Compressible flows in process equipment: Problems, methods and models

Stine Mia Rømmesmo Skrataas

Master of Science in Energy and Environment
Submission date: June 2011
Supervisor: Ivar Ståle Ertesvåg, EPT
Co-supervisor: Geir Berge, Petrell AS

Norwegian University of Science and Technology
Department of Energy and Process Engineering

Noregs teknisk-
naturvitskapelige universitet
NTNU

Fakultet for ingeniørvitskap og teknologi
Institutt for energi- og prosessteknikk

NTNU

EPT-M-2010-57

MASTEROPPGÅVE

for
Stine Mia Rømmesmo Skrataas

Våren 2011

Kompressibel strøyming i prosess-utstyr: problemstillingar, metodar og modellar

Compressible flows in process equipment: Problems, methods and models

Bakgrunn og føremål

I petroleums- og annan prosessindustri er kompressible fenomen sentrale i svært mange prosessar. I tillegg er faseovergang mellom gass, væske og faststoff ofte viktige delar av prosessane.

Kompressibel strøyming er basis for mange industrielle prosessar. Evne til å handtere sterkt kompressible fenomen kan vere avgjerande for å kunne rekne på slike problemstillingar. Kommersielle CFD-system (Computational Fluid Dynaminc) og tradisjonelle løysingsskjema er i hovedsak utvikla for inkompressible strøymingar. Jamvel litteraturen er avgrensa når ein kjem til fullt ut kompressible problem, særleg for blandingar.

Petrell AS (www.petrell.no) i Trondheim brukar eigen programvare som integrerer numerisk fluiddynamikk (CFD), termodynamikk og konstruksjonsteknikk for å rekne på strøyming, masse- og varmetransport i og omkring prosessutstyr.

Masteroppgåva er ei vidareføring av prosjektoppgåva hausten 2010, som hadde vekt på å kartlegge eksisterande teori og modellar (løysingsskjema) for handtering av kompressibel strøming. I denne oppgåva vil ein teste ut nokre av desse modellane for å kartlegge sterke og veike sider av modellane for bruk i sterkt kompressibel strømning.

I oppgåva skal kandidaten

- Setje seg inn i Petrell sin programvare Brilliant til eit nivå der ein kan gjere endringar på innlagde løysingskjema og legge inn nye skjema.
- Med utgangspukt i det utførte litteraturstudiet og i samarbeid med rettleiarane legge inn i Brilliant nokre av metodane som er eigna for kompressibel strømning.
- Saman med rettleiarane etablere nokre testeksempel som skal vere basis for samanlikning mellom metodane. Eventuelt også teste ut endringar på løysingskjema som måtte kome fram med utgangspunkt i eige arbeid.
- Gjennomføre testar, drøfte resultat og drøfte sterke og svake sider ved modellane.

Ein framdriftsplan *(Planlagde aktivitetar med tidsplan for framdrift)* for hele oppgåva skal leggast fram for faglærar/rettleiar(ar) for kommentarar innan 14 dagar etter at oppgåveteksten er utlevert.

Oppgåvesvaret skal redigerast mest mogleg som ein forskingsrapport med innhaldsliste, eit samandrag på norsk, konklusjon, litteraturliste, etc. Kandidaten skal leggje vekt på å gjere rapporten lettleseleg og oversiktleg. Det er viktig at naudsynte tilvisingar mellom stadar som svarar til kvarandre i tekst, tabellar og figurar, er gitt begge stadar. Når arbeidet skal vurderast, vert det lagt stor vekt på at resultata er godt gjennomarbeidde, at dei er oversiktleg framstilte (grafisk, tabellarisk), og at dei er utførleg drøfta.

Det er ein føresetnad at kandidaten på eige initiativ opprettar eit tilfredsstillande kontakttilhøve med faglærar og rettleiar(ar).

Kandidaten skal rette seg etter arbeidsreglement ved Petrell AS, og etter eventuelt andre pålegg frå den ansvarlege leinga. Kandidaten må ikkje gripe inn i styring og drift av anlegg, installasjonar og liknande utan etter avtale med ansvarshavande.

I samsvar med "Utfyllende reglar til studieforskrifta for teknologistudiet/sivilingeniørstudiet" ved NTNU § 20, tingar instituttet seg rett til å nytte alle resultat til undervisnings- og forskingsføremål, og dessutan til publikasjonar.

Sluttrapporten for oppgåva skal leverast innbunden i _3_ komplette eksemplar med eit "lausblad" med konsentrert samandrag med forfattarnamn og oppgåvetittel for eventuell referering i tidsskrift (maksimum éi maskinskriven side med dobbel linjeavstand. Fleire kopiar av rapporten ut over dette, til evt. medrettleiar(ar)/kontaktpersonar skal avtalast med, og evt. leverast direkte til dei det gjeld. Til faglærar/instituttet skal det også leverast ein komplett kopi på CD-ROM i Word-format eller tilsvarande

Sluttrapporten skal innleverast inn til instituttet *innan 14. juni 2011.*

Institutt for energi- og prosessteknikk, 17. januar 2011

[illegible]

Olav Bolland
Instituttleiar

Ivar Ertesvåg

Ivar S. Ertesvåg
Faglærar/rettleiar

Medrettleiar/kontaktperson:
Geir Berge, Petrell AS

Preface

This master's thesis covers 30 units, or one semester, of my degree in Master of Science and Technology at the Norwegian University of Science and Technology. The work has consisted of implementing and testing of algorithms in Brilliant. The master's thesis is a continuation of the specialisation project last semester.

I would like to thank Petrell AS and Geir Berge for his guidance, he has been very helpful. At the Department of Energy and Process Engineering, I would like to thank my supervisor Ivar Ståle Ertesvåg.

Trondheim 10/06/2011

Mia Skrataas

Stine Mia Rømmesmo Skrataas

Abstract

SIMPLE, SIMPLER, SIMPLEC and IDEAL are solution procedures originally developed for incompressible flows and staggered grids. For SIMPLE, SIMPLER and SIMPLEC, extensions for collocated grids and for treatment of flows at all speeds have already been proposed. For IDEAL, only an extension for collocated grids has been found, and an extension for treatment of flows at all speeds is proposed here. Extended versions of SIMPLE and SIMPLER are implemented in Brilliant, a multiphysics CFD-program developed by Petrell AS. These implemented algorithms are compared to the existing solution procedure in Brilliant, an extended version of the SIMPLEC algorithm. As expected, SIMPLE and SIMPLEC gave almost identical solutions for all the three presented test cases. The values given by the SIMPLER algorithm differed slightly from the values given by the two other algorithms. When simulating a shock tube, all three algorithms showed large deviations from the quasi-analytical solution in some regions of the shock tube. The SIMPLER algorithm spent the least CPU time for this simulation example, while SIMPLE and SIMPLEC spent less CPU time than SIMPLER when simulating methane flow in a pipe. Even though the CPU time was not registered for the last simulation example, a pressure relief pipe, it was noticed that the time consumption was much greater for the SIMPLER algorithm than for SIMPLE and SIMPLEC.

Sammendrag

SIMPLE, SIMPLER, SIMPLEC og IDEAL er løsningsalgoritmer som i utgangspunktet er utviklet for inkompressibel strømning og forskjøvet nettverk. Utvidelser for samlokalisert nettverk og for strømninger ved alle hastigheter har allerede blitt foreslått for SIMPLE, SIMPLER og SIMPLEC. For IDEAL er kun en samlokalisert utvidelse funnet, og en utvidelse for strømninger ved alle hastigheter er her foreslått. Utvidede versjoner av SIMPLE og SIMPLER er implementert i Brilliant, et multifysikk CFD-program utviklet av Petrell AS. Metodene implementert i denne masteroppgaven ble sammenlignet med eksisterende løsningsmetode i Brilliant, en utvidet versjon av SIMPLEC-algoritmen. Som forventet ga SIMPLE og SIMPLEC nesten identiske resultater for alle de tre presenterte problemstillingene. Verdiene av variablene beregnet med SIMPLER-algoritmen avvek noe fra verdiene gitt av de to andre algoritmene. Ved simulering av et støtrør viste alle algoritmene store avvik fra analytisk løsning i deler av røret. Simuleringen av dette støtrøret med SIMPLER tok mindre CPU-tid enn simulering med SIMPLE og SIMPLEC. SIMPLE og SIMPLEC brukte mindre CPU-tid enn SIMPLER-algoritmen da strømning av metan i et rør ble simulert. Selv om bruk av CPU-tid ikke er registrert for det siste eksempelet, et trykkavlastningsrør, var det klart at SIMPLER brukte mye mer CPU-tid enn hva SIMPLE og SIMPLEC gjorde.

Contents

List of Figures

List of Tables

Nomenclature

Greek symbols

α	Relaxation factor
Γ	Diffusion coefficient
γ	Specific heat ratio
δ	Kronecker delta
μ	Dynamic viscosity
ρ	Density
$\underline{\tau}$	Viscous stress tensor
ϕ	Scalar variable

Roman symbols

ΔA	Area shared between two adjacent CVs
a	Coefficient
$\dot{b}$	Source term
b	Coefficient
c	Speed of sound
d	Coefficient
E	Total energy
$\vec{f}$	Volume forces
I	Unit tensor
k	Conductivity
n	Normal vector
$N1$	Number of repetitions for the first inner iteration process
$N2$	Number of repetitions for the second inner iteration process
p	Pressure
q	Flux
R	Specific gas constant
S	Source term

Sa	Dependent part of the source term
Sc	Independent part of the source term
T	Temperature
t	Time
u	Velocity in x direction
u_i	Velocity in x_i direction
$\hat{u}$	Pseudo velocity
$\bar{u}$	Weighted pseudo velocity
V	Volume
v	Velocity in y direction
$\vec{v}$	Velocity vector
w	Velocity in z direction
x	Direction of movement
y	Direction of movement
z	Direction of movement

Subscripts and superscripts

$'$	Correction value
$*$	Preliminary value
C	SIMPLEC approximation
E	SIMPLE approximation
e	Energy
n	Preliminary value
NB	Neighbouring points
nb	Neighbouring surface
P	Grid point P
p	Pressure
ρ	Density
$t-1$	Previous time step

u	Velocity
x_i	Direction of movement
x_j	Direction of movement
x_k	Direction of movement

1 Introduction

1.1 Background and motivation

The rapid increase in computer power provides the opportunity for more accurate results when simulating complex problems involving fluid flows and thermodynamics. In many cases, experiments are not feasible due to economy, time consumption, safety etc. For such cases, information can be gained through simulations. In order to get reliable simulation results, there is a great need of accuracy in the solution procedure.

Brilliant is a multiphysics CFD-program developed by Petrell AS. This CFD-program shows some weaknesses for simulations involving compressible flows. The existing solution procedure in Brilliant is based on the SIMPLEC algorithm. Originally, the SIMPLEC algorithm was proposed for incompressible flows and staggered grid, whereas the algorithm used in Brilliant is extended to treat both incompressible and compressible flows on a collocated grid. The SIMPLEC algorithm contains some approximations that could influence the stability and the convergence rate. Therefore, Petrell AS would like to investigate and implement new algorithms that might improve both the stability and the convergence rate.

1.2 Limitations

The goal for this master's thesis was implementation and testing of solution algorithms in the CFD-program Brilliant by Petrell AS. This CFD-program treats both compressible and incompressible flows, and the solution procedure is based on the finite volume method and collocated grids. As the existing SIMPLEC-like solution procedure in Brilliant and its functionalities are used as a foundation for new implementations, possible algorithms are here limited to SIMPLE-like algorithms.

Even though Brilliant handles multiphase flows, the theoretical study and implementation conducted in this master's thesis are limited to single-phase.

Due to the CPU time needed for simulating the test cases and the limited time for this master's thesis, the minimal CPU time and whether or not the procedure converges for different underrelaxation factors have only been registered for two of the three simulation examples.

2 Theory

2.1 Governing equations

The governing equations are based on conservation principles, and can be expressed as follows [1].

Continuity equation:
The continuity equation expresses the conservation of mass:

$$\frac{\partial}{\partial t}(\rho) + \nabla \cdot (\rho \vec{v}) = 0 \tag{1}$$

where ρ is the density and $\vec{v}$ is the velocity.

Momentum equation:
As stated by Newton's second law of motion, the rate of change of momentum in a system equals the sum of forces acting on the system:

$$\frac{\partial \rho \vec{v}}{\partial t} + \nabla \cdot (\vec{v} \otimes \rho \vec{v}) = -\nabla p + \nabla \cdot (\underline{\tau}) + \vec{f} \tag{2}$$

where $\underline{\tau}$ is the viscous stress tensor and $\vec{f}$ is the volume forces.

The viscous stress tensor can be expressed as follows:

$$\underline{\tau} = \mu[\nabla \vec{v} + \nabla \vec{v}^T] - \frac{2}{3}\mu \nabla \cdot \vec{v} I \tag{3}$$

where μ is the dynamic viscosity and I is the unit tensor.

The expressions for $\underline{\tau}$ and $\vec{v} \otimes \vec{v}$ in tensor notation are:

$$\tau_{ij} = \mu \left[\left(\frac{\partial u_i}{\partial x_j} \right) + \left(\frac{\partial u_j}{\partial x_i} \right) \right] - \frac{2}{3} \mu \left(\frac{\partial u_k}{\partial x_k} \right) \delta_{ij} \tag{4}$$

where δ is the kronecker delta ($\delta_{ij} = 1$ when $i = j$ and $\delta_{ij} = 0$ for $i \neq j$) and x_i, x_j x_k are directions of movement.

$$\vec{v} \otimes \vec{v} = u_i u_j \tag{5}$$

Energy equation:
As stated by the first law of thermodynamics, the energy equation expresses the conservation of energy:

$$\frac{\partial \rho E}{\partial t} + \nabla \cdot (\rho E \vec{v}) = -\nabla \cdot (p\vec{v}) + \nabla \cdot (\underline{\tau} \cdot \vec{v}) + \rho \vec{f} \cdot \vec{u} - \nabla \cdot q \tag{6}$$

where $\vec{f}$ represents the volume forces, E is the sum of internal and kinetic energy per unit mass and q is the heat flux:

$$E = e + \frac{1}{2} |\vec{v}|^2 \tag{7}$$

Fourier's law for heat conduction, q:

$$q = -k \nabla T \tag{8}$$

where k is the thermal conductivity and T is the temperature.

The similar structure of Eqs. (1), (2) and (6) makes it possible to express a general transport equation. A general transport equation expresses transportation of other scalar properties needed to describe fluid dynamic problems as well, like turbulence and mass fraction.

General transport equation:
The general transport equation expresses transportation of the general quantity, ϕ:

$$\underbrace{\frac{\partial}{\partial t}(\rho\phi)}_{I} + \underbrace{\nabla \cdot (\rho \vec{v} \phi)}_{II} = \underbrace{\nabla \cdot (\Gamma \nabla \phi)}_{III} + \underbrace{\vec{f}}_{IV} \tag{9}$$

I	Transient term
II	Convective term
III	Diffusive term with the general diffusion coefficient Γ
IV	Source or sink term

Equation of state for an ideal gas:
The pressure, temperature and density for an ideal gas are related through the equation of state:

$$p = \rho R T \tag{10}$$

where R is the specific gas constant.

