13	57.96
~25,000	28.10	71.86	41.45	93.52	99.16
~41,000	119.33	294.51	117.33	368.33	325.18

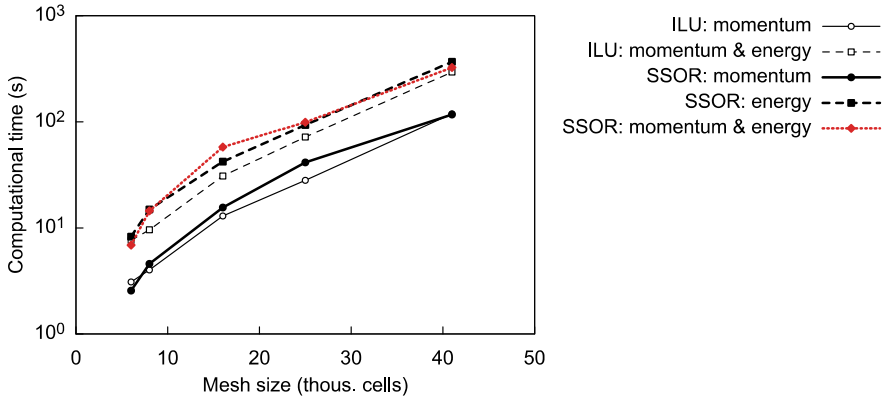


Figure 13. Comparison of the mean computational times obtained when solely the ILU preconditioning technique was used, and the overall best-case mean computational times reached with the momentum equations and/or the energy equation being preconditioned with SSOR.

4. Discussion

The aim of this study was to establish whether, in the case of simplified CFD models, the SSOR preconditioning technique can be a viable replacement for ILU. From the obtained data, it follows that SSOR should not be used in conjunction with CG to solve the pressure correction equation. When applied to the momentum and/or energy equations, computational times tend to be significantly longer even when the relaxation factors are chosen favorably (for the cases evaluated in this study, the increase was at least ~23% on average). However, because the SSOR preconditioning matrix can always be constructed, the respective techniques could be used in conjunction with ILU as a fallback option. From an engineering point of view, this would mean that ILU would be employed by default, and only in case of numerical issues would the CFD solver try to reach a converged solution using

SSOR. Such an approach would capitalize on the efficiency of ILU while maintaining reasonable numerical robustness due to the possibility of falling back to a technique with guaranteed existence of the preconditioning matrix. The resulting models would, ultimately, be much more suitable for implementation in optimization algorithms or for other use cases where large batches of simulations must be carried out without user intervention.

The best-performing combinations of numerical solution methods and SSOR forward and backward relaxation factors differ according to whether energy transport is included in the model or not. In the flow-only scenario, the momentum equations should preferably be solved using BiCGstab(3), with $\omega_F \approx 0.9$ and ω_R slightly less than ω_F, that is, both SSOR sweeps should be a little underrelaxed. Computational times obtained using other variants of BiCGstab(L) proved to be at least 80% longer. If also the energy transport is included and only the energy equation is preconditioned using SSOR, it is best to solve the momentum equations using BiCGstab(3) and the energy equation using BiCGstab(1). Here, both SSOR sweeps should be slightly overrelaxed, with $\omega_F \approx 1.2$ and $\omega_R \approx 1.1$. Similar performance can in some cases be obtained by employing BiCGstab(2) for both types of equations with the energy SSOR sweeps being overrelaxed using $\omega_F \approx \omega_R \approx 1.1$; however, a much greater solution failure probability can then be expected. If SSOR is utilized for both the momentum and the energy equations, then it is, again, preferable to use the combination of BiCGstab(3) and BiCGstab(1). The respective forward sweeps should be a little overrelaxed (ω_F between ca. 1.1 and 1.3 for the momentum, and up to ca. 1.2 for the energy), while the backward sweeps should feature ω_R slightly lower than ω_F. The best-case computational times obtained with BiCGstab(2) proved to be up to ~67% longer and, therefore, the use of this numerical solution method is discouraged in this scenario.

Supplementary Materials: The mean computational times together with other relevant information are available online at http://www.mdpi.com/1996-1073/12/12/2438/s1.

Funding: This research was funded by the Czech Republic Operational Programme Research, Development, and Education, Priority 1: Strengthening capacity for quality research, grant No. CZ.02.1.01/0.0/0.0/15_003/ 0000456 "Sustainable Process Integration Laboratory—SPIL".

Conflicts of Interest: The author declares no conflict of interest.

References

1. Turek, V.; Fialová, F.; Jegla, Z. Efficient flow modelling in equipment containing porous elements. *Chem. Eng. Trans.* **2016**, *52*, 487–492. [CrossRef]
2. Meijerink, J.A.; van der Vorst, A.H. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Math. Comput.* **1977**, *31*, 148–162. [CrossRef]
3. Chapman, A.; Saad, Y.; Wigton, L. High-order ILU preconditioners for CFD problems. *Int. J. Numer. Methods Fluids* **2000**, *33*, 767–788. [CrossRef]
4. Young, D.M. On the accelerated SSOR method for solving large linear systems. *Adv. Math.* **1977**, *23*, 215–271. [CrossRef]
5. Birken, P.; Gassner, G.; Haas, M.; Munz, C.-D. Preconditioning for modal discontinuous Galerkin methods for unsteady 3D Navier–Stokes equations. *J. Comput. Phys.* **2013**, *240*, 20–35. [CrossRef]
6. Bai, Z.-Z. On SSOR-like preconditioners for non-Hermitian positive definite matrices. *Numer. Linear Algebra* **2016**, *23*, 37–60. [CrossRef]
7. Tan, X.-Y. Shifted SSOR-like preconditioner for non-Hermitian positive definite matrices. *Numer. Algorithms* **2017**, *75*, 245–260. [CrossRef]
8. Zhang, J.-L. On SSOR-like preconditioner for saddle point problems with dominant skew-Hermitian part. *Int. J. Comput. Math.* **2019**, *96*, 782–796. [CrossRef]
9. Wang, Z.-Q. On hybrid preconditioning methods for large sparse saddle-point problems. *Linear Algebra Appl.* **2011**, *434*, 2353–2366. [CrossRef]
10. Chen, J.; Liu, Z.; Cao, N.; Yong, B. Shifted SSOR preconditioning technique for improved electric field integral equations. *Microw. Opt. Technol. Lett.* **2013**, *55*, 304–308. [CrossRef]
11. Wu, S.-L.; Li, C.-X. A modified SSOR preconditioning strategy for Helmholtz equations. *J. Appl. Math.* **2012**, *2012*, 365124. [CrossRef]

12. Zhang, L.T.; Cheng, S.H. Modified block symmetric SOR preconditioners for large sparse saddle-point problems. *Appl. Mech. Mater.* **2013**, *241–244*, 2583–2586. [CrossRef]
13. Huang, Y.-M.; Lu, D.-Y. A preconditioned conjugate gradient method for multiplicative half-quadratic image restoration. *Appl. Math. Comput.* **2013**, *219*, 6556–6564. [CrossRef]
14. Meng, Z.; Li, F.; Xu, X.; Huang, D.; Zhang, D. Fast inversion of gravity data using the symmetric successive over-relaxation (SSOR) preconditioned conjugate gradient algorithm. *Explor. Geophys.* **2017**, *48*, 294–304. [CrossRef]
15. Helfenstein, R.; Koko, J. Parallel preconditioned conjugate gradient algorithm on GPU. *J. Comput. Appl. Math.* **2012**, *236*, 3584–3590. [CrossRef]
16. Meyer, K. Technical note: A successive over-relaxation preconditioner to solve mixed model equations for genetic evaluation. *J. Anim. Sci.* **2016**, *94*, 4530–4535. [CrossRef] [PubMed]
17. Sanjuan, G.; Margalef, T.; Cortés, A. Accelerating preconditioned conjugate gradient solver in wind field calculation. In Proceedings of the 2016 International Conference on High Performance Computing & Simulation (HPCS), Innsbruck, Austria, 18–22 July 2016; Smari, W.W., Ed.; IEEE: New York, NY, USA, 2016; pp. 294–301. [CrossRef]
18. Duff, I.S.; Meurant, G.A. The effect of ordering on preconditioned conjugate gradients. *BIT Numer. Math.* **1989**, *29*, 635–657. [CrossRef]
19. DeLong, M.A.; Ortega, J.M. SOR as a preconditioner. *Appl. Numer. Math.* **1995**, *18*, 431–440. [CrossRef]
20. Chen, Y.; Zhao, Y.; Zhao, W.; Zhao, L. A comparative study of preconditioners for GPU-accelerated conjugate gradient solver. In Proceedings of the 2013 IEEE 10th International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing, Zhangjiajie, China, 13–15 November 2013; IEEE: New York, NY, USA, 2013; pp. 628–635. [CrossRef]
21. Li, J.; Zhang, N.-M. A triple-parameter modified SSOR method for solving singular saddle point problems. *BIT Numer. Math.* **2016**, *56*, 501–521. [CrossRef]
22. Pan, C. On generalized SSOR-like iteration method for saddle point problems. *WSEAS Trans. Math.* **2017**, *16*, 239–247.
23. Liang, Z.-Z.; Zhang, G.-F. Modified unsymmetric SOR method for saddle-point problems. *Appl. Math. Comput.* **2014**, *234*, 584–598. [CrossRef]
24. Wang, H.-D.; Huang, Z.-D. On a new SSOR-like method with four parameters for the augmented systems. *East Asian J. Appl. Math.* **2017**, *7*, 82–100. [CrossRef]
25. Louka, M.A.; Missirlis, N.M. A comparison of the extrapolated successive overrelaxation and the preconditioned simultaneous displacement methods for augmented linear systems. *Numer. Math.* **2015**, *131*, 517–540. [CrossRef]
26. Najafi, H.S.; Edalatpanah, S.A. On the modified symmetric successive over-relaxation method for augmented systems. *Comp. Appl. Math.* **2015**, *34*, 607–617. [CrossRef]
27. Darvishi, M.T.; Hessari, P. A modified symmetric successive overrelaxation method for augmented systems. *Comput. Math. Appl.* **2011**, *61*, 3128–3135. [CrossRef]
28. Salkuyeh, D.K.; Shamsi, S.; Sadeghi, A. An improved symmetric SOR iterative method for augmented systems. *Tamkang J. Math.* **2012**, *43*, 479–490. [CrossRef]
29. Huang, Z.-G.; Wang, L.-G.; Xu, Z.; Cui, J.-J. Preconditioned accelerated generalized successive overrelaxation method for solving complex symmetric linear systems. *Comput. Math. Appl.* **2019**, *77*, 1902–1916. [CrossRef]
30. Edalatpour, V.; Hezari, D.; Salkuyeh, D.K. Accelerated generalized SOR method for a class of complex systems of linear equations. *Math. Commun.* **2015**, *20*, 37–52.
31. Hezari, D.; Edalatpour, V.; Salkuyeh, D.K. Preconditioned GSOR iterative method for a class of complex symmetric system of linear equations. *Numer. Linear Algebra Appl.* **2015**, *22*, 761–776. [CrossRef]
32. Salkuyeh, D.K.; Hezari, D.; Edalatpour, V. Generalized successive overrelaxation iterative method for a class of complex symmetric linear system of equations. *Int. J. Comput. Math.* **2015**, *92*, 802–815. [CrossRef]
33. Pu, Z.-N.; Wang, X.-Z. Block preconditioned SSOR methods for H-matrices linear systems. *J. Appl. Math.* **2013**, *2013*, 213659. [CrossRef]
34. Kushida, N. Condition number estimation of preconditioned matrices. *PLoS ONE* **2015**, *10*, e0122331. [CrossRef] [PubMed]

35. Zhang, C.; Miao, G.; Zhu, Y. Convergence on successive over-relaxed iterative methods for non-Hermitian positive definite linear systems. *J. Inequal. Appl.* **2016**, *2016*, 156. [CrossRef]
36. Wang, H.-D.; Huang, Z.-D. On convergence and semi-convergence of SSOR-like methods for augmented linear systems. *Appl. Math. Comput.* **2018**, *326*, 87–104. [CrossRef]
37. Liang, Z.-Z.; Zhang, G.-F. On SSOR iteration method for a class of block two-by-two linear systems. *Numer. Algorithms* **2016**, *71*, 655–671. [CrossRef]
38. Zhou, L.; Zhang, N. Semi-convergence analysis of GMSSOR methods for singular saddle point problems. *Comput. Math. Appl.* **2014**, *68*, 596–605. [CrossRef]
39. Cao, G.; Huang, Y.; Song, Y. Convergence of parallel block SSOR multisplitting method for block H-matrix. *Calcolo* **2013**, *50*, 239–253. [CrossRef]
40. Yang, S.; Gobbert, M.K. The optimal relaxation parameter for the SOR method applied to the Poisson equation in any space dimensions. *Appl. Math. Lett.* **2009**, *22*, 325–331. [CrossRef]
41. Barrett, R.; Berry, M.W.; Chan, T.F.; Demmel, J.; Donato, J.; Dongarra, J.; Eijkhout, V.; Pozo, R.; Romine, C.; Van der Vorst, H. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, 2nd ed.; Society for Industrial and Applied Mathematics (SIAM): Philadelphia, PA, USA, 1994; pp. 35–49.
42. Turek, V. On improving computational efficiency of simplified fluid flow models. *Chem. Eng. Trans.* **2018**, *70*, 1447–1452. [CrossRef]
43. Van Doormaal, J.P.; Raithby, G.D. Enhancement of the SIMPLE method for predicting incompressible fluid flows. *Numer. Heat Transf.* **1984**, *7*, 147–163. [CrossRef]
44. Patankar, S.V. A calculation procedure for two-dimensional elliptic situations. *Numer. Heat Transf.* **1981**, *4*, 409–425. [CrossRef]
45. Hestenes, M.R.; Stiefel, E. Methods of conjugate gradients for solving linear systems. *J. Res. Natl. Bur. Stand.* **1952**, *49*, 409–436. [CrossRef]
46. Sleijpen, G.L.; Fokkema, D.R. BiCGstab(l) for linear equations involving unsymmetric matrices with complex spectrum. *Electron. Trans. Numer. Anal.* **1993**, *1*, 11–32.

Příloha 2

Tomáš Létal, Vojtěch Turek & Dominika Fialová. Nonlinear finite element analysis-based flow distribution model for engineering practice. *Chemical Engineering Transactions* **76**: 157–162, 2019. doi: 10.3303/CET1976027.

CHEMICAL ENGINEERING TRANSACTIONS
VOL. 76, 2019
A publication of AIDIC
The Italian Association of Chemical Engineering
Online at www.aidic.it/cet
Guest Editors: Petar S. Varbanov, Timothy G. Walmsley, Jiří J. Klemeš, Panos Seferlis

ISBN 978-88-95608-73-0; **ISSN** 2283-9216
DOI: 10.3303/CET1976027

Nonlinear Finite Element Analysis-Based Flow Distribution Model for Engineering Practice

Tomáš Létal*, Vojtěch Turek, Dominika Fialová

Institute of Process Engineering, Faculty of Mechanical Engineering, Brno University of Technology, Technická 2, 61600 Brno, Czech Republic
letal@fme.vutbr.cz

In engineering practice, it is common that heat transfer equipment containing tube bundles are designed under the assumption of uniform flow distribution. Such a flawed approach may easily lead to various operating problems (increased local fouling rates, mechanical failures, etc.) and significantly shortened service life. Accordingly, knowing the flow pattern in the bundle is crucial to proper design of the respective apparatuses. Although computational fluid dynamics (CFD) models yield very accurate data, due to their inherent computational cost they are not really suitable for evaluation of large sets of possible flow system geometries. Algebraic or otherwise greatly simplified models, on the other hand, are acceptable in terms of computational performance, but generally suffer from low accuracy and limited applicability to more complex meshes. This paper therefore proposes a computationally efficient flow distribution model whose principle is analogous to nonlinear finite element analysis (FEA). Unlike in many other simplified models, no special correction algorithms or user modifications are needed here because the underlying system of equations is solved in the matrix form and the corrector step is mesh-independent. Additionally, results provided by the model are compared to the data obtained using detailed CFD analyses of several different flow systems. Although the accuracy of the model does not match that of CFD, it can still be used at the beginning of a design process to discard the obviously unsuitable options, which would otherwise have to be evaluated via lengthy CFD simulations.

1. Introduction

Knowledge of the final fluid flow distribution among individual channels of a parallel flow system is crucial in many engineering fields. Although this information is useful primarily in the design stage, it can also provide answers in case of troubleshooting. Flow distribution data are used to assess performance and reliability of heat exchangers (in terms of fouling propensity or the resulting thermal and mechanical loading of the tube bundle), product quality (e.g. when a hydrocarbon fuel is cracked in a heated parallel flow system), etc. The most common approach to this problem nowadays is numerical investigation via standard CFD models. A multitude of such studies are therefore available ranging from those focusing on various header (Jiang et al., 2018a) or parallel flow channel (Jiang et al., 2018b) shapes, shell-and-tube (Labbadlia et al., 2017) or compact (Zhou et at., 2017) heat exchangers, microchannel (Wei et al., 2016) or fuel cell (Zhao et al., 2017) applications, solar thermal collectors (Wei et al., 2017), separation equipment (Chang et al., 2019), and other areas all the way to e.g. datacentre cooling (Yue et al., 2019). The results obtained this way are very accurate, but there is a significant cost in terms of computational complexity. Other factors that must be considered are the creation of the necessary mesh of sufficient quality and often a rather non-trivial setup of the CFD model itself. In other words, these models are suitable if a few apparatuses are to be analysed, but not in cases when a large batch of different geometries must be evaluated (e.g. when shape optimisation is to be carried out).
Modelling approaches based on CFD, or somewhat simplified CFD, can also be encountered. These are often employed to simulate less complex flow systems or when certain phenomena are less important from the modelling point of view and can thus be neglected. Here the range of studies is also very wide and includes e.g. inter-plate flow in plate heat exchangers (Yoon and Jeong, 2017), bifurcating distribution channels (Cao et al., 2018), or even ways to improve the numerical performance of the models themselves (Turek, 2018). The respective simulations, however, are still quite time-consuming with the necessary mesh creation being a

Paper Received: 11/03/2019; Revised: 27/03/2019; Accepted: 28/03/2019
Please cite this article as: Letal T., Turek V., Fialova D., 2019, Nonlinear Finite Element Analysis-Based Flow Distribution Model for Engineering Practice, Chemical Engineering Transactions, 76, 157-162 DOI:10.3303/CET1976027

relatively complex task as well. Such models are not really suitable in spite of their being partially simplified in comparison to the standard CFD ones.
The simplest flow models, on the other hand, usually employ a wire mesh instead of a fully-3D mesh and are often analytical in nature, thus also very fast. Their accuracy may suffer due to the many simplifications that are implemented, yet they are used in many engineering areas because of their efficiency and ease of automation. The most common applications include heating, ventilation, and air conditioning (HVAC) (Ye, 2017) and flow distribution systems with generally very limited numbers of rows of parallel channels (Hao et al., 2016). One can encounter even models for non-standard conditions such as supercritical flow (Liu et al., 2018). The inherent property of the models in question is that nonlinearities are included directly, which prohibits usage of matrix solvers and hence makes efficient evaluation of larger meshes very problematic.
The best research area to draw parallels from therefore seems to be water distribution networks, which, due to the sizes of such networks, necessitates matrix implementation. Here, however, the models are focused in a largely different direction. They consider simple network flow (with the network structure often being the sought result) and try to meet the specified local water demands (or aim to localise leakages, model the spreading of contaminants, etc.). In other words, these commonly work with just the network edge capacities instead of being concerned with the hydraulics-related phenomena. In rare cases though (see e.g. Dudar and Dudar, 2017), the fluid distribution network problem is approached from the perspective of finite element analysis (FEA) and such models then attempt to properly include also the pressure losses etc. The aim of the present study is to extend this modelling strategy to process and power equipment (e.g. heat exchangers) where the built-in flow systems are often much too complex for the simple flow models to be used.

2. Mathematical model

The model is based on Hooke's law applied to fluid flow in a channel, that is, mass flow rate is linearly dependent on the product of "compliance" of the respective channel (it being a function of hydraulic resistance) and pressure drop therein. Just as in case of many other simplified models, the effect of turbulence is not included. The basic equation governing the flow of a fluid with constant physical properties through a channel then is

$$\dot{m} = k\Delta p \tag{1}$$

in which $\dot{m}$ denotes mass flow rate, k compliance, and Δp pressure drop. The actual value of k for a given channel can be estimated from Eq(1) using the Darcy-Weisbach equation for Δp (White, 1998),

$$\Delta p = 0.5 f l d_{\mathrm{h}}^{-1} \rho v^2 \tag{2}$$

where f denotes the Darcy friction factor, l length of the channel, d_{h} its hydraulic diameter, ρ density of the fluid, and v its mean flow velocity. Channel cross-sectional area and other characteristics, which are necessary to estimate f, are calculated from the corresponding mesh properties.

