Deflated Krylov-Schwarz Domain Decomposition Methods for the CFD package X-stream

Interim Master's Thesis

Jarno Verkaik

Delft, 28th February 2003

Delft University of Technology
&
TNO TPD

supervisors: Prof.dr.ir. P. Wesseling (TU Delft)
Dr.ir. C. Vuik (TU Delft)
Ir. B.D. Paarhuis (TNO TPD)
Dr. A. Twerda (TNO TPD)

Contents

Preface

This interim Master's thesis is written for the degree of Master of Science for the study Applied Mathematics, faculty of Information Technology and Systems, Delft University of Technology. The graduation work is done in the unit of Numerical Analysis of the department of Applied Mathematical Analysis. The Master's project is being carried out at TNO TPD, division Models and Processes, department of Process Physics.

At Process Physics nine months of work will be done for the simulation package X-stream. The Master's project is separated into two parts: the first three months are intended for a literature study in order to decide the Master's project path, the last six months are for carrying out the chosen research. This interim Master's thesis comprises the first part of the project.

Delft, February 2003 Jarno Verkaik

1

Introduction

Mathematical simulation of flows is important for the design, optimization and trouble shooting of glass melting furnaces. To gain insight into the glass melting process physical experiments can be done. However, physical experiments are often very costly and time-consuming, and there are circumstances under which certain physical quantities cannot be measured. Simulation by Computational Fluid Dynamics (CFD) does not have these disadvantages. Although CFD only approximates the true physics, it gives engineers in the glass industry great insight into the transport phenomena occurring in glass melting furnaces.

At Process Physics at TNO TPD a CFD simulation package called X-stream is developed for the glass industry. With this package the glass melt and the combustion space can be simulated simultaneously. Besides solving the incompressible Navier-Stokes equations and the energy equation, various other physical models can be solved, for example models for turbulence, combustion and radiation.

To simulate the complex physical processes in a furnace with a high accuracy, many equations have to be solved, resulting in large computing times. To solve larger problems, parallel computing can be done in X-stream. A domain decomposition approach is used to divide the total problem into smaller problems, which can be computed in parallel on different processors.

In X-stream there is little experience with the domain decomposition algorithm used. The main goal of the Master's project research is get this experience and to improve this algorithm. Because the Navier-Stokes equations are nonlinear, solving these equations is very time-consuming. Therefore we focus in this research primarily on the Navier-Stokes equations.

The main goal of this interim Master's thesis is to gain insight in the existing methods to improve the X-stream domain decomposition algorithm and to decide the course of the Master's project.

The structure of this thesis is as follows. In Chapter 1 the mathematical model of a flow is discussed briefly, followed by the numerical model in Chapter 2. In Chapter 3 discussion of iterative methods for solving the systems of equations arising from discretization is presented. Domain decomposition is treated in Chapter 4, followed by the description of some testcases in Chapter 5. Chapter 6 concludes this thesis with a discussion of the continuation of the Master's project.

2

Mathematical model of a flow

2.1 Introduction

In Section 2.2 the physical conservation laws are discussed briefly. As far as additional physical models are concerned we restrict ourselves to a short treatment of the turbulence model in Section 2.3. In Section 2.4 the basic idea of wall functions boundary conditions at solid walls will be discussed without going into details. In Section 2.5 the connection with X-stream is given.

2.2 Basic transport equations

The conservation equations of mass, momentum, energy and chemical species for a multicomponent system can be found in various publications, for example Post [21], Twerda [25] and Verweij [27]. For the derivations of these equations we refer to standard textbooks on fluid dynamics, for example Bird [2], Batchelor [1], Ferziger & Perić [6] or Wesseling [38].

The conservation equations are given for a Cartesian coordinate system $(x_1, \ldots, x_d)$, where d is the number of space dimensions. Boldface Latin letters denote vectors, for example, $\boldsymbol{x} = (x_1, \ldots, x_d)$. We will use Einstein's summation convention: when a subscript is repeated in a term, a summation of d terms is implied. For example, the divergence of a vector field $\boldsymbol{u}$ is given by

$$\frac{\partial u_j}{\partial x_j} = \sum_{j=1}^{d} \frac{\partial u_j}{\partial x_j} \ . \tag{2.1}$$

Conservation of mass

Consider a multicomponent mixture consisting of N_s species with massfractions Y_α, $\alpha = 1, 2, \ldots, N_s$, $\sum_{\alpha=1}^{N_s} Y_\alpha = 1$. The velocity of the mixture $\boldsymbol{u}$ is related to the artificial velocities of the individual components $\boldsymbol{u}_\alpha$ by

$$\boldsymbol{u} = \sum_{\alpha=1}^{N_s} Y_\alpha \boldsymbol{u}_\alpha \ .$$

The difference between $\boldsymbol{u}_\alpha$ and $\boldsymbol{u}$ is called the diffusion velocity of species α,

$$\boldsymbol{U}_\alpha = \boldsymbol{u}_\alpha - \boldsymbol{u} \ .$$

The density of the mixture is denoted by ρ.

The conservation law of mass or the so-called continuity equation reads

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u_j)}{\partial x_j} = 0, \tag{2.2}$$

where u_j is the velocity of the mixture in j-direction.

Conservation of momentum

The momentum equations express the conservation of momentum of the flow. They read

$$\frac{\partial(\rho u_i)}{\partial t} + \frac{\partial(\rho u_i u_j)}{\partial x_j} = \frac{\partial \sigma_{ij}}{\partial x_j} + \sum_{\alpha=1}^{N_s} (\rho Y_\alpha f_i^{b,\alpha}) \ ,$$

where σ_{ij} is the stress tensor and $f_i^{b,\alpha}$ is the body force per unit mass on species α in i-direction. For Newtonian fluids σ_{ij} is given by the following constitutive relation:

$$\begin{aligned} \sigma_{ij} &= -p\delta_{ij} + \tau_{ij} \\ &= -p\delta_{ij} + 2\mu s_{ij} - (\tfrac{2}{3}\mu - \kappa_{\text{bulk}}) s_{kk}\delta_{ij} \ , \end{aligned} \tag{2.3}$$

where p is the pressure, δ_{ij} the Kronecker delta, μ the dynamic viscosity, κ_{bulk} the bulk viscosity, τ_{ij} the shear stress tensor and s_{ij} the rate-of-strain tensor, which reads

$$s_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \ .$$

The bulk viscosity κ_{bulk} will be neglected. For each species α we will choose

$$f_i^{b,\alpha} = g_i \ ,$$

where g_i is the gravitational acceleration in i-direction. Using this assumption the momentum equations becomes:

$$\frac{\partial(\rho u_i)}{\partial t} + \frac{\partial(\rho u_i u_j)}{\partial x_j} = -\frac{\partial p}{\partial x_j} + \frac{\partial \tau_{ij}}{\partial x_j} + \rho g_i. \tag{2.4}$$

The equations given by (2.4) and (2.3) are called the Navier-Stokes equations of motion.

Conservation of energy

Most of the time the fluid cannot be taken as isothermal, and conservation of energy must be taken into account. The energy equation is given by

$$\frac{\partial(\rho E)}{\partial t} + \frac{\partial(\rho u_j E)}{\partial t} = \frac{\partial(u_i \sigma_{ij})}{\partial x_j} - \frac{\partial q_j}{\partial x_j} + \sum_{\alpha=1}^{N_s} (\rho Y_\alpha u_i f_i^{b,\alpha}) + S_{\text{rad}} \ ,$$

where E is the total energy per unit mass, given by

$$E = e + \frac{1}{2} u_i u_i \ ,$$

where e is the internal energy and $\frac{1}{2}u_iu_i$ the kinetic energy. The variable q_j is the energy flux in j-direction and S_{rad} the radiative source term. Rewriting the energy equation in terms of the enthalpy h, defined as

$$h = e + \frac{p}{\rho} \ ,$$

evaluating σ_{ij} and substituting $f_i^{b,\alpha} = g_i$ yields

$$\frac{\partial}{\partial t}(\rho h + \frac{1}{2}\rho u_i u_i - p) + \frac{\partial}{\partial x_j}[\rho u_j (h + \frac{1}{2}u_i u_i)] = \frac{\partial(\tau_{ij}u_i)}{\partial x_j} - \frac{\partial q_j}{\partial x_j} + \rho u_i g_i + S_{\text{rad}} \ .$$

Now there will be made some assumptions considering the terms of the enthalpy energy equation above.

The heat flux (minus the radiation heat) can be simplified by

$$q_j = -\frac{\lambda}{c_p}\frac{\partial h}{\partial x_j} \ ,$$

where λ is the thermal conductivity and c_p the specific heat at constant pressure. For Mach numbers ($|\boldsymbol{u}|/c$, with c the local speed of sound) smaller than 0.3, the flow is almost incompressible and terms with the kinetic energy $\frac{1}{2}u_iu_i$ can be neglected. Also neglecting the viscous dissipation term $\partial(\tau_{ij}u_i)/\partial x_j$ and the gravitational term $\rho u_i g_i$, reduces the enthalpy energy equation to

$$\frac{\partial(\rho h - p)}{\partial t} + \frac{\partial(\rho u_j h)}{\partial x_j} = \frac{\partial}{\partial x_j}\left(\frac{\lambda}{c_p}\frac{\partial h}{\partial x_j}\right) + S_{\text{rad}} \ . \tag{2.5}$$

Conservation of chemical species

The mass conservation law for a specie α is given by

$$\frac{\partial(\rho Y_\alpha)}{\partial t} + \frac{\partial(\rho u_j Y_\alpha)}{\partial x_j} = -\frac{\partial}{\partial x_j}[\rho Y_\alpha (\boldsymbol{U}_\alpha)_j] + S_{\alpha,\text{reac}} \ ,$$

where $S_{\alpha,\text{reac}}$ is a source (or sink) term representing the chemical reactions. In this equation the multicomponent diffusion flux $\rho Y_\alpha(\boldsymbol{U}_\alpha)_j$ can be rewritten using Fick's law,

$$\rho Y_\alpha (\boldsymbol{U}_\alpha)_j = -\rho D_\alpha \frac{\partial Y_\alpha}{\partial x_j} \ ,$$

where D_α is the diffusion coefficient of species α. Now the conservation equation of chemical species becomes

$$\frac{\partial(\rho Y_\alpha)}{\partial t} + \frac{\partial(\rho u_j Y_\alpha)}{\partial x_j} = \frac{\partial}{\partial x_j}\left(\rho D_\alpha \frac{\partial Y_\alpha}{\partial x_j}\right) + S_{\alpha,\text{reac}} \ . \tag{2.6}$$

Note that this equation only has to be solved for $N_s - 1$ species, because $Y_{N_s} = 1 - \sum_{\alpha=1}^{N_s-1} Y_\alpha$.

State equations

We have obtained a total of $5+N_s$ conservation equations: one for mass, three for momentum (one for each coordinate direction), one for energy and N_s for chemical species. In total there are $7 + N_s$ unknowns: ρ, u_j $(j = 1, 2, 3)$, p, h, e and Y_α $(\alpha = 1, 2, \ldots, N_s)$. The set of equations is closed by two equations of state. For a gas the hydrodynamic equation of state can be used, which is derived from the ideal gas law

$$p = \rho R^0 T \sum_{i=\alpha}^{N_s} \frac{Y_\alpha}{M_\alpha} \ ,$$

where R^0 is the universal gas constant and M_i the molar mass of species i. The second equation relates the sensible enthalpy to the temperature,

$$h_{\text{sens}} = \sum_{\alpha=1}^{N_s} \left(\int_{T_{\text{ref}}}^{T} c_{p,\alpha}(\theta) d\theta \right) Y_\alpha \ ,$$

where T_{ref} is a reference temperature, and $c_{p,\alpha}$ the specific heat at constant pressure for species α. we have choosen the reference enthalpy and temperature to be zero. This equation is called the caloric equation of state.

General form of the conservation equations

The general form of the conservation equations of mass, momemtum, energy and chemical species reads

$$\frac{\partial(\rho\phi)}{\partial t} + \frac{\partial F_j}{\partial x_j} = S_\phi \ , \tag{2.7}$$

where $F_{\phi,j}$ is the total flux of quantity ϕ in x_j direction. The total flux is the sum of the convective and the diffusive flux,

$$F_j = F_j^{\text{con}} + F_j^{\text{dif}} = \rho u_j \phi - \Gamma_\phi \frac{\partial \phi}{\partial x_j} \ , \tag{2.8}$$

where Γ_ϕ is the effective transport coefficient. For example for the continuity equation $\phi = 1$, $\Gamma_\phi = 0$ and $S_\phi = 0$.

2.3 Turbulence model

For sufficiently large Reynolds numbers flows show rapid apparently random fluctuations. Such flows are called turbulent. For an introduction to turbulence see for example Nieuwstadt [18].

The timedependent Navier-Stokes equations still apply to turbulent flows and solving this coupled set of equations for the flow would give a complete description of the fluid behaviour, including the small-scale turbulence effects. However, a turbulent flow exhibits on a broad range of length and time scales, and because of this a numerical solution of the complete set of time-dependent conservation equations resolving all time and length scales is far beyond

the current computer capacity when applied to most applications. In engineering, one is fortunately not always interested in the instantaneous values of the flow variabeles, but most of the time in statistical values, such as time-averaged values.

2.3.1 Reynolds and Favre decomposition

Now two decompositions of the quantity ϕ will be discussed: Reynolds decomposition and Favre decomposition.

Reynolds decomposition

For a turbulent flow we split a quantity ϕ into a mean part and a fluctuating part,

$$\phi = \bar{\phi} + \phi' \ .$$

This decomposition is usually called the Reynolds decomposition. The Reynolds conditions are defined by

$$\begin{aligned}
&\text{(i)} \quad \overline{f+g} = \overline{f} + \bar{g} \ , \\
&\text{(ii)} \quad \overline{\alpha f} \ = \alpha \bar{f} \ , \\
&\text{(iii)} \quad \overline{\frac{\partial f}{\partial s}} \ = \frac{\partial \bar{f}}{\partial s} \ , \\
&\text{(iv)} \quad \overline{\overline{f} g} \ = \bar{f}\bar{g} \ ,
\end{aligned}$$

where f and g are fluctuating quantities and α a constant.

The three most pertinent ways are to define the average $\bar{\phi}$ are time averaging, spatial averaging and ensemble averaging. Time average is appropriate for stationary turbulence, spatial average for homogeneous turbulence, and ensemble average is the most general type of averaging.