2.2 The Finite Volume Method

There are several approaches for solving the Navier-Stokes Equations numerically. In the Finite Volume Method [1, 2], the equations are integrated over a certain volume and the domain is then divided into several control volumes, abbreviated CVs. The volume integrated conservation equations are applied to each CV. For multiphase flows, each phase can be described by a set of equations, and a source term in Eq. (1) expresses the transition between phases. By using the Gauss divergence theorem and a lower order discretisation scheme [1], the integral form of the governing equations, Eqs. (1), (2) and (9), can be expressed as follows [1]:

$$\left(\frac{\Delta(\rho V)}{\Delta t}\right)_{\mathrm{P}} + \sum_{\mathrm{nb}} (\rho u n \Delta A)_{\mathrm{nb}} = S_{\mathrm{P}} \tag{11}$$

$$a_{\mathrm{P}}(u_{\mathrm{i}})_{\mathrm{P}} - \sum_{\mathrm{NB}} a_{\mathrm{NB}}(u_{\mathrm{i}})_{\mathrm{NB}} = -\sum_{\mathrm{nb}} (p n_i \Delta A)_{\mathrm{nb}} + b_{\mathrm{P}} \tag{12}$$

$$a_{\mathrm{P}}\phi_{\mathrm{P}} - \sum_{\mathrm{NB}} a_{\mathrm{NB}}(\phi)_{\mathrm{NB}} = b_{\mathrm{P}} \tag{13}$$

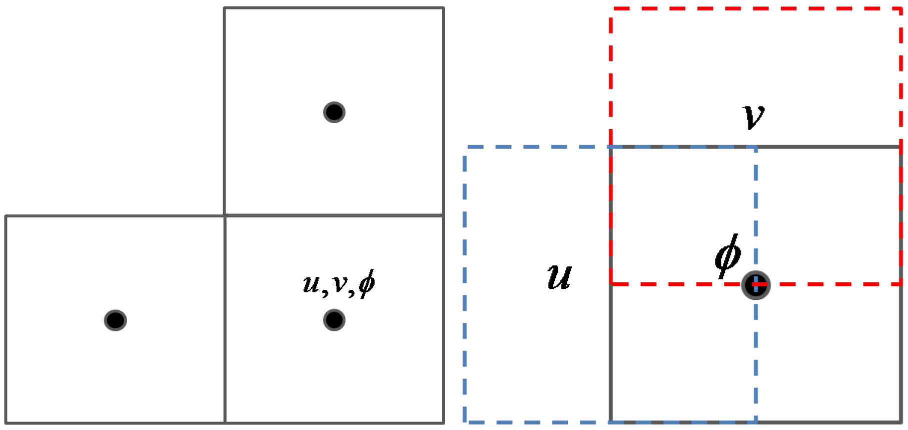
where S is a source term, the coefficient b can be calculated from preliminary and known values, $\sum_{\mathrm{nb}}$ denotes the sum over all surfaces and $\sum_{\mathrm{NB}}$ denotes the sum of neighbouring CVs. $(\rho u n \Delta A)_{\mathrm{nb}}$ in the continuity equation gives the mass flow normal to the surface nb and ΔA is the shared area between the two neighbouring control volumes. The source term, S, can be linearised as:

$$S = (Sa\phi)_{\mathrm{P}} + Sc \tag{14}$$

where $(Sa\phi)_{\mathrm{P}}$ is dependent of the variable, ϕ, in the central point and Sc is the part of the source term independent of ϕ.

The discretisation of the Navier-Stokes Equations and the derivation of Eqs. (11), (12) and (13) were discussed in more detail in the specialisation project [3].

For collocated grids, all variables are stored in the same point, which can be the midpoint of the control volume. The collocated grid is illustrated in Fig. 1a. As the momentum equation requires both the velocities and the pressure on the surfaces of each control volume, their values must be found by interpolation. If both the velocities and the pressure are linearly interpolated to the surface of the control volume, the calculated pressure field could assume an unrealistic checkerboard form [1]. In the staggered grid [4], however, the scalars are stored in the central points of the CV, and the velocities are stored on the surfaces, as illustrated in Fig 1b. With this kind of grid, there is no need for interpolation of the pressure for the momentum equation, and the unrealistic checkerboard form will not appear [5]. However, for three dimensional flows, it is necessary to define four control volumes for each control volume defined by the collocated grid, making the required storage memory large for the staggered grid compared to the collocated grid [6, 7]. Therefore, special methods for interpolating the velocities to the surfaces of the control volumes have been obtained, like the Rhie-Chow interpolation [8]. This kind of interpolation, called momentum interpolation, prevents the checkerboard form of the pressure from appearing by involving the pressure when the velocities are interpolated to the surface of the control volume. The Rhie-Cow interpolation is described in more detail in the specialisation project [3].



(a) Collocated grid. **(b)** Staggered grid.

Figure 1: Grid.

The methods for solving the Navier-Stokes Equations numerically may be divided into two main categories [9]: (1) density-based methods and (2) pressure-based methods. The density-based methods are traditionally used for solving compressible flows. In these methods, the continuity equation is used as an equation for the density, while the pressure is solved through the energy equation and state equations. This way of solving the Navier-Stokes Equations is well suited for highly compressible flows. As the Mach number decreases, however, the density changes are small and the interaction between the pressure and density weakens. Therefore, the density-based methods might lead to unstable solutions for low Mach number flows and for incompressible flows [10]. The pressure-based methods use the continuity equation as an equation for the pressure. These methods can furthermore be divided into the direct approach and the segregated approach. In the direct approach, the continuity equation and the momentum equation are solved simultaneously, which guarantees good interaction between the pressure and the velocity [9]. However, for nonlinear problems, the coefficients are functions of the solved variables and the equations have to be solved several times with updated coefficients. In addition, for three dimensional multiphase flow, the memory needed for storage of all the coefficients for the different phases and equations is large. Therefore, the direct approach is usually not favourable [5, 11]. The pressure correction method is the most common segregated method [9]. In this method, the velocity is solved from the momentum equation based on a guessed pressure field or a pressure field calculated from initial values for velocity. As the velocity field calculated from such a pressure field does not necessarily guarantee conservation of mass, it has to be corrected or updated.

The volume integrated Navier-Stokes Equations can be solved explicitly or implicitly with respect to time [1, 2]. For explicit calculations of the equations, the neighbouring variables are known from previous iteration level or from previous time step. When an equation is solved implicitly, for instance the momentum equation Eq. (12), the central velocity and the neighbouring velocities are at the same time step and iteration level, and a system of equations needs to be solved. As the explicit scheme uses neighbouring velocities known from previous time step or iteration level, the velocity in the point P can be directly calculated. The number of required operations is much larger for the implicit scheme than for the explicit scheme, hence giving a significant difference in CPU time required. As can be seen from Taylor truncation error analysis, both schemes are first order. The implicit scheme is unconditionally stable. The explicit scheme, however, is conditionally stable, hence limiting the size of the time step.

In the SIMPLE algorithm [12], a set of pressure correction equations are solved. The variables solved from this set of equations correct both the pressure and the velocity. Many algorithms have been proposed in an attempt to improve the performance of the SIMPLE algorithm. This group of SIMPLE-like algorithms is often called the SIMPLE-family. The SIMPLE algorithm and its descendants were originally proposed for incompressible and steady flows on staggered grids. However, these SIMPLE-like algorithms can be extended to also treat unsteady and compressible flows on collocated grids. Literature on extending SIMPLE, SIMPLER and SIMPLEC can be found in [10, 11]. The IDEAL algorithm has been extended to collocated grids [13], but no extension for flows at all speeds has been found. Therefore, such an extension of the IDEAL algorithm will be proposed in Sect. 2.2.4. The algorithms presented here are extensions to the original algorithms. In the following, the algorithms called SIMPLE, SIMPLER, SIMPLEC and IDEAL are extended versions of the algorithms.

2.2.1 Compressible and collocated extension of SIMPLE

In the SIMPLE (Semi-Implicit Method for Pressure-Linked Equations) algorithm by Patankar and Spalding [12], a preliminary guessed pressure field is used to solve the implicit momentum equation. Since the coefficients for the momentum equation depend on both the velocity and the density, these variables need a guessed initial value as well. As both the pressure and velocity need an initially guessed value, their preliminary values are assumed independently. The interconnection between the pressure and velocity is then neglected.

As the preliminary guessed values might be far from satisfying the momentum equation and the continuity equation, the variables must be updated. The updated values consist of preliminary values and correction values:

$$p = p^{\mathrm{n}} + p' \tag{15}$$

$$u_i = u_i^* + u_i' \tag{16}$$

$$\rho = \rho^* + \rho' \tag{17}$$

where the superscripts n and $*$ denotes preliminary values and the prime, $'$, denotes correction values.

The discretised implicit momentum equation, Eq. (12), for an arbitrary central point, P, can be expressed as follows:

$$a_{\mathrm{P}}(u_i)^*_{\mathrm{P}} - \sum_{\mathrm{NB}} a_{\mathrm{NB}}(u_i)^*_{\mathrm{NB}} = -\sum_{\mathrm{nb}}(p^{\mathrm{n}} n_i \Delta A)_{\mathrm{nb}} + (b^*_{\mathrm{m}})_{\mathrm{P}} \tag{18}$$

By subtracting the momentum equation based on preliminary variables, Eq. (18), from the momentum equation based on the corrected variables, Eq. (12), a correction-based momentum equation is obtained:

$$a_{\mathrm{P}}(u_i)'_{\mathrm{P}} - \sum_{\mathrm{NB}} a_{\mathrm{NB}}(u_i)'_{\mathrm{NB}} = -\sum_{\mathrm{nb}}(p' n_i \Delta A)_{\mathrm{nb}} + b'_{\mathrm{P}} \tag{19}$$

where the correction value of the coefficient b' is zero as the value of b can be calculated.

In order to simplify Eq. (19), the terms containing neighbouring velocity correction, $\sum_{\mathrm{NB}} a_{\mathrm{NB}}(u'_i)_{\mathrm{NB}}$, are neglected and the expression for the velocity becomes:

$$(u_i)'_{\mathrm{P}} = -\sum_{\mathrm{nb}}(p' n_i \Delta A)_{\mathrm{nb}} d^{\mathrm{E}}_{\mathrm{P}} \tag{20}$$

where $d^E = \frac{1}{a}$.

For a completely converged solution, all correction values are zero. Therefore, the omission of terms including the neighbouring velocity corrections will not influence the accuracy of the calculated variables if the solution procedure converges. However, the neglection of these terms might influence both the convergence rate and the stability [5]. By combining Eqs. (16) and (20), an equation for the total velocity based on the pressure correction and the preliminary velocity is obtained:

$$(u_i)_{\mathrm{P}} = (u_i)^*_{\mathrm{P}} - \sum_{\mathrm{nb}}(p' n \Delta A)_{\mathrm{nb}} d^{\mathrm{E}}_{\mathrm{P}} \tag{21}$$

As for the momentum equation, a correction-based continuity equation can be obtained by subtracting the continuity equation based on preliminary variables from the continuity equation based on corrected variables:

$$\begin{gathered}\left[\left(\frac{\Delta\big((\rho^* + \rho')V\big)}{\Delta t}\right)_{\mathrm{P}} + \sum_{\mathrm{nb}}((\rho^* + \rho')(u^* + u')n\Delta A)_{\mathrm{nb}} - (S^* + S')_{\mathrm{P}}\right] - \\ \left[\left(\frac{\Delta(\rho^* V)}{\Delta t}\right)_{\mathrm{P}} + \sum_{\mathrm{nb}}(\rho^* u^* n \Delta A)_{\mathrm{nb}} - {S^*}_{\mathrm{P}}\right] = 0\end{gathered} \tag{22}$$

When the accuracy of the calculated variables increases, the correction values will decrease. Terms including the multiplication of two correction values decrease faster than terms including only one correction value, and are therefore neglected:

$$\left(\frac{\Delta(\rho' V)}{\Delta t}\right)_{\mathrm{P}} + \sum_{\mathrm{nb}}((\rho^* u' + \rho' u^*)n\Delta A)_{\mathrm{nb}} = {S'}_{\mathrm{P}} \tag{23}$$

The discretised continuity equation requires the velocities on the surfaces of each control volume. Equation (20) expressed on the surface of an arbitrary CV becomes:

$$(u_i)'_{(\mathrm{P}+1/2)_i} = -\sum_{\mathrm{NB}} (pn_i \Delta A)_{\mathrm{NB}} d^{\mathrm{E}}{}_{(\mathrm{P}+1/2)_i} \tag{24}$$

For incompressible flows, the density can be calculated and no correction is needed. Therefore, the term $\rho' u^*$ in Eq. (23) equals zero. If compressible flows should be treated as well, this term can be expressed through the pressure correction. The relations between the pressure and the density for compressible flows, and the speed of sound are given by thermodynamic relations for an ideal gas [14]:

$$\rho' = \frac{\partial \rho}{\partial p} p' = \frac{1}{c^2} p' \tag{25}$$

$$c = \sqrt{\gamma R T} \tag{26}$$

where c is the speed of sound and γ is the specific heat ratio.

As the term including $\frac{u^* p'}{c^2}$ is small for low Mach numbers, this term becomes negligible compared to the term including $p^* u'$ for incompressible flows [10], and the expression for ρ' can thus be used for both compressible and incompressible flows.

The pressure correction equation is obtained by substituting Eqs. (24) and (25) into the correction-based continuity equation, Eq. (23):

$$\begin{aligned} \left(\frac{p'V}{c^2 \Delta t}\right)_{\mathrm{P}} + \sum_{\mathrm{NB}} a_{\mathrm{NB}} (p'_{\mathrm{P}} - p'{}_{\mathrm{NB}}) + \sum_{\mathrm{nb}} \left(\frac{1}{c^2} p' u^* n \Delta A\right)_{\mathrm{nb}} + \\ = (S)'{}_{\mathrm{P}} - \left(\frac{\rho' V}{\Delta t}\right)_{\mathrm{P}}^{\mathrm{t}-1} \end{aligned} \tag{27}$$

This equation involves values for the pressure correction both in CV centres and on CV surfaces. By using a differential scheme, such as the upwind scheme [1], the pressure corrections on CV surfaces can be expressed by central points, and the contributions from the surfaces are included in the central and neighbour coefficients. The pressure correction can then be expressed as:

$$a_{\mathrm{P}} p'_{\mathrm{P}} - \sum_{\mathrm{NB}} a_{\mathrm{NB}} p'_{\mathrm{NB}} = b_{\mathrm{P}} \tag{28}$$

As the neighbouring correction terms are neglected, the resulting equation tends to over-predict the value of the pressure correction [11]. To stabilise the solution procedure and enhance the convergence rate, the pressure is underrelaxed, hence updated by the sum of the old value and a fraction of the correction value. The density is updated the same way:

$$p^{\mathrm{new}} = p^{\mathrm{n}} + \alpha_{\mathrm{p}} p' \tag{29}$$

$$\rho^{\mathrm{new}} = \rho^{*} + \alpha_{\rho} \rho' \tag{30}$$

The velocity is first solved from the implicit momentum equation, Eq. (18), and corrected by the pressure correction, Eq. (21). Then, the corrected velocity and the preliminary velocity are weighted in order to get the new velocity:

$$u_i{}^{\mathrm{new}} = \alpha_{\mathrm{u}} u_i + (1 - \alpha_{\mathrm{u}}) u_i{}^{*} \tag{31}$$

As the coefficients for both the momentum equation and pressure correction equation require values for the velocity on the surface of each control volume, momentum interpolation, for instance Rhie-Chow interpolation [8], is used. Rhie-Chow interpolation is described in detail in the specialisation project [3]. The pressure in the momentum equation, Eq. (18), is linearly interpolated to the surface of the control volume.

Solution procedure for the extended SIMPLE algorithm:

1. Use initial values or values from the previous time step as preliminary values for the pressure, velocity, temperature and any other variables that might influence the coefficients or source terms. Calculate the density through thermodynamic relations, for instance the equation of state, Eq. (10). Calculate the speed of sound, for instance through the equation for an ideal gas, Eq. (26).

2. Calculate the coefficients and source terms for the momentum equation and solve the momentum equation implicitly, Eq. (18).

3. Calculate the coefficients and source terms for the pressure correction equation. Solve the pressure correction equation implicitly, Eq. (28).

4. Correct the pressure and density through Eqs. (25), (29) and (30). Update the preliminary velocity field through Eq. (21). Find a new value for the velocity by weighting the preliminary value and the updated value, Eq. (31).

5. Solve the energy equation and any other relevant transport equations through Eq. (13).

6. Update the speed of sound due to the new temperature through Eq. (26).

7. Return to step 2 with the updated values as a preliminary guess for the next iteration level. Repeat until convergence.