2.1 Structure and mathematical representation of the quasi-3D mesh

The mesh consists of nodes which are interconnected either by straight, directed edges representing virtual flow channels, or T-shaped elements ("T-joints", a special set of at least three straight edges – see further) in which the flow is divided or combined. Straight edges, as well as the straight portions of a T-shaped element, are assumed to have constant cross-sections and are uniquely identifiable by the respective boundary nodes. For an arbitrary mesh, the following additional assumptions are made:

- The pressure gradient over the edge is given by the difference in total pressures in its boundary nodes.
- Physical properties of the fluid are constant within one iteration and are obtained using the mean edge pressure and the corresponding enthalpy (or temperature).

The general set of equations that must hold for an arbitrary edge can then be written as

$$\mathbf{K}_{ij}\mathbf{p}_{ij} = \dot{\mathbf{m}}_{ij}, \quad \text{with} \quad \mathbf{K}_{ij} = \begin{bmatrix} k_{ii} & k_{ij} \\ k_{ji} & k_{jj} \end{bmatrix}, \ \mathbf{p}_{ij} = \begin{bmatrix} p_i \\ p_j \end{bmatrix}, \ \text{and} \ \dot{\mathbf{m}}_{ij} = \begin{bmatrix} \dot{m}_i \\ \dot{m}_j \end{bmatrix} \tag{3}$$

in which $\mathbf{K}_{ij}$ denotes the compliance matrix of the directed edge connecting the nodes i and j, $\mathbf{p}_{ij}$ the respective vector holding pressures in the boundary nodes, and $\dot{\mathbf{m}}_{ij}$ the vector of mass flow rates in these nodes. It can be shown that the compliance matrix generally attains the following form:

$$\mathbf{K}_{ij} = \begin{bmatrix} k_{ij} & -k_{ij} \\ -k_{ij} & k_{ij} \end{bmatrix}, \text{ where } \quad k_{ij} = \frac{2\rho_{ij} A_{ij} d_{\mathrm{h},ij}^2}{C_{ij}\mu_{ij} l_{ij}} \tag{4}$$

In the equation above, A_{ij} is the cross-sectional area of the respective edge, C_{ij} the constant from the formula for calculation of laminar friction factor, and μ_{ij} the dynamic viscosity of the fluid. Should an external acceleration

field be in effect (e.g. due to gravity), also the hydrostatic pressure head must be included via $p_{\mathrm{h},ij} = \rho_{ij}(\mathbf{a}_{ij} \cdot \mathbf{l}_{ij})$, in which $\mathbf{a}_{ij}$ and $\mathbf{l}_{ij}$ denote the external acceleration vector and the vector representing the directed edge, respectively. This yields a new set of equations replacing Eq(3),

$$\begin{bmatrix} k_{ij} & -k_{ij} \\ -k_{ij} & k_{ij} \end{bmatrix} \begin{bmatrix} p_i - p_{\mathrm{h},ij} \\ p_j \end{bmatrix} = \begin{bmatrix} \dot{m}_i \\ \dot{m}_j \end{bmatrix} \quad \text{or} \quad \mathbf{K}_{ij}\mathbf{p}_{ij} = \dot{\mathbf{m}}_{ij} + \dot{\mathbf{m}}_{\mathrm{h},ij} \tag{5}$$

where $\dot{\mathbf{m}}_{\mathrm{h},ij}$ is the mass flow rate correction vector accounting for the effect of the external acceleration field.
In case of a set of interconnected edges, there must be at least one source node and at least one (different) sink node. In other words, while in the majority of nodes the sums of mass flow rates must be zero, in the sources and sinks these attain non-zero values. The solution is then computed iteratively. At the beginning of this process, the complete set of equations is constructed from Eq(5) for all the edges in the set with the elements in the compliance matrix and the right-hand side (RHS) vector being obtained from the boundary conditions and the initial estimate. In the next step (i.e., the predictor step), the set of equations is solved, yielding a new estimate of the pressure vector. This vector corresponds to the system of equations not including any nonlinearities, which are considered in the subsequent corrector step (see Sections 2.2 and 2.3). Lastly, the elements in the compliance matrix and the RHS vector are updated, and a new predictor step is carried out. Dudar and Dudar (2017) suggest that the next-iteration (I + 1) value of k_{ij} should be calculated using $k_{ij,I+1} = k_{ij,I}(\dot{m}_{ij,I,\mathrm{corr}}/\dot{m}_{ij,I,\mathrm{lin}})$, where $\dot{m}_{ij,I,\mathrm{corr}}$ denotes the corrected mass flow rate including the effect of nonlinearities (i.e., from the corrector step) and $\dot{m}_{ij,I,\mathrm{lin}}$ the linear estimate from the predictor step. This results in relatively poor convergence behaviour and requires additional relaxation. In the present study, the elements of the compliance matrix are therefore updated using the square root of the mass flow ratio as indicated in Figure 1. Although this approach results in longer computational times, convergence is much smoother, and no additional relaxation is needed.

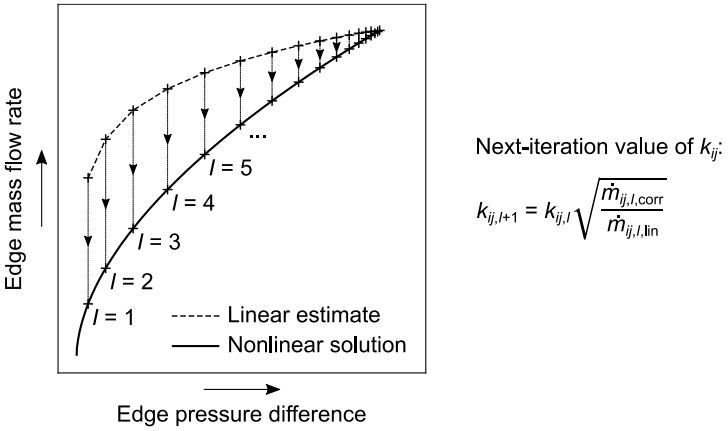


Figure 1: Typical convergence history of the mass flow rate through an edge connecting nodes i and j with the next-iteration (I + 1) value of k_{ij} being obtained using the square root of the ratio of the current corrected (nonlinear) and estimated (linear) flow rates

2.2 Including typical nonlinearities

Nonlinearities are introduced into the model for example by frictional pressure drop, minor losses, or the dependence of fluid properties on pressure (because pressure is unknown during the solution process). The most basic scenario, which will be discussed first, is flow through a straight edge where pressure changes only due to friction. This is for an arbitrary edge given by the Darcy-Weisbach equation in which the Darcy friction factor can be estimated e.g. using the Churchill approximation (Churchill, 1977). The resulting nonlinear dependence of mass flow rate on edge pressure difference is shown as the thick solid line in Figure 2. Because the pressure difference estimates are known from the predictor step and the fact that the respective dependence is monotonous, in the corrector step one can use e.g. the bisection method to quickly get the corresponding nonlinear solution, $\dot{m}_{ij,\mathrm{corr}}$. Zero mass flow rate can then be taken as one of the bounds for the bisection method while the other bound can be easily estimated using the first-order Taylor approximation (i.e., the tangent) at $\Delta p_{ij} = 0$. Should other common nonlinearities be included as well, the nonlinear solution curve would be different, but still monotonous. The same procedure to obtain $\dot{m}_{ij,\mathrm{corr}}$ would therefore be applicable.

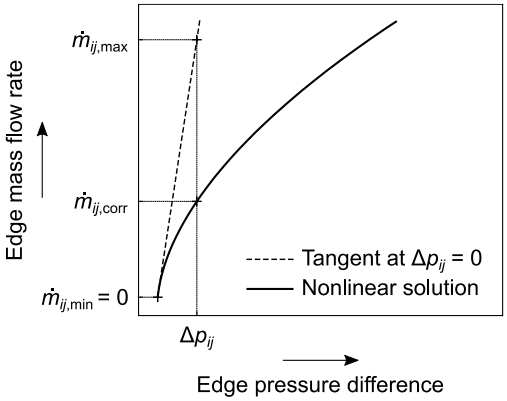


Figure 2: Estimation of bounds for the bisection method used in the corrector step for an edge connecting nodes i and j, for which the corresponding pressure difference is Δp_{ij}

2.3 T-joints

Nonlinearities are also introduced in T-joints where the fluid is split into or merged from multiple separate streams. Each T-joint comprises three or more T-connected mesh edges. Two edges represent the main channel (distributor or collector) and the remaining edges the attached branches. For modelling purposes, the main channel edges are directed towards the branch (i.e., edge flow velocity is positive if the fluid flows towards the virtual shared node) while the branch edges are directed out of the T-joint. Frictional pressure drop is dealt with just as in case of the regular (straight) edges. Minor losses are included via minor loss coefficients taken from relevant literature. The static pressure changed due to the flow being split or merged is obtained in the manner suggested by Bailey (1975).

Having a T-joint consisting of edges *ij*, *kj*, and *jl*, where the edges *ij* and *kj* form the main channel and the edge *jl* represents the branch (see the schematic in Figure 3), one can calculate the cross-sectional area ratio $R_A = A_{jl} / A_{ij} = A_{jl} / A_{kj}$ (it is assumed that $A_{ij} = A_{kj}$). Then it can be shown that the velocity ratio

$$R_v = |v_{jl}| \,/\, \max\{|v_{ij}|, |v_{kj}|\} \in [0, 2/R_A], \tag{6}$$

where v_{ij}, v_{kj}, and v_{jl} are the flow velocities in the main channel edges and the branch edge, respectively. The interval mentioned in Eq(6) also is the one to which the bisection method is applied to get the final value of R_v and thus the final edge velocities. Because the corresponding boundary node pressures (p_i, p_k, and p_l) are known from the predictor step, one can take as the initial estimate of v_{jl} for instance the value obtained from the minor loss equation with the pressure drop calculated from p_k and p_l. Eq(6) then yields for the current estimate of R_v from the bisection method the velocities v_{ij} and v_{kj} (these are interdependent because of the law of conservation of mass). Now all the velocity estimates are known and are used to check whether the main channel pressure difference, including the static pressure regain, is close enough to the one given by the known node pressures, p_i and p_k. If not, the range of the bisection method is halved accordingly, and the entire process is repeated. Example convergence history for the corrector step is shown in Figure 3.

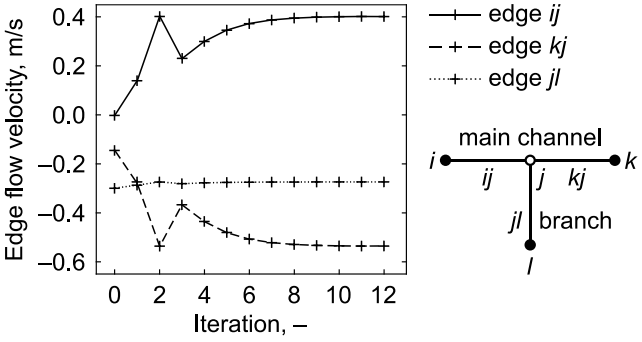


Figure 3: Typical convergence history for the corrector step carried out for a T-joint with boundary nodes i, k, and l; in this example, the flows are merged in the main channel while the respective flow direction is $i \rightarrow k$

2.4 Boundary conditions

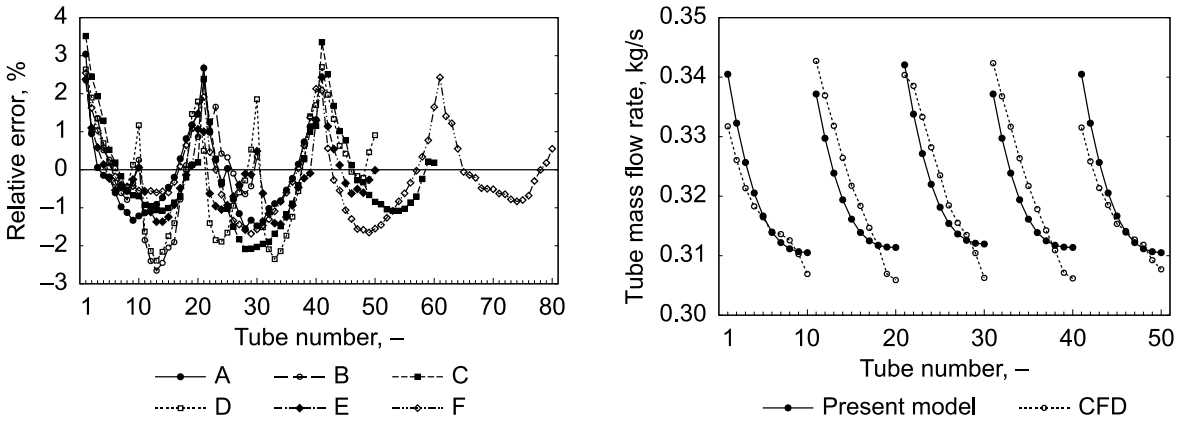
In each inlet and outlet node, either the mass flow rate or the pressure must be prescribed. Every mesh edge is assumed to be a closed channel with walls featuring a specified absolute roughness, which is then utilised for the estimation of Darcy friction factors. As of now, the system is assumed to be adiabatic. However, future enhancement of the model by adding heat transfer capability to the straight mesh elements is planned so that e.g. heat transfer through tube walls can be simulated.

3. Model validation

The model has been implemented using the NumPy scientific computing package (Oliphant, 2006). Several different flow systems similar to cross-flow tube bundles common e.g. in air coolers were evaluated (see Table 1). Both the U and Z flow arrangements were considered. The systems were adiabatic with cuboid headers and water at 300 K being used as the process medium. The obtained flow distributions within the bundles (i.e., individual tube mass flow rates) were then compared with data yielded by detailed CFD simulations. Relative tube mass flow rate errors calculated as $100(\dot{m}/\dot{m}_{\mathrm{CFD}} - 1)$ were always below 4 %. An example plot of these errors for a subset of the flow systems and a plot of the actual mass flow rates for one of them are shown in Figure 4. In terms of performance, all test runs finished in at most tens of seconds.

Table 1: Evaluated flow systems; all tubes were with inner diameters of 10 mm and lengths of 2,000 mm

Flow system	Headers (W × H × L)	Mass flow rate	Tube bundle
A	40 × 40 × 320 mm	6.4 kg/s	2 rows with 20 tubes each, 90°
B	40 × 40 × 280 mm	9.6 kg/s	3 rows with 10 tubes each, 60°
C	55 × 55 × 320 mm	19.2 kg/s	3 rows with 20 tubes each, 90°
D	55 × 55 × 280 mm	16.0 kg/s	5 rows with 10 tubes each, 60°
E	65 × 70 × 235 mm	16.0 kg/s	5 rows with 10 tubes each, 45°
F	70 × 70 × 320 mm	25.6 kg/s	4 rows with 20 tubes each, 90°



(a) relative errors (b) tube mass flow rates for flow system D

Figure 4: Tube mass flow rate relative errors with respect to the data from detailed CFD simulations of the U-arranged flow systems from Table 1 (a) and the comparison of mass flow rates through individual tubes of the U-arranged flow system D from Table 1

4. Conclusions

The model discussed in this paper represents a robust way to estimate flow distribution in a system consisting of a large set of parallel flow channels. The downsides of the present model are its being steady-state and adiabatic; however, implementation of the respective enhancements is planned for the near future.

The preliminary tests carried out using the developed computer code yielded flow distribution data with relative errors – compared to the results from detailed CFD simulations – of less than 4 %. The mass flow rate trends per individual tube rows in the bundles were not identical to those obtained using CFD, but the differences were still acceptable given the intended purpose of the model. As for its numerical performance, the time needed to evaluate a single flow system configuration never exceeded 30 s, whereas the corresponding detailed CFD simulations required units of hours to finish when run in parallel on 16 or more CPU cores.

In spite of the present model not being as accurate as CFD, its ultimate benefit lies in the fact that it can be easily automated. Process engineers can therefore use it to effortlessly estimate flow distribution in many different geometries, and then use CFD to evaluate in detail only the promising ones. In other words, the developed computer code can be considered a “pre-screening sieve” with which process engineers can save themselves significant amounts of time and effort.

Acknowledgments

This research has been supported by the project No. CZ.02.1.01/0.0/0.0/16_026/0008413 “Strategic partnership for environmental technologies and energy production”, which has been co-funded by the Czech Ministry of Education, Youth and Sports within the EU Operational Programme Research, Development and Education.

References

Bailey B., 1975, Fluid flow in perforated pipes, Journal of Mechanical Engineering Science, 17, 338–347.

Cao J., Kraut M., Dittmeyer R., Zhang L., Xu H., 2018, Numerical analysis on the effect of bifurcation angle and inlet velocity on the distribution uniformity performance of consecutive bifurcating fluid flow distributors, International Communications in Heat and Mass Transfer, 93, 60–65.

Chang Y., Wang H., Jin J., Liu Z., Lv W., 2019, Flow distribution and pressure drop in UZ-type mini-hydrocyclone group arranged in compact parallel manifolds, Experimental Thermal and Fluid Science, 100, 114–123.

Churchill S., 1977, Friction-factor equation spans all fluid and flow regimes, Chemical Engineering, 84, 91–92.

Dudar O.I., Dudar E.S., 2017, Application of 1 D finite element method in combination with laminar solution method for pipe network analysis, In: IOP Conference Series: Materials Science and Engineering, Vol 262, paper ID 012085.

Hao Y., Wang Y., Hu T., 2016, The flow distribution in the parallel tubes of the cavity receiver under variable heat flux, Applied Thermal Engineering, 108, 641–649.

Jiang Y., Qin J., Xu Y., Zhang S., Chetehouna K., Gascoin N., Bao W., 2018a, The influences of the header geometry on hydrocarbon fuel flow distribution in compact parallel channels, Aerospace Science and Technology, 79, 318–327.

Jiang Y., Xu Y., Qin J., Zhang S., Chetehouna K., Gascoin N., Bao W., 2018b, The flow rate distribution of hydrocarbon fuel in parallel channels with different cross section shapes, Applied Thermal Engineering, 137, 173–183.

Labbadlia O., Laribi B., Chetti B., Hendrick P., 2017, Numerical study of the influence of tube arrangement on the flow distribution in the header of shell and tube heat exchangers, Applied Thermal Engineering, 126, 315–321.

Liu J., Li H., Lei X., Zhang Q., Li L., 2018, An improved model on flow distributions of supercritical pressure water in parallel heated pipes, Applied Thermal Engineering, 130, 793–803.

Oliphant T.E, 2006, A guide to NumPy, Trelgol Publishing, Spanish Fork, UT, USA.

Turek V., 2018, On improving computational efficiency of simplified fluid flow models, Chemical Engineering Transactions, 70, 1447–1452.

Wei M., Boutin G., Fan Y., Luo L., 2016, Numerical and experimental investigation on the realization of target flow distribution among parallel mini-channels, Chemical Engineering Research and Design, 113, 74–84.

Wei M., Fan Y., Luo L., Flamant G., 2017, Design and optimization of baffled fluid distributor for realizing target flow distribution in a tubular solar receiver, Energy, Renewable Energy and Energy Storage Systems, 136, 126–134.

White F., 1998, Fluid Mechanics, 4th ed., McGraw-Hill Inc., New York, NY, USA.

Ye W.-B., 2017, Design method and modeling verification for the uniform air flow distribution in the duct ventilation, Applied Thermal Engineering, 110, 573–583.

Yoon W., Jeong J.H., 2017, Development of a numerical analysis model using a flow network for a plate heat exchanger with consideration of the flow distribution, International Journal of Heat and Mass Transfer, 112, 1–17.

Yue C., Zhang Q., Zhai Z., Ling L., 2019, Numerical investigation on thermal characteristics and flow distribution of a parallel micro-channel separate heat pipe in data center, International Journal of Refrigeration, 98, 150–160.

Zhao C., Yang J., Zhang T., Yan D., Pu J., Chi B., Li J., 2017, Numerical simulation of flow distribution for external manifold design in solid oxide fuel cell stack, International Journal of Hydrogen Energy, 42, 7003–7013.

Zhou J., Sun Z., Ding M., Bian H., Zhang N., Meng Z., 2017, CFD simulation for flow distribution in manifolds of central-type compact parallel flow heat exchangers, Applied Thermal Engineering, 126, 670–677.

Příloha 3

Vojtěch Turek. On improving computational efficiency of simplified fluid flow models. *Chemical Engineering Transactions* **70**: 1447–1452, 2018. doi: 10.3303/CET1870242.