In practice all problems involve inhomogeneous turbulence and often the bulk flow is said to be stationary. That is the reason why very often time averaging is used,

$$\bar{\phi}^T(\boldsymbol{x}, t) = \frac{1}{T} \int_{-T/2}^{T/2} \phi(\boldsymbol{x}, t+\tau) d\tau \ .$$

A problem of the above definition is that it does not satisfy Reynolds condition (iv). Another definition of the average that does satisfy all Reynolds conditions is ensemble averaging,

$$\overline{\phi}^E(\boldsymbol{x}, t) = \lim_{N_e \to \infty} \frac{1}{N_e} \sum_{i=1}^{N_e} \phi^{(i)}(\boldsymbol{x}, t) \ ,$$

where N_e is the total number of turbulence-experiments and $\phi^{(i)}(\boldsymbol{x}, t)$ a realization in the i-th experiment. Using the so called ergodic hypothesis for a stationary turbulent flow it is assumed that

$$\lim_{T \to \infty} \overline{\phi}^T = \overline{\phi}^E \ .$$

Favre decomposition

For turbulent flows with density variations due to temperature effects it is customary to use density weighted or Favre averaged quantities. The Favre decomposition for a quantity ϕ reads

$$\phi = \tilde{\phi} + \phi'' \ ,$$

where the Favre average $\tilde{\phi}$ is defined by

$$\tilde{\phi} = \frac{\overline{\rho\phi}}{\bar{\rho}}. \tag{2.9}$$

The main advantage above the ordinary Reynolds decomposition is that explicit density correlations are avoided in the averaged transport equations.

Relations

With the Reynolds conditions one can derive the relations

$$\bar{\bar{f}} = \bar{f} \ , \quad \tilde{\tilde{f}} = \tilde{f} \ , \quad \overline{f'} = 0 \ , \quad \widetilde{f''} = 0 \ , \quad \overline{f''} = \bar{f} - \tilde{f} \ , \quad \overline{\rho f''} = 0 \ .$$

It follows that

$$\overline{fg} = \tilde{f}\tilde{g} + \overline{f'g'} \ ,$$
$$\overline{\rho fg} = \bar{\rho}(\tilde{f}\tilde{g} + \widetilde{f''g''}) = \bar{\rho}\tilde{f}\tilde{g} + \overline{\rho f''g''} \ .$$

2.3.2 Reynolds averaged transport equations

Now Reynolds averaging combined with Favre decomposition will be applied to the conservation equations given in Section 2.2.

Conservation of mass

Taking the mean of the continuity equation (2.2) and using the Reynolds conditions gives

$$\frac{\partial\bar{\rho}}{\partial t} + \frac{\partial(\overline{\rho u_j})}{\partial x_j} = 0 \ . \tag{2.10}$$

With $\overline{\rho u_j} = \bar{\rho}\tilde{u}_j$ by (2.9) this equation becomes

$$\frac{\partial\bar{\rho}}{\partial t} + \frac{\partial(\bar{\rho}\tilde{u}_j)}{\partial x_j} = 0 \ . \tag{2.11}$$

Conservation of momentum

Reynolds averaging the Navier-Stokes equations (2.4) and using Favre decomposition gives

$$\frac{\partial(\bar{\rho}\tilde{u}_i)}{\partial t} + \frac{\partial}{\partial x_j}(\bar{\rho}\tilde{u}_i\tilde{u}_j + \widetilde{\bar{\rho}u_i''u''_j}) = -\frac{\partial\bar{p}}{\partial x_j} - \frac{\partial\bar{\tau}_{ij}}{\partial x_j} + \bar{\rho}g_i \ ,$$

where

$$\bar{\tau}_{ij} = -\mu\left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}\right) + \frac{2}{3}\mu\frac{\partial \bar{u}_i}{\partial x_j}\delta_{ij} \ .$$

This equations are called the Reynolds averaged Navier-Stokes equations.

Conservation of energy

Reynolds averaging the enthalpy energy equation (2.5) using Favre decomposition yields

$$\frac{\partial(\bar{\rho}\tilde{h})}{\partial t} + \frac{\partial}{\partial x_j}(\bar{\rho}\tilde{u}_j\tilde{h} + \widetilde{\bar{\rho}u_j''h''}) = \frac{\partial}{\partial x_j}\left(\frac{\lambda}{c_p}\frac{\partial \bar{h}}{\partial x_j}\right) + \bar{S}_{\text{rad}} \ . \tag{2.12}$$

Conservation of species

Reynolds averaging the equation of chemical species (2.6) using Favre decomposition results in

$$\frac{\partial(\bar{\rho}\tilde{Y}_\alpha)}{\partial t} + \frac{\partial}{\partial x_j}(\bar{\rho}\tilde{u}_j\tilde{Y}_\alpha + \overline{\bar{\rho}u_j'Y_\alpha'}) = \frac{\partial}{\partial x_j}\left(\rho D_\alpha\frac{\partial \tilde{Y}_\alpha}{\partial x_j}\right) + \bar{S}_{\alpha,\text{reac}} \ . \tag{2.13}$$

The terms $\widetilde{\bar{\rho}u_i''u_j''}$ (the so called Reynolds stresses), $\widetilde{\bar{\rho}u_j''h''}$ and $\widetilde{\bar{\rho}u_j''Y_\alpha''}$ introduced by the Reynolds decomposition are unknown. In order to close the problem we need a turbulence model to evaluate the unknown terms. A survey of turbulence modeling can be found in for example Wilcox [39].

There are four main turbulence models: algebraic (or zero-equation) models, one-equation models, two-equation models and second-order closure (or Reynolds stress) models. By definition, a n-equation model signifies a model that requires solution of n additional partial differential equations (PDEs). All the n-equation models are based on the Boussinesq closure hypothesis, which reads for the Reynolds decomposition using Favre decomposition

$$-\widetilde{\bar{\rho}u_i''u_j''} = 2\mu_t\tilde{s}_{ij} - \frac{2}{3}\delta_{ij}(\tilde{\rho}\tilde{k} + \mu_t\tilde{s}_{kk}) \ ,$$

where μ_t is the (dynamic) eddy viscosity, $\bar{s}_{ij}$ is the Favre averaged strain-rate tensor and $\tilde{k}$ the turbulent kinetic energy, given by $\tilde{k} = \frac{1}{2}\widetilde{u_i''u_i''}$. In second-order closure models the Boussinesq's closure hypothesis is abandoned, and differential equations are formulated for the individual components of the Reynolds stress tensor.

2.3.3 Standard k-ϵ turbulence model

In this subsection a popular two-equation turbulence model will be treated briefly, the so-called k-ϵ model. This model performs especially well for isotropic turbulent pipe flows.

In the k-ϵ model the eddy viscosity is defined as

$$\mu_t = C_\mu\overline{\rho}\tilde{k}^2/\tilde{\epsilon} \ ,$$

where C_μ is an empirical constant and ϵ the turbulent dissipation per unit mass, defined by

$$\epsilon = \frac{\mu}{\bar{\rho}}\widetilde{\frac{\partial u_i''}{\partial x_j}\frac{\partial u_i''}{\partial x_j}} \ .$$

The equations for $\tilde{k}$ and $\tilde{\epsilon}$ read

$$\frac{\partial \bar{\rho}\tilde{k}}{\partial t} + \frac{\partial(\bar{\rho}\tilde{u}_j\tilde{k})}{\partial x_j} = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_k}\right)\frac{\partial \tilde{k}}{\partial x_j}\right] + P_k - \bar{\rho}\tilde{\epsilon} \ ,$$
$$\frac{\partial \bar{\rho}\tilde{\epsilon}}{\partial t} + \frac{\partial(\bar{\rho}\tilde{u}_j\tilde{\epsilon})}{\partial x_j} = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_\epsilon}\right)\frac{\partial \tilde{\epsilon}}{\partial x_j}\right] + C_{\epsilon 1}P_k\frac{\tilde{\epsilon}}{\tilde{k}} - C_{\epsilon 2}\frac{\bar{\rho}\tilde{\epsilon}^2}{\tilde{k}} \ ,$$

where P_k is the production of kinetic energy, given by

$$P_k = -\bar{\rho}\widetilde{u_i'' u_j''}\frac{\partial \tilde{u}_i}{\partial x_j} \ .$$

The constants σ_k and σ_ϵ are turbulent Prandtl numbers and C_μ, C_k and C_ϵ empirical constants. In an analogous way the unknown terms in the turbulent conservation equations of energy (2.12) and chemical species (2.13) can be modeled by posing

$$\bar{\rho}\widetilde{u_j'' h''} = -\frac{\mu_t}{\sigma_h}\frac{\partial \tilde{h}}{\partial x_j} \ ,$$
$$\bar{\rho}\widetilde{u_j'' Y_\alpha''} = -\frac{\mu_t}{\sigma_\alpha}\frac{\partial \tilde{Y}_\alpha}{\partial x_j} \ ,$$

where σ_h and σ_α are turbulent Prandtl numbers.

2.4 Solid wall boundary conditions

Often the no-slip boundary condition is used near solid walls, which means that the velocities of the flow tend to the wall-velocities. This is only valid near solid walls if the Reynolds number is small, so viscous effects must be taken into account. Using the standard k-ϵ turbulence model (see Section 2.3.3) this causes a problem, because this model is only valid for high Reynolds numbers (Re > 5000).

One way to solve this problem is to use a low Reynolds number turbulence model. However, these models are computingly very expensive because very fine grids near the solid walls are required. A more appropriate way to solve this problem is to use the so-called wall function method, which will briefly be discussed now.

Let y^+ be a dimensionless distance to the wall, given by

$$y^+ = \frac{\rho y u_\tau}{\mu} \ ,$$

where u_τ is known as the friction velocity which is related to the wall shear-stress τ_w by

$$u_\tau = \sqrt{\frac{\tau_w}{\rho}} \ .$$

The boundary layer, starting from the solid wall can be split into three regions of interest, *i.e.*

1. viscous sublayer, $0 < y^+ < 5$,

2. buffer layer, $5 < y^+ < 30$,

3. inertial (fully turbulent) sublayer, $30 < y^+ < 1000$.

The nearest-wall grid point is often chosen in the inertial sublayer, and not in the viscous sublayer. One reason for this is that the viscous sublayer is generally not a region of our interest. Another reason is that having a grid point in this layer will cause numerical problems because of stretched grid cells.

The classical law of the wall, which is valid for the inertial sublayer, reads

$$U^+ = \frac{1}{\kappa} \ln y^+ + B \ ,$$

with κ and B dimensionless empirical constants. The constant κ is called the Von Kármán constant. Near the solid wall convection and diffusion of k and ϵ is neglected by the argument of local equilibrium between production and destruction. This leads to the following expressions of k and ϵ in the wall region,

$$k = \frac{u_\tau^2}{\sqrt{C_\mu}} \ , \quad \epsilon = \frac{u_\tau^3}{\kappa y} \ .$$

The wall function for the enthalpy can be given by

$$q_w = \rho u_\tau \frac{h - h_w}{\sigma_h} U^+ + P_j \ ,$$

where q_w is the wall heat flux, h_w the enthalpy at the wall and P_j given by

$$P_j = \frac{\pi/4}{\sin(\pi/4)} \left(\frac{A^+}{\kappa} \right)^{1/2} \frac{\sigma_h}{\sigma_{h,\mathrm{lam}}} (\sigma_{h,\mathrm{lam}} - \sigma_h) \ ,$$

where A^+ is the empirical dimensionless Van Driest constant, and $\sigma_{h,\mathrm{lam}}$ the laminar Prandtl number.

2.5 Connection with X-stream

In X-stream all the basic conservation equations as discussed in Section 2.2 can be solved for the glass melting process and the combustion process. Both two-equation and second-order closure turbulence models are integrated. For turbulent flows Reynolds decomposition is used and wall functions are applied. The other main physical models in X-stream are: combustion, radiation, batch, electrical boosting, foam, bubbling, stirring, volatilization and refractory corrosion.

3

Finite volume discretization

3.1 Introduction

Generally three discretization methods can be distinguished for discretising partial differential equations (PDEs): the finite difference method, the finite element method and the finite volume method. In this chapter the Finite Volume (FV) method will be discussed, which is in great detail treated in for example Patankar [20], Ferziger & Perić [6] or Wesseling [38].

In this chapter we will closely follow Wesseling [38] and entirely adopt his notation. For simplicity we will use a Cartesian tensor notation: summation takes place over Greek indices that occur twice in a term or product, for example

$$u_{\alpha,\alpha} = \sum_{\alpha=1}^{d} \frac{\partial u_\alpha}{\partial x_\alpha} \ .$$

In Section 3.2 first the computing grid is described, which will be used throughout this chapter. In Sections 3.3 and 3.4 respectively the discretizations of the general convection-diffusion equation and the incompressible Navier-Stokes equations are described. In Section 3.5 the connection with X-stream is given.

3.2 The computing grid

In this chapter we restrict ourselves to the two-dimensional case for sake of simplicity. Generalisation to three dimensions is straightforward. We will use a cell-centred uniform grid, see Figure 3.1. The rectangular domain $\Omega = L_1 \times L_2$ is subdivided in rectangular cells of size $h_1 \times h_2$. The computing grid G is the set of cell-centres:

$$G = \{\boldsymbol{x} \in \Omega : \boldsymbol{x} = \boldsymbol{x_j} \ , \quad \boldsymbol{j} = (j_1, j_2) \ , \quad j_\alpha = 1, 2, \ldots, m_\alpha \ , \quad m_\alpha = L_\alpha / h_\alpha\} \ ,$$

with $\boldsymbol{x_j}$ defined by

$$\boldsymbol{x_j} = (x_j^1, x_j^2) \ , \quad x_j^1 = (j_1 - 1/2)h_1 \ , \quad x_j^2 = (j_2 - 1/2)h_2 \ .$$

The cell with center $\boldsymbol{x_j}$ is denoted by $\Omega_{\boldsymbol{j}}$. Define

$$\boldsymbol{e}_1 \equiv (1/2, 0) \ , \quad \boldsymbol{e}_2 \equiv (0, 1/2) \ .$$

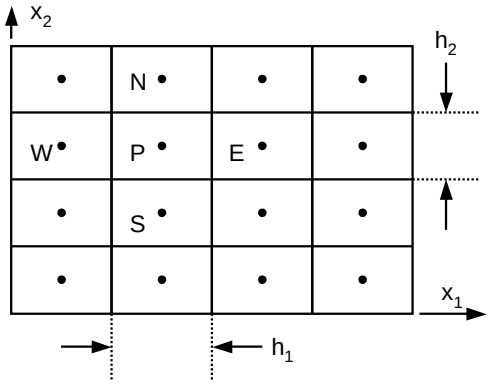

Figure 3.1: A cell-centred grid. (• grid points; – finite volume boundaries.)