2.2.2 Compressible and collocated extension of SIMPLER

Since the coefficients in the discretised Navier-Stokes Equations depend on the velocity, both the pressure and the velocity require an initial guess in the SIMPLE algorithm. The initial velocity and pressure are then assumed independently, and their interconnection is neglected. In SIMPLER (SIMPLE Revised) by Patankar [5], the initial pressure and velocity are connected by introducing the pseudo velocity, and hence solving an equation for the pressure, based on an initially guessed velocity field. As the coefficients for the momentum equation do not contain any contribution from the pressure, only the density and the velocity need an initial guess.

The pseudo velocity is defined as:

$$(\hat{u}_i)^*_{\mathrm{P}} = \frac{\sum\limits_{\mathrm{NB}}(au_i^n)_{\mathrm{NB}} + b_{\mathrm{P}}}{a_{\mathrm{P}}} \tag{32}$$

where the contribution from the neighbouring velocities, $(au_i^n)_{\mathrm{NB}}$, are known from a preliminary guessed velocity field.

The continuity equation requires an expression for the velocities on the surfaces of the control volume. The pseudo velocity on the surface can be found by interpolation, and the explicit momentum equation on the surface of the CV can be expressed as:

$$(u_i)_{(\mathrm{P}+1/2)_i} = (\hat{u}_i)^*_{(\mathrm{P}+1/2)_i} - \sum_{\mathrm{NB}}(p^n n_i \Delta A)_{\mathrm{NB}} d^{\mathrm{E}}{}_{(\mathrm{P}+1/2)_i} \tag{33}$$

An equation for the pressure is obtained by substituting the explicit momentum equation into the continuity equation, Eq. (11):

$$\begin{aligned}\left(\frac{\rho V}{\Delta t}\right)_{\mathrm{P}} + \sum_{\mathrm{nb}}\left[(\rho n \Delta A)_{\mathrm{nb}}\left((\hat{u}_i)^*_{\mathrm{nb}} - \sum_{\mathrm{NB}}(p^n n_i \Delta A)_{\mathrm{NB}} d^{\mathrm{E}}{}_{\mathrm{nb}}\right)\right] \\ = S_{\mathrm{P}} + \left(\frac{\rho V}{\Delta t}\right)^{\mathrm{t}-1}_{\mathrm{P}}\end{aligned} \tag{34}$$

For incompressible flows, the density can be calculated. For compressible flows, on the other hand, the density is expressed through the pressure, using the relation in Eq. (25). The pressure equation can be expressed as follows:

$$a_{\mathrm{P}} p_{\mathrm{P}} - \sum_{\mathrm{NB}} a_{\mathrm{NB}} p_{\mathrm{NB}} = b_{\mathrm{P}} \tag{35}$$

This pressure field is regarded as a preliminary pressure field for the rest of the solution procedure, which coincides with the solution procedure for the SIMPLE algorithm. However, in the SIMPLER algorithm, the pressure correction corrects only the velocity and density, while the pressure is kept unchanged.

Solution procedure for the extended SIMPLER algorithm:

1. Use initial values or values from the previous time step as preliminary values for the velocity, temperature, pressure and any other variables that might influence coefficients or source terms. The pressure is used to calculate the density through thermodynamic relations, for instance the equation of state, Eq. (10). Calculate the speed of sound, for instance through Eq. (26) for ideal gases.
2. Calculate the coefficients and source terms for the momentum equation.
3. Calculate the pseudo velocities through Eq. (32).
4. Calculate the coefficients and source terms for the pressure equation. Solve the pressure equation, Eq. (35) implicitly, and use this pressure field as a preliminary pressure field for the rest of the iteration level.
5. Follow Step 2 through Step 7 for the SIMPLE algorithm, but do not correct the pressure in SIMPLE's Step 4.
6. Return to Step 2 with the updated values as a preliminary guess for the next iteration level. Repeat until convergence.

2.2.3 Compressible and collocated extension of SIMPLEC

The SIMPLEC algorithm (SIMPLE Consistent) by Van Doormaal and Raithby [15] follows the same procedure as the SIMPLE algorithm. However, the correction contributions from neighbouring velocities are approximated rather than neglected. In the SIMPLE algorithm, the velocity correction terms from neighbouring control volumes are neglected while the velocity correction term from the central point is kept. The values of the velocity correction from neighbouring CVs are then assumed to be negligible compared to the velocity correction in the central point. However, nearby velocity corrections are more likely of the same magnitude.

In order to obtain the pressure correction equation, $\sum_{\mathrm{NB}} a_{\mathrm{NB}}(u_i)'_{\mathrm{P}}$ is subtracted from both sides of Eq. (19):

$$\left(a_{\mathrm{P}} - \sum_{\mathrm{NB}} a_{\mathrm{NB}}\right)(u_i)'_{\mathrm{P}} - \sum_{\mathrm{NB}} a_{\mathrm{NB}}(u'_{\mathrm{NB}} - (u_i)'_{\mathrm{P}}) = -\sum_{\mathrm{nb}}(p' n_i \Delta A)_{\mathrm{nb}} d^{\mathrm{E}}_{\mathrm{P}} \quad (36)$$

Assuming that the velocity corrections in nearby CVs are of the same magnitude, the term $\sum_{\mathrm{NB}} a_{\mathrm{NB}}(u'_{\mathrm{NB}} - (u_i)'_{\mathrm{P}})$ can be neglected. The velocity correction can be expressed as:

$$(u_i)'_{\mathrm{P}} = -\sum_{\mathrm{nb}}(p' n_i \Delta A)_{\mathrm{nb}} d^{\mathrm{C}}_{\mathrm{P}} \quad (37)$$

where:

$$d^C = \frac{1}{a - \sum\limits_{\mathrm{NB}} a_{\mathrm{NB}}} \quad (38)$$

Solution procedure for the extended SIMPLEC algorithm:

The SIMPLEC algorithm follows the same solution procedure as the SIMPLE algorithm. However, the coefficient d^{C} differs from the coefficient d^{E} used in the SIMPLE algorithm. In the coefficient for the SIMPLEC algorithm, the correction contribution from neighbouring velocities is subtracted from the central coefficient.

2.2.4 Compressible and collocated extension of IDEAL

In the IDEAL (Inner Doubly Efficient Algorithm for Linked Equations) algorithm, Sun et al. [16, 17] claim to almost fully overcome both of the two main approximations in the SIMPLE algorithm:(1) The preliminary velocity and pressure are assumed independently and (2) the velocity correction from neighbouring CVs can be neglected. Each iteration level in IDEAL consists of two inner iteration processes.

First inner iteration process:

The first inner iteration process connects the preliminary pressure and velocity fields. As for the SIMPLER algorithm, pseudo velocities, Eq. (32), are calculated from known preliminary velocities. However, the pseudo velocity used in the IDEAL algorithm is weighted between the pseudo velocity described in Sect. 2.2.2, and the preliminary velocity:

$$(\bar{u}_i)^*_{\mathrm{P}} = \alpha_u (\hat{u}_i)^*_{\mathrm{P}} + (1 - \alpha_u)(u_i)^*_{\mathrm{P}} \quad (39)$$

As described in Sect.2.2.1, the coefficients for the momentum equation requires the velocities on the surface of the control volumes. These velocities are found by momentum interpolation for the collocated grid. For the pseudo velocities, on the other hand, linear interpolation is used. The pressure equation based on these pseudo velocities, Eq. (35), is then solved. When the preliminary pressure field is known, the explicit momentum equation gives updated values for the velocity field:

$$(u_i)_{\mathrm{P}} = \alpha_u(\hat{u})^*_{\mathrm{P}} - \sum_{\mathrm{nb}}(pn_i\Delta A)_{\mathrm{nb}} d^{\mathrm{E}}_{\mathrm{P}} + (1-\alpha_u)(u_i)^*_{\mathrm{P}} \tag{40}$$

where the preliminary velocity and the pseudo velocity are weighted in order to get the updated velocity.

The velocities calculated from the explicit momentum equation are then regarded as new preliminary velocities, $(u_i)^* = (u_i)$. This velocity field is used to update the pseudo velocities, but the coefficients and source terms both remain unchanged. Once more, the pressure equation is solved and updated velocities are found. The first inner iteration process is repeated N1 times, and the resulting pressure field from this iteration process is regarded as a preliminary pressure field for the second inner iteration process, $p^{\mathrm{n}} = p$. The resulting velocity field is solved from the implicit momentum equation, Eq. (18).

Second inner iteration process:

The second inner iteration process follows the same procedure as the first. For this inner iteration process, the resulting velocity from the first inner iteration process is used as a preliminary velocity field. The pseudo velocity is continuously updated due to the new velocity from the previous iteration step. This inner iteration process is repeated N2 times.

The number of repetitions for the first and second inner iteration processes, N1 and N2, are adjusted ensuring that the continuity equation and the momentum equation are almost satisfied for each iteration level.

In the compressible extensions of SIMPLE, SIMPLER and SIMPLEC, the density is corrected through the pressure correction and the speed of sound. As the pressure correction equation is not solved for the IDEAL algorithm, the pressure correction is not known. For the compressible extension of the IDEAL algorithm, the density can be updated due to changes in the pressure, hence giving a correction-like update:

$$p' = p - p^{\mathrm{n}} \tag{41}$$

When the change in pressure is calculated, the density is updated as described in Sect. 2.2.1.

Solution procedure for the extended IDEAL algorithm:

1. Use initial values or values from the previous time step as preliminary values for the velocity, temperature, pressure and any other variables that might influence the coefficients and source terms. Calculate the density through thermodynamic relations, for instance the equation of state, Eq. (10). Calculate the speed of sound, for instance through Eq. (26) for an ideal gas.
2. Calculate the coefficients and source terms for the momentum equation. Calculate pseudo velocities based on the preliminary velocities through Eqs. (32) and (39).
3. Calculate the coefficients for the pressure equation and solve the pressure equation implicitly, Eq. (35).
4. Calculate the velocity field based on this pressure field through the explicit momentum equation, Eq. (40).
5. Update the pseudo velocity due to the updated velocity, Eqs. (32) and (39).
6. Repeat Step 3 through Step 5 N1 times, and use the resulting pressure field as a preliminary pressure field for Step 7.
7. Solve the implicit momentum equation, Eq. (18), based on the preliminary pressure field.
8. Update the pseudo velocity due to the new velocity from the previous step, Eqs. (32) and (39).
9. Solve the pressure equation implicitly, Eq. (35).
10. Calculate the velocity through the explicit momentum equation, Eq. (40).
11. Update the pseudo velocity based on the new velocity from the previous step.
12. Repeat Step 9 through Step 11 N2 times.
13. Solve the energy equation and any other relevant transport equations through Eq. (13).
14. Calculate the speed of sound, Eq. (26), and the change in pressure, Eq. (41).
15. Update the density due to the change in pressure through Eqs. (17) and (25).

16. Return to Step 2 with the updated values as a preliminary guess for the next iteration level. Repeat until convergence.

2.2.5 Comparison of the solution algorithms

In the specialisation project [3], several of the algorithms in the SIMPLE-family were described and evaluated. Due to literature presenting promising results for some test cases for incompressible flows, the IDEAL algorithm was regarded as the most interesting algorithm for Brilliant. As SIMPLE and SIMPLER share some features with SIMPLEC and IDEAL, both SIMPLE and SIMPLER are also described in this master's thesis. The only difference between SIMPLE and SIMPLEC is the calculation of the coefficient d in the pressure correction equation. Neither SIMPLER nor IDEAL corrects the pressure through the pressure correction, and they both solve the pressure field through the pressure equation.

As mentioned in the previous section, the two main assumptions in the SIMPLE algorithm are [16]: (1) The preliminary velocity and pressure are assumed independently and (2) the velocity correction contribution from neighbouring control volumes can be neglected in the pressure correction equation. None of these assumptions will affect the values of the calculated variables if the solution procedure converges. However, both assumptions might influence the convergence rate and the stability [5, 18].

In the SIMPLER algorithm, the preliminary pressure and velocity fields are connected by solving an equation for the pressure based on the pseudo velocities. The pressure correction equation is solved, and both the velocity and density are corrected by the pressure correction. As in the SIMPLE algorithm, the velocity correction contributions from neighbouring CVs are neglected in order to obtain the pressure correction equation.

As the SIMPLEC algorithm follows the same procedure as the SIMPLE algorithm, the preliminary velocity and pressure field are assumed independently. However, the second approximation is improved in the SIMPLEC algorithm, by neglecting less significant terms than those neglected in the SIMPLE algorithm.

In the algorithms solving the pressure correction equation, assumptions about the velocity correction contribution from neighbouring control volumes are made in order to simplify the pressure correction equation. In the IDEAL algorithm, Sun et al. [16, 17] claim to almost overcome both of the two main assumptions made in the SIMPLE algorithm. The preliminary pressure field is calculated based on preliminary velocities and the pressure correction equation is not solved at all.

For all the presented algorithms, the convergence rate depends on when and how often the coefficients, source terms and density are updated. The accuracy of the density again depends on when and how often the energy equation is solved. In the IDEAL algorithm, both the velocity field and the pressure field are updated several times each iteration level. However, the coefficients and hence the density are only updated once.

3 Existing and implemented numerical procedures in Brilliant

Computational Fluid Dynamics (CFD) is a generic term for calculations and analysis of fluids in motion and related topics, like heat transfer and chemical reactions [1]. The system to be analysed can be divided into smaller parts by a grid, and the discretised Navier-Stokes Equations are applied to each CV or node in the grid.

Brilliant is a multiphysics CFD-program developed by Petrell AS. The program code is object orientated and written in C++. Brilliant is used to analyse fluid flow, fires, gas leakages or dispersion, radiation, conduction in solid material and stress analysis. This CFD-program is also used as a platform for the simulation program VessFire. VessFire is used in analysis of thermo-mechanical response during blow-down of process segment and equipment.

3.1 Grid and discretisation

In structured grids [1], the coordinates of neighbouring control volumes are automatically known. Information about each CV in the domain can easily be stored in a three dimensional array due to geometric position. However, as structured grids involve lines or curves that cannot be broken, very complex geometries are poorly represented by a structured grid. In unstructured grids [1], the control volume could have any shape, but tetrahedral or hexahedral elements are the most common elements in three dimensional problems. Even though many CVs can meet along a line, the surface of a CV is usually shared with only one neighbouring CV. The unstructured grid gives larger geometric flexibility than the structured grid. It is, however, more complicated to store

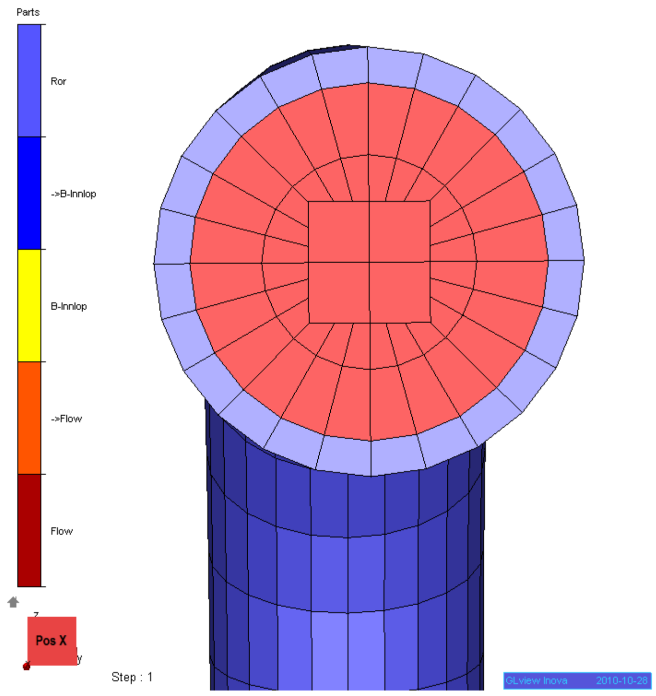


Figure 2: Grid in Brilliant, viewed with GL view Inova.

information related to the CVs in an organised way.