CHEMICAL ENGINEERING TRANSACTIONS
VOL. 70, 2018
A publication of AIDIC
The Italian Association of Chemical Engineering
Online at www.aidic.it/cet
Guest Editors: Timothy G. Walmsley, Petar S. Varbanov, Rongxin Su, Jiří J. Klemeš

ISBN 978-88-95608-67-9; **ISSN** 2283-9216
DOI: 10.3303/CET1870242

On Improving Computational Efficiency of Simplified Fluid Flow Models

Vojtěch Turek

Sustainable Process Integration Laboratory – SPIL, NETME Centre, Faculty of Mechanical Engineering, Brno University of Technology – VUT Brno, Technická 2, 616 69 Brno, Czech Republic
turek@fme.vutbr.cz

Single-core computational efficiency of several iterative methods for numerical solution of nonsymmetric linear systems originating from simplified fluid flow models is evaluated together with different preconditioning techniques in terms of solution behaviour and the overall rate of convergence. A previously developed simplified 3D CFD model, which involves the solution of linear systems of orders usually from units to tens of thousands, is used to generate test cases. The respective Java software application employs the Parallel Colt linear algebra library. This is to ensure that the corresponding sparse matrix computations needed in different solution and preconditioning methods are carried out in as efficient a manner as possible to minimize the influence of the computer implementation itself. Measures (warm-up phase etc.) are taken to eliminate the effects of JVM-specific behaviour such as JIT compilation.

1. Introduction

Improving performance is crucial when it comes to computational methods used to evaluate objective functions in direct optimization algorithms. Because the simplified 3D CFD model proposed by Turek et al. (2016) is intended for just this purpose, i.e., fast evaluation of many possible flow system geometries in the early stage the apparatus design process, finding ways to further shorten computational time and improve convergence should be considered. This paper aims to do this by applying preconditioning techniques to the numerical methods used to obtain solutions of the underlying systems of linearized equations.
Even though the simplified 3D CFD model has a lot in common with the classical CFD models implemented in various simulation tools, it utilises a largely simplified and rather coarse mesh while turbulence is also modelled in a simplified manner. Consequently, the numerical behaviour of the model is more erratic and significantly more prone to divergence. The respective preconditioning techniques must therefore be selected carefully with respect to the specifics of the model. This, however, is not possible by just relying on literature discussing the classical CFD approach such as the monographs by Saad (2003) or Bruaset (1995) focusing on iterative methods or preconditioning techniques, respectively. Similarly, numerical performance-oriented investigations (e.g. Norris, 2000) or various papers presenting the respective preconditioning performance comparisons (e.g. Ma, 2000) are of limited use here because they were carried out with the classical CFD models. In other words, the only way to properly choose preconditioning techniques for the numerical solution methods involved in the simplified CFD model is to perform the respective benchmarks.
Additionally, only single-core performance is considered in this paper. The reason for such a limitation is that, given the simplified meshes containing relatively small amounts of cells, the computational overhead resulting from splitting a task over multiple cores would be significant. It is therefore much more economical in terms of CPU time to limit each computation to a single core and run these computations asynchronously on multiple cores within a suitable optimisation algorithm.

2. Benchmarking procedure

The paper by Turek et al. (2016), which discussed in detail the simplified 3D CFD model, also presented a Java software where this model had been implemented. Because originally the software did not include any

Please cite this article as: Turek V., 2018, On improving computational efficiency of simplified fluid flow models , Chemical Engineering Transactions, 70, 1447-1452 DOI:10.3303/CET1870242

preconditioning or benchmarking capabilities, these had been added. In keeping with (Goetz, 2005), however, the following measures were taken so that the results were not affected by Java-specific behaviour:

- 30 warm-up runs (i.e., full computations from initialization to converged state including all the related diagnostic printouts, data saving, etc.) were always carried out. This was to make sure that all Java initializations and compilations have been finished before the timing phase.
- The code was run with “XX:+PrintCompilation” and “verbose:gc” JVM options to check whether enough warm-up runs were carried out (see the item above). Similarly, “Xbatch” was used to serialize the compiler with the application.
- Even though the benchmarks were run with no other applications being opened at the same time, 50 test runs were carried out to eliminate the effect of various tasks which the OS performs in the background. Means and standard deviations were then calculated from the obtained data (no outliers were encountered in any of the tests).

For consistency and efficiency’s sake, all the involved numerical routines and sparse matrix computations utilised the Parallel Colt linear algebra library (Wendykier and Nagy, 2010). Moreover, the benchmarks were run on two disparate machines to find if this made any difference in terms of single core computational efficiency. These were as follows: (a) server: Intel Xeon E5-2698 v4 with 128 GB RAM, (b) laptop: Intel Core i5-6300U with 16 GB RAM.

2.1 Evaluated solution and preconditioning methods

The following commonly used iterative methods for numerical solution of linear systems were tested: conjugate gradient (CG) (Hestenes and Stiefel, 1952), conjugate gradient on the normal residuals (CGNR) (Saad, 2003), and bi-conjugate gradient stabilized with minimization of residuals over L-dimensional subspaces, L = 1, …, 8, (BiCGstab(L)) (Sleijpen and Fokkema, 1993). Other methods with smoother convergence such as generalized minimal residual (GMRES) (Saad and Schultz, 1986), quasi-minimal residual (QMR) (Freund and Nachtigal, 1991), or iterative refinement (IR) (Moler, 1967) were not considered because their rates of convergence generally are much slower than those of CG-based methods.

The solution methods listed above were used either with no preconditioner, or with one of these five preconditioner types when applicable: diagonal (i.e., Jacobi) (van der Vorst, 2003), incomplete LU factorization (ILU) (Meijerink and van der Vorst, 1977), dual-threshold incomplete LU factorization (ILUT) (Saad, 1994), symmetric successive overrelaxation (SSOR) (Young, 1977), and incomplete Cholesky factorization (ICC) (Manteuffel, 1980). Please note that in this study only the default relaxation parameters of 1.0 for the SSOR forward and backward sweeps were used, i.e., effectively the method reverted to the Gauss-Seidel form.

As for newer solution methods and preconditioning techniques, these generally are various multigrid implementations using several increasingly coarsened grids (e.g. Ruggiu et al., 2018) or are explicitly tailored to multi-core usage (e.g. Esmaily et al., 2018). The former approach is not feasible in case of the already much simplified meshes, while the latter one is out of the scope of this paper for the reasons discussed earlier.

2.2 Test cases

The flow system used for benchmarking purposes was a Z-arranged one with identical cuboid headers (width × height × length = 40 × 40 × 380 mm) and a tube bundle containing two inline rows of 20 tubes each. The respective tube length was 500 mm while the inner diameter was 10 mm. Two meshes of different finenesses were considered. The coarser one consisted of roughly 6,000 cells while the finer one contained roughly 25,000 cells. Boundary conditions were as follows: mass flow inlet with the rate set to 0.5 kg/s of water at 300 K and pressure outlet with the boundary pressure of 101,325 Pa. Heat flux of 25 kW was defined solely on the tube walls if the energy equation was enabled, otherwise all the walls were adiabatic. Considering other CFD model parameters, steady-state simulations were carried out using the SIMPLEC pressure-velocity coupling (van Doormaal and Raithby, 1984), Power Law discretization scheme (Patankar, 1981), and standard scaled residual limits (10^{-3} for continuity and momentum, 10^{-6} for energy).

Because CG and ICC require symmetric, positive-definite matrices, these two methods were only used for the pressure correction equation. Although SSOR together with its forward-only or backwards-only versions were, too, originally meant for such matrices, they have been shown by Birken et al. (2013) and Woźnicki (2001), respectively, to work quite well even when the respective matrix was not symmetric. What is more, construction of the SSOR preconditioning matrix cannot lead to breakdown (contrary to incomplete factorization methods) and therefore this technique was evaluated with all types of equations.

The set of test cases (i.e., combinations of the solution and preconditioning methods) was built in successive steps with respect to the obtained results. First, all feasible solution methods were evaluated on the server using the coarser mesh without preconditioning being applied to find the baseline performances. These combinations were subsequently also tested with only the pressure solver, or only the momentum solvers, being preconditioned. Promising combinations of the solution and preconditioning methods that were thus

obtained were evaluated in the third step. The fourth step in the benchmarking process constituted testing of the best combinations found so far, but this time with the energy equation being enabled and solved both without any preconditioning (again to find baseline performances) and with various preconditioners in place. This totalled to 934 cases.

Any combination which failed to converge within 1,000 iterations or 10 minutes was deemed unsuitable. The remaining 214 combinations were tested using the finer mesh, for which the suitability thresholds were 2,000 iterations or 20 minutes. The best 10 combinations in terms of (a) flow only and (b) flow and energy transport were in the end also evaluated using each mesh on the laptop along with the corresponding baseline combinations (i.e., without any preconditioning being applied). This constituted 60 additional cases. In summary, 1,148 cases were evaluated on the server and 60 cases on the laptop.

3. Results

The results are split into four categories covering the two meshes (coarser and finer) and two simulation scenarios (flow only and flow & energy transport). Overall rate of convergence is discussed first and then a few comments on smoothness of convergence and solution behaviour in general are provided.

3.1 Overall rate of convergence

Figures 1 through 4 show the ten best-performing combinations of the solution and preconditioning methods for each mesh and simulation scenario. Each of the graphs contains mean computational time and the corresponding standard deviation obtained using both the server and the laptop. Average improvements over the respective baseline combinations (those with no preconditioning being applied) are also provided. These values are the averages of the speedups obtained using the two machines and, if t_A denoted the actual computational time and t_B the baseline time, are computed as $t_B / t_A - 1$.

As can be seen from Figure 1, which corresponds to flow only and the coarser mesh, the best approach was to solve the pressure correction equation with CG preconditioned using ILU and the momentum equations with BiCGstab(3) using the same preconditioner. A converged solution was reached in 3.0 s with this setup. Compared to the respective average baseline time of 15.7 s, this yields the improvement of 4.0. It is also apparent that in almost all the listed cases the laptop performance was slightly better. The reason for this might lie in the fact that the laptop CPU core ran during benchmarks at much higher clock rate than the server CPU core (2.93 GHz vs. 2.20 GHz).

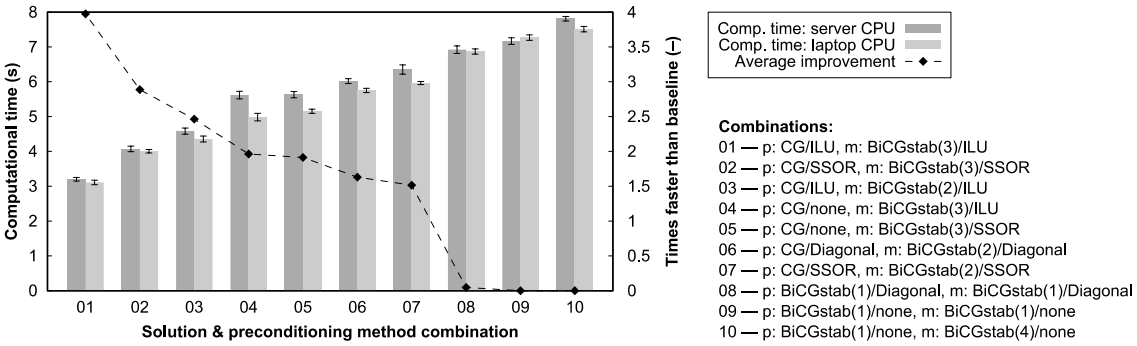


Figure 1: Ten best-performing combinations of solution and preconditioning methods in case of simulation of only the flow using the coarser mesh (roughly 6,000 cells)

The results were very similar when the finer mesh was used (see Figure 2). Again, the best combination was CG/ILU for the pressure correction equation and BiCGstab(3)/ILU for the momentum equations, with the respective computational time being 55.7 s. Given this combination's baseline time of 308.6 s, the average speedup was 4.5. Furthermore, Figure 2 shows that even in this case the laptop CPU slightly overperformed the server one.

When the energy equation was enabled while dealing with the coarser mesh (see Figure 3), in all the ten best cases CG/ILU was used to solve the pressure correction equation; however, BiCGstab(2)/ILU overperformed BiCGstab(3)/ILU. As for the energy solver, the best combination used BiCGstab(2)/SSOR. The corresponding speedup was 3.6 (on average, computational time decreased from 36.7 s to 8.0 s). The notable increase in computational time over the flow-only scenarios was caused primarily by much more iterations being needed for the energy scaled residual to reach the standard limit of 10^{-6}. Additionally, here the server CPU performed slightly better than the laptop one. This might be because larger amounts of varied data and

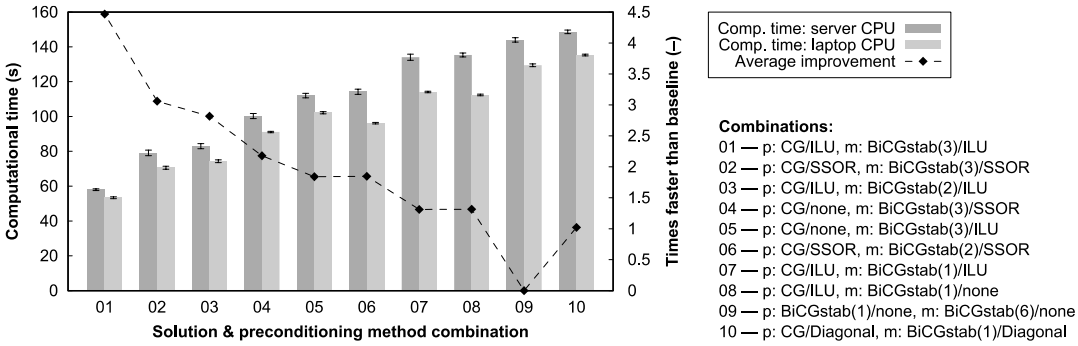


Figure 2: Ten best-performing combinations of solution and preconditioning methods in case of simulation of only the flow using the finer mesh (roughly 25,000 cells)

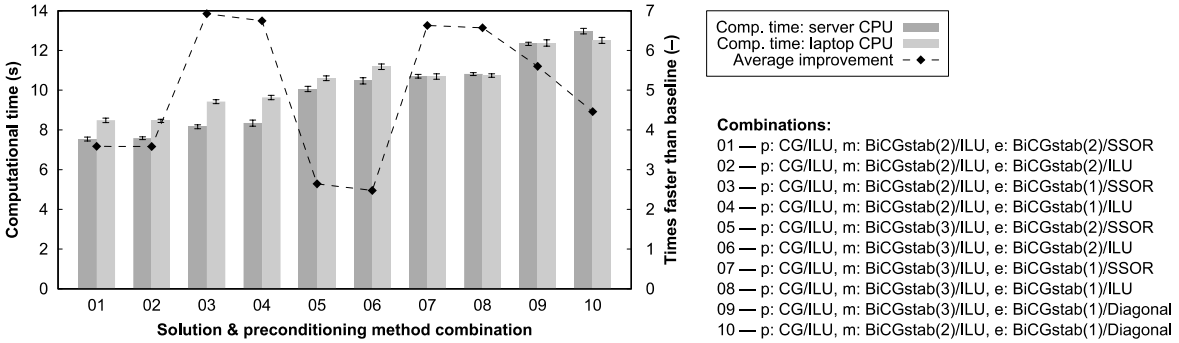


Figure 3: Ten best-performing combinations of solution and preconditioning methods in case of simulation of flow and energy transfer using the coarser mesh (roughly 6,000 cells)

instructions were being handled and the fact that the server CPU had an order of magnitude more L1, L2, and L3 caches.

Results related to the finer mesh, which are shown in Figure 4, are quite similar to those shown in Figure 3 in terms of the pressure correction and momentum solvers. Still, there are significant differences in the utilized energy solvers – one can see many combinations with various unpreconditioned versions of BiCGstab(*L*) and, in case of the last combination, CGNR with diagonal preconditioning in place. The speedups, therefore, vary notably, i.e., combinations with all solvers being preconditioned perform roughly 10 to 17 times faster than their baseline variants, while with unpreconditioned energy solver the speedups are only about 3. In case of the best-performing combination, the average computational time dropped from 722.7 s to 69.2 s. It should also be mentioned that a much larger percentage of cases resulted without user intervention in divergence when the energy equation was enabled. Given the purpose of the original simulation tool and its having to be as autonomous as possible, the respective combinations were discarded as unsuitable.

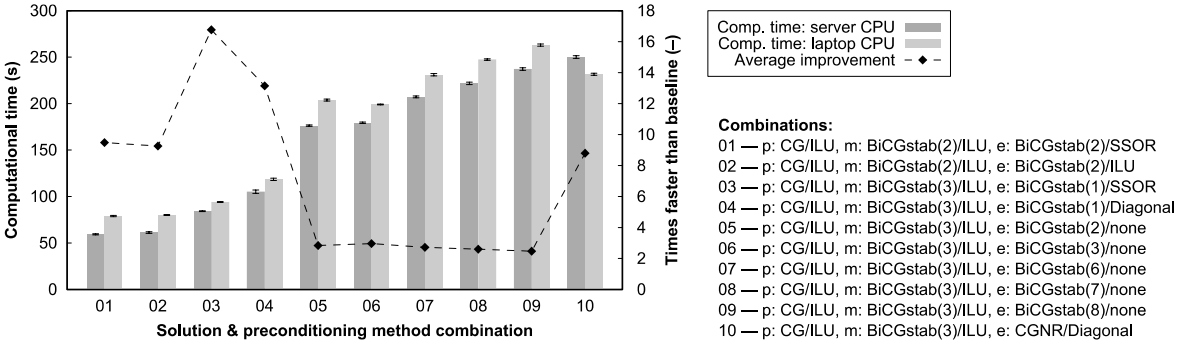


Figure 4: Ten best-performing combinations of solution and preconditioning methods in case of simulation of flow and energy transfer using the finer mesh (roughly 25,000 cells)

As for the ICC and ILUT preconditioning methods, these proved to be wholly unsuitable due to their slow convergence. What is more, these two preconditioners were markedly more likely to lead to divergence than ILU or SSOR.

3.2 Smoothness of convergence & general solution behaviour

In general, if a combination of preconditioned solvers reached a converged solution, then the convergence was faster, yet not necessarily smoother, than in case of the corresponding unpreconditioned combination (see Figures 5 and 6). However, it was also true that divergence was much more common among preconditioned solvers than when unpreconditioned solvers were employed. ILU provided marginally smoother convergence than SSOR and significantly smoother convergence than diagonal preconditioning. Considering BiCGstab(*L*), smoothness of convergence increased with increasing value of *L* (see Figure 5).

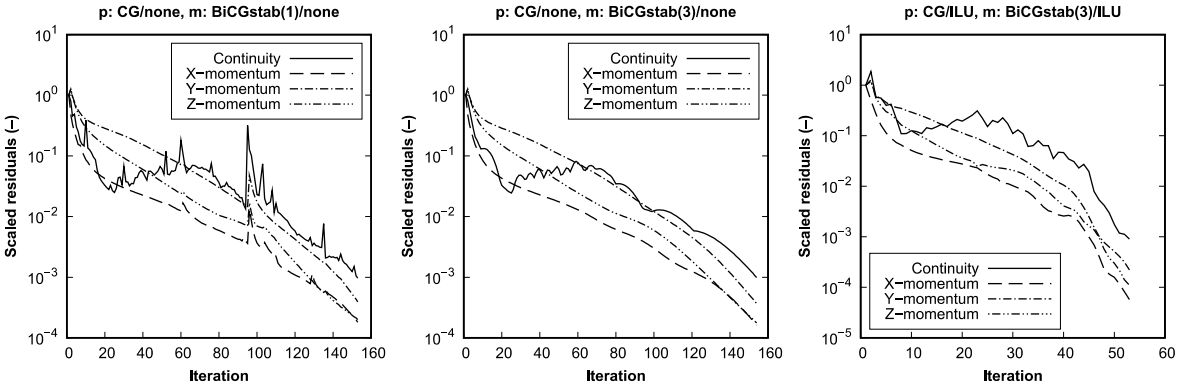


Figure 5: Comparison of scaled residual plots corresponding to three different combinations of solution and preconditioning methods in case of simulation of only the flow using the coarser mesh (roughly 6,000 cells)

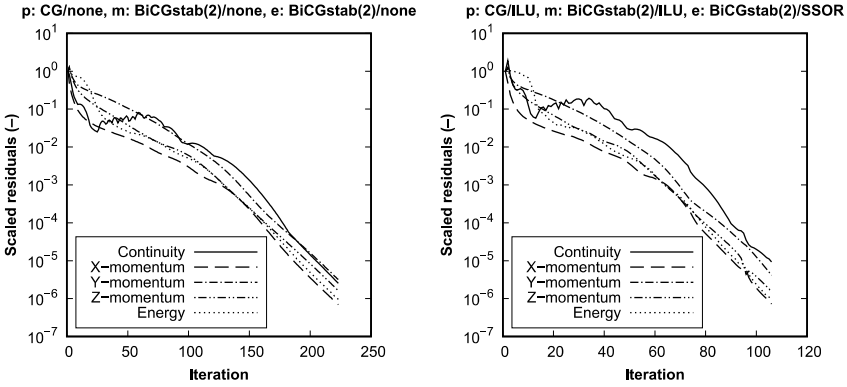


Figure 6: Comparison of scaled residual plots corresponding to a combination of unpreconditioned solution methods and the same methods preconditioned using ILU (pressure correction, momentum) and SSOR (energy) in case of the coarser mesh (roughly 6,000 cells)

4. Conclusions

The data presented in the previous paragraphs clearly suggest that in terms of the discussed simplified 3D CFD simulations best results are provided by different combinations of solution and preconditioning methods depending on whether only the flow or flow and energy transport are modelled. As for flow only, the best performance was reached using CG as the pressure correction solver together with BiCGstab(3) for momentum, both with ILU preconditioning being applied. The speedups compared to the respective baseline combination were ca. 4–5 depending on mesh fineness. If energy transport was included into the simulations, then the best results were obtained using CG/ILU for pressure correction, BiCGstab(2)/ILU for momentum, and BiCGstab(2)/SSOR or BiCGstab(2)/ILU for energy. The speedups corresponding to this combination were ca. 4 for the coarser mesh and ca. 10 in case of the finer mesh. Still, further tests should be carried out with respect to the selection of relaxation parameter $\omega \in (0, 2)$ in SSOR which, if chosen properly, can drastically improve performance.