The value of a quantity φ in $\boldsymbol{x_j}$ is denoted by $\varphi_{\boldsymbol{j}}$, and $\varphi_{\boldsymbol{j}+\boldsymbol{e}_1}$ is located at a cell face, namely at

$$\boldsymbol{x}_{\boldsymbol{j}+\boldsymbol{e}_1} = (j_1 h_1, (j_2 - 1/2)h_2) \ .$$

The cell at the 'east' side of $\Omega_{\boldsymbol{j}}$ is designated by $\Omega_{\boldsymbol{j}+2\boldsymbol{e}_1}$.

3.3 The convection-diffusion equation

3.3.1 Problem description

Taking $\rho = 1$, dropping the subscript of Γ_φ, and renaming S_ϕ by q, the convection-diffusion equation given by equation (2.7) and (2.8) can be written as

$$\frac{\partial \varphi}{\partial t} + (u_\alpha \varphi)_{,\alpha} - (\Gamma \varphi_{,\alpha})_{,\alpha} = q \ , \quad \boldsymbol{x} \in \Omega \subset \mathbb{R}^d \ , \quad 0 < t \leq T \ .$$

with T a given time. We assume that this equation is linear, with the quantity φ to be the only unknown.

The dimensionless form of the above equation is

$$\frac{\partial \varphi}{\partial t} + \mathcal{L}\varphi = q \ , \quad \mathcal{L}\varphi \equiv (u_\alpha \varphi)_{,\alpha} - (\varepsilon \varphi_{,\alpha})_{,\alpha} \ , \tag{3.1}$$

where $\varepsilon \equiv \mathrm{Pe}^{-1}$ with Pe the Péclet number, given by

$$\mathrm{Pe} = \frac{UL}{\Gamma} \ ,$$

with L and U typical length and velocity scales.

The convection-diffusion equation is a so-called parabolic equation. However for $0 < \Gamma \ll 1$, or $\mathrm{Pe} \gg 1$ hyperbolic aspects are dominant. A kind of mixture of parabolic and hyperbolic behaviour is typical for the convection-diffusion equation.

Initial conditions

The following initial condition is required at time $t = 0$, *i.e.*

$$\varphi(0, \boldsymbol{x}) = \varphi_0(\boldsymbol{x}) \ , \quad \boldsymbol{x} \in \Omega \ .$$

Boundary conditions

One has to specify suitable boundary conditions (BCs) in order that the problem is well-posed. First of all the BCs have to be chosen such that the problem has an unique solution. For a second order equation, such as the general convection diffusion equation, this means that we have to prescribe exactly one BC at each part of the domain Ω. Second, the problem is well-posed if small perturbations in the data do not cause large changes in the solution. Suitable boundary conditions for $\varepsilon \ll 1$ are given by, see Wesseling [38],

$$\begin{aligned}
\varphi(t,\boldsymbol{x}) &= f_{\text{in}}(t,\boldsymbol{x}) \ , \quad \boldsymbol{x} \in \partial\Omega_{\text{in}} \ \ \text{(Dirichlet)} \ , \\
\varphi(t,\boldsymbol{x}) &= f_{\text{out}}(t,\boldsymbol{x}) \ , \ \boldsymbol{x} \in \partial\Omega_{\text{out}} \ \text{(Dirichlet)} \ , \ \ \text{or} \ , \\
\frac{\partial\varphi(t,\boldsymbol{x})}{\partial n} &= g_{\text{out}}(t,\boldsymbol{x}) \ , \ \boldsymbol{x} \in \partial\Omega_{\text{out}} \ \text{(Neumann)} \ ,
\end{aligned}$$

where $\boldsymbol{n}$ is the outward unit normal on the boundary $\partial\Omega$, $\partial\Omega_{\text{in}}$ is the inflow boundary (where $u_j n_j < 0$) and $\partial\Omega_{\text{out}}$ the outflow boundary ($u_j n_j > 0$).

3.3.2 Spatial and temporal discretization

Spatial discretization

We will now integrate Equation (3.1) over a cell $\Omega_{\boldsymbol{j}}$. Integrating the first term in the left-hand side (LHS) of (3.1) and using the midpoint rule gives

$$\int_{\Omega_j} \frac{\partial\varphi}{\partial t} d\Omega \approx h_1 h_2 \frac{\partial\varphi_j}{\partial t} \ .$$

Integrating the second term in the LHS of (3.1) and using Gauß' divergence theorem yields

$$\begin{aligned}
L_h\varphi_{\boldsymbol{j}} \equiv \textstyle\int_{\Omega_j} \mathcal{L}\varphi d\Omega &= \left[\textstyle\int_{\boldsymbol{x}_{j+e_1-e_2}}^{\boldsymbol{x}_{j+e_1+e_2}} - \int_{\boldsymbol{x}_{j-e_1-e_2}}^{\boldsymbol{x}_{j-e_1+e_2}}\right](u^1\varphi - \varepsilon\varphi_{,1})dx_2 \\
&\quad + \left[\textstyle\int_{\boldsymbol{x}_{j-e_1+e_2}}^{\boldsymbol{x}_{j+e_1+e_2}} - \int_{\boldsymbol{x}_{j-e_1-e_2}}^{\boldsymbol{x}_{j+e_1-e_2}}\right](u^2\varphi - \varepsilon\varphi_{,2})dx_1 \\
&= F^1|_{\boldsymbol{j}-\boldsymbol{e}_1}^{\boldsymbol{j}+\boldsymbol{e}_1} + F^1|_{\boldsymbol{j}-\boldsymbol{e}_2}^{\boldsymbol{j}+\boldsymbol{e}_2} \ ,
\end{aligned}$$

where L_h is defined as a discrete operator, and F^α is the (numerical) flux. The surface integrals in the above equation will be approximated by the midpoint rule. The diffusive flux, for example at the 'east' side, can be approximated by

$$\int_{\boldsymbol{x}_{j+e_1-e_2}}^{\boldsymbol{x}_{j+e_1+e_2}} (-\varepsilon\varphi_{,1})dx_2 \approx -\varepsilon(\varphi_{\boldsymbol{j}+2\boldsymbol{e}_1} - \varphi_{\boldsymbol{j}})\frac{h_2}{h_1} \ .$$

The convective flux can be approximated by an interpolation scheme, as will be described in section 3.3.3.

Integrating the source term in the RHS of Equation (3.1) yields

$$\int_{\Omega_j} q d\Omega \approx \hat{q}_{\boldsymbol{j}} \equiv h_1 h_2 q(\boldsymbol{x}_{\boldsymbol{j}}) \ .$$

Discretization in time

Applying spatial discretization to Equation (3.1) results in

$$\frac{d\varphi_j}{dt} + L_h\varphi_{\boldsymbol{j}} = \hat{q}_{\boldsymbol{j}} \ .$$

For discretization in time one can for example choose a first order (implicit) Euler backwards scheme,

$$(\varphi_{\boldsymbol{j}}^{(n)} - \varphi_{\boldsymbol{j}}^{(n-1)})/\tau + L_h\varphi_{\boldsymbol{j}}^{(n)} = \hat{q}_{\boldsymbol{j}}^{(n)} \ , \quad n = 1, \ldots, N \ , \quad N \equiv T/\tau \ ,$$

where τ is the time step.

3.3.3 Interpolation practices

Central and upwind discretization

There are various interpolation methods for approximating the convective flux. One way is to use a central difference scheme (CDS), which reads for a uniform grid,

$$(u\varphi)_{\mathrm{cds},\boldsymbol{j}+\boldsymbol{e}_1} \approx \frac{1}{2}u_{\boldsymbol{j}+\boldsymbol{e}_1}(\varphi_{\boldsymbol{j}} + \varphi_{\boldsymbol{j}+2\boldsymbol{e}_1}) \ .$$

The CDS is $\mathcal{O}(h_\alpha^2)$ accurate. Another way is to use a upwind difference scheme (UDS), and approximate $u\varphi$ by the value of the node upstream, *i.e.*

$$(u\varphi)_{\mathrm{uds},\boldsymbol{j}+\boldsymbol{e}_1} \approx \begin{cases} u_{\boldsymbol{j}+\boldsymbol{e}_1}\varphi_{\boldsymbol{j}} & \text{if } \ u_{\boldsymbol{j}+\boldsymbol{e}_1} > 0 \ , \\ -u_{\boldsymbol{j}+\boldsymbol{e}_1}\varphi_{\boldsymbol{j}+\boldsymbol{2e_1}} & \text{if } \ u_{\boldsymbol{j}+\boldsymbol{e}_1} \leq 0 \ . \end{cases}$$

The UDS is $\mathcal{O}(h_\alpha)$ accurate.

Spurious wiggles

Writing the convection-diffusion equation in non-conservation form one can formulate te so-called maximum principle, which can give us a priori information. By the sign of the source term this principle tells us if the exact solution has a local maximum or minimum. If this is true, wiggles in the numerical solution must be regarded as numerical artefacts.

Discretization of the convection-diffusion equation leads to a FV scheme. If this scheme is of the so-called positive type, which can be verified by the scheme's stencil, a discrete maximum principle can be formulated that gives us the conditions for which numerical wiggles may occur.

One can verify that for the convection-diffusion equation with a constant velocity field the UDS is positive for all Péclet numbers and satisfies the discrete maximum principle, so with this scheme no wiggles occur. On the other hand one can verify that the central scheme introduces wiggles for

$$p_{\boldsymbol{j}+\boldsymbol{e}_\alpha} \equiv \frac{|u_{\boldsymbol{j}+\boldsymbol{e}_\alpha}|h_\alpha}{\varepsilon} \geq 2 \ , \tag{3.2}$$

where p is called the dimensionless mesh Péclet number.

Hybrid scheme

In order to have a discretisation scheme that doe not introduce wiggles, an option is to choose an UDS. However, this is not a favourable option because this scheme is only first order accurate and it also introduces numerical diffusion. Because the central scheme is second order it is a good idea to use the upwind scheme for regions where $p > 2$ and the central scheme elsewhere. Doing this we get the so-called hybrid difference scheme (HDS), see for a more detailed description Patankar [20] or Wesseling [38],

$$(u\varphi)_{\mathrm{hds},\boldsymbol{j}+\boldsymbol{e}_\alpha} \approx s(p_{\boldsymbol{j}+\boldsymbol{e}_\alpha})(u\varphi)_{\mathrm{uds},\boldsymbol{j}+\boldsymbol{e}_\alpha} + (1 - s(p_{\boldsymbol{j}+\boldsymbol{e}_\alpha}))(u\varphi)_{\mathrm{cds},\boldsymbol{j}+\boldsymbol{e}_\alpha} \ ,$$

where $s(p_{j+e_\alpha})$ is a switch function with the mesh Péclet number defined by (3.2).

For iterative convergence in nonlinear cases it is advisable to choose $s(p)$ such that it switches smoothly between 0 and 1. Furthermore one can show that the HDS is $\mathcal{O}(h_\alpha)$ accurate.

Defect correction

Defect correction (also known as deferred correction) is a iterative method to improve the accuracy of a lower order discretization, without having to solve for a higher order discretization. More details on defect correction can be found in Wesseling [38]. Let the system of equations corresponding to a lower and a higher order discretization for the stationary convection-diffusion equation be denoted by, respectively,

$$\bar{L}_h\bar{\varphi}_h = \bar{q}_h \ , \quad L_h\psi_h = q_h \ .$$

Defect correction is given by

$$\begin{aligned} &\bar{L}_h\varphi_h^{(0)} = \bar{q}_h \ , \\ &\bar{L}_h\varphi_h^{(k)} = q_h + \beta(\bar{L}_h - L_h)\varphi_h^{(k-1)}, \ k = 1, \ldots, \ , \end{aligned}$$

where $0 \leq \beta \leq 1$ is a blending factor. If $\bar{L}_h$ is first order accurate (for example UDS) and L_h is a second order scheme (for example CDS), one can show that for $\beta = 1$ $\varphi_h^{(1)}$ is of second order accuracy.

Compared to a second order scheme, defect correction has the advantage that the resulting system has better properties when solved with an iterative solver.

3.3.4 Convergence, consistency and stability

In order that the numerical scheme used with the temporal discretization is a good scheme, it must be convergent. How this can be accomplished will now be discussed briefly.

Global and local truncation error

Truncation errors are errors that are caused by truncation (to truncate means to shorten by cutting off) of an infinite process. The process we consider is where the maximum mesh size goes to zero.

The global truncation error $\boldsymbol{e}^{(n)}$ is defined by

$$\boldsymbol{e}^{(n)} \equiv \boldsymbol{\varphi}_{\text{ex}}^{(n)} - \boldsymbol{\varphi}^{(n)} \ ,$$

where $\boldsymbol{\varphi}_{\text{ex}}^{(n)}$ is the algebraic vector which contains the exact solutions at the gridpoints, at time t_n. The local truncation error $\boldsymbol{r}^{(n)}$ is defined by

$$\boldsymbol{r}^{(n)} \equiv L_h \boldsymbol{e}^{(n)} \ ,$$

where L_h is a discrete operator.

Convergence, consistency and stability

Let $\|\cdot\|$ be some norm. A numerical scheme is convergent if $\|\boldsymbol{e}^{(n)}\| \downarrow 0,\ n = 1, \ldots, T/\tau$ for $h \downarrow 0$ and $\tau \downarrow 0$. A scheme is called consistent if $\|\boldsymbol{r}^{(n)}\| \downarrow 0,\ n = 1, 2, \ldots, T/\tau$ and $h \downarrow 0$.