Brilliant uses a grid where the number of neighbouring CVs is arbitrary, here called an irregular grid. Control volumes with different shapes are combined in order to give a good representation of the domain. Information about the neighbouring control volumes is directly connected to each control volume. This kind of grid makes it possible to split or combine CVs during the simulation. All variables are stored in the midpoint of the cells, and the velocities on the surfaces are found by Rhie-Chow interpolation. Figure 2 shows a pipe gridded in Brilliant. The irregular grid gives a good approximation of the geometry of the pipe.

In Brilliant the finite volume method is applied for solving problems involving fluids in motion, and the finite element method is applied for stress analysis. Lower order differencing schemes [1] are used for the discretisation of convective terms, and the pressure and velocity are coupled through the SIMPLEC algorithm.

3.2 Changes implemented in the solution procedure

Previously, the only solution procedure implemented in Brilliant was an extended version of the SIMPLEC algorithm. However, both the SIMPLE algorithm and the SIMPLER algorithm were implemented in Brilliant as a part of this master's thesis. Since the SIMPLE-like algorithms more or less follow the same solution procedure, the existing SIMPLEC algorithm was used as a foundation for the implementation of both SIMPLE and SIMPLER. It was made an attempt of implementing the IDEAL algorithm as well, but the implemented procedure diverged even for simple test cases.

3.2.1 Pressure correction and momentum equation in SIMPLEC

As discussed in the specialisation project fall 2010 [3], Brilliant shows some instability problems for simulations involving compressible fluid flows. The main motivation for the specialisation project was finding literature on algorithms that might improve both the stability and the convergence rate compared to the existing SIMPLEC-like algorithm.

Early 2011 Petrell AS made some changes in the existing solution procedure. The changes made in solution procedure are shown in Fig. 3: (1) The implicit momentum equation in each direction were solved repeatedly with updated coefficients and (2) the pressure correction equation was solved repeatedly with updated pressure, density, velocities and coefficients until convergence.

Because of (1), the velocities in one direction were repeatedly solved with updated coefficients due to the newly calculated value for the velocity. The coefficients were updated due to changes in the velocity in one direction at the time, and the rest of the variables were kept unchanged. In some cases, the velocity in one direction could be forced to change due to deviations in another variable. Because of (2), the continuity equation was solved repeatedly with updated coefficients, making the variables satisfy only the continuity equation and not the momentum equation. Both (1) and (2) could result in large errors being brought into the next equation to be solved, and hence a possible source to instability.

When loops A, B, C and D in Fig. 3 were removed, both the momentum equation and the pressure correction equation were solved only once each time entering Loop F.

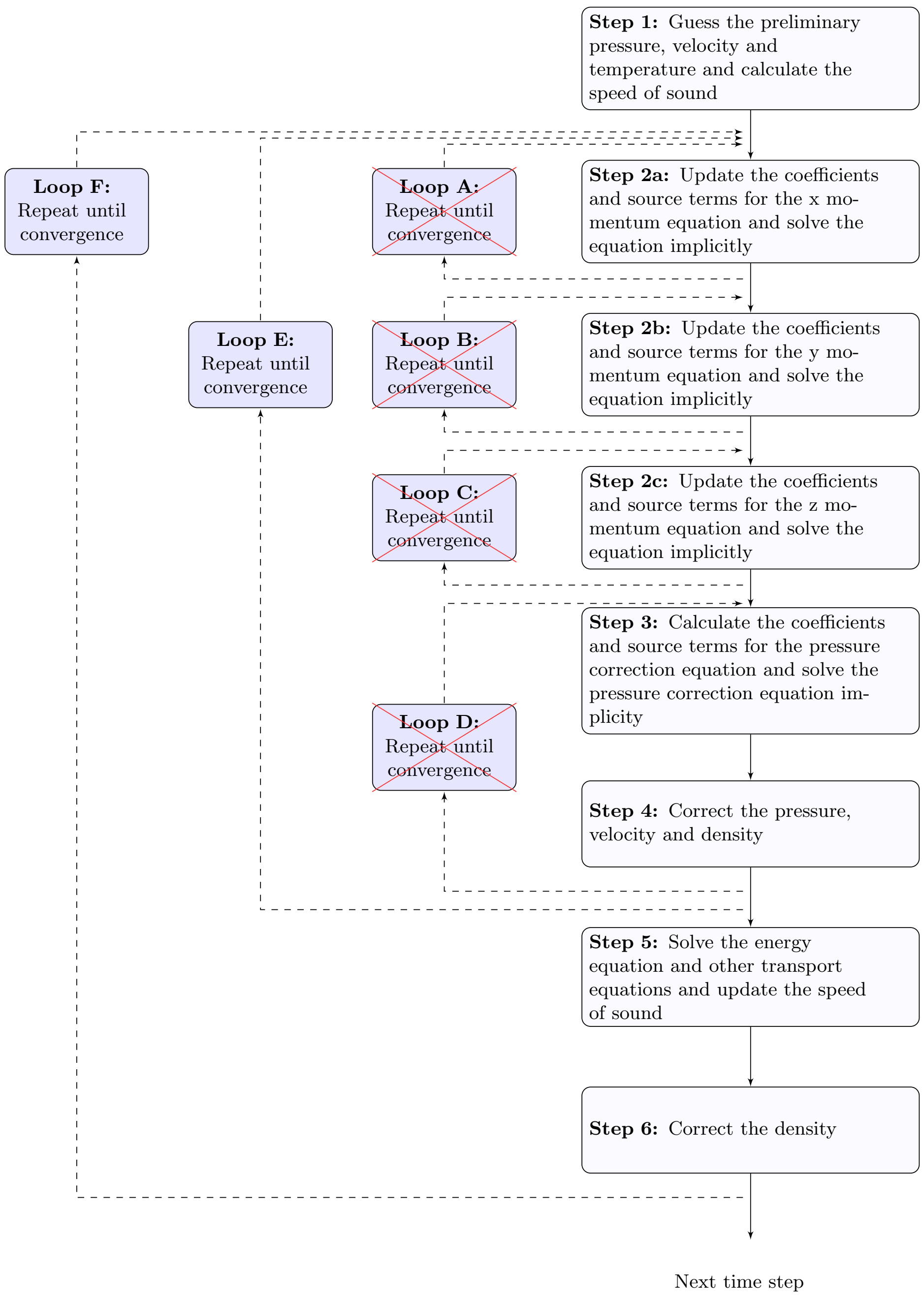
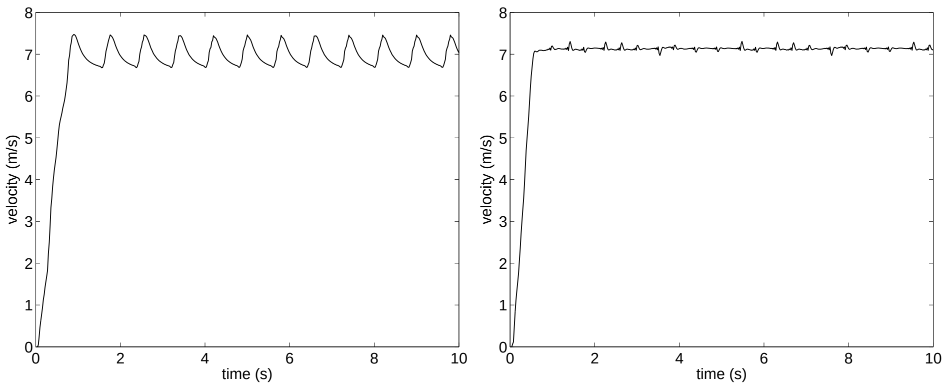

Step 1: Guess the preliminary pressure, velocity and temperature and calculate the speed of sound
Loop F: Repeat until convergence
Loop A: Repeat until convergence
Step 2a: Update the coefficients and source terms for the x momentum equation and solve the equation implicitly
Loop E: Repeat until convergence
Loop B: Repeat until convergence
Step 2b: Update the coefficients and source terms for the y momentum equation and solve the equation implicitly
Loop C: Repeat until convergence
Step 2c: Update the coefficients and source terms for the z momentum equation and solve the equation implicitly
Step 3: Calculate the coefficients and source terms for the pressure correction equation and solve the pressure correction equation implicity
Loop D: Repeat until convergence
Step 4: Correct the pressure, velocity and density
Step 5: Solve the energy equation and other transport equations and update the speed of sound
Step 6: Correct the density
Next time step

Figure 3: Solution procedure in Brilliant, changes made by Petrell early 2011.

Figure 4 shows a simulation example from the specialisation project, where the central outlet velocity of air in a 5 m long pipe with a diameter of 0.0193 m is plotted. The fluid starts at rest, but with a fixed flow rate at the inlet and a fixed pressure at the outlet. A sudden jump in mass flow results in an oscillating outlet velocity.



(a) Old solution procedure **(b)** Updated solution procedure

Figure 4: Outlet velocity in the central point of the pipe.

As shown in Fig. 4 the changes made to the solution procedure resulted in a more stable central velocity for the pipe flow example from the specialisation project. However, the result calculated by use of the updated solution procedure still contains some small instabilities which could cause problems for more complex simulations.

3.2.2 Implementation of the SIMPLE algorithm

As described in Sect. 2.2.3, the SIMPLE algorithm and the SIMPLEC algorithm follow the same solution procedure. However, the SIMPLE algorithm neglects the velocity correction contribution from neighbouring CVs, whereas the SIMPLEC algorithm approximates them. The solution procedures for SIMPLE and SIMPLEC can be illustrated by Fig. 3.

The modified solution procedure for the SIMPLEC algorithm in Brilliant was used as a foundation for the implementation of the SIMPLE algorithm. Only one minor modification is implemented in order to obtain the SIMPLE algorithm.

Changes implemented in the existing solution procedure:

1. As the coefficient d^{E} differs from the coefficient d^{C}, the contribution from neighbouring velocity corrections removed from the calculation of d.

The rest of the solution procedure for the SIMPLE algorithm coincides with the solution procedure for the existing SIMPLEC-like algorithm, and was kept unchanged.

3.2.3 Implementation of the SIMPLER algorithm

As described in Sect. 2.2.2, the SIMPLER algorithm does not follow the exact same solution procedure as SIMPLE and SIMPLEC. The SIMPLER algorithm connects the preliminary velocity and pressure by solving an equation for the pressure based on guessed preliminary velocities. In addition to the implementation of the coefficients for the pressure equation, the calculation of pseudo velocities is also implemented. However, as illustrated in Fig. 5, large parts of the solution procedures coincide.

For the implementation of the SIMPLER algorithm, the existing solution procedure for the SIMPLEC algorithm in Brilliant was used as a foundation. Steps R1 and R2 shown in Fig. 5 had to be implemented in front of the existing procedure. As for the SIMPLE algorithm, the correction contribution from neighbouring velocities is neglected in the pressure correction equation.

Changes implemented in the existing solution procedure:

1. The calculation of the pseudo velocities as described in Sect. 2.2.2, including special treatment for the control volumes adjacent to the boundaries, is implemented.

2. The calculation of coefficients for the pressure equation, including special treatment for the control volumes adjacent to the boundaries, is implemented.

3. As for the SIMLPE algorithm, the coefficient d^{E} differs from d^{C}, and the contribution from neighbouring velocity corrections is removed from the calculation of d.

4. The pressure correction corrects only the velocity and the density, and the correction of the pressure is removed.

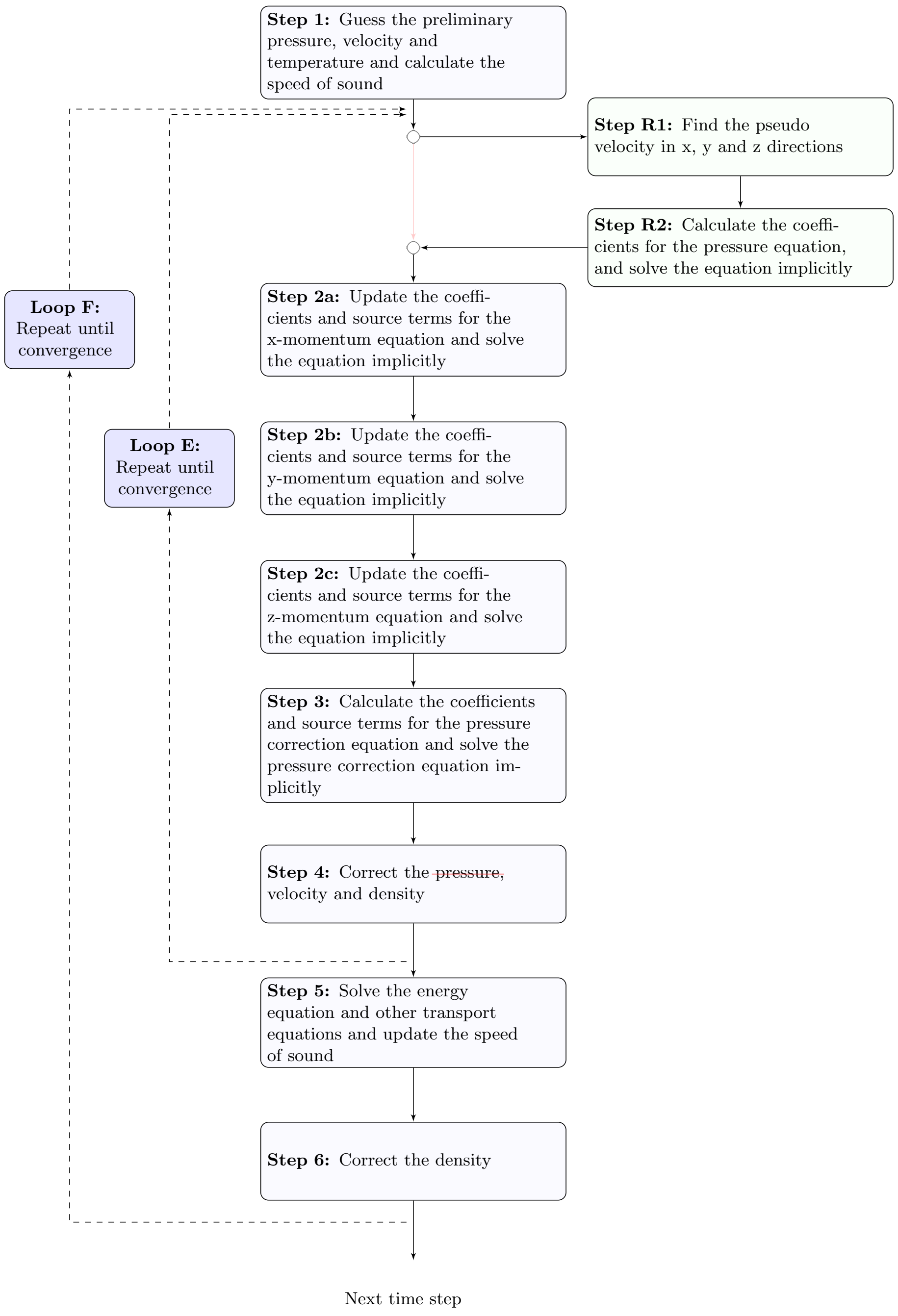

Step 1: Guess the preliminary pressure, velocity and temperature and calculate the speed of sound
Step R1: Find the pseudo velocity in x, y and z directions
Step R2: Calculate the coefficients for the pressure equation, and solve the equation implicitly
Loop F: Repeat until convergence
Step 2a: Update the coefficients and source terms for the x-momentum equation and solve the equation implicitly
Loop E: Repeat until convergence
Step 2b: Update the coefficients and source terms for the y-momentum equation and solve the equation implicitly
Step 2c: Update the coefficients and source terms for the z-momentum equation and solve the equation implicitly
Step 3: Calculate the coefficients and source terms for the pressure correction equation and solve the pressure correction equation implicitly
Step 4: Correct the ~~pressure,~~ velocity and density
Step 5: Solve the energy equation and other transport equations and update the speed of sound
Step 6: Correct the density
Next time step

Figure 5: Implementation of SIMPLER.