Considering the two different CPUs, it is obvious that they performed almost identically in terms of single-core computations involving the small simplified meshes.

Acknowledgment

This research has been supported by EU project Sustainable Process Integration Laboratory – SPIL, funded as project No. CZ.02.1.01/0.0/0.0/15_003/0000456, by Czech Republic Operational Programme Research, Development, and Education, Priority 1: Strengthening capacity for quality research.

References

Birken P., Gassner G., Haas M., Munz C.-D., 2013, Preconditioning for modal discontinuous Galerkin methods for unsteady 3D Navier–Stokes equations, Journal of Computational Physics, 240, 20–35.

Bruaset A.M., 1995, A Survey of Preconditioned Iterative Methods, CRC Press, Boca Raton, FL, USA

Esmaily M., Jofre L., Mani A., Iaccarino G., 2018, A scalable geometric multigrid solver for nonsymmetric elliptic systems with application to variable-density flows, Journal of Computational Physics, 357, 142–158.

Freund R., Nachtigal N., 1991, QMR: a quasi-minimal residual method for non-Hermitian linear systems, Numerische Mathematik, 60, 315–339.

Goetz B., 2005, Java theory and practice: anatomy of a flawed microbenchmark, IBM Corporation <www.ibm.com/developerworks/java/library/j-jtp02225> accessed 10.01.2018.

Hestenes M.R., Stiefel E., 1952, Methods of conjugate gradients for solving linear systems, Journal of Research of the National Bureau of Standards, 49, 409–436.

Ma S., 2000, Comparisons of the ILU(0), Point-SSOR, and SPAI preconditioners on the CRAY-T3E for nonsymmetric sparse linear systems arising from PDEs on structured grids, The International Journal of High Performance Computing Applications, 14, 39–48.

Manteuffel T.A., 1980, An incomplete factorization technique for positive definite linear systems, Mathematics of Computation, 34, 473–497.

Meijerink J.A., van der Vorst A.H., 1977, An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix, Mathematics of Computation, 31, 148–162.

Moler C.B., 1967, Iterative refinement in floating point, Journal of the ACM, 14, 316–321.

Norris S.E., 2000, A Parallel Navier Stokes Solver for Natural Convection and Free Surface Flow, PhD Thesis, University of Sydney, Australia.

Patankar S.V., 1981, A calculation procedure for two-dimensional elliptic situations, Numerical Heat Transfer, 4, 409–425.

Ruggiu A.A., Weinerfelt P., Nordström J., 2018, A new multigrid formulation for high order finite difference methods on summation-by-parts form, Journal of Computational Physics, 359, 216–238.

Saad Y., 1994, ILUT: a dual threshold incomplete LU factorization, Numerical Linear Algebra with Applications, 1, 387–402.

Saad Y., 2003, Iterative Methods for Sparse Linear Systems, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

Saad Y., Schultz M., 1986, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM Journal on Scientific and Statistical Computing, 7, 856–69.

Sleijpen G.L., Fokkema D.R., 1993, BiCGstab(l) for linear equations involving unsymmetric matrices with complex spectrum, Electronic Transactions on Numerical Analysis, 1, 11–32.

Turek V., Fialová D., Jegla Z., 2016, Efficient flow modelling in equipment containing porous elements, Chemical Engineering Transactions, 52, 487–492.

van der Vorst H.A., 2003, Iterative Krylov Methods for Large Linear Systems, Cambridge University Press, Cambridge, UK.

van Doormaal J.P., Raithby G.D., 1984, Enhancement of the SIMPLE method for predicting incompressible fluid flows, Numerical Heat Transfer, 7, 147–163.

Wendykier P., Nagy J.G., 2010, Parallel Colt: a high-performance Java library for scientific computing and image processing, ACM Transactions on Mathematical Software, 37, 31:1–31:22.

Woźnicki Z.I., 2001, On performance of SOR method for solving nonsymmetric linear systems, Journal of Computational and Applied Mathematics, 137, 145–76.

Young D.M., 1977, On the accelerated SSOR method for solving large linear systems, Advances in Mathematics, 23, 215–271.

Příloha 4

Vojtěch Turek, Dominika Fialová & Zdeněk Jegla. Efficient flow modelling in equipment containing porous elements. *Chemical Engineering Transactions* **52**: 487–492, 2016. doi: 10.3303/CET1652082.

CHEMICAL ENGINEERING TRANSACTIONS
VOL. 52, 2016
A publication of AIDIC
The Italian Association of Chemical Engineering
Online at www.aidic.it/cet
Guest Editors: Petar Sabev Varbanov, Peng-Yen Liew, Jun-Yow Yong, Jiří Jaromír Klemeš, Hon Loong Lam

ISBN 978-88-95608-42-6; **ISSN** 2283-9216
DOI: 10.3303/CET1652082

Efficient Flow Modelling in Equipment Containing Porous Elements

Vojtěch Turek*, Dominika Fialová, Zdeněk Jegla

Institute of Process Engineering, Faculty of Mechanical Engineering, Brno University of Technology, Technická 2, 616 69 Brno, Czech Republic
turek@fme.vutbr.cz

Possible ways to model flow distribution and pressure drop in process and power equipment containing porous elements (e.g. tube sheets of dense tube bundles of recuperative heat exchangers, matrices of regenerative heat exchangers, packed beds of catalytic converters, etc.) are discussed. A simplified, hybrid flow distribution and pressure drop model based on finite volume method and applicable to a wide variety of equipment is proposed. Given the fact that the model in question is intended to be utilised primarily in optimisation algorithms, emphasis is placed on both obtaining accurate-enough data within relatively short time frames and ease of automation of the process. A proof-of-concept implementation of the model is discussed alongside improvements regarding solution convergence and solution efficiency in general that are critical in order to make the model commercially viable given its intended application.

1. Introduction

Often times it is necessary to optimise various parts of flow distribution systems for better performance and reliability. With the advent of more and more powerful computing hardware it may seem that utilising detailed 3D flow models based on Computational Fluid Dynamics (CFD) is the right choice for the mentioned purpose whether one is designing new equipment (Wei et al., 2015), analysing flow behaviour in an existing apparatus (Khaled et al., 2016), or making modifications to flow distribution-related parts in an already operated piece of equipment (Facão, 2015). But is it really so?

In some cases (e.g. Anbumeenakshi and Thansekhar 2016) it may be beneficial to forgo modelling altogether and carry out an experimental investigation. If only a few flow system geometries need to be evaluated with the best possible accuracy (as was done for example by Chen et al., 2015) then it certainly is appropriate to utilise CFD and accept that, given the nature of such models, the necessary data will not be available in minutes but rather in hours, tens of hours, or even days. The same applies to situations where employing a simplified model just is not practicable – be it due to the complexity of the flow system geometry (Li et al., 2014) or because constructing a simplified model would probably be non-trivial (as discussed for instance by Pendyala et al. 2015 in case of nanofluid flows). If, however, an engineer deals with the initial part of a design process of relatively common equipment and needs to estimate the behaviour of tens or hundreds of possible flow system geometries (or distribution scenarios) then, clearly, making use of detailed CFD models just to reject the majority of options would not make much sense. This is where simplified and fast, albeit less accurate flow models are much more fitting. These, though, are rarely discussed in the literature and if they are, they either tend to be tailored to specific flow system geometries and thus suffer from limited flexibility (see for example the model developed by Turek et al. (2011) which is applicable to double-U-tube heat exchanger modules) or only a cursory outline is mentioned as a side note and the main focus is still on detailed CFD modelling – as is e.g. the case of the aforementioned paper by Facão (2015).

Considering the porous nature of various elements of process and power equipment, by that we mean in this paper any element that can be for the purposes of flow distribution modelling reasonably represented by a porous structure. It can therefore be a packed bed of a catalytic converter (Turek et al., 2014), a matrix

Please cite this article as: Turek V., Fialová D., Jegla Z., 2016, Efficient flow modelling in equipment containing porous elements, Chemical Engineering Transactions, 52, 487-492 DOI:10.3303/CET1652082

in a regenerative heat exchanger (Drobnič et al., 2016), a plate-fin block of a recuperative heat exchanger (Liu et al., 2015), a dense tube bundle (Turek et al., 2015), or any other similar structure.
The following text discusses the two most common flow distribution and pressure drop modelling approaches, that is, detailed 3D Computational Fluid Dynamics-based modelling and simplified 1D or 2D modelling, together with their advantages and disadvantages. Then a proposed hybrid modelling approach is described and its benefits as well as not-yet-resolved caveats are reviewed.

2. Challenges of detailed 3D CFD flow distribution modelling

In flow distribution systems the inlet stream splits into multiple, often thousands of streams and, consequently, combines back into a single outlet stream. Hence, the flow usually exhibits strong turbulence and, as the authors of this paper can attest, numerical flow simulations of these systems therefore are highly unstable in the first ca. 50 to 100 iterations. This brings about the need to bootstrap every such computation (unless, of course, one is willing to deal with meshes consisting of tens of millions of cells). In other words, if one is to obtain a converged solution with a moderately-sized mesh then:

(i) Mesh quality (in terms of cell skewness, aspect ratio, and smoothness) must be reasonable and the solution must be initialised with a relatively accurate estimate – e.g. in ANSYS Fluent (ANSYS, 2015) Hybrid Initialisation is often much more suitable than Standard Initialisation.
(ii) Lower under-relaxation factors may be necessary in the first phase of the solution process. Similarly, relatively generous solution variable limits are recommended to prevent unnecessary value cut-offs which may lead to divergence.
(iii) It is good practice to start with a minimal setup – flow equations (momentum and pressure) together with turbulence included e.g. via the k-ε model (Launder and Spalding, 1974) or the Realizable k-ε model as recommended by Chen and Sparrow (2009). First-order differencing schemes such as the First Order Upwind Scheme (Courant et al., 1952), the Hybrid Scheme (Spalding, 1972), or the Power Law Scheme (Patankar, 1981) are good fits here as well as the straightforward SIMPLE pressure-velocity coupling (Patankar, 1972).There is virtually no point in attempting solution initialisation with a pressure-velocity coupling providing faster convergence because its benefit would be lost due to the current solution being very far from its stabilised state. It should also be noted that reversed flow is almost always encountered at the outlet(s) at the very beginning of the computation should the outlet zones be relatively short (3–6 times the hydraulic diameter of the outlet header).
(iv) Once a converged intermediate solution is obtained, one can switch to a higher-order differencing scheme – e.g. QUICK (Leonard, 1979) – and, again, run the computation until convergence is reached. It is not uncommon to still encounter reversed flow at the outlet(s) this far into the computation in spite of the fact that later on in the converged solution no reversed flow will be observed there.
(v) As the next step, under-relaxation factors can be gradually made larger while the solution is allowed to converge after each change. Analogously, the solution variable limits, if modified, can be gradually returned to their default values while making sure a converged state is reached after each change.
(vi) At this point one can include other equations such as the energy equation. If the fluid undergoes no significant temperature change in the distribution system then this step may be omitted given our aim being to obtain an accurate-enough estimate of flow distribution in as little time as possible.

As is apparent from the steps listed above, this is by no means a straightforward process. Also, even though automation is possible to a certain degree (e.g. in ANSYS Fluent one can employ journal files), one still has to monitor the computation closely and make impromptu adjustments to avoid divergence, unnecessarily slow convergence, or unsuitable y^+ values.
The second challenge concerning flow distribution modelling lies in flow instability. There is no guarantee that the steady-state solution which one has obtained earlier truly reflects the reality, that is, that flow rates through individual flow channels (e.g. the tubes of the bundle) do not fluctuate too much in time. This is why carrying out a transient simulation in addition to the steady one is always recommended. Should one opt to do so, it might be beneficial to switch to a faster-convergence pressure-velocity coupling – e.g. PISO (Issa, 1986) – after the steady-state solution has been obtained. Still, the relative benefit of such a decision depends on the actual flow system behaviour because in many cases a “one-iteration-per-time-step” simulation process results from a careful steady solution initialisation and as such a faster-convergence pressure-velocity coupling may prove to be slower due to its increased computational cost.
In summary, with a detailed 3D CFD model one gets very accurate data compared to various simplified modelling approaches (although some simplified models have been shown to provide solutions very similar to those obtained with detailed CFD models – see e.g. (Turek et al., 2015)). The process is still rather laborious and requires a lot of time and almost constant supervision. This is why it makes sense to develop simplified, fast, and automatable flow distribution models which can then be used effortlessly by process engineers.

3. Challenges of simplified 1D and 2D flow distribution modelling

Vast majority of porous elements present in process and power equipment do not feature one dominant dimension – a tube bundle, for instance, rarely consists of just one or two tube rows. As a consequence, (quasi-)1D flow distribution models, that is, models consisting of multiple interconnected 1D submeshes, usually cannot describe the porous elements in question accurately enough. Still, for a very limited set of scenarios (quasi-)1D models may serve quite well (Turek et al., 2011).

Two-dimensional models, on the other hand, can handle various flow distribution geometries very well (see for example the aforementioned paper by Turek et al. (2015)). Flow variables corresponding to the mesh elements in such (quasi-)2D models, however, are almost always evaluated sequentially because simultaneous (or segregated) evaluation of velocity, pressure and other fields is rather challenging in terms of convergence on the grounds of problematic interconnection of the different-dimension submeshes via source terms. That being said, this type of models usually suffers from a major downside being the necessity to (i) carefully construct the respective computer code including a suitable and numerically sound way of making flow variable corrections, (ii) often times use significant under-relaxation factors slowing down already very slow convergence, and, last but not least, (iii) change non-negligible portions of the code each time a non-trivial change is made to the flow system geometry. Still, with respect to accuracy (quasi-)1D and (quasi-)2D models – if applied carefully to a specific class of flow distribution systems with a very limited range of possible geometry changes – often are more than satisfactory for fast and effortless preliminary evaluation of simpler flow systems.

In reality, though, this is seldom the case. Process engineers need a tool that can be used to preliminarily evaluate many different types of flow systems without the need to change the respective computer code for it to meet the current requirements.

4. Hybrid modelling approach

It is obvious from the previous paragraphs that both the detailed 3D CFD modelling and simplified (quasi-)1D and (quasi-)2D modelling approaches provide benefits as well as suffer from shortcomings. It would therefore be of great interest to process engineers to have available a tool that would (i) provide relatively accurate flow distribution data in reasonable time frames and (ii) allow effortless solution automation of flow distribution problems which, inherently, are numerically ill-posed.

The former requirement is straightforward – fast flow evaluation at the cost of too low an accuracy is as pointless as obtaining quite an accurate solution within an unacceptably long time frame. The second requirement, however, is more complex and includes several factors. Given the needs of process engineers, the respective tool must be able to not only manage the solution process so that a converged solution is obtained with as little user intervention as possible but also generate a suitable mesh from several key pieces of data (such as the main dimensions of the evaluated flow distribution system) and statistically process the obtained results in case of unsteady computation (e.g. time-averaging of tube mass flow rates with respect to a maximum allowed fluctuation error).

The following text is tailored to the specifics of dense tube bundles (i.e., tube bundles with larger ratios of total tube cross-sectional area to tube sheet area) but it can easily be generalised to other types of porous elements.

4.1 Implemented simplification measures

The starting point for the hybrid modelling approach is a 3D CFD model including mass, momentum, and energy transport with turbulence being tentatively neglected. In other words, the initial model works with a 3D mesh and solves the momentum equations to get the u, v, and w velocity fields and the pressure correction equation (the SIMPLE pressure-velocity coupling is utilised) to get the pressure field.

The primary simplification lies in mesh consisting solely of cuboid cells because this not only significantly reduces the number of cells required to span the flow system geometry but also greatly simplifies the manner in which gradients etc. are calculated internally when matrix coefficients are being generated. Lower overall number of cells provides obvious reduction in computation time. As for the gradients etc., the cuboid nature of the cells ensures that normal components of all face fluxes are inherently equal to the x-, y-, and z-fluxes themselves.

The disadvantage of a cuboid-cell mesh is obvious; one can hardly use such cells to fully span non-cuboid elements. Still, there is relatively easy remedy for this – any tube can simply be replaced by a set of $N \times N$ cuboid cell layers resulting in the identical cross-sectional area (see Figure 1). If there is a non-zero heat flux prescribed for the tube surface then the original value must obviously be scaled to match the new “tube” surface area by the factor of $F_q = \pi\, d_2 \,/\, (4\, N\, l_{tc})$ in which d_2 denotes the tube outer diameter, N the number of cells spanning one side of the square tube mesh cross-section, and l_{tc} the respective tube cell side length.

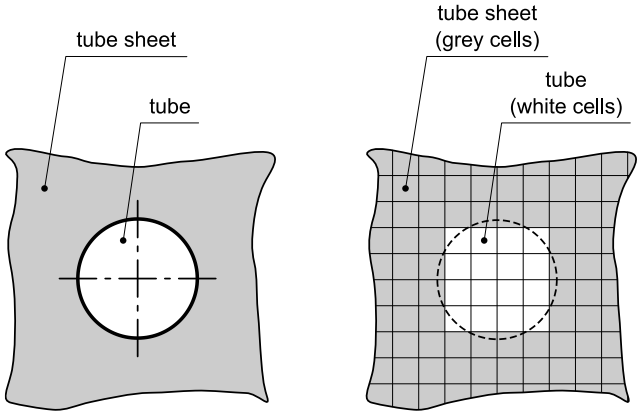


Figure 1: Top view of the original tube geometry and the replacement tube mesh consisting of layers of 4 × 4 cuboid cells

There is one more benefit regarding cuboid-cell meshes, namely the ease with which such meshes can be generated. Given the common arrangements of tubes in bundles (90 °, 60 °, and 45 °) it can easily be seen that if one is to span the flow distribution system geometry without cell face splitting, the fineness of the resulting mesh is governed by a single parameter – the number of cells spanning one side of the square tube mesh cross-section, *N*. The spaces between tube rows and also between individual tubes in rows are then spanned accordingly with cells sized as close to the tube cells as possible.

Considering the actual value of *N*, the lower limit generally is two. The reason for this is that in a simplified model it usually is sufficient to utilise the SIMPLE pressure-velocity coupling which operates on a staggered grid. With this in mind, it is obvious that with a single cell comprising tube cross-section the momentum transport from the distribution header to the tubes and from the tubes to the collection header would suffer thus rendering the computation even more unstable and likely to diverge. Using too large a value of *N* – say, more than six – also may not be a good idea because this would defeat the purpose of simplified modelling (size of the resulting mesh would most probably be of the order of units of millions of cells). In any case, one can further optimise the mesh by employing a double sided, uneven (e.g. successive ratio) longitudinal spacing in the tubes or a single sided, uneven spacing in the headers in directions normal to tube sheets. This modification, which is trivial to implement, provides two benefits: (i) the number of cells is further reduced and (ii) mesh quality is improved because mesh fineness is greater in areas where gradients of flow variables are likely to be large and lower where no significant changes are expected.