Let $\boldsymbol{\delta}^{(0)}$ be a hypothetical arbitrary perturbation of $\boldsymbol{\varphi}^{(0)}$, and let $\boldsymbol{\delta}^{(n)}$ be the resulting perturbation of $\boldsymbol{\varphi}^{(n)}$. Now a numerical scheme is called stable if $\boldsymbol{\delta}^{(n)}$ remains bounded as $n \to \infty$ for all $\boldsymbol{\delta}^{(0)}$. Two useful definitions of stability are so-called zero-stability and absolute stability. A scheme is called zero-stable if there exists a bounded function $C(T)$ and a function $\tau_0(h)$ such that for arbitrary $\boldsymbol{\delta}^{(0)}$

$$\|\boldsymbol{\delta}^{(T/\tau)}\|_h \leq C(T)\|\boldsymbol{\delta}^{(0)}\|_h$$

for all $\tau \leq \tau_0(h)$ and all $h \leq h_0$ for some fixed h_0. The naming 'zero-stability' refers to the fact that the limit $h \downarrow 0$ is considered. A scheme is called absolutely stable if there exists a constant C and a function $\tau_0(h)$ such that

$$\|\boldsymbol{\delta}^{(n)}\|_h \leq C\|\boldsymbol{\delta}^{(0)}\|_h$$

for h fixed, all $n > 0$ and all $\tau \leq \tau_0(h)$. Absolute stability differs from zero-stability by the fact that h is fixed. It is favourable te have zero-stability because when the scheme is consistent (which is often not so difficult to prove) we can apply Lax's equivalence theorem, which reads

$$\text{zero-stability} + \text{consistency} \Rightarrow \text{convergence} \ ,$$

$$\text{zero-stability} \Leftarrow \text{convergence} \ .$$

The main difficulty is to prove zero- or absolute stability. But under certain conditions one can use Fourier or Von Neumann stability analysis to prove stability.

3.4 The incompressible Navier-Stokes equations

3.4.1 Problem description

Incompressible flow

The velocity field $\boldsymbol{u}(t, \boldsymbol{x})$ of the flow satisfies

$$\boldsymbol{u}(t, \boldsymbol{x}) = \frac{\partial \boldsymbol{x}(t, \boldsymbol{y})}{\partial t} \ . \tag{3.3}$$

A physical property ϕ of a material particle is called a material property. The time derivative of a material property is called the total derivative, and is denoted by $D\phi/Dt$. All material particles have some ϕ, so ϕ is defined everywhere in the flow, and therefore is a scalar field $\phi(t, \boldsymbol{x})$. We have

$$\frac{D\phi}{Dt} = \frac{\partial}{\partial t}\phi[t, \boldsymbol{x}(t, \boldsymbol{y})] = \frac{\partial \phi}{\partial t} + \frac{\partial x_\alpha(t, \boldsymbol{y})}{\partial t}\frac{\partial \phi}{\partial x_\alpha} ,$$

which can be written with (3.3) as

$$\frac{D\phi}{Dt} = \frac{\partial \phi}{\partial t} + u_\alpha \phi_{,\alpha} .$$

An incompressible flow for a constant density is a flow for which

$$\frac{D\rho}{Dt} = 0 .$$

Combining this with the dimensionless form of the continuity equation (2.1) this results in

$$u^\alpha_{,\alpha} = 0 . \tag{3.4}$$

For incompressible flow with variable density $D\rho/Dt \neq 0$, $(\rho u^\alpha)_{,\alpha} = 0$ is applied.

The Navier-Stokes equations

The dimensionless incompressible Navier-Stokes equations for a Newtonian rheology read:

$$\frac{\partial(\rho u^\alpha)}{\partial t} + (\rho u^\alpha u^\beta)_{,\beta} = -p_{,\alpha} + \sigma^{\alpha\beta}_{,\beta} + f^\alpha, \quad \sigma^{\alpha\beta} = \mathrm{Re}^{-1}(u^\alpha_{,\beta} + u^\beta_{,\alpha}) , \tag{3.5}$$

where $\boldsymbol{f}$ is a body force, $\sigma^{\alpha\beta}$ is the deviatoric stress tensor, and Re the dimensionless Reynolds number, defined by

$$\mathrm{Re} = \frac{\rho_0 U L}{\mu} ,$$

where ρ_0 is a suitable value for the density, U respectively L typical length and velocity scales, and μ is the dynamic viscosity.

Note that the Navier-Stokes equations are nonlinear, by the second term in the LHS of Equation (3.5).

Initial conditions

For the momentum equations the following initial condition is required:

$$u^\alpha(0, \boldsymbol{x}) = w^\alpha(\boldsymbol{x}) , \quad \boldsymbol{x} \in \Omega .$$

No-slip condition

At a solid surface we have the no-slip condition

$$u^\alpha(t, \boldsymbol{x}) = v^\alpha(t, \boldsymbol{x}) , \quad \boldsymbol{x} \in \Omega_{\mathrm{sw}} ,$$

with $v^\alpha(\boldsymbol{x})$ the local wall velocity.

Free surface or symmetry plane conditions

At a free surface the tangential stress components are zero. Consider for the two-dimensional case the special case that the free surface is fixed at $y = a =$ constant. In that case the normal velocity and the tangential stress $\sigma^{\alpha\beta}$, $\alpha \neq \beta$ are zero:

$$v(t, x, a) = 0 \ , \quad u_y(t, x, a) = 0 \ .$$

These conditions may also arise at a plane of symmetry.

Inflow conditions

Because the momentum equations resemble convection-diffusion equations for the velocities, we use the same inflow BCs as in Section 3.3.1, *i.e.* Dirichlet conditions.

Outflow conditions

At the outflow boundary, apart from the pressure there is usually not enough physical information available on which to base a sufficient number of BCs. Because of the resemblance of Equation (3.5) to the convection-diffusion equation, it can be shown directly by applying singular perturbation analysis that the 'wrong' information generated by the artificial outflow BC propagates upstream by a distance of $\mathcal{O}(\mathrm{Re}^{-1})$. So for highly turbulent flow, $\mathrm{Re} \gg 1$, the outflow condition does not have significantly influence on the solution, but for laminar low Reynolds flow this is not the case. In order that the problem is well-posed it is advisable to choose a (homogeneous) Neumann outflow BC for the stationary case, see Wesseling [38].

Compatibility condition

At each part of the boundary $\partial\Omega$ one has to prescribe exactly one boundary condition. If for each part a normal velocity $u^\alpha n^\alpha(t, \boldsymbol{x})$ is prescribed with $\boldsymbol{n}$ the outer normal, then it follows from Equation (3.4) and Gauß' theorem that the compatibility condition must be satisfied:

$$\int_{\partial\Omega} u^{\mathrm{nor}}(t, \boldsymbol{x}) dS = 0 \ .$$

One can show that in order for Equations (3.5) to be well-posed, the normal velocity component of the prescribed initial velocity field $\boldsymbol{u}_0(\boldsymbol{x})$ and the prescribed normal velocity component must match at $t = 0$,

$$u_0^\alpha(\boldsymbol{x}) n^\alpha = u_\alpha^{\mathrm{nor}}(0, \boldsymbol{x}) n^\alpha .$$

on parts of $\partial\Omega$ where the normal velocity is prescribed.

3.4.2 Discretization on a colocated grid

Colocated and staggered grids

There are two ways to arrange the unknowns on the grid: colocated arrangement and staggered arrangement, see Figure 3.2. When all discrete unknowns are located in the cell-centres,

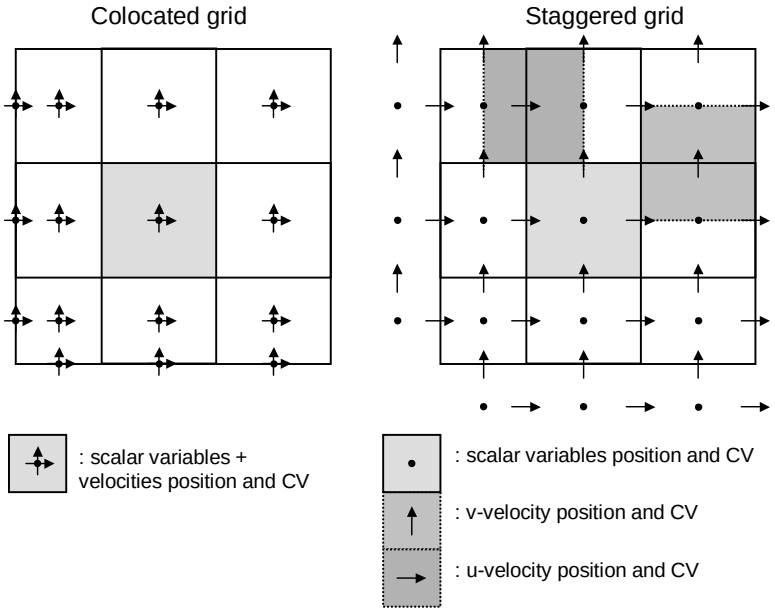

Figure 3.2: A twodimensional domain with a colocated grid arrangement (left) and a staggered grid arrangement (right). (CV: control volume or cell.)

the grid is called a colocated grid (colocate = to locate together). When the pressure is located in the cell-centres and the velocity components are located at the cell face centres, the grid is called staggered.

We will restrict ourselves to a colocated grid and to incompressible flow with constant density $\rho = 1$.

Discretization of the continuity equation

FV discretization of Equation (3.4) gives

$$\int_{\Omega_j} u^\alpha_{,\alpha} d\Omega \approx h_2 u^1 |_{\boldsymbol{j}-\boldsymbol{e}_1}^{\boldsymbol{j}+\boldsymbol{e}_1} + h_1 u^2 |_{\boldsymbol{j}-\boldsymbol{e}_2}^{\boldsymbol{j}+\boldsymbol{e}_2} = 0 \ . \tag{3.6}$$

Because the velocity components are situated in the cell centres, the cell face values in the above equation need to be interpolated, *e.g.* by applying a CDS,

$$h_2 u^1 |_{\boldsymbol{j}-2\boldsymbol{e}_1}^{\boldsymbol{j}+2\boldsymbol{e}_1} + h_1 u^1 |_{\boldsymbol{j}-2\boldsymbol{e}_2}^{\boldsymbol{j}+2\boldsymbol{e}_2} = 0 \ .$$

Discretization of the momentum equations

For simplicity we take $\boldsymbol{f} = 0$ in (3.5), so we have to discretize

$$\frac{\partial u^\alpha}{\partial t} + \bar{F}^{\alpha\beta}_{,\beta} + p_{,\alpha} = 0 \ , \quad \bar{F}^{\alpha\beta} = u^\alpha u^\beta - \sigma^{\alpha\beta} \ .$$

FV discretization yields

$$\int_{\Omega_j} \{\frac{\partial u^\alpha}{\partial t} + \bar{F}^{\alpha\beta}_{,\beta} + p_{,\alpha}\} d\Omega \approx h_1 h_2 \frac{\partial u^\alpha_{\boldsymbol{j}}}{\partial t} + h_2 \bar{F}^{\alpha 1} |_{\boldsymbol{j}-\boldsymbol{e}_1}^{\boldsymbol{j}+\boldsymbol{e}_1} + h_2 \bar{F}^{\alpha 2} |_{\boldsymbol{j}-\boldsymbol{e}_2}^{\boldsymbol{j}+\boldsymbol{e}_2} + h_\gamma p |_{\boldsymbol{j}-\boldsymbol{e}_\alpha}^{\boldsymbol{j}+\boldsymbol{e}_\alpha} \ , \tag{3.7}$$

with $\gamma \neq \alpha$. Just as for the discretization of the continuity equation the cell face values have to be interpolated between cell centre values. For the pressure this is done as

$$p_{\boldsymbol{j}+\boldsymbol{e}_\alpha} \approx \frac{1}{2}(p_{\boldsymbol{j}} + p_{\boldsymbol{j}+2\boldsymbol{e}_\alpha}) \ .$$

When $\alpha = \beta$, the deviatoric viscous stress is approximated by

$$(\sigma^{\alpha\alpha})_{\boldsymbol{j}+\boldsymbol{e}_1} = (2\mathrm{Re}^{-1}u^\alpha_{,\alpha})_{\boldsymbol{j}+\boldsymbol{e}_1} \approx 2\mathrm{Re}^{-1}_{\boldsymbol{j}+\boldsymbol{e}_1}(u^\alpha_{\boldsymbol{j}+2\boldsymbol{e}_1} - u^\alpha_{\boldsymbol{j}})/h_\alpha \ .$$

For the inertia terms one can use the CDS,

$$(u^\alpha u^\beta)_{\boldsymbol{j}+\boldsymbol{e}_\beta} \approx \frac{1}{2}[(u^\alpha u^\beta)_{\boldsymbol{j}} + (u^\alpha u^\beta)_{\boldsymbol{j}+2\boldsymbol{e}_\beta}] \ .$$

Spurious checkerboard pattern

Using the CDS to approximate the terms in Equations (3.6) and (3.7), causes a spurious checkerboard pattern. Assuming that Re = constant and neglecting the boundary conditions, one can show that

$$u^\alpha_{\boldsymbol{j}} = (-1)^{j_1+j_2}\exp\{-\frac{12}{\mathrm{Re}}(h_1^{-2} + h_2^{-2})t\} \ , \quad p = (-1)^{j_1+j_2} \ ,$$

is a solution of the discretised Navier-Stokes equations (3.7). This shows that if $\mathrm{Re} \gg 1$ the checkerboard patron is damped slowly for the velocity, but not at all for the pressure. A way to avoid checkerboard patterns is to use a staggered grid instead of a colocated grid. If we do not want this, another option is to use the pressure-weighted interpolation method, which will be discussed next.

Pressure weighted interpolation method

The pressure weighted interpolation (PWI) method (or Rhie & Chow interpolation after its inventors) is to approximate the cell face velocities as follows,

$$u^\alpha_{\boldsymbol{j}+\boldsymbol{e}_\alpha} = \frac{1}{2}(u^\alpha_{\boldsymbol{j}} + u^\alpha_{\boldsymbol{j}+2\boldsymbol{e}_\alpha}) + (\frac{h_\beta}{4a^\alpha}\Delta^\alpha p)|^{\boldsymbol{j}+2\boldsymbol{e}_\alpha}_{\boldsymbol{j}} \quad \text{(no summation)} \ , \tag{3.8}$$

where $\Delta^\alpha p_{\boldsymbol{j}} = p_{\boldsymbol{j}+2\boldsymbol{e}_\alpha} - 2p_{\boldsymbol{j}} + p_{\boldsymbol{j}-2\boldsymbol{e}_\alpha}$, $\beta \neq \alpha$. See Appendix A.1 for a derivation. This discretization method is $\mathcal{O}(h_1^2 + h_2^2)$ accurate. The second term in the RHS can be interpreted as a regularizing term that excludes spurious patterns.