The rest of the solution procedure for the SIMPLER algorithm coincides with the solution procedure for the existing SIMPLEC-like algorithm, and was kept unchanged.

In order to enclose the domain, the calculations of both the pseudo velocities and the coefficients for the pressure equation need special treatment for the control volumes adjacent to the boundaries. The treatment of these control volumes depends on the kind of boundary condition given. Implementation of the boundary conditions will be described in detail in Sect. 3.2.5.

Some of the implemented functionalities are used by both SIMPLER and IDEAL. Therefore, some of the functionalities used by the SIMPLER algorithm were changed during the implementation of the IDEAL algorithm. The pseudo velocity used by the IDEAL algorithm is weighted between the pseudo velocity described for the SIMPLER algorithm, Sect. 2.2.2, and the preliminary velocity. In order to find the pseudo velocities on the surfaces of each control volume, Rhie-Chow interpolation was first used for the SIMPLER algorithm. However, as described in Sect. 2.2.4, the pseudo velocities on the surfaces are found by linear interpolation for the IDEAL algorithm. The functionalities used by the IDEAL algorithm were tested and used for the SIMPLER algorithm as well.

3.2.4 Implementation of the IDEAL algorithm

As illustrated in Fig. 6 and described in Sect. 2.2.4, the solution procedure for the IDEAL algorithm contains large differences from the existing SIMPLEC-like algorithm in Brilliant. The IDEAL algorithm connects the preliminary pressure and velocity through an inner iteration process, where the pseudo velocities, implicit pressure equation and an explicit momentum equation are solved repeatedly. The pressure correction equation is not solved at all, and the density is therefore updated due to changes in the calculated pressure field.

Because of the large differences between the solution procedures for SIMPLEC and IDEAL, the existing procedure was not used as a foundation, and a completely new solution procedure was implemented.

Implementation of the IDEAL algorithm:

1. As illustrated in Fig. 6, a completely new solution procedure is implemented.
2. As for SIMLPE and SIMPLER, the coefficient d^{E} differs from the coefficient d^{C}, and the contribution from neighbouring velocity corrections is

removed from the calculation of d.

3. The calculation of the pseudo velocity used by the IDEAL algorithm is slightly different from the pseudo velocity implemented during the work on the SIMPLER algorithm. The pseudo velocity used by the IDEAL algorithm is weighted between the pseudo velocity implemented for the SIMPLER algorithm and the preliminary velocity.

4. The calculation of the explicit momentum equation, including special treatment for the boundaries, is implemented.

5. Since the pressure correction equation is not solved, the density is updated due to the difference between the preliminary pressure field and the updated pressure field.

6. The existing test for convergence was partly based on the pressure correction, and a new test of convergence is needed.

Even though the solution procedure for the IDEAL algorithm contains large differences from the procedure for both SIMPLEC and SIMPLER, some of the existing functionality could be used by the IDEAL algorithm as well, for instance the implicit momentum equation, implicit pressure equation, scalar equations and the calculation of the pseudo velocities.

The treatment of the boundary conditions for the explicit momentum equation is described in detail in the next section.

During this master's thesis, it was made an attempt to implement the IDEAL algorithm for flows at all speeds and collocated grids. However, the solution procedure diverged even for simple test cases. The reason for the divergence is believed to be problems with the boundary conditions for the explicit momentum equation. The implemented boundary conditions, described in the next section, give a slightly overpredicted velocity at the outlet, hence emptying the mass in the domain.

3.2.5 Implementation of boundary conditions

In order to enclose the domain, boundary conditions are needed. For the equations implemented during the work on this master's thesis, two boundary conditions are implemented in Brilliant: (1) A given mass flow boundary and (2) a given pressure boundary. The boundary with the given mass flow is used for the inflow area of the domain and the boundary with the given pressure is used for the outflow area.

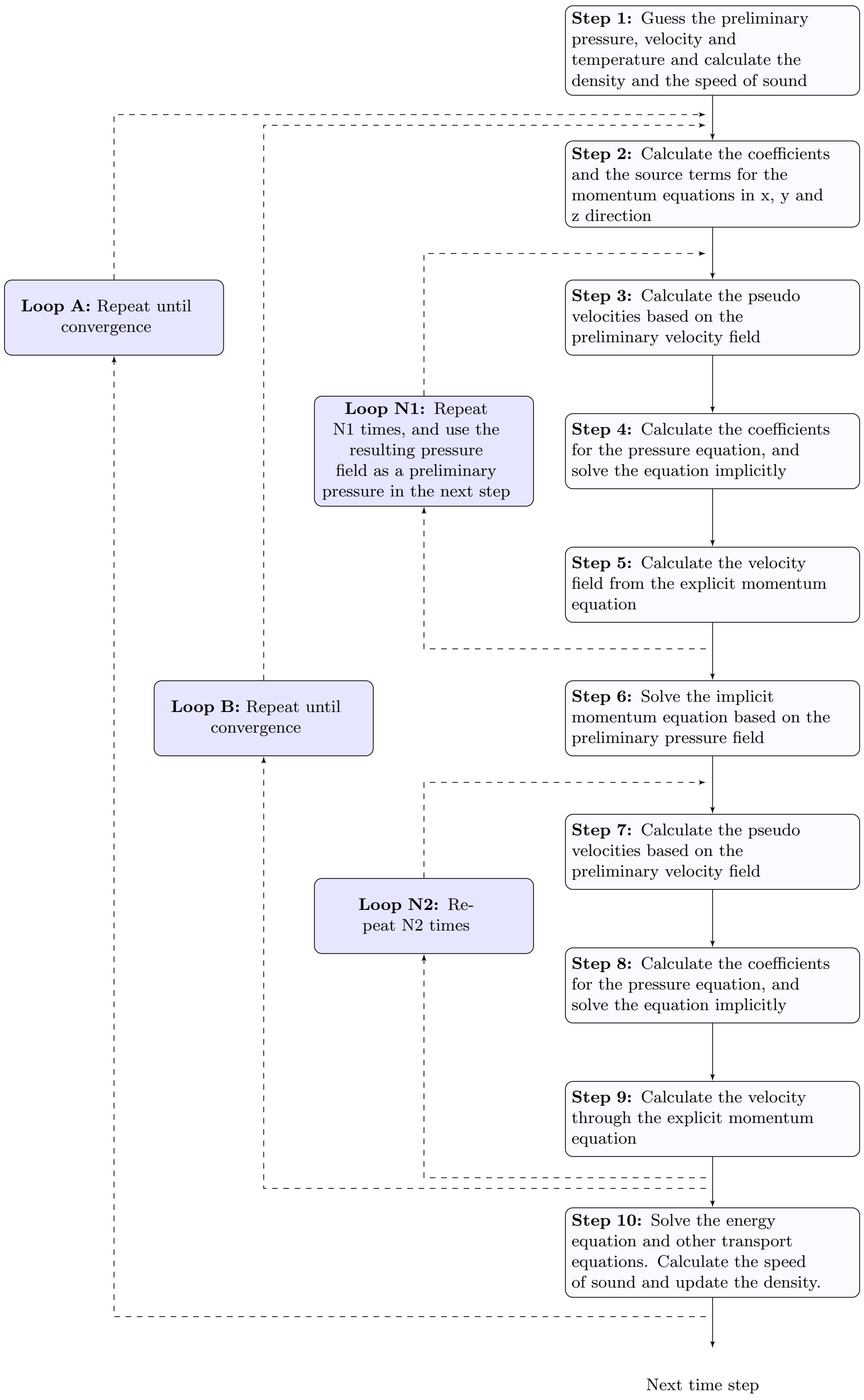

Step 1: Guess the preliminary pressure, velocity and temperature and calculate the density and the speed of sound
Step 2: Calculate the coefficients and the source terms for the momentum equations in x, y and z direction
Step 3: Calculate the pseudo velocities based on the preliminary velocity field
Step 4: Calculate the coefficients for the pressure equation, and solve the equation implicitly
Step 5: Calculate the velocity field from the explicit momentum equation
Step 6: Solve the implicit momentum equation based on the preliminary pressure field
Step 7: Calculate the pseudo velocities based on the preliminary velocity field
Step 8: Calculate the coefficients for the pressure equation, and solve the equation implicitly
Step 9: Calculate the velocity through the explicit momentum equation
Step 10: Solve the energy equation and other transport equations. Calculate the speed of sound and update the density.
Loop A: Repeat until convergence
Loop N1: Repeat N1 times, and use the resulting pressure field as a preliminary pressure in the next step
Loop B: Repeat until convergence
Loop N2: Repeat N2 times
Next time step

Figure 6: Implementation of IDEAL.

Both the implemented coefficients and pseudo velocities need to be treated differently for the control volumes adjacent to the boundaries than for the CVs in the rest of the domain. Some of the unknowns are given on the boundary and others are not specified. An adiabatic boundary condition, which means that the first order derivative equals zero, is already implemented in the matrix solver used by Brilliant. Since the variables do not change over the adiabatic boundary, the neighbouring coefficient is set to zero and it gives no contribution to the central coefficient.

Pseudo velocity

In order to calculate the pseudo velocities, the coefficients from the momentum equation and the neighbouring velocities must be known. As the existing implicit momentum equation in Brilliant is used, boundary conditions for the coefficients in the momentum equation are already implemented.

For the boundary with the given mass flow, the velocity can be calculated, and there is no need for special treatment for the control volumes adjacent to this boundary.

As the boundary at the outlet only contains information about the pressure, the velocity is unknown. The neighbouring coefficients for the control volumes represented by the boundaries are then set to zero, and the pseudo velocity depends only on the upstream velocities. It should be noticed that if back flow occurs at the outlet, these velocities would no longer represent the upstream velocities. So, if back flow occurs, the domain should be expanded until the back flow no longer appears at the outlet.

In order to calculate the coefficients for the pressure equation for the control volume adjacent to the outlet boundary, the pseudo velocity on the given pressure boundary is required. This pseudo velocity is calculated on the basis of the pseudo velocity in the control volume adjacent to the boundary and the change in pressure from the adjacent control volume to the known pressure boundary:

$$(\bar{u}_i)^*_{(\mathrm{P}+1/2)} = (\bar{u}_i)^*_{\mathrm{P}} + (p_{\mathrm{P}} - p_{\mathrm{B_p}}) d^{\mathrm{E}}_{\mathrm{P}} (nA)_{(\mathrm{P}+1/2)} \tag{42}$$

where the central coefficient on the boundary is unknown, and the upstream central coefficient is used instead.

Pressure equation

The continuity equation requires the velocity on each surface of the control volume. In order to obtain the pressure equation, the momentum equation with known pseudo velocities is, for each surface, inserted into the continuity equation. For the boundary with the given mass flow, the velocity can be calculated, and there is no need for inserting the momentum equation into the continuity equation for this surface:

$$\begin{gathered}\left(\frac{\rho V}{\Delta t}\right)_{\mathrm{P}} + \sum_{\substack{\mathrm{NB}\\ \mathrm{NB}\neq\mathrm{B_{MF}}}} \left(a_{\mathrm{NB}}(p^{\mathrm{n}}_{\mathrm{P}} - p^{\mathrm{n}}_{\mathrm{NB}})\right) + \sum_{\substack{\mathrm{nb}\\ \mathrm{nb}\neq\mathrm{B_{MF}}}} (\rho n \Delta A \bar{u}^{*})_{\mathrm{nb}} = \\ (S_{\mathrm{c}})_{\mathrm{P}} + \left(\frac{\rho V}{\Delta t}\right)^{\mathrm{t}-1}_{\mathrm{P}} - (u\rho n A)_{\mathrm{B_{MF}}}\end{gathered} \tag{43}$$

The summation in Eq. (43) includes all neighbouring surfaces, except for the boundary with the given mass flow. This neighbouring coefficient is set to zero, and gives no contribution to the central coefficient.

For the control volume adjacent to the given pressure boundary, the neighbouring pressure and coefficient are known and therefore included in the constant term:

$$a_{\mathrm{P}} p_{\mathrm{P}} - \sum_{\substack{\mathrm{NB}\\ \mathrm{NB}\neq\mathrm{B_p}}} a_{\mathrm{NB}} p_{\mathrm{NB}} = b - a_{\mathrm{B_p}} p_{\mathrm{B_p}} \tag{44}$$

where the summation includes all neighbouring control volumes, except for the boundary with the known pressure.

Explicit momentum equation

In order to calculate the velocity in the control volumes adjacent to the given mass flow boundary from the explicit momentum equation, the pressure on the boundary must be known. As the pressure is known for all the control volumes inside the domain, the pressure on the given mass flow boundary is found by extrapolation.

For the control volume adjacent to the pressure boundary, the boundary condition is already included in the calculation of the pseudo velocity, and no special treatment is needed.

In order to examine whether or not the mass is conserved for a given iteration level, the convective term on all surfaces of each control volume in the domain are summarised. For the boundary with the given mass flow, the convective term is given. However, for the given pressure boundary, the velocity is not known. The velocity on this boundary is calculated by the central velocity in the control volume adjacent to the given pressure boundary and the change in pressure from this control volume to the given pressure boundary:

$$(u_i)_{(\mathrm{P}+1/2)} = (u_i)_{\mathrm{P}} + (p_{\mathrm{P}} - p_{\mathrm{B_p}})d^{\mathrm{E}}_{\mathrm{P}}(nA)_{(\mathrm{P}+1/2)} \tag{45}$$

where the central coefficient on the boundary is unknown. Therefore, the central coefficient for the control volume adjacent to the given pressure boundary is used.

4 Results

Three examples are simulated by use of the implemented versions of SIMPLE and SIMPLER: (1) Methane pipe flow, (2) a shock tube and (3) a pressure relief pipe. The variables calculated by use of SIMPLE and SIMPLER are compared to the variables calculated by use of the existing SIMPLEC algorithm. For the shock tube problem, a quasi-analytical solution can be obtained, and the variables calculated by all three algorithms are compared to this solution.

As discussed in Sect. 3.2.3, some of the functionalities used by the SIMPLER algorithm were adjusted during the implementation of the IDEAL algorithm. The modified algorithm is used for the simulation examples in Sects. 4.1, 4.2 and 4.3. In Sect. 4.4, the two variants of SIMPLE-like algorithms are compared.

When finding the minimum CPU time needed by an algorithm for simulating a given case, the underrelaxation factors must be adjusted. The optimal value of the underrelaxation factors may vary from algorithm to algorithm, and their values must be adjusted individually in order to ensure the optimal performance for all algorithms [19]. The robustness of an algorithm can be measured by the algorithm's ability to converge for a wide range of underrelaxation factors [20].

Both the robustness and the minimum CPU time needed are investigated for the first two simulation examples. When the robustness is investigated, the different underrelaxation factors are given the same value, $\alpha_{\mathrm{u,v,w,e,}\ \rho,\ \mathrm{p',\ p}} = \alpha$. Due to the large amount of CPU time needed for simulating the pressure relief pipe, the minimum CPU time needed and robustness are not investigated for this simulation example.

4.1 Simulation example 1: Methane pipe flow

A 150 m long pipe with a diameter of 0.5 m is filled with methane. The methane starts at rest, and a sudden jump in mass flow is applied to the inlet after 0.5 sec. After the sudden jump, the mass flow fixed at 0.1 kg/s throughout the rest of the simulation. The temperature at the inlet is 281 K, and the pressure at the outlet is 100000 Pa. Figure 7 shows the grid and the geometry in GL View Inova.