As mentioned above, in most flow distribution problems the SIMPLE pressure-velocity coupling should suffice. One could make use of a faster-convergence coupling – e.g. SIMPLER (Patankar, 1980), SIMPLEC (Van Doormaal and Raithby, 1984), or PISO – but the actual benefit in terms of shortening of the evaluation time may well be cancelled due to its larger computational complexity. What is more, with respect to the nature of flow distribution problems such couplings may even hinder convergence in the initial phase of the computation when it is not yet stabilised enough.

With respect to differencing schemes, again, there is virtually no point in employing other than the first order ones (upwind, hybrid, or power law). Although it is true that with a second order scheme such as QUICK the accuracy of the resulting data would be somewhat higher, given the nature of the model one is after here (a simplified one) it is debatable whether the associated increase in computational complexity and hence also increase in computational time is warranted.

With regard to the fact that temperature changes stemming from energy dissipation are rather insignificant, solving the energy equation is only necessary if heat transfer is in effect in some part of the flow distribution system (e.g. in the tubes). Solving this equation for adiabatic systems would lead only to a marginal increase in accuracy but at the cost of significant increase in computational cost.

Lastly, let us focus on boundary conditions. Because we are interested in maintaining the total mass flow rate through the entire system, the mass flow inlet boundary condition must be implemented (in some cases the velocity inlet boundary condition is also acceptable but the actual velocity then has to be calculated from the prescribed total mass flow rate). Second, the pressure outlet boundary condition must be available so that pressure frame can be specified for the solution. Third, we need the stationary wall boundary condition – e.g. of the “no slip” type. Frictional pressure drop as well as heat transfer in case of heated/cooled walls can then be easily modelled in a simplified way via source terms.

In summary, to obtain a simplified 3D CFD model covering virtually any reasonable flow distribution system it is enough to solve the usual momentum and pressure correction equations coupled via SIMPLE and supplemented by source terms accounting for changes in gravitational potential energy, pressure losses due to friction, and possible additional minor losses in the porous structure. Should heat transfer be in effect then, obviously, also the energy equation is required with fluid heating or cooling in the respective cells again being included in a simplified manner via source terms.

4.2 Computational efficiency concerns

In each major iteration of the simplified CFD solver we must solve – at least approximately – four or five rather large systems of linearized equations. Each of these systems is of the rank of roughly the number of cells in the mesh, that is, each of the matrices contains tens to hundreds of thousands of rows. It is therefore essential that the numerical methods employed for this purpose are as efficient as possible. This can be achieved by utilising highly-optimised computer codes such as ATLAS (Whaley et al., 2001), BLAS (Dongarra, 2002), or LAPACK (Anderson et al., 1999). Further increase in efficiency can be accomplished by splitting the matrix-solving code over multiple threads.

Many specialised linear algebra libraries based on such high-performance codes are available for programmers' convenience. For example, the testing CFD code – written by the authors of this paper in Oracle Java (Oracle, 2016) – makes use of the MTJ-N package (Halliday, 2016) which acts as an intermediary between pure Java code and native implementations of BLAS and LAPACK. Being given a sample flow system, this code was able to automatically generate a suitable mesh consisting of roughly 55,000 cells and provide a fully converged steady-state solution in ca. 130 s (126 iterations) while being run in a single-core mode on a regular office computer.

4.3 Potential caveat & future work

Admittedly, in many flow distribution scenarios the convergence of simplified 3D CFD models is rather slow (or even problematic) regardless the actual under-relaxation factors. It therefore still remains to be seen whether including the effect of turbulence into the model would provide substantial improvement in terms of convergence so that the increased computational cost would be justified. This, of course, renders the discussed type of models less practicable; however, possible computationally cheap ways of convergence improvement including the major iteration-specific selection of internal matrix solver type and its setup are currently researched by the authors. As for model validation and benchmarks, these activities are planned for the future but neither of them has yet been carried out for obvious reasons.

5. Conclusions

It has been shown that, compared to both 3D CFD modelling and simplified (quasi-)1D and (quasi-)2D modelling the proposed hybrid approach provides significant benefits. First, hybrid models are universally applicable – just as the usual 3D CFD models are – but one is not required to deal with mesh creation (which is often non-trivial) because in case of cuboid-cell meshes this can easily be automated.

Second, the model places emphasis on short evaluation times at the cost of lower, but still good-enough accuracy. Even though these times are somewhat longer compared to (quasi-)1D and (quasi-)2D models due to decidedly larger meshes and more complex nature of the employed equations, this is remedied by the fact that virtually no modifications to the hybrid model are necessary if flow system geometry is changed (apart from the portion which handles mesh creation but that is in most cases a rather simple affair). In contrast, a relatively minor change in geometry can result in a lot of necessary changes in a (quasi-)1D or (quasi-)2D model due to their being tailor-made for specific flow systems.

Lastly, although no benchmark or validation has been carried out as of yet due to the still not fully resolved issues with convergence, one can reasonably expect the resulting data to be at least as accurate as those obtained with lower-dimension models. In any case, the main advantage, that is, the universality of such models combined with very easy generation of meshes and short computational times, remains and promises to yield a valuable flow distribution evaluation tool once the convergence issues have been overcome.

Acknowledgement

The authors gratefully acknowledge financial support provided by Technology Agency of the Czech Republic within the research project No. TE02000236 "Waste-to-Energy (WtE) Competence Centre".

References

Anbumeenakshi C., Thansekhar M.R., 2016, Experimental investigation of header shape and inlet configuration on flow maldistribution in microchannel. Exp. Therm. Fluid Sci., 75, 156–161.

Anderson E., Bai Z., Bischof C., Blackford S., Demmel J., Dongarra J.J., Du Croz J., Greenbaum A., Hammarling S., McKenney A., Sorensen D., 1999, LAPACK Users' Guide, 3rd Ed., Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

ANSYS Inc., 2015, ANSYS Fluent User's Guide. ANSYS, Inc., Canonsburgh, USA.

Chen A., Sparrow E.M., 2009, Turbulence modeling for flow in a distribution manifold. Int. J. Heat Mass Transfer, 52, 1573–1581.

Chen Y., Wang Y., Bao Z., Zhang Q., Li X.-Y., 2016, Numerical investigation of flow distribution and heat transfer of hydrocarbon fuel in regenerative cooling panel. Appl. Therm. Eng., 98, 628–635.

Courant R., Isaacson E., Rees M., 1952, On the solution of non-linear hyperbolic differential equations by finite differences, Comm. Pure Appl. Math., 5, 243–255.

Dongarra J.J., 2002, Basic Linear Algebra Subprograms technical forum standard, Int. J. High Performance Applications and Supercomputing, 16, 1–111.

Drobnič B., Oman J., Tuma M., 2006, A numerical model for the analyses of heat transfer and leakages in a rotary air preheater. Int. J. Heat Mass Tran., 49, 5001–5009.

Facão J., 2015, Optimization of flow distribution in flat plate solar thermal collectors with riser and header arrangements. Sol. Energy, 120, 104–112.

Halliday S., 2016, Matrix-Toolkits-Java, <github.com/fommil/matrix-toolkits-java>, accessed 28.02.2016.

Issa R.I., 1986, Solution of the implicitly discretized fluid flow equation by operator splitting, J. Comput. Phys., 62, 40–65.

Khaled M., Ramadan M., Shaito A., Hage H.E., Harambat F., Peerhossaini H., 2016, Parametric analysis of heat exchanger thermal performance in complex geometries — Effect of air velocity and water flow distributions, Heat Transfer Eng., 37, 1027–1037, DOI: 10.1080/01457632.2015.1104166.

Launder B.E., SpaldingD.B., 1974, The numerical computation of turbulent flows, Comput. Method. Appl. M., 3, 269–289.

Leonard B.P., 1979, A stable and accurate convective modelling procedure based on quadratic upstream interpolation, Comput. Method. Appl. M., 19, 59–98.

Li L., Ma T., Xu X.Y., Zeng M., Wang Q.W., 2014, Study on heat transfer and pressure drop performances of airfoil-shaped printed circuit heat exchanger. Chem. Eng. Trans., 39, 895–900.

Liu J., Zhang S., Zhao X., Yi G., Zhou Z., 2015, Influence of fin arrangement on fluid flow and heat transfer in the inlet of a plate-fin heat exchanger, J. Zhejiang Univ. Sci. A, 16, 279–294.

Oracle, 2016, What is Java? <www.java.com/en/download/whatis_java.jsp>, accessed 30.03.2016.

Patankar S.V., 1980, Numerical Heat Transfer and Fluid Flow, McGraw-Hill, New York, NY, USA.

Patankar S.V., 1981, A calculation procedure for two-dimensional elliptic situations, Num. Heat Transfer, 4, 409–425.

Patankar S.V., Spalding D.B., 1972, A calculation procedure for heat, mass, and momentum transfer in three-dimensional parabolic flows, Int. J. Heat Mass Transfer, 15, 1787–1806.

Pendyala R., Chong J.L., Ilyas S.U., 2015, CFD analysis of heat transfer performance in a car radiator with nanofluids as coolants. Chem. Eng. Trans., 45, 1261–1266.

Spalding D.B., 1972, A novel finite-difference formulation for differential expressions involving both first and second derivatives, Int. J. Num. Methods Eng., 4, 551–559.

Turek V., Bébar L., Jegla Z., 2014, Simplified pressure drop and flow distribution modelling in radial catalytic converters. Chem. Eng. Trans., 39, 853–858.

Turek V., Fialová D., Jegla Z., Kilkovský B., 2015, Efficient 2D model of flow distribution in dense tube bundles, Chem. Eng. Trans., 45, 1177–1182.

Turek V., Hájek J., Jegla Z., Stehlík P., 2011, Optimum design of fluid distribution systems in heat exchangers, Asia-Pac. J. Chem. Eng., 6, 750–759.

Van Doormaal J.P., Raithby G.D., 1984, Enhancement of the SIMPLE method for predicting incompressible fluid flows, Numer. Heat Transfer, 7, 147–163.

Wei M., Fan Y., Luo L., Flamant G., 2015, CFD-based evolutionary algorithm for the realization of target fluid flow distribution among parallel channels, Chem. Eng. Res. Des., 100, 341–352.

Whaley R.C., Petitet A., Dongarra J.J., 2001, Automated empirical optimization of software and the ATLAS Project, Parallel Comput., 27, 3–35.

Příloha 5

Vojtěch Turek, Ladislav Bébar & Zdeněk Jegla. Simplified pressure drop and flow distribution modelling in radial catalytic converters. *Chemical Engineering Transactions* **39**: 853–858, 2014. doi: 10.3303/CET1439143.

CHEMICAL ENGINEERING TRANSACTIONS
VOL. 39, 2014
A publication of AIDIC
The Italian Association of Chemical Engineering
www.aidic.it/cet
Guest Editors: Petar Sabev Varbanov, Jiří Jaromír Klemeš, Peng Yen Liew, Jun Yow Yong

ISBN 978-88-95608-30-3; **ISSN** 2283-9216
DOI: 10.3303/CET1439143

Simplified Pressure Drop and Flow Distribution Modelling in Radial Catalytic Converters

Vojtěch Turek*, Ladislav Bébar, Zdeněk Jegla

Institute of Process and Environmental Engineering, Faculty of Mechanical Engineering, Brno University of Technology, Technická 2, 616 69 Brno, Czech Republic
turek@fme.vutbr.cz

A simplified model of a radial hydrocarbon reforming catalytic converter is presented. It focuses on fluid flow, distribution along the annular catalytic packed bed, and overall pressure drop. As for the chemistry aspect of the problem or possible issues due to pore blockages, these are not considered in the paper.
The model is based on a set of several well-known equations such as the continuity equation, Euler's equation for inviscid flow, equation of state, etc. and is built with utilization in geometry optimization algorithms in mind. An optimization software implementing the discussed model and a comparison of results and experimental data related to a series of existing catalytic converters are presented as well.

1. Introduction

Radial catalytic converters are typically employed in petroleum refineries to convert low-octane intermediates into higher-quality products, such as high-octane gasoline, with hydrogen being a by-product (Rahimpour et al., 2013). This process is highly endothermic (Zagoruiko et al., 2014) and therefore the entire catalytic conversion unit usually consists of several converters connected in series each of which is preceded by a fired heater where the feed is re-heated.
Pressure drops in packed beds of catalytic converters tend to be quite significant which, combined with pressure drops in fired heaters, induces considerable pumping costs. Converters must therefore be designed in such a way that the resulting pressure drops are as low as possible. Moreover, flow distribution along the catalytic packed bed should be as uniform as possible – especially if regeneration of the catalyst (Bartholomew, 2000) is not performed. This helps to achieve uniform aging of the catalyst and consequently longer service life of the catalytic bed.
Neither the chemistry aspect of the problem (except for an estimated temperature gradient in the packed bed) nor pore blockages (Jiménez-García et al., 2013) are considered in this paper. What is of interest to us is pressure change modelling as pressure gradients also largely influence flow distribution. Earlier, pressure drop of a packed bed had usually been estimated en bloc using for instance the Ergun equation as discussed in (Green and Perry, 2008); however, this did not enable incorporation of incremental changes in temperature, viscosity, etc. into the models or reasonable prediction of flow distribution. Current modelling approaches therefore favor direct numerical simulation, often via customized finite-element or finite-volume methods as described for instance by Palle and Aliabadi (2013) in case of regular packing or by Mousazadeh et al. (2013) for random particle distribution. Some of the models even consider effective transport properties of packed layers as discussed by Bertei et al. (2013). Computational fluid dynamics (CFD) is very popular as well (see e.g. Zhou et al., 2013), especially in case of automotive industry applications (Kumar and Mazumder, 2010). Other models such as those based on artificial neural networks can also be encountered (Zamaniyan et al., 2013). Nonetheless, for optimization purposes one needs a simple-enough model if reasonable optimization times are to be expected. Hence, although one-dimensional simplified models of packed beds exist for chemical phenomena (Srinivasan and Depcik, 2013a), heat and mass transport (Srinivasan and Depcik, 2013b), or both (Saouli et al., 2011), these are still quite complex and their implementation in geometry optimization algorithms would lead to optimization times that are prohibitively long if used in engineering practice. What is more, the implementation itself

Please cite this article as: Turek V., Bébar L., Jegla Z., 2014, Simplified pressure drop and flow distribution modelling in radial catalytic converters, Chemical Engineering Transactions, 39, 853-858 DOI:10.3303/CET1439143

would certainly not be easy. A simple yet accurate enough pressure drop and flow distribution model allowing fast and effortless geometry optimization is therefore presented in this paper.

2. Description of the catalytic converter unit

Figure 1 shows a typical radial catalytic converter. Desulfurized naphtha feedstock enters the converter at the top through the inlet duct. Then it flows into the inlet dome through the inlet distributor and continues into the channels along the perimeter of the annular catalytic packed bed. Converted feed then leaves the unit through the central outlet duct.
Both cylindrical surfaces of the packed bed consist of two layers. Rigid outer layer made of a perforated steel sheet prevents deformations of the bed. Inner layer – a wire mesh screen – keeps the packing from falling through the holes in the outer perforated sheet.

3. Mathematical model

Let us first consider flow through a simple channel. Figure 2 shows a segment of a straight channel with the control volume being delimited by a dotted line. Pressure, density, and flow velocity at the inlet are denoted as p_1, ρ_1, and v_1, respectively. Channel area at the inlet into the control volume is A_1. Quantities at the outlet from the control volume are denoted analogously with the subscript "2". Since the segment delimited by the control volume is of a very short length dl, we can approximate the outlet cross-sectional area by $A_2 = A_1 + (\partial A / \partial l)\,\mathrm{d}l$ and proceed similarly for the remaining quantities.
We must also consider friction, dF_F, standard gravity, g, the rate at which fluid leaves the control volume through the wall of the channel, $\delta \mathrm{d}l$, and the angle of inclination of the segment from the direction of g, φ. Please note that δ is positive for outflow and negative for inflow.
The first equation that we will need is the continuity equation, that is

$$\rho v A = \text{const.} \tag{1}$$

After writing this equation for the segment in Figure 2, expanding it, and removing higher-order terms, we get

$$\frac{1}{\rho}\frac{\partial \rho}{\partial l} + \frac{1}{v}\frac{\partial v}{\partial l} + \frac{1}{A}\frac{\partial A}{\partial l} + \frac{\delta}{\rho v A} = 0. \tag{2}$$

Next, by performing a balance of forces acting on the fluid delimited by the control volume, we obtain

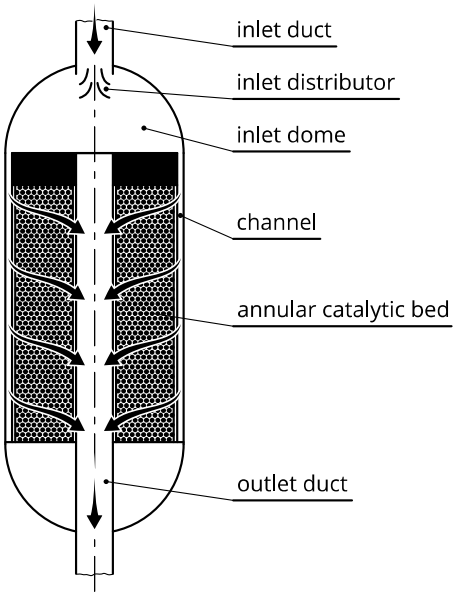


Figure 1: Cross-section of a typical radial catalytic converter

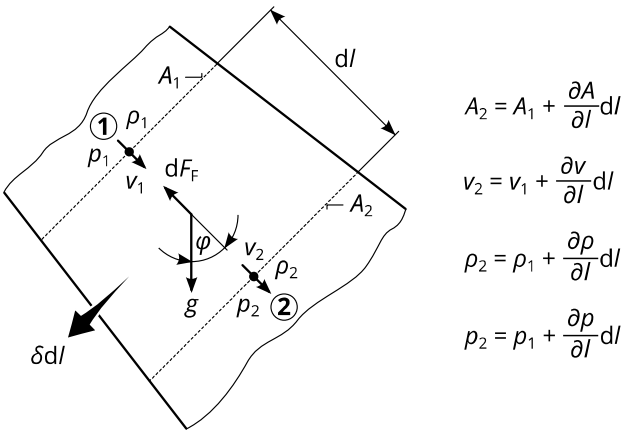


Figure 2: Short segment of an inclined channel with variable cross-section and porous wall

$$v\frac{\partial v}{\partial l}+\frac{1}{\rho}\frac{\partial p}{\partial l}-g\cos\varphi+\frac{1}{\rho A}\frac{\mathrm{d}F_{\mathrm{F}}}{\mathrm{d}l}=0 \tag{3}$$

which, in fact, is the Euler's equation for steady one-dimensional compressible inviscid flow. The last term in this equation can be replaced for convenience – per the Darcy-Weisbach equation (Brown, 2002) – by specific energy loss,

$$\frac{1}{\rho A}\frac{\mathrm{d}F_{\mathrm{F}}}{\mathrm{d}l}=\frac{\mathrm{d}E}{\mathrm{d}l}=\lambda\frac{\mathrm{d}l}{D_{\mathrm{h}}}\frac{v^2}{2}. \tag{4}$$

Here, λ denotes Darcy friction factor and D_{h} hydraulic diameter of the channel. The modified equation thus is

$$v\frac{\partial v}{\partial l}+\frac{1}{\rho}\frac{\partial p}{\partial l}-g\cos\varphi+\lambda\frac{\mathrm{d}l}{D_{\mathrm{h}}}\frac{v^2}{2}=0. \tag{5}$$

Finally and most importantly, we need to find the equation governing pressure change in the segment. To do so, we will utilize two other equations, namely the equation of state,

$$p=\rho RT, \tag{6}$$

and the first law of thermodynamics for an adiabatic process,

$$\frac{\partial q}{\partial l}=c_{\mathrm{v}}\frac{\partial T}{\partial l}-\frac{p}{\rho^2}\frac{\partial \rho}{\partial l}. \tag{7}$$

Therefore, by combining Eqs. (5) through (7) we get

$$\frac{1}{\rho}\frac{\partial \rho}{\partial l}-\frac{1}{\rho v^2}\frac{\partial p}{\partial l}+\frac{g\cos\varphi}{v^2}-\frac{\lambda}{2D_{\mathrm{h}}}+\frac{1}{A}\frac{\partial A}{\partial l}+\frac{\delta}{\rho vA}=0. \tag{8}$$

Since in this simplified case heat is generated solely due to friction, we can rewrite the above equation as

$$\frac{\partial p}{\partial l}\left(\frac{1}{\kappa p}-\frac{1}{\rho v^2}\right)=\frac{\lambda}{2D_{\mathrm{h}}}\left(1+\frac{\kappa-1}{\kappa}\frac{\rho v^2}{p}\right)-\frac{g\cos\varphi}{v^2}-\frac{1}{A}\frac{\partial A}{\partial l}-\frac{\delta}{\rho vA} \tag{9}$$

in which $\kappa = c_{\mathrm{p}} / c_{\mathrm{v}}$ denotes heat capacity ratio. From here, it immediately follows that

$$\frac{\partial p}{\partial l}=\frac{\frac{\lambda}{2D_{\mathrm{h}}}\left(1+\frac{\kappa-1}{\kappa}\frac{\rho v^2}{p}\right)-\frac{g\cos\varphi}{v^2}-\frac{1}{A}\frac{\partial A}{\partial l}-\frac{\delta}{\rho vA}}{\frac{1}{\kappa p}-\frac{1}{\rho v^2}}. \tag{10}$$

At this point it is easy to obtain equation for pressure change in a segment of the length Δl in the discretized representation of the geometry. Other quantities are calculated accordingly with difference versions of the equations above.