Substitution of (3.8) in (3.6) result in the following discretization of $u^{\ \alpha}_{,\alpha}$ with the PWI method:

$$\begin{aligned}
&+ h_2 u^1|^{\boldsymbol{j}+2\boldsymbol{e}_1}_{\boldsymbol{j}-2\boldsymbol{e}_1} + h_1 u^2|^{\boldsymbol{j}+2\boldsymbol{e}_2}_{\boldsymbol{j}-2\boldsymbol{e}_2} \\
&+ h_2^2[(\frac{1}{2a^1}\Delta^1 p)_{\boldsymbol{j}+2\boldsymbol{e}_1} - (\frac{1}{a^1}\Delta^1 p)_{\boldsymbol{j}} + (\frac{1}{2a^1}\Delta^1 p)_{\boldsymbol{j}-2\boldsymbol{e}_1}] \\
&+ h_1^2[(\frac{1}{2a^2}\Delta^2 p)_{\boldsymbol{j}+2\boldsymbol{e}_2} - (\frac{1}{a^2}\Delta^2 p)_{\boldsymbol{j}} + (\frac{1}{2a^2}\Delta^2 p)_{\boldsymbol{j}-2\boldsymbol{e}_2}] = 0 \ .
\end{aligned} \tag{3.9}$$

See Appendix A.2 for a derivation of this equation.

Boundary conditions and pressure

A disadvantage of the PWI method is that it requires some further specification of conditions at boundaries, beyond what is given for the differential equations. Let $(j = 1, j_2)$, so that the 'west' face of $\Omega_{\boldsymbol{j}}$ is part of the boundary. When the PWI method (3.8) is applied in (3.6) and (3.7), $p_{\boldsymbol{j}-2\boldsymbol{e}_1}$ occurs, referring to a grid point outside the grid G. This value is approximated by extrapolation from the interior,

$$p_{\boldsymbol{j}-2\boldsymbol{e}_1} = 2p_{\boldsymbol{j}} - p_{\boldsymbol{j}+2\boldsymbol{e}_1} \ ,$$

which is artificial since the differential equations are not accompanied by a boundary condition for the pressure.

When the pressure distribution is required at the boundaries this can be obtained by extrapolation from the interior, for example by

$$p_{\boldsymbol{j}-\boldsymbol{e}_1} = \frac{3}{2}p_{\boldsymbol{j}} - \frac{1}{2}p_{\boldsymbol{j}+2\boldsymbol{e}_1} \ .$$

For more details we refer to Wesseling [38].

Summary of equations

After spatial discretization, the following system of ordinary differential equations is obtained,

$$\begin{aligned} \frac{du}{dt} + N(\boldsymbol{u}) + G\boldsymbol{p} &= \boldsymbol{b}_1(t) \ , \\ D\boldsymbol{u} + C\boldsymbol{p} &= \boldsymbol{b}_2(t) \ , \end{aligned} \tag{3.10}$$

where N is a nonlinear algebraic operator arising from the discretization of the inertia and viscous terms, G is a linear algebraic operator representing the discretization of the pressure gradient, D and C are linear algebraic operators corresponding to the velocity terms and pressure terms, respectively in the PWI discretization of the continuity equation. The terms $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ are source terms, arising from the boundary conditions and body forces. The system (3.10) contains both differential and algebraic systems and is therefore called a differential-algebraic system (DAS).

3.5 Connection with X-stream

In X-stream the governing equations are discretized with the FV method on a colocated grid. The grid is structured and boundary-fitted, see for details Ferziger & Perić [6] or Wesseling [38]. Defect correction is applied: upwind discretization is blended with central discretization. The PWI method is used to resolve the checkerboard problem for solving the (nondimensionless) incompressible Navier-Stokes equations.

4

Iterative solution methods

4.1 Introduction

The algebraic systems arising from FV discretization are generally very large and sparse, because many grid points are required for accuracy. Therefore iterative methods are more efficient and demand far less storage than direct methods, especially for the three-dimensional case.

In this chapter we restrict ourselves entirely to iterative methods. In Section 4.2 iterative methods for solving linear equations are discussed, without going into details. In Section 4.3 iterative solution methods for solving the non-linear system for the stationary incompressible Navier-Stokes equations will be discussed. In Section 4.4 the connection with X-stream is given.

4.2 Methods for solving linear equation systems

There exists various methods for solving linear algebraic systems, each having their own advantages and disadvantages. Is this section, first the principle of basic iterative methods is discussed in Subsection 4.2.1. Next, a class of efficient solution methods called Krylov subspace methods in Subsection 4.2.2. These methods combined with deflation are discussed in Subsection 4.2.3. In Subsection 4.2.4 the SIP method is treated.

4.2.1 Basic iterative methods

Stationary iterative methods

Consider the linear algebraic $n \times n$ system

$$A\boldsymbol{y} = \boldsymbol{b} \ . \tag{4.1}$$

Let a new iterand $\boldsymbol{y}^{(k)}$ be computed by

$$\boldsymbol{y}^{(k)} = B\boldsymbol{y}^{(k-1)} + \boldsymbol{c} \ , \tag{4.2}$$

where the matrix B and the vector $\boldsymbol{c}$ remain to be chosen. We want that the iterative method is convergent, *i.e.* $\lim_{k\to\infty} \boldsymbol{y}^{(k)} = \boldsymbol{y}$, so it is obviously necessary that $\boldsymbol{y}$ is a stationary point of the iteration process (4.2). Therefore we have $\boldsymbol{y} = B\boldsymbol{y} + \boldsymbol{c}$, and hence $\boldsymbol{c} = (I - B)A^{-1}\boldsymbol{b}$. Rewriting (4.2) yields

$$M\boldsymbol{y}^{(k)} = N\boldsymbol{y}^{(k-1)} + \boldsymbol{b} \ , \tag{4.3}$$

with $M \equiv A(I-B)^{-1}$ and $N \equiv MB$, so $M-N=A$, *i.e.* $M-N$ is a splitting of A. Equation (4.3) can be rewritten as

$$M\delta\boldsymbol{y} = \boldsymbol{b} - A\boldsymbol{y}^{(k-1)} \ , \quad \boldsymbol{y}^{(k)} = \boldsymbol{y}^{(k-1)} + \delta\boldsymbol{y} \ . \tag{4.4}$$

Application of under- or overrelaxation with relaxation parameter α means that the RHS of the first equation in (4.4) is multiplied by α, so

$$M\delta\boldsymbol{y} = \alpha(\boldsymbol{b} - A\boldsymbol{y}^{(k-1)}) \ , \quad \boldsymbol{y}^{(k)} = \boldsymbol{y}^{(k-1)} + \delta\boldsymbol{y} \ . \tag{4.5}$$

A method to solve (4.1) of the form (4.3), (4.4) or (4.5) is called a basic iterative method (BIM).

Convergence and stop criterion

Let I denote the $n \times n$ identity matrix, then we have

$$B = M^{-1}N = I - M^{-1}A \ ,$$

and the global truncation error $\boldsymbol{e}^{(k)} \equiv \boldsymbol{y} - \boldsymbol{y}^{(k)}$ statisfies

$$\boldsymbol{e}^{(k)} = B\boldsymbol{e}^{(k-1)} \ ,$$

so convergence of a BIM is governed by B, which is called the iteration matrix. We have

$$\|\boldsymbol{e}^{(k)}\| \leq \|B^k\|\|\boldsymbol{e}^{(0)}\| \ ,$$

where $\|\cdot\|$ denotes a certain norm. Let $\sigma(B)$ denote the eigenvalues of B and define the spectral radius of B by $\rho(B) \equiv \max\{|\sigma(B)|\}$. One can show that the spectral radius satisfies

$$\rho(B) = \lim_{k\to\infty} \|B^k\|^{1/k} \ ,$$

and hence we see that we have convergence if and only if $\rho(B) < 1$. The smaller $\rho(B)$, the faster the convergence.

When the difference between two successive iterates is used as a stop criterion one can show that this does not give a good indication of the precision achieved. It is better to derive a stop criterion from the k-th residual, defined as $\boldsymbol{r}^{(k)} \equiv \boldsymbol{b} - A\boldsymbol{y}^{(k)}$. For example, a suitable stop criterion could be

$$\|\boldsymbol{r}^{(k)}\|/\|\boldsymbol{b}\| < \varepsilon \ ,$$

for a certain fixed tolerance $\varepsilon > 0$.

Convergence for the case that A is a special matrix

If the matrix A satisfies certain conditions one can prove that the BIM (4.3) is convergent. The first condition is that the splitting

$$A = M - N$$

is a so-called regular splitting, which is by definition the case if $M^{-1} \geq 0$ and $N \geq 0$. We will call the splitting convergent if (4.3) converges. The other condition is that A is a so-called Stieltjes matrix or M-matrix, which is defined as follows. The $n \times n$ matrix A is called an M-matrix if $a_{ij} \leq 0$, $i \neq j$, $i, j = 1, 2, \ldots, n$, A non-singular and $A^{-1} \geq 0$. Now one can prove that the BIM (4.3) is convergent if the splitting $A = M - N$ is regular and A is a M-matrix.

Often, it is not so easy to verify that A is a M-matrix, because the condition $A^{-1} \geq 0$ is hard to check. Fortunately, one can prove the M-matrix property indirectly. If A is a so-called K-matrix which is irreducible one can prove that A is a M-matrix. The definition of a K-matrix is as follows. A matrix A is called a K-matrix if

$$a_{ii} > 0, \;\; i = 1, 2, \ldots, n \;,$$
$$a_{ij} \leq 0, \;\; i, j = 1, 2, \ldots, n, \quad j \neq i \;,$$

and

$$\sum_j a_{ij} \geq 0, \quad i = 1, 2, \ldots, n \;,$$

with strict inequality for at least one i. A matrix is called irreducible if the corresponding system does not consist of subsystems that are independent of each other.

4.2.2 Krylov subspace methods

Define the computing work W to solve the linear system (4.1) as

$$W = \mathcal{O}(N^{\alpha}) \;,$$

with N the total number of equations and α a certain number. Now it can be shown that for BIMs, for discretizations of elliptic PDEs, in general $\alpha \approx 2$. This means that in general BIMs converge slowly, but fortunately BIMs can be accelerated. There are two ways to do this: multigrid acceleration and Krylov subspace acceleration. Multigrid methods bring α down to the ideal case $\alpha = 1$, but are in general more difficult to implement. For a survey on (geometric) multigrid methods see for example Wesseling [37]. Krylov subspace methods come close to $\alpha = 1$ and are much easier to implement. A detailed survey on Krylov subspace methods can be found in for example Golub & Van Loan [9], Saad [22] or Vuik [28] (in Dutch).

In this chapter we restrict ourselves entirely to Krylov methods.

Basic idea of Krylov methods

Multiplying (4.3) by M^{-1} gives

$$\boldsymbol{y}^{(k)} = M^{-1} N \boldsymbol{y}^{(k-1)} + M^{-1} \boldsymbol{b} \;, \tag{4.6}$$

and the exact solution $\boldsymbol{y}$ satisfies

$$\boldsymbol{y} = M^{-1} N \boldsymbol{y} + M^{-1} \boldsymbol{b} \;. \tag{4.7}$$

When we subtract (4.7) from (4.6) we can write

$$\boldsymbol{y}^{(k)} - \boldsymbol{y} = M^{-1} N (\boldsymbol{y}^{(k-1)} - \boldsymbol{y}) = (M^{-1} N)^k (\boldsymbol{y}^{(0)} - \boldsymbol{y}) = (I - M^{-1} A)^k (\boldsymbol{y}^{(0)} - \boldsymbol{y}) \;. \tag{4.8}$$

Define the polynomal of degree k by

$$p_k(X) = (I - X)^k \ , \tag{4.9}$$

where X is a $n \times n$ matrix. By this we see that $p_k(0) = I$, where '0' denotes the $n \times n$ matrix containing only zeros. Now we can write (4.8) as

$$\boldsymbol{y}^{(k)} = p_k(M^{-1}A)(\boldsymbol{y}^{(0)} - \boldsymbol{y}) + \boldsymbol{y} \ ,$$

which can be written as

$$\begin{aligned} \boldsymbol{y}^{(k)} &= \boldsymbol{y}^{(0)} + [I - p_k(M^{-1}A)]A^{-1}\boldsymbol{r}^{(0)} \ , \\ &= \boldsymbol{y}^{(0)} + [I - p_k(M^{-1}A)](M^{-1}A)^{-1}M^{-1}\boldsymbol{r}^{(0)} \ , \\ &= \boldsymbol{y}^{(0)} + q_{k-1}(M^{-1}A)M^{-1}\boldsymbol{r}^{(0)}, \end{aligned}$$

where the polynomal q_k is defined as

$$q_k(X) = [I - p_k(X)]X^{-1} \ .$$

Note that $q_k(0) = I$. So we can write the iterand $\boldsymbol{y}^{(k)}$ as

$$\begin{aligned} \boldsymbol{y}^{(k)} &= \boldsymbol{y}^{(0)} + q_{k-1}(M^{-1}A)M^{-1}\boldsymbol{r}^{(0)} \ , \\ &= \boldsymbol{y}^{(0)} + a_0 M^{-1}\boldsymbol{r}^{(0)} + a_1(M^{-1}A)(M^{-1}\boldsymbol{r}^{(0)}) + \dots \\ &\quad + a_{k-1}(M^{-1}A)^{k-1}(M^{-1}\boldsymbol{r}^{(0)}) \ , \end{aligned}$$

where $a_0, a_1, \dots, a_{k-1}$ are certain coeffients. So we see that

$$\boldsymbol{y}^{(k)} \in \boldsymbol{y}^{(0)} + \text{span}\{M^{-1}\boldsymbol{r}^{(0)}, M^{-1}A(M^{-1}\boldsymbol{r}^{(0)}), \dots, (M^{-1}A)^{k-1}(M^{-1}\boldsymbol{r}^{(0)})\} \ .$$

The space

$$K^{(k)}(A; \boldsymbol{r}^{(0)}) \equiv \text{span}\{\boldsymbol{r}^{(0)}, A\boldsymbol{r}^{(0)}, \dots, A^{k-1}\boldsymbol{r}^{(0)}\}$$

is called the Krylov space of dimension k, belonging to A and $\boldsymbol{r}^{(0)}$. For BIMs one has

$$\boldsymbol{y}^{(k)} \in \boldsymbol{y}^{(0)} + K^{(k)}(M^{-1}A; M^{-1}\boldsymbol{r}^{(0)}) \ .$$

Methods that look for optimal approximations to $\boldsymbol{y} - \boldsymbol{y}^{(0)}$ in $K^{(k)}(A; \boldsymbol{r}^{(0)})$ are called Krylov subspace methods.