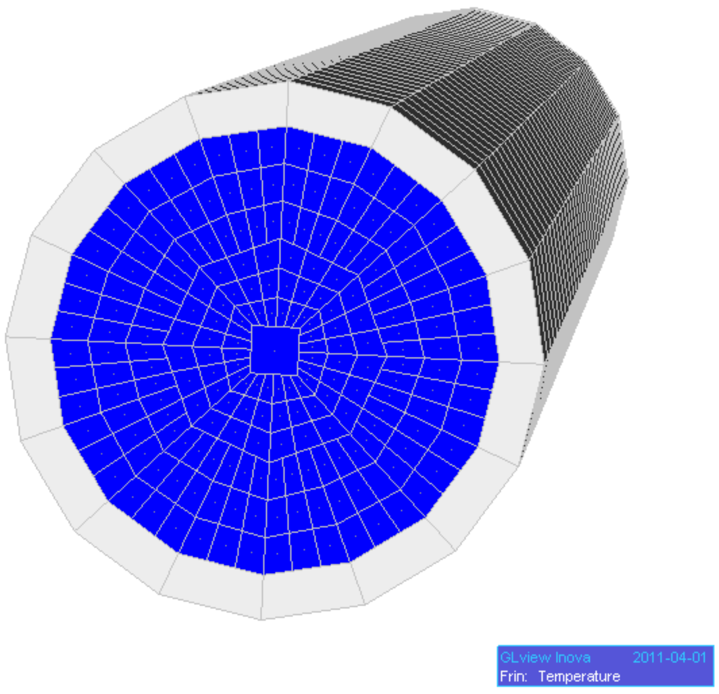
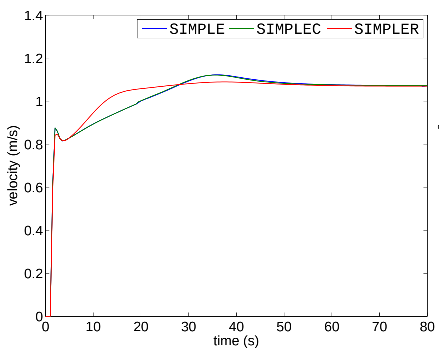
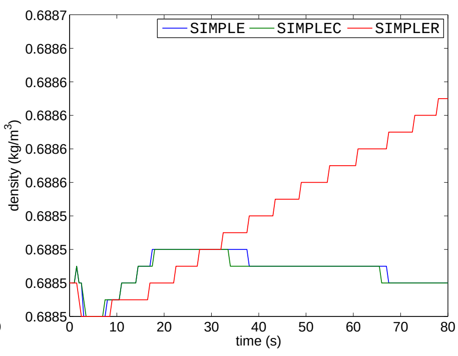


Figure 7: Grid and geometry of the pipe.

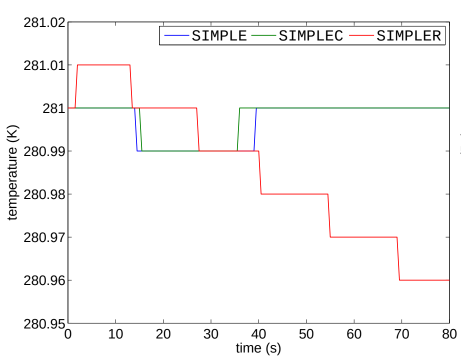
In Fig. 8 the density, temperature, pressure and velocity calculated by use of the extended versions of SIMPLE, SIMPLER and SIMPLEC are plotted in the central point of the pipe's outlet. All the four variables calculated by using SIMPLE and SIMPLEC are distributed almost identically. The steady velocity calculated by use of the SIMPLER algorithm coincides with the steady velocity calculated by SIMPLE and SIMPLEC. Some time after the sudden jump in mass flow, however, the velocity calculated by use of the SIMPLER algorithm is slightly greater than the velocity calculated by the two other algorithms. Even though the velocity given by the SIMPLER algorithm stations some time after the sudden jump in mass flow, the density keeps increasing gradually, while the temperature decreases gradually. This behaviour is not seen in the temperature or density given by SIMPLEC and SIMPLE. However, the changes in both the temperature and density calculated by the SIMPLER algorithm are very small.



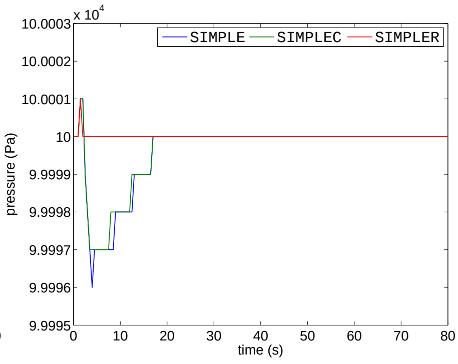
(a) Velocity



(b) Density



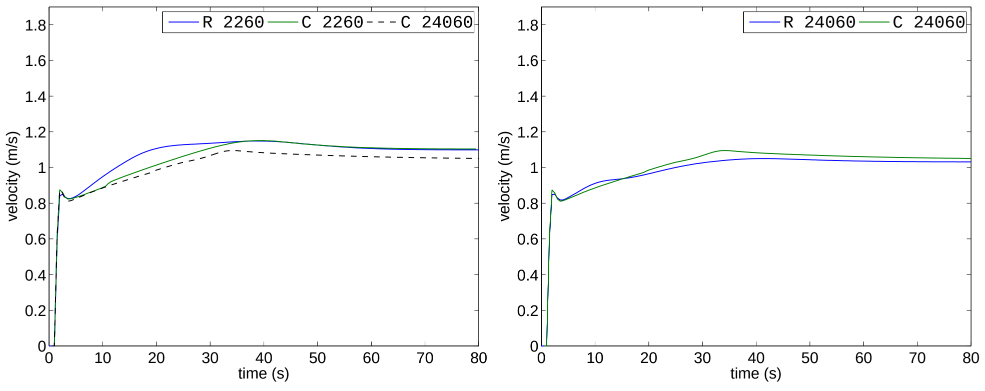
(c) Temperature



(d) Pressure

Figure 8: Properties in the midpoint of the outlet.

The methane pipe flow simulation was conducted on a relatively fine grid, resulting in 11490 solved control volumes. SIMPLEC and SIMPLER were also tested on both finer and coarser grid, with 2260 and 24060 solved CVs respectively. The velocity in the central point of the outlet is plotted in Fig. 9, where R gives the result calculated by the SIMPLER algorithm and C gives the result calculated by the SIMPLEC algorithm.



(a) Velocity on coarse grid **(b)** Velocity on fine grid

Figure 9: SIMPLER and SIMPLEC with finer and coarser grid.

For the coarse grid in Fig. 9, the velocities calculated by SIMPLER and SIMPLEC are compared to the velocity calculated by the SIMPLEC algorithm on the fine grid. Both SIMPLER and SIMPLEC give inaccurate results when simulating the methane pipe flow on the coarsest grid. For both algorithms, the calculated steady velocity is slightly greater than the steady velocity calculated by the SIMPLEC algorithm on the fine grid. For the SIMPLER algorithm, the largest deviation appears some time after the sudden jump in mass flow. Here, the velocity calculated by the SIMPLER algorithm is a bit greater than the velocities calculated by the SIMPLEC algorithm on both the coarse and the fine grid.

When simulating the methane pipe flow on the finest grid, the steady velocity calculated by use of the SIMPLER algorithm is slightly lower than the steady velocity calculated by the SIMPLEC algorithm. However, in total, the differences between the velocities calculated by the two algorithms are relatively small for the fine grid.

Table 1: Minimum CPU time for simulation example 1.

	CPU time	α_u	$\alpha_{p'}$	α_ρ	α_e	α_p
SIMPLE	1050.41	0.9	0.9	0.8	1	-
SIMPLER	7873.97	0.9	0.9	0.5	0.5	0.9
SIMPLEC	1095.66	0.9	0.9	0.3	0.3	-

Table 1 gives the minimum CPU time when simulating the methane pipe flow on the medium-sized grid by use of SIMPLE, SIMPLEC and SIMPLER. In order to find the minimum CPU time needed by the different algorithms, the underrelaxation factors are adjusted. The SIMPLE algorithm uses the least CPU time, closely followed by the SIMPLEC algorithm. However, the CPU time needed by the SIMPLER algorithm for this simulation example is very large compared to the CPU time spent when simulating by SIMPLE and SIMPLEC.

Table 2 gives the range of convergence for all three algorithms with various underrelaxation factors. Both SIMPLEC and SIMPLER converge for all tested values of α. Even though the SIMPLE algorithm used the least CPU time with optimal underrelaxation factors, it is the only algorithm not converging for all values of α. The SIMPLE algorithm converges for values of α in the range $0.9 - 0.1$.

Table 2: Robustness example 1.

	Convergence?	α
SIMPLE		
	No	1.0
	Yes	0.9-0.1
SIMPLER		
	Yes	1.0-0.1
SIMPLEC		
	Yes	1.0-0.1

4.2 Simulation example 2: Shock tube

In this example, nitrogen starts at rest in a 1 m long tube with height and width of 0.1 m. The tube consists of fluid only, and has no walls. The initial condition is divided in two at the length 0.5 m; a high pressure side and a low pressure side. The high pressure side is initiated at 180000 Pa and 2.145 kg/m^3, while the low pressure side is initiated at 100000 Pa and 1.191 kg/m^3. The initial temperature is 283 K. Figure 10 shows the grid and the geometry in GL View Inova.

This simulation example with slightly different initial conditions is known as Sod's shock tube problem [21]. Assuming a polytropic equation of state and isentropic flow everywhere, except across the shock, a quasi-analytical solution of the shock tube can be obtained [22, 23]. Until the shock and the expansion fan have reached the two ends of the tube, the solution can be divided into five different regions. The regions are: An undisturbed high pressure side, a rarefaction wave, two constant-regions separated by a contact discontinuity and an undisturbed low pressure side separated from the second constant region by a shock. The velocity, density and pressure calculated when simulating the shock tube by use of SIMPLE, SIMPLER and SIMPLEC are compared to the values calculated by the quasi-analytical solution. The equations used for calculating the quasi-analytical values of the velocity, pressure and density are presented in App. A.

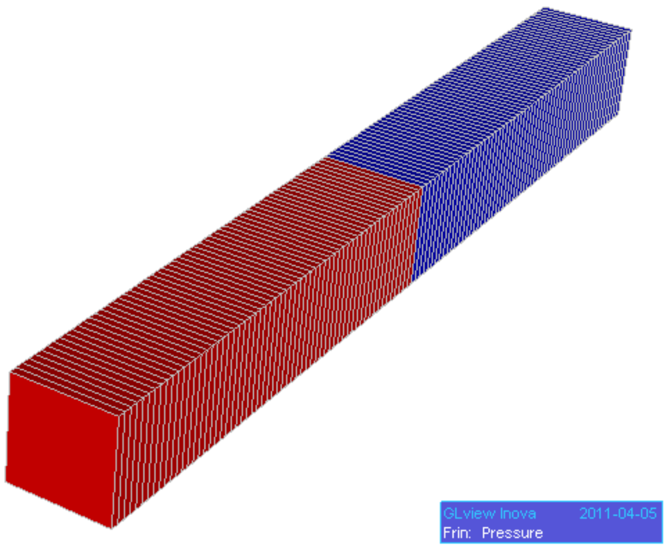
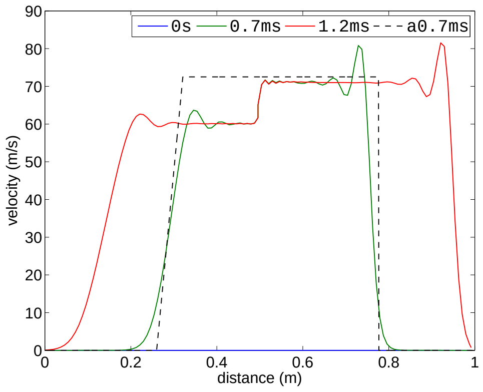
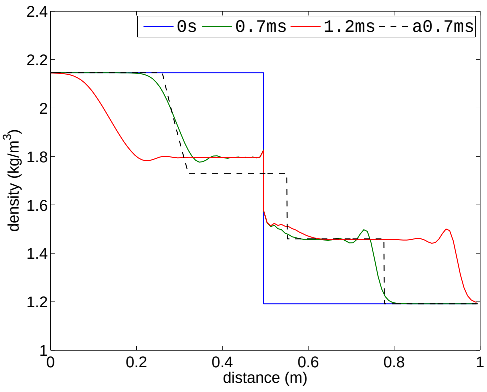


Figure 10: Grid and geometry of the shock tube.

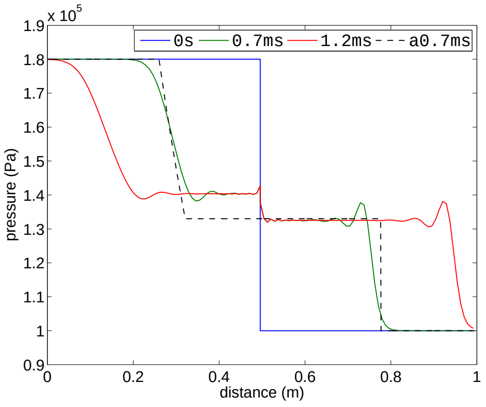
Figures 11, 12 and 13 give the velocity, pressure and density distributions over the length of the shock tube calculated by use the extended versions of respectively SIMPLE, SIMPLEC and SIMPLER. The distribution of the different variables for the time 0 ms, 0.7 ms and 1.2 ms after the initial state are plotted in the figures. In addition, the analytical solutions 0.7 ms after the initial state are plotted in each figure as well.



(a) Velocity



(b) Density

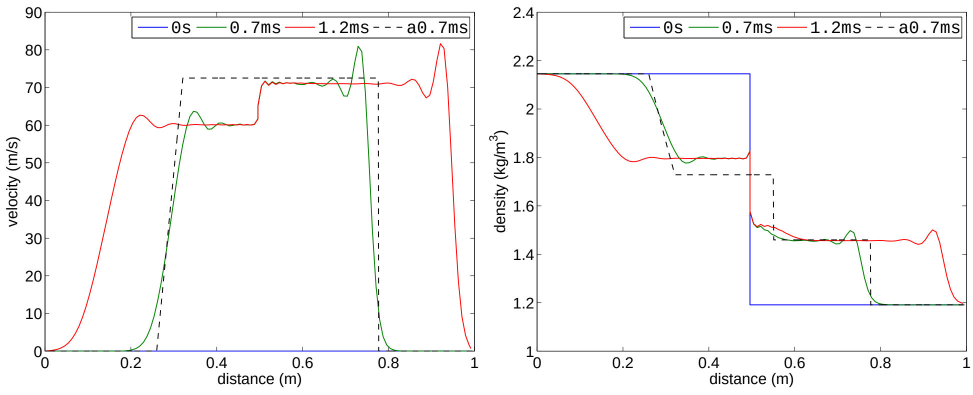


(c) Pressure

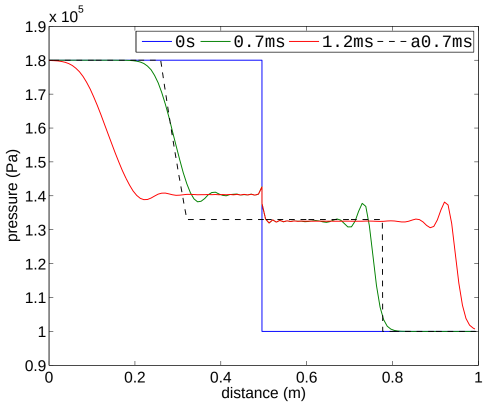
Figure 11: Shock tube calculations using the SIMPLE algorithm, compared to the analytical solution after 0.7 ms (a0.7ms).

Between the rarefaction wave and the shock, the velocity calculated by the quasi-analytical solution is constant. The velocity calculated by use of the SIMPLE algorithm, however, contains a sudden jump in the midpoint of the shock tube, where the high and the low pressure side initially were separated. The value of the velocity on the high density side of the constant regions is very low compared to the value of the quasi-analytical velocity, and the value of the velocity on the low density side of constant regions is only slightly lower than the quasi-analytical velocity. Both the pressure and density calculated by the SIMPLE algorithm contain a sudden jump in the midpoint of the tube as well, and the density contains no sudden change where the quasi-analytical solution gives the contact discontinuity. For all three calculated variables, the SIMPLE algorithm gives more accurate values on the low density side of the constant regions than on the high density side. Compared to the quasi-analytical solution, the location of both the shock and rarefaction wave calculated by use of the SIMPLE algorithm is quite good. However, all three variables calculated by the SIMPLE algorithm contain oscillations in vicinity to both the shock and the rarefaction wave.