Considering minor losses (flow through the perforated steel sheets, wire meshes, etc.), these can be calculated via the standard equation

$$\Delta p=\zeta\frac{\rho v^2}{2} \tag{11}$$

with coefficients of hydraulic resistance, ζ, computed according to the formulas available in (Idelchik, 2001).

It is assumed that the bed is packed regularly (Afandizadeh and Foumeny, 2001). Pressure changes therein are calculated in the same manner as in Eq. (11) since the usual equations for en bloc estimation of pressure drop (especially Ergun equation) are not suitable for prediction of pressure drop in case of turbulent flow regime (Allen et al., 2013). Temperature changes caused by energy dissipation due to induced turbulence are then estimated similarly as in Eq. (4), i.e., via specific energy loss. Such an approach allows us to incorporate gradual changes in temperature, pressure, viscosity, and other quantities.

4. Software implementation of the model

The model has been implemented in a geometry optimization tool (see Figure 3) developed in Java (Oracle, 2014). The entire geometry is discretized automatically and a quasi-1D mesh is produced based on the provided input data. Equations are then solved sequentially in an upwind-like (Patankar, 1980, Section 5.2-2) iterative manner until convergence is reached. Considering physical properties of the fluid (density, dynamic viscosity, etc.), these are always calculated ad hoc with respect to actual temperature and pressure.

The tool is primarily intended for catalytic bed geometry optimization with the domain being given by ranges of characteristic dimensions (inner and outer diameters and length of the catalytic bed). Even though extensive evaluation of all possible geometries in the domain would certainly yield an optimum, the process would be rather time-consuming. An appropriate optimization method is therefore selected according to the actual dimension of the domain. The Golden Section Method (Ravindran et al., 2006, pp. 51–53) is employed for a one-dimensional domain. Two-dimensional domains are explored using the Hooke and Jeeves method (Ravindran et al., 2006, pp. 92–97) enhanced with intelligent selection of initial estimate and with an algorithm for adaptive pattern step length. A 3D extension of this enhanced Hooke and Jeeves method is used for three-dimensional optimization domains. Additionally, the application can take full advantage of contemporary multi-core processors – it splits the optimization domain into several smaller domains so that all available processor cores are utilized and that optimization times are shortened even further.

Based on our tests, both objective functions, that is, pressure drop and relative standard deviation from uniform flow distributions, are smooth and unimodal for any reasonable set of input data. It is therefore safe to use the above-mentioned optimization methods.

4.1 Comparison of obtained data and industrial-case geometries

In order to get a sense of how well the optimization tool discussed in the previous section performs, the authors compared its outputs to several existing catalytic converters connected in series. Although these converters have been designed before wide-spread availability of CFD and other numerical tools requiring substantial computing power, all design decisions were then made according to both experience and extensive prototype testing to ensure as good a performance as possible while retaining low overall pressure drop.

Catalytic bed of the first converter was required to be 14.5 m^3 in volume and have inner diameter of 0.65 m. Characteristic size of particles making up the packed bed was 1.9 mm. Mass flow rate of the

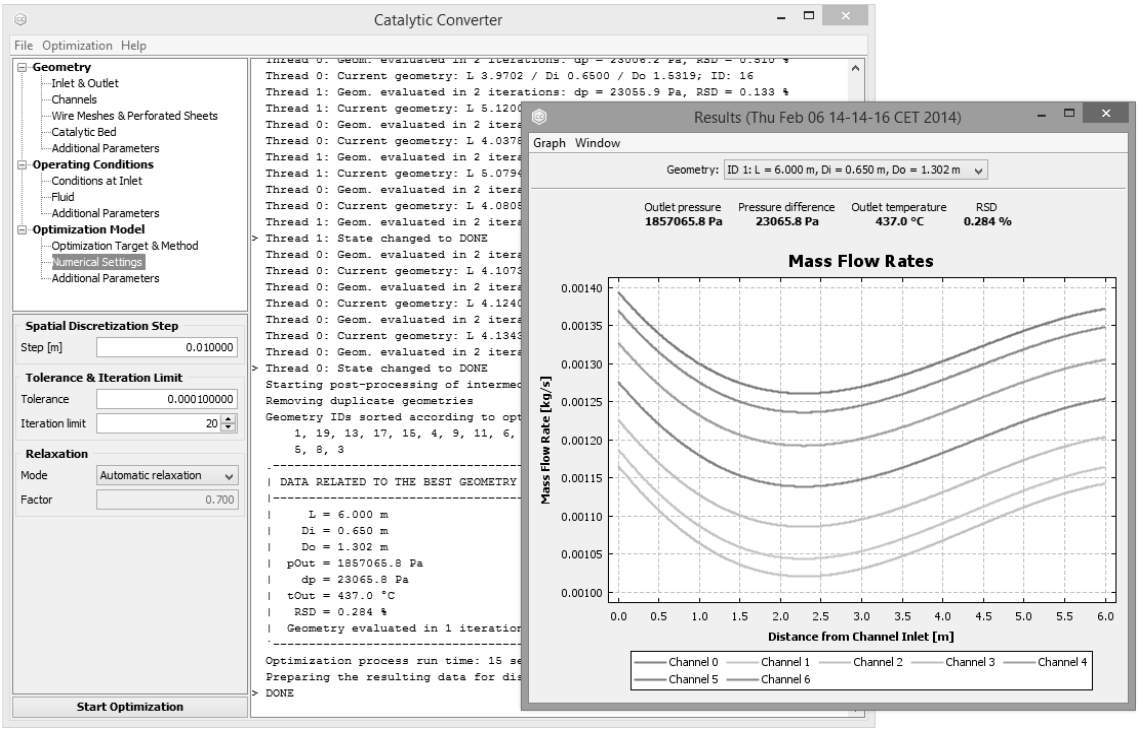


Figure 3: Screenshot of the developed geometry optimization tool with sample results being displayed

hydrocarbon mixture being converted was 52.44 kg s^{-1} and – according to the operator – its temperature dropped from 505 °C to 437 °C in the converter. The second converter in the series contained packed bed with volume of 36.3 m^3 that consisted of the same type of particles and had the same inner diameter as in case of the first apparatus. Here, temperature of the stream dropped from 505 °C (after re-heating) to 470 °C. As for the third converter, its catalytic bed was required to be 72.5 m^3 in volume. Again, the same type of particles was used as packing and identical inner diameter of the bed was required. Temperature drop in this case, however, was even smaller – from 505 °C to 495 °C.

Results yielded by the optimization tool are shown in Table 1 alongside data provided by the operator of the three converters mentioned above. It can be seen that there is a good agreement between the data sets. Of course, this is not sufficient as a model validation (for that an extensive testing is planned) but it hints that the model might perform rather well in its present state.

It should also be noted that relative standard deviation from uniform flow distribution, being a measure of uniformity of flow distribution along the catalytic bed, is quite low for all three converters. The catalyst inside the packed beds should therefore be aging uniformly.

Table 1: Comparison of data obtained with the geometry optimization tool and data provided by the operator of the catalytic converters

	Converter I		Converter II		Converter III	
	opt. tool	operator	opt. tool	operator	opt. tool	operator
Catalytic Bed						
inner diameter	0.650 m		0.650 m		0.650 m	
outer diameter	2.034 m	2.060 m	2.882 m	2.920 m	3.083 m	3.100 m
length	4.967 m	4.830 m	5.854 m	5.700 m	10.163 m	10.050 m
Operating conditions						
inlet temperature	505 °C		505 °C		505 °C	
outlet temperature	437 °C		470 °C		495 °C	
inlet pressure	1,834.0 kPa		1,692.0 kPa		1,571.0 kPa	
outlet pressure	1,801.9 kPa	1,803.7 kPa	1,666.7 kPa	1,667.5 kPa	1,548.4 kPa	1,550.1 kPa
pressure drop	32.1 kPa	30.3 kPa	25.3 kPa	24.5 kPa	22.6 kPa	20.9 kPa
RSD*	0.30 %	N/A	0.15 %	N/A	1.16 %	N/A

*Relative standard deviation from uniform flow distribution

5. Future work

Although according to preliminary tests it seems that the presented model performs quite well, it still needs to be validated properly. In other words, until performance of the model is validated against a large-enough set of experimental data or at least data obtained via CFD evaluations, it cannot be considered to be production-ready.

6. Conclusions

A simple mathematical model allowing easy and very fast evaluation of catalytic converter pressure drop and flow distribution uniformity has been described. According to preliminary tests, the model seems to be quite accurate but proper validation is still necessary before it is production-ready. Also, a user-friendly geometry optimization tool developed in Java has been presented. The tool takes advantage of modern multi-core processors and splits each optimization task to an optimum number of cores to shorten optimization time as much as possible.

Once a validated model is available, chemical engineers will be able to use the respective optimization tool to quickly and effortlessly design radial catalytic converters with low pressure drops and as uniform flow distributions along catalytic beds as possible. Not only that such converters will be cheaper to operate because of lower pumping costs, they will also require less maintenance due to more uniform catalyst ageing.

Nomenclature

A cross-sectional area of a channel, m^2
c_p specific heat capacity at constant pressure, J kg^{-1} K^{-1}
c_v specific heat capacity at constant volume, J kg^{-1} K^{-1}
D_h hydraulic diameter, m
E energy, J
F_F friction, N
g standard gravity, m s^{-2}
l length, m

p pressure, Pa
R specific gas constant, J kg^{-1} K^{-1}
T thermodynamic temperature, K
v flow velocity, m s^{-1}
δ amount of fluid that leaves control volume through a wall, kg m^{-1} s^{-1}
ζ coefficient of hydraulic resistance, –
κ heat capacity ratio, –
λ Darcy friction factor, –
ρ density, kg m^{-3}
φ angle of inclination of a segment from the direction of g, °

Acknowledgement

The authors gratefully acknowledge financial support provided within the project “NETME CENTRE PLUS” (National Sustainability Programme I / LO1202) which is co-funded by the Ministry of Education, Youth and Sports of the Czech Republic.

References

Afandizadeh S., Foumeny E., 2001, Design of packed bed reactors: Guides to catalyst shape, size, and loading selection, Appl. Therm. Eng., 21, 669–682.
Allen K.G., von Backström T.W., Kröger D.G., 2013, Packed bed pressure drop dependence on particle shape, size distribution, packing arrangement and roughness, Powder Technol., 246, 590–600.
Bartholomew C., 2000, Catalyst Deactivation and Regeneration, Kirk-Othmer Encyclopedia of Chemical Technology, Eds. Kroschwitz J.I., Howe-Grant M., John Wiley & Sons, Hoboken, NJ, USA.
Bertei A., Nucci B., Nicolella C., 2013, Effective transport properties in random packings of spheres and agglomerates, Chem. Eng. Trans., 32, 1531–1536.
Brown G., 2002, The history of the Darcy-Weisbach equation for pipe flow resistance, Environmental and Water Resources History, 34–43.
Green D.W., Perry R.H., Eds., 2008, Perry’s Chemical Engineering Handbook, 8th Ed., McGraw-Hill, New York, NY, USA.
Idelchik I.E., 2001, Handbook of Hydraulic Resistance, 3rd Ed., Begell House Publishers, Redding, CT, USA.
Jiménez-García G., de Lasa H., Quintana-Solórzano R., Maya-Yescas R., 2013, Catalyst activity decay due to pore blockage during catalytic cracking of hydrocarbons, Fuel, 110, 89–98.
Kumar A., Mazumder S., 2010, Toward simulation of full-scale monolithic catalytic converters with complex heterogeneous chemistry, Comput. Chem. Eng., 34, 135–145.
Mousazadeh F., van Den Akker H.E., Mudde R.F., 2013, Direct numerical simulation of an exothermic gas-phase reaction in a packed bed with random particle distribution, Chem. Eng. Sci., 100, 259–265.
Oracle, 2014, What is Java? <www.java.com/en/download/whatis_java.jsp>, accessed on 17.01.2014.
Palle S., Aliabadi S., 2013, Direct simulation of structured wall bounded packed beds using hybrid FE/FV methods, Comput. Fluids, 88, 730–742.
Patankar S.V., 1980, Numerical Heat Transfer and Fluid Flow, Hemisphere Publishing Corp., Washington, D.C., USA.
Rahimpour M.R., Jafari M., Iranshahi D., 2013, Progress in catalytic naphtha reforming process: A review, Appl. Energ., 109, 79–93.
Ravindran A., Ragsdell K., Reklaitis G., 2006, Engineering Optimization: Methods and Applications, 2nd Ed., John Wiley & Sons, Hoboken, NJ, USA.
Saouli O., Bencheikh-Lehocine M., Hassen Meniai A., 2011, 1-D reactive transport modeling in heterogeneous porous media, Chem. Eng. Trans., 24, 415–420.
Srinivasan A., Depcik C., 2013a, One-dimensional pseudo-homogeneous packed-bed reactor modeling: I. Chemical species equation and effective diffusivity, Chem. Eng. Technol., 36, 22–32.
Srinivasan A., Depcik C., 2013b. One-dimensional pseudo-homogeneous packed-bed reactor modeling: II. Energy equation and effective thermal conductivity, Chem. Eng. Technol., 36, 379–389.
Zagoruiko A.N., Belyi A.S., Smolikov M.D., Noskov A.S., 2014, Unsteady-state kinetic simulation of naphtha reforming and coke combustion processes in the fixed and moving catalyst beds, Catal. Today, 220–222, 168–177.
Zamaniyan A., Joda F., Behroozsarand A., Ebrahimi H., 2013, Application of artificial neural networks (ANN) for modeling of industrial hydrogen plant, Int. J. Hydrogen Energ., 38, 6289–6297.
Zhou X., Duan Y., Huai X., Li X., 2013, 3D CFD modeling of acetone hydrogenation in fixed bed reactor with spherical particles, Particuology, 11, 715–722.

Příloha 6

Vojtěch Turek, Jiří Hájek, Zdeněk Jegla & Petr Stehlík. Optimum design of distribution systems in heat exchangers. *Asia-Pacific Journal of Chemical Engineering* **6**(5): 750–759, 2011. doi: 10.1002/apj.516.

ASIA-PACIFIC JOURNAL OF CHEMICAL ENGINEERING
Asia-Pac. J. Chem. Eng. 2011; **6**: 750–759
Published online 7 October 2010 in Wiley Online Library
(wileyonlinelibrary.com) DOI:10.1002/apj.516

Curtin University

Special theme research article

Optimum design of fluid distribution systems in heat exchangers

Vojtěch Turek, Jiří Hájek, Zdeněk Jegla and Petr Stehlík*

Institute of Process and Environmental Engineering, Faculty of Mechanical Engineering, Brno University of Technology, Technická 2, 616 69 Brno, Czech Republic

Received 22 March 2010; Revised 17 August 2010; Accepted 19 August 2010

ABSTRACT: Uniform flow distribution is often required by process equipment for its correct and reliable operation. Due to the fact that the resulting distribution is influenced by shapes of splitting and combining manifolds, flow distribution uniformity can be considerably increased by careful design. In case of high-temperature applications, uniform flow distribution is usually crucial. The aim of this paper is to provide a formula for coefficient of static regain for parallel flow systems containing linearly tapered manifolds with rectangular cross sections connected by double U-tubes. This formula was derived by means of approximating data obtained by evaluation of several baseline configurations of such distribution systems using fluid flow modelling software ANSYS® FLUENT®. Sample fluid distributions found using the proposed formula are shown alongside corresponding data computed by ANSYS FLUENT. Also, a practical computational tool for shape optimization of the investigated class of manifolds is presented. © 2010 Curtin University of Technology and John Wiley & Sons, Ltd.

KEYWORDS: heat exchanger; CFD modelling; splitting manifold; combining manifold; U-tubes

INTRODUCTION

In a broader perspective, the motivation for studying optimum design of splitting and combining manifolds is closely connected with the design of heat exchanger networks (HENs). A well-designed HEN can bring substantial energy and financial savings and, thus, there is a lot of effort devoted to this topic. The design process consists of three main stages:

(1) Targeting,[1] which is a preliminary technical economic analysis determining optimum heat recovery;
(2) Synthesis,[2] during which an optimum HEN layout is found; and
(3) Detailed design of individual pieces of process equipment (i.e. selection of suitable heat exchanger types to be used[3] and their design in terms of required heat duties, allowed pressure losses, necessary compatibility of media with materials that exchanger parts are made of, their fouling,[4] etc.).

However, it is clear that an excellent HEN layout alone cannot guarantee optimal performance of the network and, therefore, optimization of important parts of heat exchangers should be done in the detailed design stage as well. In other words, the overall HEN design problem (Fig. 1) can be split into selection of process stream matches (I), detailed design of heat exchangers (II), and optimization of specific exchanger parts (III).

Considering fluid distribution systems in heat exchangers, optimization usually means that these are designed in such a way that ensures as uniform a flow distribution as possible. Importance of a uniform flow distribution can be even greater in case of high-temperature media being used as process fluids, since serious damage to the system may occur due to uneven thermal expansion. Thus, being able to predict the resulting flow distribution and optimize the shape of splitting and combining manifold not only can lead to better exchanger performance but can also prevent its total destruction.

Background research

Most papers concerning fluid distribution are related to manifold systems with constant cross sections. One of the first papers on this topic[5] described the analytical successive branch-by-branch approach decomposing an entire splitting or combining manifold into

**Correspondence to*: Vojtěch Turek, Institute of Process and Environmental Engineering, Faculty of Mechanical Engineering, Brno University of Technology, Technická 2, 616 69 Brno, Czech Republic. E-mail: turek@upei.fme.vutbr.cz

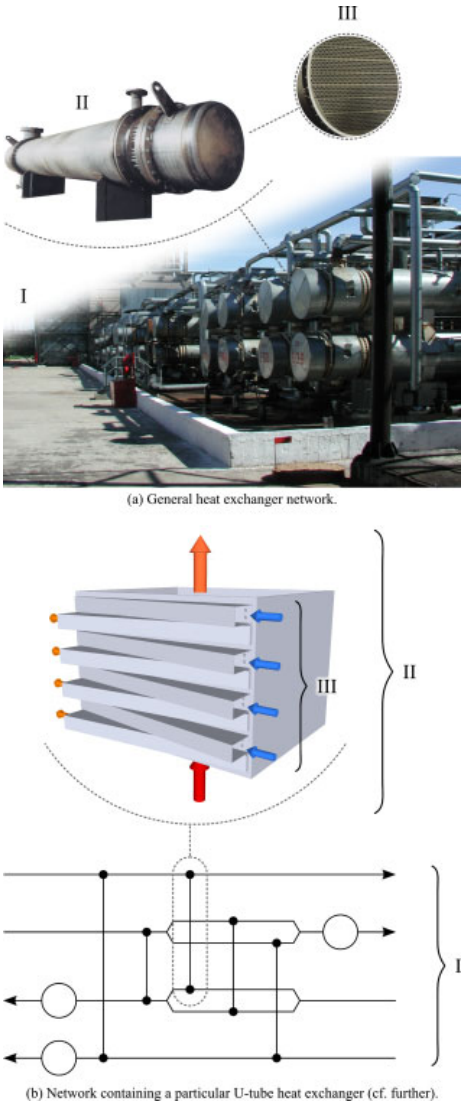


Figure 1. Design problem 'HEN → individual unit → specific part of the unit'. This figure is available in colour online at www.apjChemEng.com.

control volumes around the discharge orifices and dealing with each of them separately. Nevertheless, only manifolds with uniformly spaced lateral pipes and circular cross sections distributing fluid into (or collecting fluid from) atmosphere or some other constant-pressure environment were discussed. Another paper[6] dealing with uniformly perforated manifolds having constant circular cross sections investigated the influence of axial velocity of fluid on the actual direction of discharge. Also, friction coefficients were assumed to be calculated rather than given as constants in advance. Other existing algebraic models describe either mere division or combination of flows in manifolds with circular[7] or rectangular[8] cross sections, or parallel flow systems with such manifolds, e.g. in solar panels[9] or fuel cells.[10,11]

More advanced models cannot be solved analytically, since the utilized equations (momentum conservation law, etc.) contain differential and integral terms. There are models for division or combination of flows in case of manifolds with circular[12] and rectangular[13] cross sections. As for complete parallel systems, both finite-difference[14] and differential[15,16] models have been developed. There also exist models for microchannel parallel systems,[17] systems in fuel cells,[18,19] and in electronic cooling modules.[20,21] Models supporting two-phase flow exist as well.[22,23]

All the models mentioned so far assume manifolds have constant cross sections. However, the amounts of fluid flowing through branches of a manifold (i.e. the resulting distribution) depend on pressure differences between ends of U-tubes, which can be greatly influenced by longitudinal manifold cross-sectional variability. This is due to the fact that the differences themselves are given by pressure profiles in the manifolds and any change of a pressure profile (caused, for instance, by a locally convergent or divergent shape of the manifold) must influence the lateral flow rates. An appropriate design of the manifolds can thus increase flow distribution uniformity. Nevertheless, research papers dealing with fluid distribution systems containing manifolds with variable cross sections[24,25] are rare and, to the best of the authors' knowledge, only cases of varying rectangular cross-sectional height have been investigated so far. Moreover, although the models presented therein can be implemented directly using common programming languages (C++, Java, etc.), they require discretization of the respective governing equations which introduces a considerable amount of additional work. Also, a computational tool based on one of these models would probably need significantly more time to yield a solution than a tool making use of the simpler successive branch-by-branch approach with algebraic equations only.