Preconditioning

Consider the BIM (4.3). This can be rewritten as

$$\boldsymbol{y}^{(k)} = M^{-1}N\boldsymbol{y}^{(k-1)} + M^{-1}\boldsymbol{b} = B\boldsymbol{y}^{(k-1)} + M^{-1}\boldsymbol{b} \ ,$$

which is called a preconditioned Richardson's iteration. The Richardson's iteration can be viewed as a technique for solving the system

$$(I - B)\boldsymbol{y} = M^{-1}A \ ,$$

or equivalently by substituting B

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

This system, which has the same solution as the original system, is called a preconditioned system and M is called the (left) preconditioner. In literature the term preconditoner has a double meaning. Besides referring to the preconditioning matrix, the entire BIM is often called a preconditioner.

One requirement for M is that $M^{-1}\boldsymbol{x}$ is inexpensive to solve. Another requirement is that M^{-1} is spectrally equivalent to A^{-1}. The preconditioner has to be chosen in such a way that $M^{-1}A$ resembles more the identity matrix and therefore has a smaller condition number than A, that is, $\kappa(M^{-1}A) \equiv \lambda_{\max}/\lambda_{\min}$ is significantly smaller than $\kappa(A)$. Also the preconditioner has te be chosen such that the eigenvalue spectrum of $M^{-1}A$ is more clustered than the spectrum of A.

The preconditioned GCR method

For a general symmetric or nonsymmetric matrix A the generalized conjugate residual (GCR) Krylov subspace method can be used to solve the linear system $A\boldsymbol{y} = \boldsymbol{b}$. For the norm $\|\cdot\|$ we take the Euclidian norm, defined as

$$\|\boldsymbol{x}\|_2 \equiv \sqrt{(\boldsymbol{x},\boldsymbol{x})} \equiv (x_1^2 + \ldots + x_n^2)^{1/2} \ .$$

The preconditioned GCR (PGCR) method is given by the following algorithm, see for example Saad [22] or Wesseling [38].

Algorithm 4.1 (PGCR). Given a general symmetric or non-symmetric $n \times n$ matrix A, a vector $\boldsymbol{b}$, a preconditioner M, and an initial guess $\boldsymbol{y}^0$ ($A\boldsymbol{y}^{(0)} \approx \boldsymbol{b}$). This algorithm solves the linear system $A\boldsymbol{y} = \boldsymbol{b}$.

$\boldsymbol{r}^{(0)} = \boldsymbol{b} - A\boldsymbol{y}^{(0)}$;
for $k = 1, \ldots,$ convergence **do**
  Solve $M\boldsymbol{s}^{(k)} = \boldsymbol{r}^{(k-1)}$;
  $\boldsymbol{v}^{(k)} = A\boldsymbol{s}^{(k)}$;
  $[\boldsymbol{v}^{(k)}, \boldsymbol{s}^{(k)}] :=$ **orthonorm**$[\boldsymbol{v}^{(k)}, \boldsymbol{s}^{(k)}, \{\boldsymbol{v}^{(1)}, \ldots, \boldsymbol{v}^{(k-1)}\}, \{\boldsymbol{s}^{(1)}, \ldots, \boldsymbol{s}^{(k-1)}\}]$;
  $\beta = (\boldsymbol{r}^{(k-1)}, \boldsymbol{v}^{(k)})$;
  $\boldsymbol{y}^{(k)} = \boldsymbol{y}^{(k-1)} + \beta\boldsymbol{s}^{(k)}$;
  $\boldsymbol{r}^{(k)} = \boldsymbol{r}^{(k-1)} - \beta\boldsymbol{v}^{(k)}$;
end for
$\boldsymbol{y} \approx \boldsymbol{y}^{(k-1)}$;

The routine **orthonorm** refers to the orthonormalization process used. It is left open in the above algorithm, because there are several ways to do this. Three main orthonormalization methods can be distinguished: modified Gram-Schmidt (MGS), reorthogonalized classical Gram-Schmidt (RCGS) and Householder, see for example Vuik & Frank [29] or Vuik e.a [34]. The MGS method is most common used and for this method the routine **orthnonorm** becomes

function $[\boldsymbol{v}, \boldsymbol{s}] = \textbf{orthonorm}[\boldsymbol{v}, \boldsymbol{s}, \{\boldsymbol{v}^{(1)}, \ldots, \boldsymbol{v}^{(k-1)}\}, \{\boldsymbol{s}^{(1)}, \ldots, \boldsymbol{s}^{(k-1)}\}]$
  for $j = 1, \ldots, k-1$ **do**
    $\alpha = (\boldsymbol{v}, \boldsymbol{v}^{(j)})$;
    $\boldsymbol{v} := \boldsymbol{v} - \alpha \boldsymbol{v}^{(j)}$;
    $\boldsymbol{s} := \boldsymbol{s} - \alpha \boldsymbol{s}^{(j)}$;
  end for
  $\boldsymbol{v} := \boldsymbol{v} / \|\boldsymbol{v}^{(k)}\|_2$;
  $\boldsymbol{s} := \boldsymbol{s} / \|\boldsymbol{s}^{(k)}\|_2$;

Computational work

The orthonormalization process used in Algorithm 4.1 could make the PGCR algorithm expensive. First of all, the vectors $\boldsymbol{v}^{(1)}, \ldots, \boldsymbol{v}^{(k)}$ and $\boldsymbol{s}^{(1)}, \ldots, \boldsymbol{s}^{(k)}$ need to be stored in memory and with every iteration the memory usage increases. Secondly, the orthonormalization work of the vectors increases with every iteration.

The storage and computing work can be reduced by applying the following techniques:

- Restarting: stop the PGCR algorithm after k_{res} iterations and remove $\boldsymbol{v}^{(1)}, \ldots, \boldsymbol{v}^{(k_{\text{res}})}$ and $\boldsymbol{s}^{(1)}, \ldots, \boldsymbol{s}^{(k_{\text{res}})}$.

- Truncation: allow only k_{trunc} vectors $\boldsymbol{v}^j$ and $\boldsymbol{s}^j$, and replace an old vector by a new one.

When restarting or truncation is applied the optimality property of the PGCR algorithm gets lost.

Robustness and convergence

Inspection of the PGCR algorithm in combination with MGS orthonormalization shows that break-down can occur if $\|\boldsymbol{v}^{(k)}\|_2 = 0$ or $\|\boldsymbol{s}^{(k)}\|_2 = 0$. This can happen if $\boldsymbol{r}^{(k-1)} = \boldsymbol{r}^{(k-2)}$, which is unlikely to happen in practice, or if the exact solution is reached, *i.e.* $\boldsymbol{y}^{(k-1)} = \boldsymbol{y}$. This means that the (P)GCR method is very robust.

Concerning convergence of Krylov subspace methods one can show that convergence is monotone,

$$\|\boldsymbol{r}^{(k)}\| \leq \|\boldsymbol{r}^{(k-1)}\| \ .$$

The preconditioned CG method

In the special case that A is a symmetric positive definite (SPD) matrix, using the so-called preconditioned conjugate gradient (PCG) method is less expensive than the PGCR method. Compared to the PGCR method much literature on the PCG method can be found, see for example Golub & Van Loan [9], Saad [22] or Wesseling [38]. The PCG method is given by the following algorithm.

Algorithm 4.2 (PCG). Given a SPD $n \times n$ matrix A, a vector $\boldsymbol{b}$, a SPD preconditioner M, and an initial guess $\boldsymbol{y}^{(0)}$ ($A\boldsymbol{y}^{(0)} \approx \boldsymbol{b}$), this algorithm solves the linear system $A\boldsymbol{y} = \boldsymbol{b}$.

$\boldsymbol{r}^{(0)} = \boldsymbol{b} - A\boldsymbol{y}^{(0)}$;
Solve $M\boldsymbol{q}_1 = \boldsymbol{r}^{(0)}$;
$\boldsymbol{s}^{(0)} = \boldsymbol{q}_1$;
for $k = 1, \ldots,$ convergence **do**
  $\alpha = (\boldsymbol{r}^{(k-1)}, \boldsymbol{q}_1)/(A\boldsymbol{s}^{(k-1)}, \boldsymbol{s}^{(k-1)})$;
  $\boldsymbol{y}^{(k)} = \boldsymbol{y}^{(k-1)} + \alpha\boldsymbol{s}^{(k-1)}$;
  $\boldsymbol{r}^{(k)} = \boldsymbol{r}^{(k-1)} - \alpha A\boldsymbol{s}^{(k-1)}$;
  Solve $M\boldsymbol{q}_2 = \boldsymbol{r}^{(k)}$;
  $\beta = (\boldsymbol{r}^{(k)}, \boldsymbol{q}_2)/(\boldsymbol{r}^{(k-1)}, \boldsymbol{q}_1)$;
  $\boldsymbol{s}^{(k)} = \boldsymbol{q}_2 + \beta\boldsymbol{s}^{(k-1)}$;
  $\boldsymbol{q}_1 := \boldsymbol{q}_2$;
end for
$\boldsymbol{y} \approx \boldsymbol{y}^{(k-1)}$;

Computational work, robustness and convergence

Because no orthonormalization is needed, an iteration with the PCG method is far less expensive compared to the PGCR method, and less memory is required. Break-down does not occur in computing α and β, because A and M are SPD. Therefore the PCG algorithm is robust. Concerning convergence, one can show that

$$\|\boldsymbol{y}^{(k)} - \boldsymbol{y}\|_{M^{-1}A} \leq 2\left[\frac{\sqrt{\kappa(M^{-1}A)} - 1}{\sqrt{\kappa(M^{-1}A)} + 1}\right]^k \|\boldsymbol{y}^{(0)} - \boldsymbol{y}\|_{M^{-1}A} \ ,$$

where $\|\boldsymbol{x}\|_{M^{-1}A} \equiv \sqrt{(M^{-1}A\boldsymbol{x}, \boldsymbol{x})}$.

4.2.3 Deflated Krylov subspace methods

The deflation method is originally proposed in Nicolaides [17]. In the late nineties this method had a rebirth as an application for solving elliptic layered problems with extreme contrasts in the coefficients, see for example Vuik *et al.* [30] or Vuik *et al.* [36]. The matrix of the system arising from discretization was ill-conditioned because of the large jumps in the coefficients. It was observed that solving the system with a conventional (preconditioned) Krylov subspace method gave erratic convergence behaviour. Removing the smallest eigenvalues of the matrix by the deflation technique solved this problem.

Now the basic idea of deflation will be discussed briefly.

Basic idea of deflation

Consider the linear algebraic $n \times n$ system

$$A\boldsymbol{y} = \boldsymbol{b} \ , \tag{4.10}$$

where A is an general matrix. Let P and Q be given by

$$P \equiv I - AZ(Y^TAZ)^{-1}Y^T \ ,$$
$$Q \equiv I - Z(Y^TAZ)^{-1}Y^TA \ ,$$

where Z and Y are suitable matrices. Note that for a symmetric matrix A and $Y = Z$, the Q is equal to P^T. The matrices P and Q have the following properties,

- P and Q are projectors, thus $P^2 = P$ and $Q^2 = Q$,
- $PAZ = Y^T P = 0$, $Y^T AQ = QZ = 0$,
- $PA = AQ$.

To solve the system (4.10) using deflation, note that $\boldsymbol{y}$ can be written as

$$\boldsymbol{y} = (I - Q)\boldsymbol{y} + Q\boldsymbol{y}$$

and that $(I - Q)\boldsymbol{y} = Z(Y^T AZ)^{-1}Y^T A\boldsymbol{y} = Z(Y^T AZ)^{-1}Y^T \boldsymbol{b}$ can be computed immediately. In light of the identity $AQ = PA$ we can solve the deflated system

$$PA\boldsymbol{y} = P\boldsymbol{b} \ .$$

Because $PAZ = 0$ and therefore $\mathrm{Ker}(PA)$ does not contain only the zero vector, this system does not have an unique solution. We denote this non-unique solution by $\tilde{\boldsymbol{y}}$, resulting in

$$PA\tilde{\boldsymbol{y}} = P\boldsymbol{b} \ . \tag{4.11}$$

Although this system does not have an unique solution it can be shown that $Q\,\tilde{\boldsymbol{y}}$ is unique and is equal to $Q\boldsymbol{y}$ (see for a proof Vermolen & Vuik [26] for the case that A is SPD).

Deflation can be combined with preconditioning. Suppose M is a suitable preconditioner of A, then (4.11) can be replaced by: solve $\tilde{\boldsymbol{y}}$ from

$$M^{-1}PA\tilde{\boldsymbol{y}} = M^{-1}P\boldsymbol{b}$$

and form $Q\tilde{\boldsymbol{y}}$, or solve $\tilde{\boldsymbol{u}}$ from

$$PAM^{-1}\tilde{\boldsymbol{u}} = P\boldsymbol{b} \ ,$$

and form $QM^{-1}\tilde{\boldsymbol{u}}$. Both systems can be solved by a Krylov subspace method.

Note that when the deflated system (4.11) is solved with a Krylov subspace method one always has to take $\tilde{\boldsymbol{y}}^{(0)} = 0$. Also note that when an initial guess $\boldsymbol{y}^{(0)} \neq \boldsymbol{0}$ is chosen, one has to solve the deflated system

$$PA\tilde{\boldsymbol{v}} = P\boldsymbol{f} \ ,$$

where $\boldsymbol{v} \equiv \boldsymbol{y} - \boldsymbol{y}^{(0)}$ and $\boldsymbol{f} \equiv \boldsymbol{b} - A\boldsymbol{y}^{(0)}$ in order that system (4.10) is solved rightly.

We will restrict ourselves to the deflated PGCR algorithm (DPGCR) for the case that A and M are general matrices, and to the deflated PCG algorithm (DPCG) for the case A and M are SPD. The following algorithm is the deflated variant of Algorithm (4.1).

Algorithm 4.3 (DPGCR). Given a general symmetric or non-symmetric $n \times n$ matrix A, a vector $\boldsymbol{b}$, a preconditioner M, projectors P and Q, and an initial guess $\boldsymbol{y}^{(0)}$ ($A\boldsymbol{y}^{(0)} \approx \boldsymbol{b}$). This algorithm solves the linear system $A\boldsymbol{y} = \boldsymbol{b}$.