As can be seen from Figs. 11 and 12, the values of all three variables calculated using the SIMPLEC algorithm are almost identical to the values calculated by use of the SIMPLE algorithm, thus giving the same remarks as those discussed in previous paragraph.



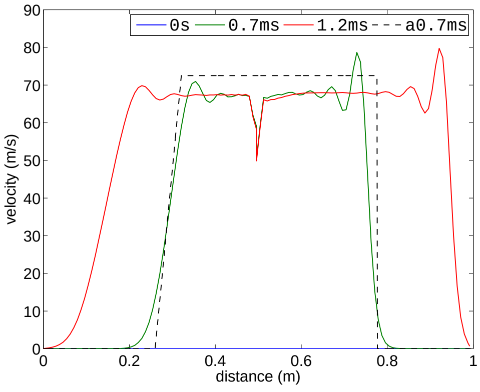
(a) Velocity **(b)** Density



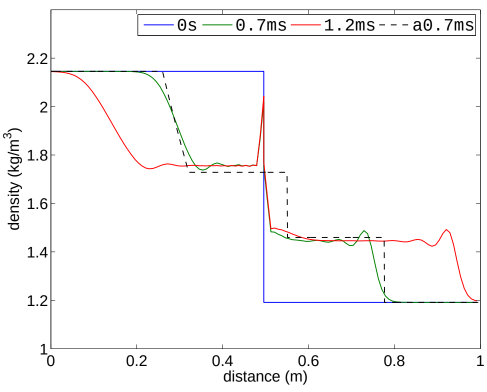
(c) Pressure

Figure 12: Shock tube calculations using the SIMPLEC algorithm, compared to the analytical solution after 0.7 ms (a0.7ms).

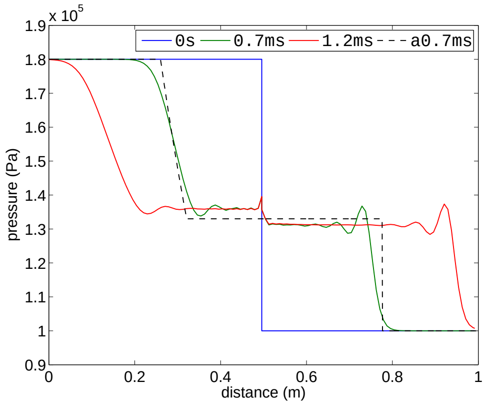
In the two constant regions of the shock tube, the value of the velocity calculated by the SIMPLER algorithm is low compared to the velocity given by the quasi-analytical solution. In the central point of the tube, the velocity given by the SIMPLER algorithm contains a large undershoot, while the density contains a large overshoot. As for SIMPLE and SIMPLEC, there is no sudden change in the density where the quasi-analytical solution places the contact discontinuity, but the location of both the shock and the rarefaction wave is quite good. All three variables calculated by the SIMPLER algorithm contain oscillations in vicinity of both the shock and the rarefaction wave. Except for the oscillations and a small jump in the central point of the tube, the pressure calculated from the SIMPLER algorithm contains only small deviations from the quasi-analytical solution.



(a) velocity



(b) density



(c) pressure

Figure 13: Shock tube calculations using the SIMPLER algorithm, compared to the analytical solution after 0.7 ms (a0.7ms).

Table 3: Minimum CPU time for simulation example 2.

	CPU time	α_u	$\alpha_{p'}$	α_ρ	α_e	α_p
SIMPLE	150.141	1	1	0.9	0.8	-
SIMPLER	133.544	1	1	0.4	1	1
SIMPLEC	150.362	0.9	1	1	1	-

Table 3 gives the minimum CPU time spent when simulating the shock tube by use of SIMPLE, SIMPLER and SIMPLEC. In order to find the minimum CPU time needed by the different algorithms, the underrelaxation factors were adjusted. SIMPLE and SIMPLEC spend approximately the same amount of CPU time for the simulation of the shock tube. The SIMPLER algorithm spends less CPU time than the others.

In Tab. 4, the range of convergence for the shock tube problem by use of the three different algorithms is given. For the simulation of the shock tube example, all algorithms converged for all tested values of the underrelaxation factor, α.

Table 4: Robustness example 2.

	Convergence?	α
SIMPLE		
	Yes	1.0-0.1
SIMPLER		
	Yes	1.0-0.1
SIMPLEC		
	Yes	1.0-0.1

4.3 Simulation example 3: Pressure relief pipe

In this example, methane flows from a tank and into a 19.9 m long pressure relief pipe, with an inner diameter of 0.36354 m. The initial pressure and temperature in the tank is 15395800 Pa and 264 K, and the initial mass flow into the tank is 80.503 kg/s. Both the mass flow from the tank and the pressure decrease with time. The temperature decreases to a certain minimum before it slowly increases throughout the simulation. The grid and geometry of the pipe can be seen in Fig. 14.

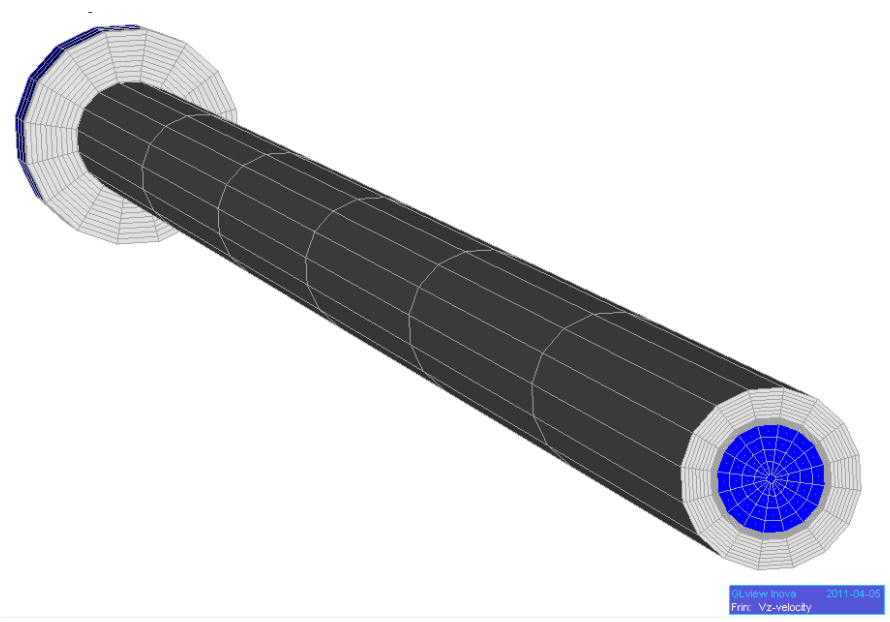
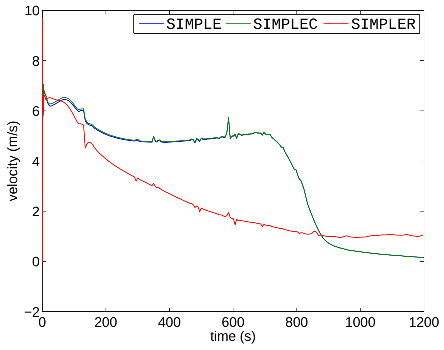


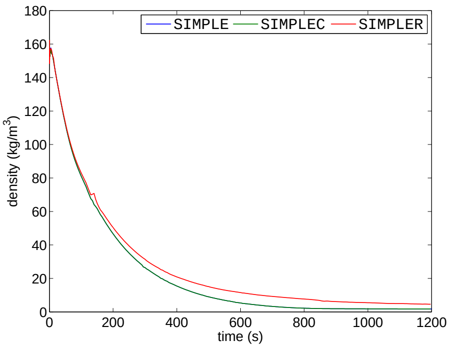
Figure 14: Grid for the pressure relief pipe.

In Fig. 15 the velocity, density, temperature and pressure in the central point of the outlet calculated by all three algorithms are plotted. As for the two previous simulation examples, SIMPLE and SIMPLEC give almost identical results for all the four calculated variables. The pressure calculated using the SIMPLER algorithm give only minor deviations from the pressure calculated by the two other algorithms. As the pressure is given on the outlet boundary, this result was expected. However, the velocity, temperature and density given by the SIMPLER algorithm deviates highly from the values calculated by use of SIMPLE and SIMPLEC. Due to the large differences in the calculated variables, it was further examined and verified that the mass was conserved for all algorithms. The velocity and temperature calculated by all algorithms contain oscillations and sudden changes. As the calculated temperature and velocity contain large instabilities, errors might be brought into the next time step, hence a possible source of errors for all three algorithms. For all algorithms, and especially for the SIMPLER algorithm, the need of underrelaxation was great, indicating poor stability for the simulation of the pressure relief pipe. This might cause the large differences in the calculated velocities and temperatures.

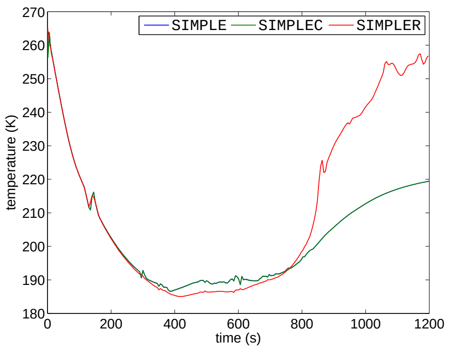
In order to find the minimum CPU time for the different algorithms, the underrelaxation factors have to be optimised. The CPU time needed for simulation example 3 is much larger than for the two other simulation examples, and the underrelaxation factors were only adjusted until convergence was reached. Even though the underrelaxation factors were not optimised, it is noticed that the CPU time spent by the SIMPLER algorithm is about two to three times greater than the CPU time needed for this simulation example using SIMPLE and SIMPLER.



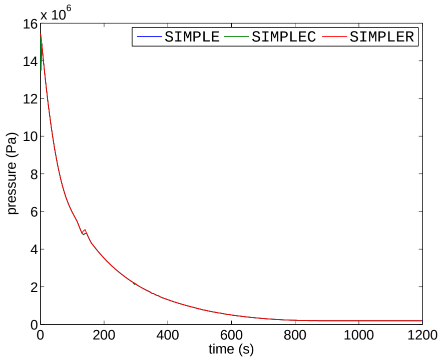
(a) Velocity



(b) Density



(c) Temperature



(d) Pressure

Figure 15: Properties in the centre of the outlet of the pressure relief pipe.

Due to the CPU time needed for this simulation example, the ability to converge for different underrelaxation factors was not recorded either. However, more adjustments are needed in order to reach convergence for the SIMPLER algorithm than for both SIMPLE and SIMPLEC.

4.4 Importance of small adjustments in SIMPLER

First, the SIMPLER algorithm was implemented with a pseudo velocity as described in Sect. 2.2.2. The pseudo velocities were interpolated to the surfaces of the control volumes by Rhie-Chow interpolation and the velocities on the pressure boundary were set equal to the upstream velocity. As earlier discussed, some of the functionalities changed during the work on the IDEAL algorithm were used for the SIMPLER algorithm as well:

1. The pseudo velocity is now weighted between the pseudo velocity described in Sect. 2.2.2 and the preliminary velocity.
2. The pseudo velocities are now linearly interpolated to the surfaces of the control volumes.
3. The pseudo velocity on the known pressure boundary is now found as described in Sect. 3.2.5.

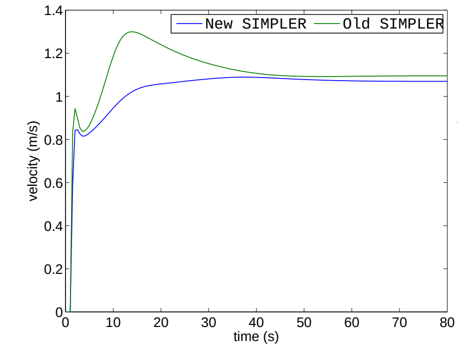
In order to compare the two different variants of the SIMPLER algorithm, simulations of the methane pipe flow and the shock tube are conducted by both algorithms.

4.4.1 Methane pipe flow

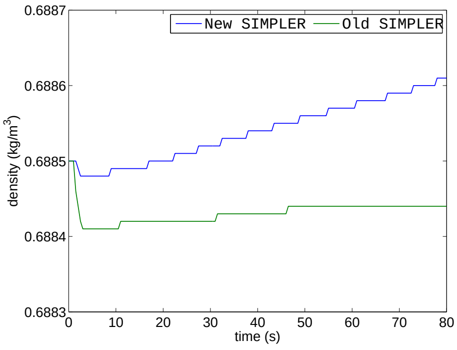
In Fig. 16, the velocity, density, temperature and pressure calculated by the old and the modified SIMPLER algorithm are plotted. The velocity calculated from the old SIMPLER algorithm contains a large overshoot some time after the sudden jump in mass flow, and the steady velocity is greater than the velocity calculated by the modified SIMPLER algorithm. The density calculated from the modified SIMPLER algorithm is increasing, while the temperature decreases. The changes, however, are small.

4.4.2 Shock tube

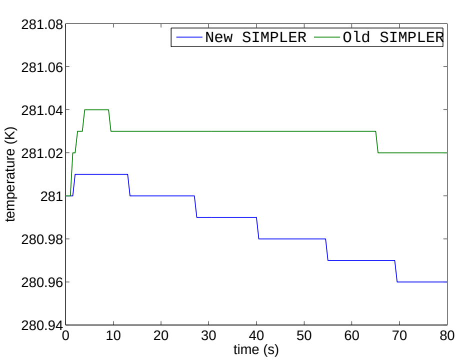
In Fig. 17, the velocity, density and pressure calculated by the old and the modified SIMPLER algorithm are plotted. There are only small differences in the pressure and density calculated by use of the two algorithms. For the velocity, however, greater differences can be seen. The velocity calculated from both algorithms contain a large undershoot where the high and low pressure side initially were separated, and both algorithms give oscillations in vicinity to the shock and the rarefaction wave. Except for the oscillations and undershoots, the velocity calculated from the modified SIMPLER algorithm is more or less constant between the rarefaction wave and the shock. However, the calculated



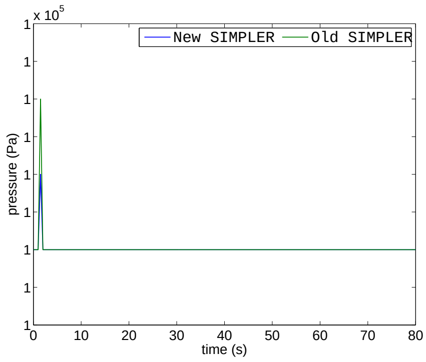
(a) velocity



(b) density



(c) temperature



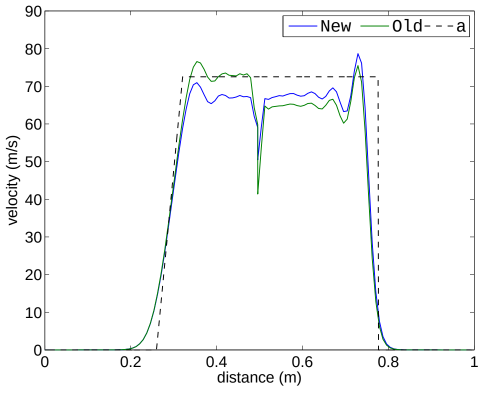
(d) pressure

Figure 16: Properties in the midpoint of the outlet.

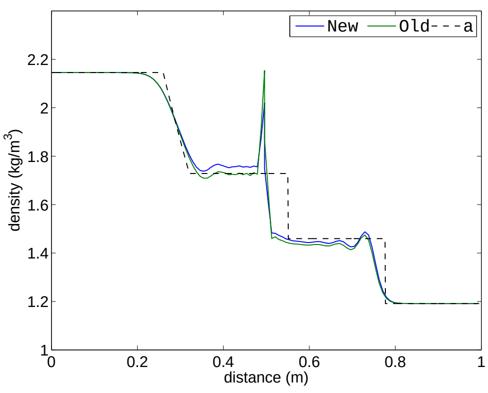
value is lower than the value calculated from the analytical solution. In this region, the velocity calculated from the old SIMPLER algorithm is greater on the initial high pressure side than on the initial low pressure side. Both algorithms give a large overshoot in the calculated density in the midpoint of the shock tube. The instabilities in both the velocity and density are smaller for the modified SIMPLER algorithm than for the old one.