Modelled distribution system

The modelled fluid distribution system is a part of a made-to-measure heat exchanger for high-temperature applications. Typically, this exchanger is used to preheat fluidizing and combustion air and consists of several stacked identical distribution systems (also referred to as exchanger sections, Fig. 2) mounted horizontally in a vertical hexahedral shell (Fig. 1(b) depicting a shell containing four distribution systems with linearly tapered manifolds). Both manifolds have variable rectangular cross sections and are stepless, since these are easy to manufacture by welding together several parts cut from a sheet metal plate. Process fluid to

DOI: 10.1002/apj

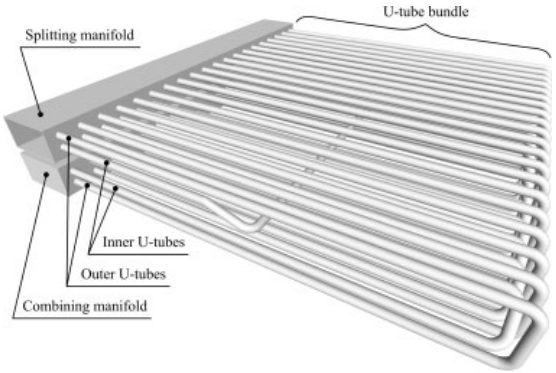


Figure 2. Modelled fluid distribution system.

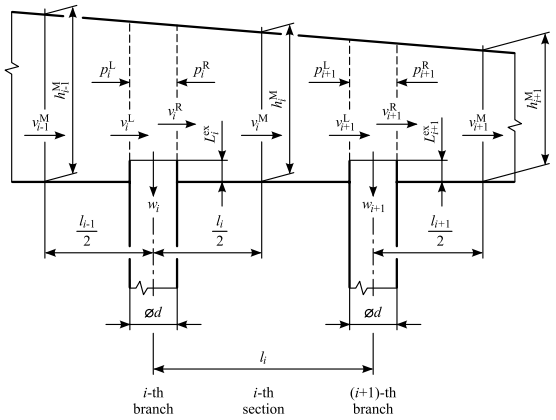


Figure 3. Scheme of a manifold with variable rectangular cross section.

be heated up flows from splitting manifolds through U-tube bundles (each of which consists of 20 pairs of U-tubes with the same diameter) into combining manifolds and then out of the sections. Splitting and combining manifolds are of the same shape, though combining ones are 'reversed' (in general, cross-sectional area of a splitting manifold decreases in flow direction, whereas cross-sectional area of a combining one increases). Hot flue gas flows across the U-tube bundles. It is clear that uniform distribution of process fluid into U-tubes is important for the exchanger's performance and can also prevent damage caused by e.g. uneven thermal expansion.

ANALYTICAL MODEL

This work presents a model restricted to constant-density fluids. The model is built upon several fundamental equations. The first of these is the Darcy–Weisbach equation, Ref. [26], p. 340, governing pressure loss due to friction,

$$p_{i+1}^{\mathrm{L}} - p_i^{\mathrm{R}} = -f_i \frac{l_i}{D_{\mathrm{h},i}^{\mathrm{M}}} \rho \frac{\left(v_i^{\mathrm{M}}\right)^2}{2} \tag{1}$$

where f_i denotes Darcy friction factor for the i-th manifold section, $D_{\mathrm{h},i}^{\mathrm{M}}$ hydraulic diameter in its middle, and ρ density of the fluid (for nomenclature of local mean fluid velocities and other terms see Fig. 3). It is assumed that the cross-sectional area of each branch is small enough compared with the manifold cross-sectional area at that branch, and thus it is neglected in the equation.

In laminar flow, which can occur in long straight segments of U-tubes in case of a low mass flow rate, Darcy friction factor depends solely on Reynolds number, Ref. [27], p. 160,

$$f_i = \frac{64}{\mathrm{Re}_i} \tag{2}$$

whereas in case of a turbulent flow, it also depends on absolute roughness of the inner surface of the manifold section, ε, and its hydraulic diameter. To get the friction factor, one can either solve the implicit Colebrook–White equation[28]

$$\frac{1}{\sqrt{f_i}} = -2\log_{10}\left(\frac{\varepsilon}{3.7D_{\mathrm{h},i}^{\mathrm{M}}} + \frac{2.51}{\mathrm{Re}_i\sqrt{f_i}}\right) \tag{3}$$

or use an explicit approximation (e.g. Churchill,[29] Serghides,[30] or Haaland[31] approximation). In the present model, Churchill approximation

$$A_i = \left[-2.457\ln\left(\left(\frac{7}{\mathrm{Re}_i}\right)^{0.9} + \frac{0.27\varepsilon}{D_{\mathrm{h},i}^{\mathrm{M}}}\right)\right]^{16}$$

$$B_i = \left(\frac{37530}{\mathrm{Re}_i}\right)^{16}$$

$$f_i = 8\left[\left(\frac{8}{\mathrm{Re}_i}\right)^{12} + \frac{1}{(A_i + B_i)^{1.5}}\right]^{\frac{1}{12}} \tag{4}$$

is used since it gives reasonable values of the friction factor[32] while retaining computational simplicity.

Pressure changes caused by changes of manifold cross section are computed using the Bernoulli equation

$$\frac{(v_i^{\mathrm{R}})^2}{2} + gz_i + \frac{p_i^{\mathrm{R}}}{\rho} = \frac{(v_{i+1}^{\mathrm{L}})^2}{2} + gz_{i+1} + \frac{p_{i+1}^{\mathrm{L}}}{\rho} \tag{5}$$

and the continuity equation

$$b_i h_i v_i^{\mathrm{R}} = b_{i+1} h_{i+1} v_{i+1}^{\mathrm{L}} \tag{6}$$

where b_i, h_i, b_{i+1}, and h_{i+1} denote width and height of manifold cross section at the i-th and $(i+1)$-th branch,

DOI: 10.1002/apj

respectively. As of now, the effect of gravitational field is neglected and thus gravitational terms gz_i and gz_{i+1} are not present in the final version of Eqn (5). Combining Eqns (1, 5, and 6) then yields a new equation for pressure loss in the i-th manifold section

$$p_{i+1}^{\mathrm{L}} - p_i^{\mathrm{R}} = \underbrace{-f_i \frac{l_i (b_i^{\mathrm{M}} + h_i^{\mathrm{M}})}{2 b_i^{\mathrm{M}} h_i^{\mathrm{M}}} \rho \frac{(v_i^{\mathrm{M}})^2}{2}}_{\text{friction}} + \underbrace{\frac{\rho (b_i^{\mathrm{M}} h_i^{\mathrm{M}} v_i^{\mathrm{M}})^2}{2} \left[\frac{1}{(b_i h_i)^2} - \frac{1}{(b_{i+1} h_{i+1})^2} \right]}_{\text{change of cross section}} \quad (7)$$

where b_i^{M} denotes width of manifold in the middle of i-th section.

Another fundamental equation[6]

$$p_i^{\mathrm{R}} - p_i^{\mathrm{L}} = \frac{C_{\mathrm{r},i}}{2} \rho \left[(v_i^{\mathrm{L}})^2 - (v_i^{\mathrm{R}})^2 \right] \quad (8)$$

governs pressure changes caused by changes in fluid momentum near the branches. $C_{\mathrm{r},i}$ denotes coefficient of static regain for the i-th branch which has been defined as the ratio of the difference in static pressure between the flow just upstream and just downstream of the branch to the difference in dynamic pressures (see further). This coefficient corrects for the effect of discharge angle (fluid flowing out of the manifold does not generally lose all its original axial velocity and therefore it does not discharge perpendicularly to the manifold axis).

Minor losses in the manifolds are computed using the equation

$$\Delta p = \xi \rho \frac{v^2}{2} \quad (9)$$

where ξ denotes a coefficient of hydraulic resistance[33] and v mean fluid velocity. Computation of pressure losses in U-tubes utilizes Eqns (1, 2, 4, and 9). Since the fluid is heated up while flowing through the U-tubes, these are divided into three sectors with gradually increasing temperatures. Fluid density and dynamic viscosity (necessary to calculate Reynolds number) are computed according to the temperature in the respective sector rather than assumed to be constant. In case of water being fed into the splitting manifold of the exchanger section, for instance, the DIPPR105 equation[34]

$$\rho = \frac{0.14395}{0.0112^{1+\left(1-\frac{T}{649.727}\right)^{0.05107}}} \quad (10)$$

could be used to calculate density and the Vogel equation[35]

$$\mu = e^{-3.7188 + \frac{578.919}{-137.546+T}} \quad (11)$$

to calculate dynamic viscosity (in both cases T denotes thermodynamic temperature).

Mass flow rates $\dot{m}_i^{\mathrm{I}}$ and $\dot{m}_i^{\mathrm{O}}$ through the i-th inner and outer U-tube (Fig. 2), respectively, are found using the equation

$$\dot{m}_i^{\mathrm{I}} + \dot{m}_i^{\mathrm{O}} = \rho b_i h_i (v_i^{\mathrm{L}} - v_i^{\mathrm{R}}). \quad (12)$$

NEW FORMULA FOR COEFFICIENT OF STATIC REGAIN

Since the original formula for coefficient of static regain[6] was derived for manifolds with constant circular cross section, it had to be modified. Computational fluid dynamics (CFD) modelling software ANSYS FLUENT[36] was used to evaluate several baseline profiles per each manifold profile type (constant cross section and linear change of cross-sectional width and height). Obtained results were then approximated using a new formula for coefficient of static regain that would be applicable to such cases. This formula,

$$C_{\mathrm{r},i} = \left[3.54 \left(1 - \frac{L_i^{\mathrm{ex}}}{h_i} \right) + 4.33 \frac{b_i - b_i^{\mathrm{fs}}}{b_i} - 0.69 \right] + \left[0.30 + 1.49 \log_{10} \left(\frac{d}{D_{\mathrm{h},i}} \right)^2 + 370 D_{\mathrm{h},i}^{\mathrm{diff}} \frac{n-i}{n-1} \right] \log_{10} \frac{v_i^{\mathrm{L}}}{v_i^{\mathrm{L}} - v_i^{\mathrm{R}}} + + [155 \exp(-890 D_{\mathrm{h,avg}}^{\mathrm{diff}}) - 155] D_{\mathrm{h},i}^{\mathrm{diff}} \quad (13)$$

seems to work quite well in case of linear changes of cross-sectional width and height (Figs 4–7). As for the terms used in the above equation, L_i^{ex} denotes exsertion of the i-th U-tube pair, b_i^{fs} free space between the U-tubes and sidewalls of the manifold, $D_{\mathrm{h},i}$ hydraulic diameter of the manifold at the i-th branch, $D_{\mathrm{h},i}^{\mathrm{diff}}$ difference in hydraulic diameter between the $(i-1)$-th and the i-th manifold section, n number of branches, and

$$D_{\mathrm{h,avg}}^{\mathrm{diff}} = \frac{1}{n} \sum_{i=1}^{n} D_{\mathrm{h},i}^{\mathrm{diff}} \quad (14)$$

the average difference in hydraulic diameter.

Sample results

Figures 4–7 show mass flow rates through U-tubes in four different fluid distribution systems with an equal total mass flow rate $\dot{m}_{\mathrm{tot}} = 55\ \mathrm{kg \cdot s^{-1}}$ of water. Predictions using formula 13 are compared with that computed by ANSYS FLUENT CFD software. Since in the detailed simulations the flow is unsteady and

DOI: 10.1002/apj

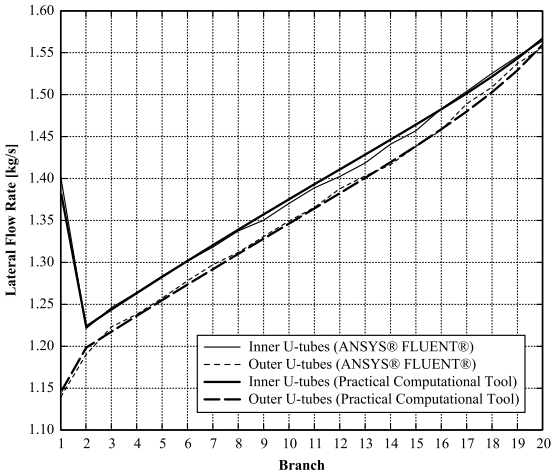


Figure 4. Lateral flow rates of water through U-tubes in case both manifolds have constant cross section 169 × 287 mm (width × height). Maximum relative error is less than 1%.

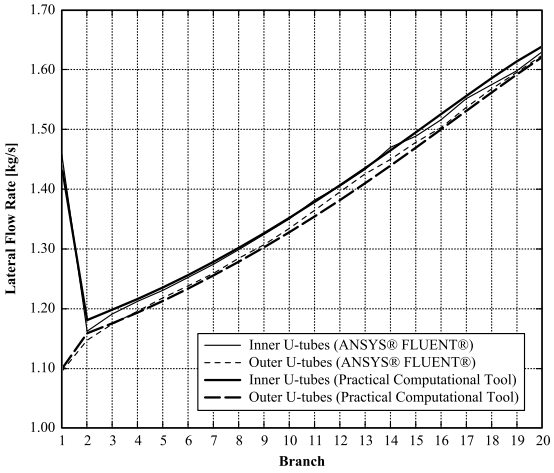


Figure 5. Lateral flow rates of water through U-tubes in case open end dimensions of both manifolds are 169 × 287 mm (width × height) and closed end dimensions are 140 × 170 mm. Maximum relative error is less than 2%.

it takes approximately 10 s before flow rates stabilize (with slight oscillations in the quasi-steady state), all the corresponding flow rates displayed in the figures are arithmetic means of flow rates through that particular U-tube at times $t \geq 10$ s.

PRACTICAL COMPUTATIONAL TOOL

A practical computational tool (Fig. 8) allowing process engineers to easily find the best possible distributor

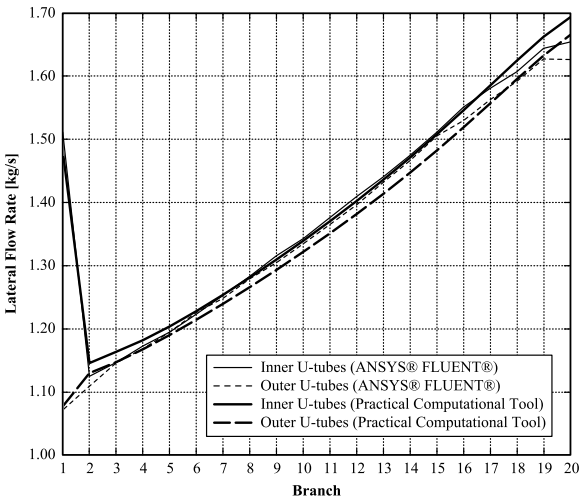


Figure 6. Lateral flow rates of water through U-tubes in case open end dimensions of both manifolds are 169 × 287 mm (width × height) and closed end dimensions are 140 × 95 mm. Maximum relative error is less than 3%.

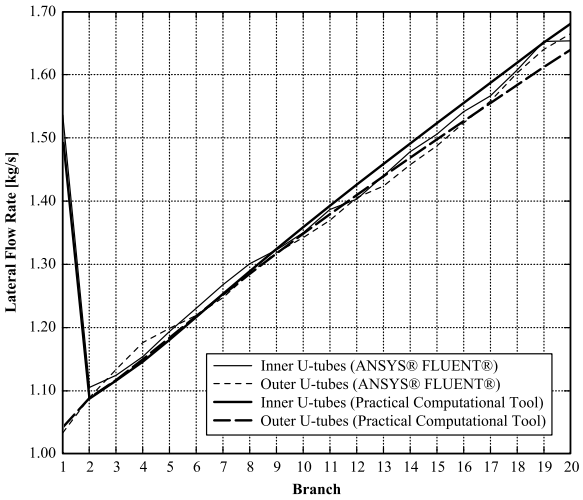


Figure 7. Lateral flow rates of water through U-tubes in case open end dimensions of both manifolds are 169 × 287 mm (width × height) and closed end dimensions are 140 × 50 mm. Maximum relative error is less than 3%.

and collector shape (in terms of flow distribution uniformity) within given dimension ranges is being developed using Java™ Development Kit. To the best of the authors' knowledge, no other tool for direct shape optimization of the studied flow system is available at the moment. Although the tool requires Java™ Runtime Environment to be installed, this minor drawback is remedied by the fact that it can be used on a wide range of architectures and operating systems.

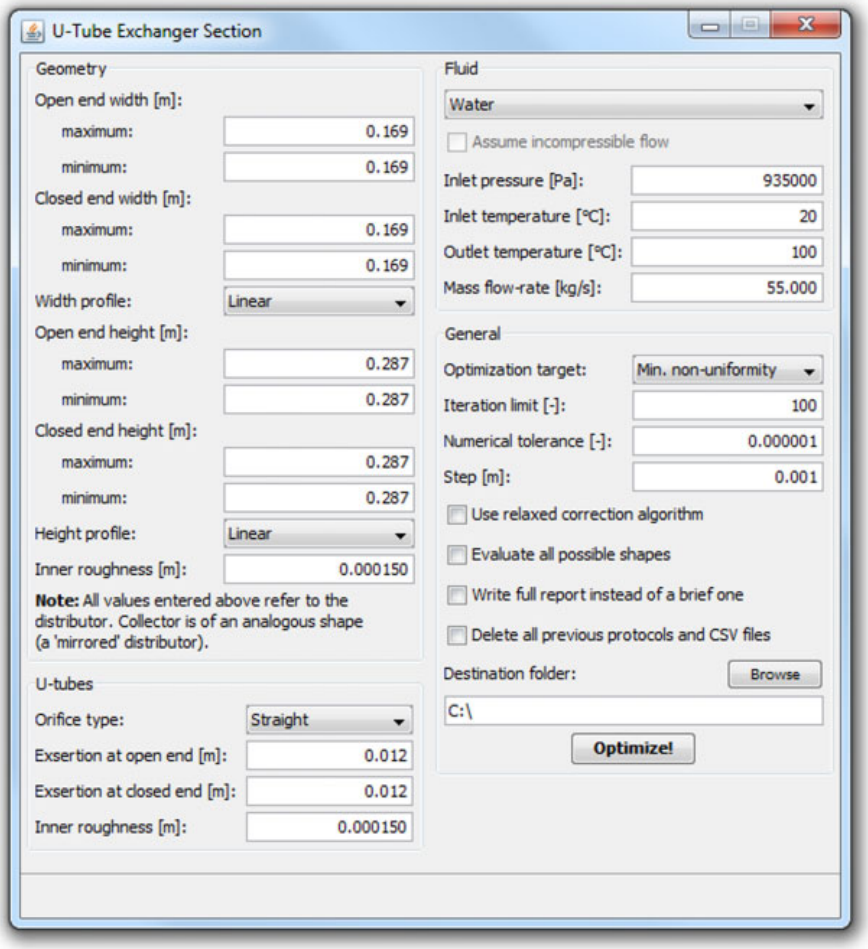


Figure 8. Practical computational tool. This figure is available in colour online at www.apjChemEng.com.