$\boldsymbol{r}^{(0)} = \boldsymbol{b} - A\boldsymbol{y}^{(0)}$
$\tilde{\boldsymbol{y}}^{(0)} = \boldsymbol{0}$;
$\tilde{\boldsymbol{r}}^{(0)} = P\boldsymbol{r}^{(0)}$;
for $k = 1, \ldots,$ convergence **do**

$\boldsymbol{s}^{(k)} = M^{-1}\tilde{\boldsymbol{r}}^{(k-1)}$;
$\boldsymbol{v}^{(k)} = PA\boldsymbol{s}^{(k)}$
$[\boldsymbol{v}^{(k)}, \boldsymbol{s}^{(k)}] := \mathbf{orthonorm}[\boldsymbol{v}^{(k)}, \boldsymbol{s}^{(k)}, \{\boldsymbol{v}^{(1)}, \ldots, \boldsymbol{v}^{(k-1)}\}, \{\boldsymbol{s}^{(1)}, \ldots, \boldsymbol{s}^{(k-1)}\}]$;
$\beta = (\tilde{\boldsymbol{r}}^{(k-1)}, \boldsymbol{v}^{(k)})$;
$\tilde{\boldsymbol{y}}^{(k)} = \tilde{\boldsymbol{y}}^{(k-1)} + \beta\boldsymbol{s}^{(k)}$;
$\tilde{\boldsymbol{r}}^{(k)} = \tilde{\boldsymbol{r}}^{(k-1)} - \beta\boldsymbol{v}^{(k)}$;
end for
$\boldsymbol{y} \approx (I - Q)A^{-1}\boldsymbol{r}^{(0)} + Q\tilde{\boldsymbol{y}}^{(k-1)} + \boldsymbol{y}^{(0)}$;

The deflated variant of Algorithm (4.2) is given by the following algorithm.

Algorithm 4.4 (DPCG). Given a general symmetric or non-symmetric $n \times n$ matrix A, a vector $\boldsymbol{b}$, a preconditioner M, projectors P and Q for the case $Y = Z$, and an initial guess $\boldsymbol{y}^{(0)}$ ($A\boldsymbol{y}^{(0)} \approx \boldsymbol{b}$). This algorithm solves the linear system $A\boldsymbol{y} = \boldsymbol{b}$.

$\boldsymbol{r}^{(0)} = \boldsymbol{b} - A\boldsymbol{y}^{(0)}$;
$\tilde{\boldsymbol{y}}^{(0)} = \mathbf{0}$;
$\tilde{\boldsymbol{r}}^{(0)} = P\boldsymbol{r}^{(0)}$;
$\boldsymbol{p}^{(0)} = \boldsymbol{s}^{(0)} = M^{-1}\tilde{\boldsymbol{r}}^{(0)}$;
for $k = 1, \ldots,$ convergence **do**
$\alpha = (\tilde{\boldsymbol{r}}^{(k-1)}, \boldsymbol{s}^{(k-1)})/(\boldsymbol{p}^{(k-1)}, PA\boldsymbol{p}^{(k-1)})$;
$\tilde{\boldsymbol{y}}^{(k)} = \tilde{\boldsymbol{y}}^{(k-1)} + \alpha\boldsymbol{p}^{(k-1)}$;
$\tilde{\boldsymbol{r}}^{(k)} = \tilde{\boldsymbol{r}}^{(k-1)} - \alpha PA\boldsymbol{p}^{(k-1)}$;
$\boldsymbol{s}^{(k)} = M^{-1}\tilde{\boldsymbol{r}}^{(k)}$;
$\beta = (\tilde{r}^{(k)}, \boldsymbol{s}^{(k)})/(\tilde{\boldsymbol{r}}^{(k-1)}, \boldsymbol{s}^{(k-1)})$;
$\boldsymbol{p}^{(k)} = \boldsymbol{s}^{(k)} + \beta\boldsymbol{p}^{(k-1)}$
end
$\boldsymbol{y} \approx (I - Q)A^{-1}\boldsymbol{r}^{(0)} + Q\tilde{\boldsymbol{y}}^{(k-1)} + \boldsymbol{y}^{(0)}$;

Choosing the subspaces Z and Y

There are various possibilities to choose the subspaces Y and Z. In general application Y is often chosen equal to Z. The columns of Z are usually called deflation vectors. There are various ways to choose these vectors. For example approximate eigenvectors can be chosen. We will not go into further details, but refer to for example Vuik *et al.* [30] and Vuik *et al.* [36].

4.2.4 SIP

The Strongly Implicit Procedure (SIP) is described in for example Ferziger & Perić [6] and Kim & Lee [12]. This method is especially designed for algebraic equations that are discretizations of PDEs and does not apply to generic systems of equations.

The SIP method is based on the splitting, $A = M - N$, with M being an incomplete LU (ILU) factorization of A. This means that we want to construct a lower triangular matrix L_{I} and an upper triangular matrix U_{I} with $\mathrm{diag}(U_{\mathrm{I}}) = I$, such that

$$M = L_{\mathrm{I}}U_{\mathrm{I}} \approx A \ ,$$

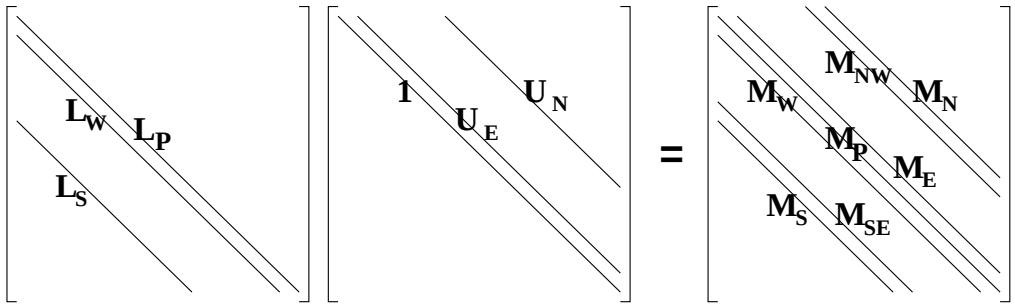

Figure 4.1: Incomplete LU factorization for the SIP method.

so the nonzero pattern of $L_{\mathrm{I}}U_{\mathrm{I}}$ corresponds to the nonzero pattern of A. Therefore we have

$$M = L_{\mathrm{I}}U_{\mathrm{I}} = A - N \ ,$$

with N the remainder matrix. Substituting $M = L_{\mathrm{I}}U_{\mathrm{I}}$ and N in the BIM (4.3) yields the following iterative method,

$$L_{\mathrm{I}}U_{\mathrm{I}}\boldsymbol{y}^{(k)} = N\boldsymbol{y}^{(k-1)} + \boldsymbol{b} \ , \tag{4.12}$$

or, equivalently,

for $k = 1, \ldots$ convergence **do**
  $\boldsymbol{r}^{(k-1)} = \boldsymbol{b} - A\boldsymbol{y}^{(k-1)}$;
  Solve $L_{\mathrm{I}}\boldsymbol{q}^{(k)} = \boldsymbol{r}^{(k-1)}$;
  Solve $U_{\mathrm{I}}\delta\boldsymbol{y} = \boldsymbol{q}^{(k)}$;
  $\boldsymbol{y}^{(k)} = \boldsymbol{y}^{(k-1)} + \delta\boldsymbol{y}$;
end for

In order that (4.12) is convergent, A must be an M-matrix or an irreducible K-matrix and the splitting must be regular, *i.e.* $(L_{\mathrm{I}}U_{\mathrm{I}})^{-1} \geq 0$ and $N \geq 0$.

Now the SIP method will be derived for a two-dimensional uniform grid, as defined in Section 3.2, and a five-point computing molecule. The ILU factorisation is in the form as in Figure 4.1 and a row of M corresponding to the (i,j)-the grid point is given by

$$\begin{aligned}
M_{\mathrm{S}}^{i,j} &= L_{\mathrm{S}}^{i,j} \ , \\
M_{\mathrm{SE}}^{i,j} &= L_{\mathrm{S}}^{i,j}U_{\mathrm{E}}^{i,j-1} \ , \\
M_{\mathrm{W}}^{i,j} &= L_{\mathrm{W}}^{i,j} \ , \\
M_{\mathrm{P}}^{i,j} &= L_{\mathrm{S}}^{i,j}U_{\mathrm{N}}^{i,j-1} + L_{\mathrm{W}}^{i,j}U_{\mathrm{E}}^{i-1,j} + L_{\mathrm{P}}^{i,j} \ , \\
M_{\mathrm{E}}^{i,j} &= L_{\mathrm{P}}^{i,j}U_{\mathrm{E}}^{i,j} \ , \\
M_{\mathrm{NW}}^{i,j} &= L_{\mathrm{W}}^{i,j}U_{\mathrm{N}}^{i-1,j} \ , \\
M_{\mathrm{N}}^{i,j} &= L_{\mathrm{P}}^{i,j}U_{\mathrm{N}}^{i,j} \ .
\end{aligned} \tag{4.13}$$

The (i,j)-the component of $N\boldsymbol{y}$ is

$$\begin{aligned}
(N\boldsymbol{y})_{i,j} = {} & N_{\mathrm{P}}^{i,j}y_{i,j} + N_{\mathrm{S}}^{i,j}y_{i,j-1} + N_{\mathrm{W}}^{i,j}y_{i-1,j} + N_{\mathrm{E}}^{i,j}y_{i+1,j} + N_{\mathrm{N}}^{i,j}y_{i,j+1} \\
& M_{\mathrm{SE}}^{i,j}y_{i+1,j-1} + M_{\mathrm{NW}}^{i,j}y_{i-1,j+1} \ .
\end{aligned} \tag{4.14}$$

Now $y_{i+1,j-1}$ and $y_{i-1,j+1}$ will be approximated using Taylor series expansion. For $y_{i+1,j-1}$ this yields

$$\begin{aligned} y_{i+1,j-1} &\approx y_{i,j} + h_1 \frac{dy}{dx_1} - h_2 \frac{dy}{dx_2} \ , \\ &\approx y_{i,j} + h_1 \frac{y_{i+1,j} - y_{i,j}}{h_1} - h_2 \frac{y_{i,j} - y_{i,j-1}}{h_2} \ , \\ &= y_{i+1,j} + y_{i,j-1} - y_{i,j} \ , \end{aligned} \tag{4.15}$$

and analogous for $y_{i-1,j+1}$ this gives

$$y_{i-1,j+1} \approx y_{i,j+1} + y_{i-1,j} - y_{i,j} \ , \tag{4.16}$$

Multiplying the RHS of (4.15) and (4.16) with a parameter $0 < \alpha < 1$ results in the following approximations

$$\begin{aligned} y_{i+1,j-1} &\approx \alpha(y_{i+1,j} + y_{i,j-1} - y_{i,j}) \ , \\ y_{i-1,j+1} &\approx \alpha(y_{i,j+1} + y_{i-1,j} - y_{i,j}) \ , \end{aligned}$$

With this approximations (4.14) can be rewritten as

$$\begin{aligned} (N\boldsymbol{y})_{i,j} \approx\ & (N_{\mathrm{P}}^{i,j} - \alpha M_{\mathrm{SE}}^{i,j} - \alpha M_{\mathrm{NW}}^{i,j}) y_{i,j} + \\ & (N_{\mathrm{S}}^{i,j} + \alpha M_{\mathrm{SE}}^{i,j}) y_{i,j-1} + (N_{\mathrm{W}}^{i,j} + \alpha M_{\mathrm{NW}}^{i,j}) y_{i-1,j} + \\ & (N_{\mathrm{E}}^{i,j} + \alpha M_{\mathrm{SE}}^{i,j}) y_{i+1,j} + (N_{\mathrm{N}}^{i,j} + \alpha M_{\mathrm{NW}}^{i,j}) y_{i,j+1} \ . \end{aligned} \tag{4.17}$$

Setting each of the coefficients in the RHS of (4.17) to zero, then it follows from (4.13) that the entries of N can be expressed in those of L_{I} and U_{I},

$$\begin{aligned} N_{\mathrm{S}}^{i,j} &= -\alpha M_{\mathrm{SE}}^{i,j} = -\alpha L_{\mathrm{S}}^{i,j} U_{\mathrm{E}}^{i,j-1} \ , \\ N_{\mathrm{W}}^{i,j} &= -\alpha M_{\mathrm{NW}}^{i,j} = -\alpha L_{\mathrm{W}}^{i,j} U_{\mathrm{N}}^{i-1,j} \ , \\ N_{\mathrm{P}}^{i,j} &= \alpha(M_{\mathrm{SE}}^{i,j} + M_{\mathrm{NW}}^{i,j}) = \alpha(L_{\mathrm{S}}^{i,j} U_{\mathrm{E}}^{i,j-1} + L_{\mathrm{W}}^{i,j} U_{\mathrm{N}}^{i-1,j}) \ , \\ N_{\mathrm{E}}^{i,j} &= -\alpha M_{\mathrm{SE}}^{i,j} = -\alpha L_{\mathrm{S}}^{i,j} U_{\mathrm{E}}^{i,j-1} \ , \\ N_{\mathrm{N}}^{i,j} &= -\alpha M_{\mathrm{NW}}^{i,j} = -\alpha L_{\mathrm{W}}^{i,j} U_{\mathrm{N}}^{i-1,j} \ . \end{aligned} \tag{4.18}$$

Now, using $M = A - N$, (4.13) and (4.18), this results in

$$\begin{aligned} L_{\mathrm{S}}^{i,j} &= A_{\mathrm{S}}^{i,j} / (1 + \alpha U_{\mathrm{E}}^{i,j-1}) \ , \\ L_{\mathrm{W}}^{i,j} &= A_{\mathrm{W}}^{i,j} / (1 + \alpha U_{\mathrm{N}}^{i-1,j}) \ , \\ L_{\mathrm{P}}^{i,j} &= A_{\mathrm{P}}^{i,j} + \alpha(L_{\mathrm{S}}^{i,j} U_{\mathrm{E}}^{i,j-1} + L_{\mathrm{W}}^{i,j} U_{\mathrm{N}}^{i-1,j}) - L_{\mathrm{S}}^{i,j} U_{\mathrm{N}}^{i,j-1} - L_{\mathrm{W}}^{i,j} U_{\mathrm{N}}^{i-1,j} \ , \\ U_{\mathrm{E}}^{i,j} &= (A_{\mathrm{E}}^{i,j} - \alpha L_{\mathrm{S}}^{i,j} U_{\mathrm{N}}^{i,j-1}) / L_{\mathrm{P}}^{i,j} \ , \\ U_{\mathrm{N}}^{i,j} &= (A_{\mathrm{N}}^{i,j} - \alpha L_{\mathrm{W}}^{i,j} U_{\mathrm{E}}^{i-1,j}) / L_{\mathrm{P}}^{i,j} \ . \end{aligned} \tag{4.19}$$

The approximations in (4.19) are second order accurate when $\alpha = 1$, but for this α the SIP method is generally not convergent. That is why often $0.92 < \alpha < 0.96$ is chosen. Entries of (4.19) whose indices are outside the index boundaries should be set equal to zero.