4.5 Discussion of the simulation examples

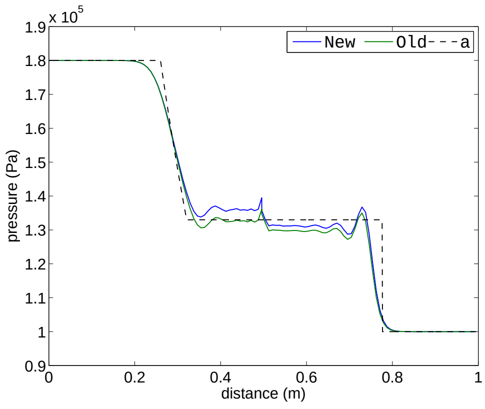
For all three simulation examples, the values of the variables calculated by use of the extended versions of SIMPLE and SIMPLEC are almost identical. As discussed in Sect. 2.2.5, SIMPLE and SIMPLEC follow the same solution procedure, and none of the assumptions made in the two algorithms should affect



(a) Velocity



(b) Density



(c) Pressure

Figure 17: Shock tube calculations using the modified and the old SIMPLER algorithm.

the values of the calculated variables if the procedure converges. As both SIMPLE and SIMPLEC converged for all three simulation examples, the results were expected to coincide. The SIMPLER algorithm, however, does not follow the exact same solution procedure as SIMPLE and SIMPLEC, and the results given by the extended SIMPLER algorithm differ from the results given by the two other algorithms.

For the methane pipe flow example, the variables calculated using SIMPLER and SIMPLEC on the coarse grid are compared to the variables calculated by use of the SIMPLEC algorithm on the fine grid. The velocities calculated by both SIMPER and SIMPLEC on the coarse grid are only a bit greater than the velocity calculated on the fine grid. On the fine grid, the velocity calculated using the SIMPLER algorithm only contains small deviations from the velocity calculated by use of the SIMPLEC algorithm.

For the shock tube, the variables calculated using SIMPLE, SIMPLER and SIMPLEC are compared to a quasi-analytical solution. The variables calculated by use of all three algorithms deviate a lot from the quasi-analytical solution in some regions of the shock tube. None of the algorithms manage to deal with the contact discontinuity, and all variables calculated by use of the three algorithms contain sudden jumps where the high and low pressure side initially were separated. All variables oscillate in vicinity of the shock and rarefaction wave as well. The velocity and density given by the SIMPLER algorithm contain large overshoots and undershoots in the midpoint of the tube. However, the most accurate calculated variable is the pressure calculated by use of the SIMPLER algorithm. The reason for this is unknown, but it should be noticed that this is the only variable not corrected by the pressure correction.

In order to reach convergence, the need for underrelaxation is greater for the SIMPLER algorithm than for SIMPLE and SIMPLEC when simulating the pressure relief pipe. The velocity and temperature calculated by use of the SIMPLER algorithm deviate greatly from the velocity and temperature calculated using SIMPLE and SIMPLER. All variables given by the three algorithms contain oscillations and sudden changes. As no experimental data is known for this simulation example, it is difficult to discuss the accuracy of the different algorithms.

When finding the minimal CPU time needed by the different algorithms, the underrelaxation factors were adjusted in order to find the optimal performance of each algorithm. For the simulation of the methane pipe flow, the CPU time needed by the SIMPLER algorithm is very large compared to the CPU time needed by the two other algorithms. For the shock tube the CPU time needed by the SIMPLER algorithm is slightly lower than the CPU time needed by SIMPLE and SIMPLEC. Because of the limited time for this master's thesis, and the time required to simulate the pressure relief pipe, the minimum CPU time for simulation example 3 is not found. Still, when the underrelaxation factors were adjusted until convergence was reached, it was noticed that the CPU time used by the SIMPLER algorithm was large compared to the CPU time used by both SIMPLE and SIMPLEC.

Many comparisons between the SIMPLE-like algorithms have been conducted in the past. In the specialisation project [3], some of these comparisons were discussed. The different comparisons favour different algorithms, making it difficult to suggest how the algorithms are expected to perform compared to one another. Here, the amount of CPU time needed by the SIMPLER algorithm was great compared to the CPU time needed by SIMPLE and SIMPLEC for two out of three simulation examples. The extended versions of SIMPLE and SIMPLER were implemented during the work on this master's thesis. Only minor changes were needed in order to change the solution procedure from the existing SIMPLEC-like algorithm to the extended SIMPLE algorithm. The SIMPLER algorithm solves an equation for the pressure, and larger changes were needed in order to implement the extended version of this algorithm. For such iterative processes, the performance can greatly depend on where and how often certain quantities are updated. As shown in Sect. 4.4, only minor changes can greatly improve the performance of the algorithms.

5 Conclusion and further work

SIMPLE and SIMPLEC follow the same solution procedure, but they differ in the calculation of the coefficients in the pressure correction equation. SIMPLE neglects the velocity correction contribution from neighbouring control volumes, whereas SIMPLEC approximates them. As correction terms tend towards zero when the solution procedure converges, this difference should not affect the values of the calculated variables for a converged procedure. As expected, all the variables calculated by use of the extended versions of SIMPLE and SIMPLEC are almost identical for the three presented simulation examples; methane pipe flow, a shock tube and a pressure relief pipe. The solution procedure for the SIMPLER algorithm differs from the procedure for SIMPLE and SIMPLEC, as SIMPLER solves an equation for the pressure based on the preliminary velocity. As the pressure is solved from the pressure equation, the pressure is not corrected through the pressure correction equation. The values of the variables calculated by use of the extended SIMPLER algorithm differ from those calculated by use of the two other algorithms.

When simulating the shock tube, the SIMPLER algorithm uses less CPU time than SIMPLE and SIMPLEC. For the two other simulation examples, however, the SIMPLER algorithm needs a great amount of CPU time compared to SIMPLE and SIMPLEC. For the methane pipe flow and for the shock tube, almost no underrelaxation was needed in order to reach convergence. For the pressure relief pipe, however, the need of underrelaxation was great for all algorithms, especially for the SIMPLER algorithm.

All three algorithms were compared to a quasi-analytical solution for the simulation of the shock tube. The SIMPLER algorithm gives large overshoots and undershoots in the midpoint of the shock tube, where the high and low pressure side initially were separated. The results calculated by use of SIMPLE and SIMPLEC also show large deviations from the quasi-analytical solution in some parts of the shock tube. All variables calculated by the three algorithms give oscillations in vicinity to the rarefaction wave and the shock, and none of the algorithms manage to deal with the contact discontinuity. It is noticed that the most accurate calculated variable is the only variable not corrected by the pressure correction, the pressure calculated by use of the SIMPLER algorithm.

An attempt of implementing the extended IDEAL algorithm was made. However, the solution procedure diverged even for simple test cases. This is probably due to problems with the boundary conditions for the explicit momentum equation. Therefore, further work on this topic should investigate the boundary conditions. It should be tested where and how often the density and coefficients should be updated in order to lower the CPU time needed for the IDEAL algorithm. For the SIMPLER algorithm only minor changes affected the performance greatly, and it should be investigated if other adjustments could further improve the performance of this algorithm. Both IDEAL and SIMPLER should be extended to also treat multiphase flow.

References

[1] H.K. Versteeg and W. Malalasekera. *An introduction to computational fluid dynamics.* Pearson Education Limited, 2007.

[2] J.H. Ferziger and M. Perić. *Computational methods for fluid dynamics.* Springer, 1999.

[3] S.M.R. Skrataas. Gjennomgang av numeriske algoritmer for kompressible strø mninger. Technical report, NTNU, 2010.

[4] F.H. Harlow and J.E. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of Fluids*, 8(12):2182–2189, 1965.

[5] S.V. Patankar. *Numerical heat transfer and fluid flow.* Hemisphere Publishing Corp, 1980.

[6] H. F. Meier, J. J. N. Alves, and M. Mori. Comparison between staggered and collocated grids in the finite-volume method performance for single and multi-phase flows. *Computers & Chemical Engineering*, 23(3):247 – 262, 1999.

[7] B. Yu, W-Q. Tao, J-J. Wei, Y. Kawaguchi, T. Tagawa, and H. Ozoe. Discussion on momentum interpolation method for collocated grids of incompressible flow. *Numerical Heat Transfer Part B: Fundamentals*, 42(2):141–166, 2002.

[8] C.M Rhie and W.L. Chow. Numerical study of the turbulent flow past an airfoil with trailing edge separation,. *AIAA Journal*, 21(11):1525–1532, 1983.

[9] W.Q. Tao and Y.L. Qu, Z.G.and He. A novel segeregated algorithm for incompressible fluid flow and heat transfer problems - CLEAR (Coupled and Linked Equations Algorithm Revised) part I: Mathematical formulation and solution procedure. *Numerical Heat Transfer, Part B*, 45(1):1–17, 2004.

[10] I. Demirdžić, Ž. Lilek, and M. Perić. A collocated finite volume method for predicting flows at all speeds. *International Journal for Numerical Methods in Fluids*, 16(12):1029–1050, 1993.

[11] F. Moukalled and M. Darwish. A unified formulation of the segeregated class of algorithms for fluid flow at all speeds. *Numerical Heat Transfer Part B: Fundamentals*, 37(1):103–139, 2000.

[12] S.V. Patankar and D.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.

[13] D.L. Sun, Z.G. Qu, Y.L. He, and W.Q. Tao. Implementation of an efficient segregated algorithm-ideal on 3d collocated grid system. *Chinese Science Bulletin*, 54:929–942, 2009.

[14] M.J. Moran and H.N. Shapiro. *Fundamentals of engineering thermodynamics*. John Wiley and Sons Ltd., 2006.

[15] J.P. Van Doormaal and G.D. Raithby. Enhancements of the simple method for predicting incompressible fluid flows. *Numerical heat transfer*, 7(2):147 – 163, 1984.

[16] D.L. Sun, Z.G. Qu, Y.L. He, and W.Q. Tao. An efficient segregated algorithm for incompressible fluid flow and heat transfer problems ideal (inner doubly iterative efficient algorithm for linked equations) part i: Mathematical formulation and solution procedure. *Numerical heat transfer. Part B, Fundamentals*, 53(1):1 – 17, 2008.

[17] D.L. Sun, Z.G. Qu, Y.L. He, and W.Q. Tao. An efficient segregated algorithm for incompressible fluid flow and heat transfer problems—ideal (inner doubly iterative efficient algorithm for linked equations) part ii: Mathematical formulation and solution procedure. *Numerical heat transfer. Part B, Fundamentals*, 53(1):18 – 38, 2008.

[18] D.L. Sun, Z.G. Qu, Y.L. He, and W.Q. Tao. Performance analysis of ideal algorithm for three-dimensional incompressible fluid flow and heat transfer problems. *International Journal for Numerical Methods in Fluids*, 61(10):1132–1160, 2009.

[19] J.J. McGuirk and J.M.L.M. Palma. The efficiency of alternative pressure-correction formulations for incompressible turbulent flow problems. *Computers & Fluids*, 22(1):77 – 87, 1993.

[20] M. Zeng and W. Tao. A comparison study of the convergence characteristics and robustness for four variants of simple-family at fine grids. *Engineering Computations*, 20(3):320–340, 2009.

[21] G. A. Sod. A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. *Journal of Computational Physics*, 27(1):1 – 31, 1978.

[22] J. F. Hawley, L. L. Smarr, and J. R. Wilson. A numerical study of non-spherical black hole accretion. i equations and test problems. *Astrophysical Journal, Part 1*, 277:296 – 311, 1984.

[23] C. Pfrommer, V. Springel, T. A. Enßlin, and M. Jubelgas. Detecting shock waves in cosmological smoothed particle hydrodynamics simulations. *Monthly Notices of the Royal Astronomical Society*, 367(1):113–131, 2006.

A Shock tube problem

For the shock tube described in Sect. 4.2, a quasi-analytical solution can be obtained by assuming polytropic equation of state and isentropic flow everywhere, except across the shock. The shock tube can be divided into five regions: (5) An undisturbed high pressure side, (4) a rarefaction wave, (3) a region with constant properties separated from (2) another region with constant properties by a contact discontinuity and (1) an undisturbed low pressure side separated from the second constant region by a shock.

The equations used for the calculation of the analytical solution are here shown. Deviations of the equations can be found in [23] and [22].

The polytropic equation of state and the speed of sound:

$$p = K\rho^{\gamma} \tag{46}$$

$$c^2 = \gamma p/\rho \tag{47}$$

By combining Eqs. (47) and (46) with equations for conservation of mass, momentum and energy, equations for the density, velocity and pressure are obtained:

$$\rho = \begin{cases} \rho_5 & \text{if } x \le -c_5 t \\ \rho_5 \left(-g^2 \frac{x}{c_5 t} + (1-g^2)\right)^{2/(\gamma-1)} & \text{if } -c_5 t < x \le -v_t t \\ \rho_3 & \text{if } -v_t t < x \le v_2 t \\ \rho_2 & \text{if } v_2 t < x \le v_s t \\ \rho_1 & \text{if } x > v_s t \end{cases} \tag{48}$$

$$p = \begin{cases} p_5 & \text{if } x \le -c_5 t \\ p_5 \left(-g^2 \frac{x}{c_5 t} + (1-g^2)\right)^{2\gamma/(\gamma-1)} & \text{if } -c_5 t < x \le -v_t t \\ p_3 & \text{if } -v_t t < x \le v_2 t \\ p_2 = p_3 & \text{if } v_2 t < x \le v_s t \\ p_1 & \text{if } x > v_s t \end{cases} \tag{49}$$

$$v = \begin{cases} 0 & \text{if } x \le -c_5 t \\ (1-g^2)\left(\frac{x}{t} + c_5\right) & \text{if } -c_5 t < x \le -v_t t \\ v_3 & \text{if } -v_t t < x \le v_2 t \\ v_2 = v_3 & \text{if } v_2 t < x \le v_s t \\ 0 & \text{if } x > v_s t \end{cases} \tag{50}$$

where v_t is the propagation speed of the tail of the rarefaction wave and v_s is the shock speed.

$$g^2 = \frac{\gamma - 1}{\gamma + 1} \tag{51}$$

The post shock pressure, p_2, and velocity, v_2, are found in the intersection between possible post shock states and possible values for the rarefaction wave:

$$\left(\frac{p_2}{p_1} - 1\right) \sqrt{\frac{1 - g^2}{\gamma(p_2/p_1 + g^2)}} - \frac{2c_5}{(\gamma - 1)c_1} \left(1 - \left(\frac{p_2}{p_5}\right)^{(\gamma-1)/(2\gamma)}\right) = 0 \tag{52}$$

When the post shock pressure, $p_2 = p_3$, is known, the density on each side of the contact discontinuity can be found from the polytropic equation of state and the Rankine-Hugoniot conditions:

$$\rho_3 = \rho_5 \left(\frac{p_2}{p_5}\right)^{1/\gamma} \tag{53}$$

$$\rho_2 = \rho_1 \left(\frac{p_2 + g^2 p_1}{p_1 + g^2 p_2}\right) \tag{54}$$

Values for the velocity in the rarefaction wave are obtained by a rarefaction wave equation:

$$v_2 = v_3 = \frac{2c_5}{\gamma - 1} \left(1 - \left(\frac{p_2}{p_5}\right)^{(\gamma-1)/2\gamma}\right) \tag{55}$$

The speed of the tail of the rarefaction wave, v_t, and the speed of the shock, v_s, are obtained from Eq. (50) and the conservation of mass:

$$v_t = c_5 - \frac{v_2}{1 - g^2} \tag{56}$$

$$v_s = \frac{v_2}{1 - \rho_1/\rho_1} \tag{57}$$