The following parameters must be entered before an optimization process can be started:

(1) Dimension ranges of width and height at the open and closed ends of the manifolds (Fig. 9);
(2) Which fluid flows through the system;
(3) Roughness of inner surface of the manifolds and U-tubes;
(4) Pressure and temperature of the fluid at distributor inlet;
(5) Temperature of the fluid at collector outlet;
(6) Mass flow rate of the fluid; and
(7) Other necessary input data (iteration limit per one shape, numerical tolerance, etc.).

The tool can already be used for manifolds with constant cross section and with linearly variable cross-sectional width and height. Considering orifice types, only the exserted one (Fig. 10(a)) is available at the moment. Extension of the code enabling conical (Fig. 10(b)) and circular bellmouth orifice types (Fig. 10(c)) will be added to the application later.

The actual computational algorithm finding flow distribution for a particular splitting and combining manifold consists of these steps:

(1) Select an initial estimate of mass flow rates through distributor sections.
(2) Compute current lateral flow rates through inner and outer U-tubes using flow rates through distributor sections.
(3) Save current lateral flow rates into a reference variable.
(4) Find pressure profile along the distributor, compute pressure losses in U-tubes, and find pressure profile along the collector. In case the pressure at a certain collector branch computed using pressure at the previous branch and pressure losses in the

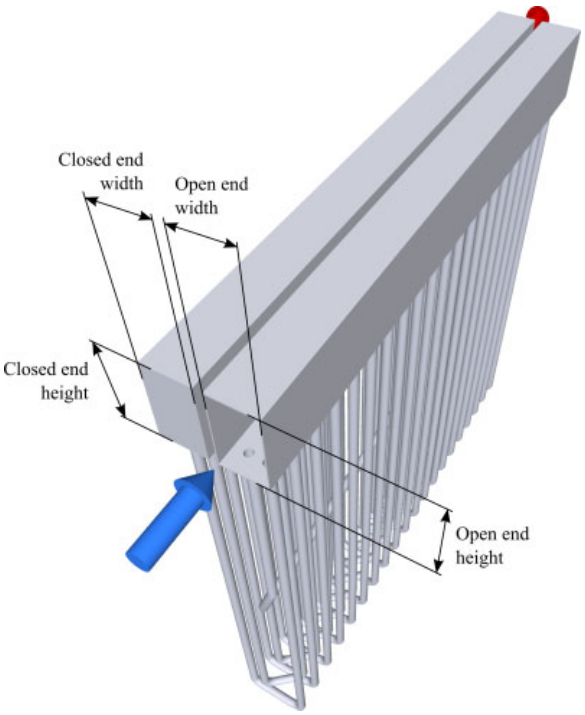


Figure 9. Open end and closed end dimensions of manifolds. This figure is available in colour online at www.apjChemEng.com.

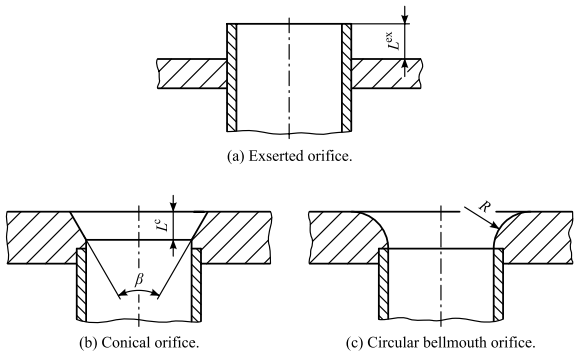


Figure 10. U-tube orifice types.

corresponding collector section differs from the pressures at the outlet of this U-tube pair, modify flow rates through these U-tubes appropriately.

(5) Check and normalize lateral flow rates. This is necessary due to modifications made in Step 4.

(6) Compare current lateral flow rates to the previous ones saved in Step 3. In case the greatest difference is small enough (e.g. less than 10^{-6}), the solution has been found; otherwise go to Step 3.

(7) Considering the ideal mass flow rate through one U-tube

$$\dot{m}_{\mathrm{id}} = \frac{\dot{m}_{\mathrm{tot}}}{2n} \tag{15}$$

where $\dot{m}_{\mathrm{tot}}$ denotes the total mass flow rate through the entire system, compute the relative standard deviation from flow uniformity

$$\delta = \frac{100}{\dot{m}_{\mathrm{id}}} \sqrt{\frac{1}{2n} \sum_{i=1}^{n} \left[(\dot{m}_i^{\mathrm{I}} - \dot{m}_{\mathrm{id}})^2 + (\dot{m}_i^{\mathrm{O}} - \dot{m}_{\mathrm{id}})^2 \right]}. \tag{16}$$

Compute also the total pressure loss in the distribution system

$$\Delta p_{\mathrm{tot}} = p_{\mathrm{in}}^{\mathrm{dist.}} - p_{\mathrm{out}}^{\mathrm{coll.}}$$

with $p_{\mathrm{in}}^{\mathrm{dist.}}$ denoting pressure at distributor inlet and $p_{\mathrm{out}}^{\mathrm{coll.}}$ pressure at collector outlet.

Clearly, the objective is to ensure fluid distribution is as uniform as possible, i.e. to minimize the above-mentioned relative standard deviation from flow uniformity. To do so, only width and height at the open and closed ends of the manifolds are manipulated within the given dimension ranges. The built-in optimization algorithm is similar to the brute force approach; however, search space specified by width and height dimension ranges is not evaluated entirely. It is discretized with a user-defined step and flow distributions are found only for shapes yielded by this discretization. Since relative standard deviation from flow uniformity behaves well while changing one dimension only, the Golden Section Method, Ref. [37], pp. 51–52, is used whenever possible to lower the number of necessary evaluations. Although Fig. 11 suggests that some more efficient two-dimensional search algorithm might be applicable, this issue should be researched further to ensure that such an algorithm can provide global optimum for any reasonable set of input data.

Example optimization problem

Consider water flowing through the distribution system at a steady flow rate 40 kg·s^{-1}. Let the dimension ranges at open and closed ends of linearly tapered manifolds be as specified in Table 1 with U-tube orifices being exserted 12 mm into the manifolds. Also, let the discretization step be 1 mm.

For such input data, the best possible shape of both manifolds is a constant one with cross-sectional dimensions 170 × 300 mm (width × height). Relative standard deviation from flow uniformity then is 7.1% and total pressure loss is 2190 Pa. Flow rates through

DOI: 10.1002/apj

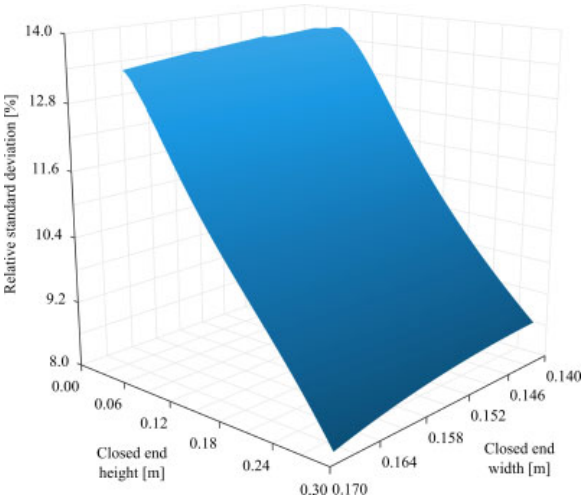


Figure 11. Relative standard deviation from flow uniformity for a linearly tapered manifold. Open end cross section has constant dimensions, 169 × 287 mm (width × height), closed end width varies from 140 to 169 mm, and closed end height from 50 to 287 mm. Total mass flow rate 55 kg·s^{-1} of water was assumed while computing data for this graph. This figure is available in colour online at www.apjChemEng.com.

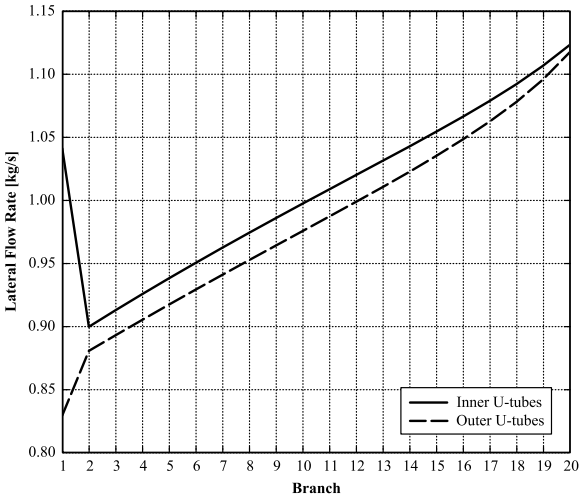


Figure 12. Lateral flow rates through U-tubes in case of the example distribution system with optimized manifolds.

Table 1. Dimension ranges for manifolds in the example distribution system.

	Open end		Closed end	
	Width	Height	Width	Height
Minimum	170	200	150	100
Maximum	170	300	170	300

All values are in mm.

U-tubes are shown in Fig. 12. Despite the 'almost brute force' approach, global optimum was obtained in approximately 500 s on a Microsoft Window 7 computer with a single-core AMD Athlon 64 3200+ CPU.

FUTURE WORK

The current version of the developed computational tool has several drawbacks that should be eliminated by future development:

(1) Additional width and height profile types will be investigated and added to the available ones if found suitable. The authors would also like to fine-tune the formula for coefficient of static regain to be applicable to arbitrarily shaped manifolds. Moreover, they would like to construct an algorithm that would yield the best width and height profile ensuring minimum possible nonuniformity (though the resulting profiles could prove themselves to be more costly and harder to manufacture than the profiles already available).
(2) An algebraic model for compressible fluids should be constructed.
(3) Even though optimization times are fairly short already, it would be great if they could be further reduced (for instance, by means of implementing a two-dimensional optimization algorithm or an intelligent algorithm selecting initial estimates of flow rates).
(4) Accuracy of pressure drop prediction should be verified.
(5) Although neglecting the effect of gravitational field should not produce significant errors, incorporating gravitational terms into the equations would increase precision.
(6) Since flow instability and branch proximity can notably influence uniformity of the distribution and heat exchanger fouling rate, these problems are considered to be analyzed.
(7) The presented computational tool can be regarded as sufficiently accurate; however, experimental testing is planned to further improve quality of results it provides.

CONCLUSIONS

A new formula for coefficient of static regain for parallel flow systems containing linearly tapered manifolds with rectangular cross sections connected by double U-tubes was presented. Although according to comparison with data from ANSYS FLUENT it seems that the formula performs quite well, physical experiments are

DOI: 10.1002/apj

planned to be performed to fine-tune it even further. Performance of the formula in case of other manifold profile types with nonlinear change of cross-sectional width and height has not been verified yet; however, preliminary tests indicate that the amount of necessary fine-tuning should not be too large.

The analytical model described in this paper can be used for constant-density fluids only and therefore an upgraded version for compressible fluids shall be constructed.

A practical computational tool for shape optimization of manifolds in the discussed distribution system was presented. Currently, the implemented computational algorithm takes a uniform distribution as an initial estimate. Since it may cause an increase in the number of iterations necessary to reach solution, this problem will be researched further.

NOMENCLATURE

Symbols

A	Identifier used in Churchill approximation of Darcy friction factor (−)
b	Width of a manifold cross section at a branch (m)
B	Identifier used in Churchill approximation of Darcy friction factor (−)
b^{fs}	Free space between U-tubes and sidewalls of a manifold (m)
C_{r}	Coefficient of static regain (−)
d	Inner diameter of a U-tube (m)
D_{h}	Hydraulic diameter (m)
$D_{\mathrm{h}}^{\mathrm{diff}}$	Difference in hydraulic diameter between two adjacent manifold sections (m)
$D_{\mathrm{h,avg}}^{\mathrm{diff}}$	Average difference in hydraulic diameter (m)
f	Darcy friction factor (−)
g	Standard gravity ($\mathrm{m\cdot s^{-2}}$)
h	Height of a manifold cross section at a branch (m)
l	Length of a manifold section (m)
L^{ex}	Exsertion of a U-tube pair into a manifold (m)
$\dot{m}^{\mathrm{I}}$	Mass flow rate through an inner U-tube ($\mathrm{kg\cdot s^{-1}}$)
$\dot{m}^{\mathrm{O}}$	Mass flow rate through an outer U-tube ($\mathrm{kg\cdot s^{-1}}$)
$\dot{m}_{\mathrm{id}}$	Ideal mass flow rate through one U-tube ($\mathrm{kg\cdot s^{-1}}$)
$\dot{m}_{\mathrm{tot}}$	Total mass flow rate through the entire distribution system ($\mathrm{kg\cdot s^{-1}}$)
n	Number of lateral branches (−)
p	Pressure (Pa)
$p_{\mathrm{in}}^{\mathrm{dist.}}$	Pressure at distributor inlet (Pa)
$p_{\mathrm{out}}^{\mathrm{coll.}}$	Pressure at collector outlet (Pa)
Re	Reynolds number (−)
t	Time (s)
T	Thermodynamic temperature (K)
v	Mean fluid velocity in a manifold section ($\mathrm{m\cdot s^{-1}}$)
w	Mean fluid velocity in a U-tube ($\mathrm{m\cdot s^{-1}}$)
z	Vertical distance of a manifold cross section from a reference plane (m)

Greek letters

δ	Relative standard deviation from flow uniformity (%)
Δp	Minor loss (Pa)
Δp_{tot}	Total pressure loss in the entire distribution system (Pa)
ε	Absolute roughness of inner surface (m)
μ	Dynamic viscosity (Pa·s)
ξ	Coefficient of hydraulic resistance (−)
ρ	Density ($\mathrm{kg\cdot m^{-3}}$)

Superscripts

L	Quantity just upstream of a branch
M	Quantity in the middle of a section
R	Quantity just downstream of a branch

Acknowledgement

The authors gratefully acknowledge the financial support of the Ministry of Education, Youth and Sports of the Czech Republic within the framework of the research plan No. MSM 0021630502 'Waste and Biomass Utilization Focused on Environment Protection and Energy Generation'.

REFERENCES

[1] J. Klemeš, F. Friedler, I. Bulatov, P. Varbanov. *Sustainability in the Process Industry: Integration and Optimization*, McGraw-Hill: New York, **2010;** 352 pp.
[2] F. Friedler. *Chem. Eng. Trans.,* **2009;** *18*, 1–26. DOI 10.3303/CET0918001.
[3] B. Master, K. Chunangad, B. Boxma, D. Král, P. Stehlík. *Heat Transfer Eng.,* **2006;** *27*, 4–11. DOI 10.1080/01457630600671960.
[4] Z. Jegla, B. Kilkovský, P. Stehlík. *Heat Transfer Eng.,* **2010;** *31*, 757–765. DOI 10.1080/01457630903500932.
[5] A. Acrivos, B.D. Babcock, R.L. Pigford. *Chem. Eng. Sci.,* **1959;** *10*, 112–124. DOI 10.1016/0009-2509(59)80030-0.
[6] B.J. Bailey. *J. Mech. Eng. Sci.,* **1975;** *17*, 338–347. DOI 10.1243/JMES_JOUR_1975_017_048_02.
[7] F. Lu, Y. Luo, S. Yang. *J. Hydrodyn. Ser. B,* **2007;** *20*, 179–185. DOI 10.1016/S1001-6058(08)60044-X.
[8] H. Fu, A.P. Watkins, M. Yianneskis. *Int. J. Numer. Methods Fluids,* **1994;** *18*, 871–886. DOI 10.1002/fld.1650180906.
[9] G.F. Jones, N. Lior. *Sol. Energy,* **1994;** *52*, 289–300. DOI 10.1016/0038-092X(94)90496-0.
[10] J. Koh, H. Seo, C.G. Lee, Y. Yoo, H.C. Lim. *J. Power Sources,* **2003;** *115*, 54–65. DOI 10.1016/S0378-7753(02)00615-8.
[11] W. Zhang, P. Hu, X. Lai, L. Peng. *J. Power Sources,* **2009;** *194*, 931–940. DOI 10.1016/j.jpowsour.2009.05.033.
[12] D.K. Chandraker, N.K. Maheshwari, D. Saha, V. Venkat Raj. *Exp. Therm. Fluid Sci.,* **2002;** *27*, 11–24. DOI 10.1016/S0894-1777(02)00202-9.
[13] M.A. Habib, R. Ben-Mansour, S.A. Said, M.S. Al-Qahtani, J.J. Al-Bagawi, K.M. Al-Mansour. *Comput. Fluids,* **2009;** *38*, 677–690. DOI 10.1016/j.compfluid.2008.07.004.
[14] A.B. Datta, A.K. Majumdar. *Int. J. Heat Fluid Flow,* **1980;** *2*, 253–262. DOI 10.1016/0142-727X(80)90019-3.
[15] R.A. Bajura, E.H. Jones. *J. Fluids Eng.,* **1976;** *98*, 654–666.
[16] Z. Miao, T. Xu. *Appl. Therm. Eng.,* **2006;** *26*, 396–402. DOI 10.1016/j.applthermaleng.2005.06.013.
[17] K.C. Toh, X.Y. Chen, J.C. Chai. *Int. J. Heat Mass Transfer,* **2002;** *45*, 5133–5141. DOI 10.1016/S0017-9310(02)00223-5.

[18] J. Wang. *Int. J. Hydrogen Energy,* **2008;** *33*, 6339–6350. DOI 10.1016/j.ijhydene.2008.08.020.
[19] J. Nie, Y. Chen. *Int. J. Hydrogen Energy,* **2010;** *35*, 3183–3197. DOI 10.1016/j.ijhydene.2010.01.050.
[20] S.H. Choi, S. Shin, Y.I. Cho. *Int. Commun. Heat Mass Transfer,* **1993;** *20*, 221–234. DOI 10.1016/0735-1933(93)90050-6.
[21] S.H. Choi, S. Shin, Y.I. Cho. *Int. Commun. Heat Mass Transfer,* **1993;** *20*, 607–617. DOI 10.1016/0735-1933(93)90073-5.
[22] L. Pustylnik, D. Barnea, Y. Taitel. *Chem. Eng. Sci.,* **2010;** *65*, 2552–2557. DOI 10.1016/j.ces.2009.12.032.
[23] M. Ablanque, C. Oliet, J. Rigola, C.D. Pérez-Segarra, A. Oliva. *Int. J. Therm. Sci.,* **2010;** *49*, 909–921. DOI 10.1016/j.ijthermalsci.2009.11.005.
[24] S. Kim, E. Choi, Y.I. Cho. *Int. Commun. Heat Mass Transfer,* **1995;** *22*, 329–341. DOI 10.1016/0735-1933(95)00024-S.
[25] J.C. Tong, E. Sparrow, J.P. Abraham. *Appl. Therm. Eng.,* **2009;** *29*, 3552–3560. DOI 10.1016/j.applthermaleng.2009.06.010.
[26] F.M. White. *Fluid Mechanics* 4th edn, McGraw-Hill: New York, **1998;** 826 pp.
[27] R. Darby. *Chemical Engineering Fluid Mechanics, Revised and Expanded*, Marcel Dekker, Inc.: New York, **2001;** 580 pp.
[28] C.F. Colebrook. *J. Inst. Civ. Eng.,* **1939;** *11*, 133–156.
[29] S.W. Churchill. *Chem. Eng.,* **1977;** *84*, 91–92.
[30] T.K. Serghides. *Chem. Eng.,* **1984;** *91*, 63–64.
[31] S.E. Haaland. *J. Fluids Eng.,* **1983;** *105*, 89–90. DOI 10.1115/1.3240948.
[32] B.J. Schorle, S.W. Churchill, M. Shacham. *Ind. Eng. Chem. Fundam.,* **1980;** *19*, 228–230. DOI 10.1021/i160074a019.
[33] I.E. Idelchik. *Handbook of Hydraulic Resistance* 3rd edn, Begell House Publishers: Redding, **2001;** 790 pp.
[34] Dortmund Data Bank Software and Separation Technology GmbH. Saturated Liquid Density (online). Available from www.ddbst.com/en/online/Online_Calc_den_Form.php. (Accessed 11 March 2010).
[35] Dortmund Data Bank Software and Separation Technology GmbH. Saturated Liquid Viscosity (online). Available from www.ddbst.com/en/online/Online_Calc_visc_Form.php. (Accessed 11 March 2010).
[36] FLUENT User's Guide, Fluent Inc.: Lebanon, USA, **2006;** 2501 pp.
[37] A. Ravindran, K.M. Ragsdell, G.V. Reklaitis. *Engineering Optimization: Methods and Applications* 2nd edn, John Wiley & Sons: Hoboken, **2006;** 688 pp.