4.3 Solving the stationary incompressible Navier-Stokes equations

There are various methods for iteratively solving the Navier-Stokes equations. A popular method used in engineering is the Semi-Implicit Method for Pressure-Linked Equations (SIMPLE). Many improved variants of SIMPLE has been proposed, like for example the SIMPLE Revised (SIMPLER) method. However, in engineering literature, for example Patankar [20] or Ferziger & Perić [6], it is not easy to verify which algebraic systems are actually solved because the presented algorithms are overrun with details. Therefore, a more mathematically convenient way is to present them in a so-called distributive iteration framework.

In this section we will only focus on SIMPLE. First the original SIMPLE algorithm is written in an linear algebra framework. Next we will show that SIMPLE can be written as classical distributive iterative method.

Deleting the time dependence in (3.10) gives the following system to be solved:

$$\begin{bmatrix} N & G \\ D & C \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{p} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \end{bmatrix} . \tag{4.20}$$

This nonlinear system has to be solved iteratively. First this system has to be linearized.

To linearize $N(\boldsymbol{u})$ various methods can be chosen, for example the Newton-Raphson method or the Picard method. We will restrict ourselves only to Picard iteration. The non-linear terms in the momentum equations can be linearized as follows,

$$(u^\alpha u^\beta)_j^{(k)} \approx (u_j^\alpha)^{(k-1)} (u_j^\beta)^{(k)} .$$

As a consequence $N(\boldsymbol{u})$ gets replaced by $L^{(k-1)}\boldsymbol{u}^{(k)}$, with $L^{(k-1)}$ a matrix that depends on $\boldsymbol{u}^{(k-1)}$. By this (4.20) becomes

$$\begin{bmatrix} L^{(k-1)} & G \\ D & C \end{bmatrix} \begin{bmatrix} \boldsymbol{u}^{(k)} \\ \boldsymbol{p}^{(k)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \end{bmatrix} . \tag{4.21}$$

This system will be denoted by $A\boldsymbol{y} = \boldsymbol{b}$. The Navier-Stokes equations can be solved by the following algorithm.

Algorithm 4.5 (Picard iteration). Let $\boldsymbol{u}^{(0)}$ and $\boldsymbol{p}^{(0)}$ be initial guesses of respective the velocity and pressure field. Then this algorithm solves the nonlinear system (4.20) for the stationary incompressible Navier-Stokes equations as follows.

for $k = 1, \ldots,$ convergence **do**
  Construct $L = L(\boldsymbol{u}^{(k-1)})$;
  Solve system (4.21);
end do
$\boldsymbol{y} \approx \boldsymbol{y}^{(k-1)}$;

The linear system 4.21 cannot be easily iteratively solved, because C in A is not a K-matrix and therefore A is not an M-matrix. This means that it is not trivial to design a convergent iterative method to solve this system.

The SIMPLE algorithm in its original form is obtained by performing only one iteration to solve (4.21) in each Picard iteration. Let $L^{(k-1)}$ be denoted by L for brevity, and let $\hat{L} \equiv \mathrm{diag}(L)$. Then the following algorithm describes SIMPLE.

Algorithm 4.6 (SIMPLE). Given initial guesses $\boldsymbol{u}^{(0)}$ and $\boldsymbol{p}^{(0)}$ of respective the velocity and pressure field. Then this algorithm solves the nonlinear system (4.20) for the stationary incompressible Navier-Stokes equations.

for $k = 1, \ldots,$ convergence **do**
  $L := L(\boldsymbol{u}^{(k-1)})$;
  Solve $L\boldsymbol{u}^{(k)} = \boldsymbol{b}_1 - G\boldsymbol{p}^{(k-1)}$;
  Solve $(C - D\hat{L}^{-1}G)\delta\boldsymbol{p} = \boldsymbol{b}_2 - D\boldsymbol{u}^{(k)}$;
  $\boldsymbol{u}^{(k)} := \boldsymbol{u}^{(k)} - \alpha_u \hat{L}^{-1}G\delta\boldsymbol{p}$;
  $\boldsymbol{p}^{(k)} = \boldsymbol{p}^{(k-1)} + \alpha_p \delta\boldsymbol{p}$;
end for

Distributive iteration

A linear system

$$A\boldsymbol{y} = \boldsymbol{b} \ , \tag{4.22}$$

is postconditioned as follows:

$$A\bar{B}\bar{\boldsymbol{y}} = \boldsymbol{b} \ , \quad \boldsymbol{y} = \bar{B}\bar{\boldsymbol{y}} \ , \tag{4.23}$$

where $\bar{B}$ is the postconditioning matrix. The matrix $\bar{B}$ is chosen in such a way that (4.23) is easier to solve iteratively than (4.22). For example one can choose $A\bar{B}$ to be an M-matrix, while A is not. Splitting the matrix $A\bar{B}$ in (4.23) yields

$$A\bar{B} = \bar{M} - \bar{N} \ ,$$

corresponding to the following splitting of the original matrix,

$$A = \bar{M}\bar{B}^{-1} - \bar{N}\bar{B}^{-1} \ .$$

Defining $M \equiv \bar{M}\bar{B}^{-1}$ and $N \equiv \bar{N}\bar{B}^{-1}$ and substituting this in the BIM (4.3) gives

$$\bar{M}\bar{B}^{-1}\boldsymbol{y}^{(k)} = \bar{N}\bar{B}^{-1}\boldsymbol{y}^{(k-1)} + \boldsymbol{b} \ . \tag{4.24}$$

When we subtract $\bar{M}\bar{B}^{-1}\boldsymbol{y}^{(k-1)}$ from the LHS and RHS of (4.24) we can rewrite this equation as

$$\bar{M}\bar{B}^{-1}(\boldsymbol{y}^{(k)} - \boldsymbol{y}^{(k-1)}) = \boldsymbol{b} - A\boldsymbol{y}^{(k-1)} \ .$$

Multiplying the RHS of this equation with a relaxation parameter α gives

$$\boldsymbol{y}^{(k)} = \boldsymbol{y}^{(k-1)} + \alpha\bar{B}\bar{M}^{-1}(\boldsymbol{b} - A\boldsymbol{y}^{(k-1)}) \ . \tag{4.25}$$

Let $\boldsymbol{y}^{(0)}$ be a certain initial guess, then the distributive iteration procedure is as follows,

for $k = 1, \ldots,$ convergence **do**
  $\boldsymbol{r}^{(k-1)} = \boldsymbol{b} - A\boldsymbol{y}^{(k-1)}$;
  Solve $\bar{M}\delta\boldsymbol{y} = \boldsymbol{r}^{(k-1)}$;
  $\delta\boldsymbol{y} := \bar{B}\delta\boldsymbol{y}$;
  $\boldsymbol{y}^{(k)} = \boldsymbol{y}^{(k-1)} + \alpha\delta\boldsymbol{y}$;
end for

The method (4.25) is called a distributive iteration because the correction $\bar{M}^{-1}(\boldsymbol{b} - A\boldsymbol{y}^{(k-1)})$ is distributed by multiplication with $\alpha\bar{B}$, over the elements of $\boldsymbol{y}^{(k)}$.

Distributive iteration for the stationary Navier-Stokes equations

Let A be the matrix of the system (4.21). We choose a matrix $\bar{B}$ in (4.25) such that $A\bar{B}$ is of block-triangular form,

$$A\bar{B} = \begin{bmatrix} Q & 0 \\ R & S \end{bmatrix} .$$

A possible choice for $\bar{B}$ is

$$\bar{B} = \begin{bmatrix} I & \bar{B}_{12} \\ 0 & \bar{B}_{22} \end{bmatrix} , \tag{4.26}$$

resulting in

$$A\bar{B} = \begin{bmatrix} L & L\bar{B}_{12} + G\bar{B}_{22} \\ D & D\bar{B}_{12} + C\bar{B}_{22} \end{bmatrix} .$$

Choosing $\bar{B}$ such that $L\bar{B}_{12} + G\bar{B}_{22} = 0$ gives

$$\bar{B}_{12} = -L^{-1}G\bar{B}_{22} ,$$

resulting in

$$A\bar{B} = \begin{bmatrix} L & 0 \\ D & (C - DL^{-1}G)B_{22} \end{bmatrix} . \tag{4.27}$$

The SIMPLE method is obtained by choosing $\bar{B}_{22} = I$, so that (4.27) becomes

$$A\bar{B} = \begin{bmatrix} L & 0 \\ D & C - DL^{-1}G \end{bmatrix} .$$

A suitable approximation $\bar{M}$ of $A\bar{B}$ is

$$\bar{M} = \begin{bmatrix} L & 0 \\ D & C - D\hat{L}^{-1}G \end{bmatrix} , \tag{4.28}$$

By this and $\bar{B}_{22} = I$ the matrix $\bar{B}$ given by (4.26) becomes

$$\bar{B} = \begin{bmatrix} I & -\hat{L}^{-1}G \\ 0 & I \end{bmatrix} . \tag{4.29}$$

Now the SIMPLE method is given by (4.25) with $\bar{M}$ (4.28) and $\bar{B}$ (4.29) and the SIMPLE algorithm in the distributed iteration framework is as follows.

Algorithm 4.7 (SIMPLE as a distributed iteration). Let $\boldsymbol{u}^{(0)}$ and $\boldsymbol{p}^{(0)}$ be respectively initial estimations of the velocity and pressure field. This algorithm solves the non-linear system (4.20) for the stationary incompressible Navier-Stokes equations.

for $k = 1, \ldots,$ convergence **do**
$L := L(\boldsymbol{u}^{(k-1)})$, $\hat{L} = \mathrm{diag}(L)$;
$\boldsymbol{r}_u^{(k)} = \boldsymbol{b}_1 - L\boldsymbol{u}^{(k)} - G\boldsymbol{p}^{(k)}$; \\ $\boldsymbol{r}^{(k)} = \boldsymbol{b} - \boldsymbol{y}^{(k)}$
$\boldsymbol{r}_p^{(k)} = \boldsymbol{b}_2 - D\boldsymbol{u}^{(k)} - C\boldsymbol{p}^{(k)}$;
Solve $L\delta\boldsymbol{u} = \boldsymbol{r}_u^{(k)}$; \\ Solve $\bar{M}\delta\boldsymbol{y} = \boldsymbol{b} - A\boldsymbol{y}^{(k)}$
Solve $(C - D\hat{L}^{-1}G)\delta\boldsymbol{p} = \boldsymbol{r}_p^{(k)} - D\delta\boldsymbol{u}$;

$\delta \boldsymbol{u} := \delta \boldsymbol{u} - \hat{L}^{-1} G \delta \boldsymbol{p}$; $\quad \backslash\backslash \delta \boldsymbol{y} = \bar{B} \delta \boldsymbol{y}$
$\boldsymbol{u}^{(k)} = \boldsymbol{u}^{(k-1)} + \alpha_u \delta \boldsymbol{u}$ $\quad \backslash\backslash\ \boldsymbol{y}^{(k+1)} = \boldsymbol{y}^{(k)} + \alpha \delta \boldsymbol{y}$
$\boldsymbol{p}^{(k)} = \boldsymbol{p}^{(k-1)} + \alpha_p \delta \boldsymbol{p}$;
end for

Many variants of the SIMPLE method can also be described in the presented distributed iteration framework, see for example Vuik *et al.* [31] or Vuik & Saghir [35] for a description of the SIMPLER method.

Convergence, computing work and stopcriterion

One can show that SIMPLE converges slowly and that the computing work is of $\mathcal{O}(N^2)$, just like BIMs. Fortunately, like BIMs, the SIMPLE method lends itself well for acceleration by Krylov subspace methods or multigrid.

Besides making the residuals small it is recommendable to make the velocity field also sufficiently divergence free, *i.e.* to make $D\boldsymbol{u}^{(k)}$ sufficiently small, for instance

$$\|D\boldsymbol{u}^{(k)}\|_\infty < \varepsilon V/H \ , \quad 0 < \varepsilon \ll 1 \ ,$$

with $\|\boldsymbol{x}\|_\infty \equiv \max\{|x_1|, \ldots, |x_n|\}$, V and H typical magnitudes for the velocity and domain, and ε a certain fixed tolerance. See for more details Wesseling [38].

4.4 Connection with X-stream

In X-stream the SIP method, the CG method and the Tri-Diagonal Matrix Algorithm (TDMA) are used to solve linear sytems of equations. Details on the TDMA method can be found in for example Patankar [20] or Ferziger & Perić [6]. The CG method is not much used because the involving systems are generally not SPD. In most applications the TDMA method performs less well than SIP, and therefore generally the SIP method is used.

In X-stream the incompressible Navier-Stokes equations are solved with SIMPLE in combination with defect correction. This is done by a domain decomposition approach, as will be discussed in the next chapter.

5

Domain decomposition methods

5.1 Introduction

The term domain decomposition (DD) is used differently by specialists in numerical analysis of PDEs. In parallel computing it means decomposing data from a computing model among the processors in a distributed memory computer. In asymptotic analysis DD refers to the determination of which PDEs to solve. In preconditioning methods DD refers only to the solution method for the algebraic system of equations arising from the discretisation. Note that all three of these interpretations may actually occur in one problem. We will restrict ourselves to DD methods as iterative solution methods for solving PDEs based on a decomposition of the spatial domain of the problem into several subdomains.

There are several motivations for using DD:

- Ease of parallelization and good parallel performance.
- Simplification of problems on complicated geometry.
- Different physical models can be used in different subdomains.
- Local grid refinement can be implemented with more ease.
- Reduction of memory requirements, because the subproblem can be much smaller than the total problem.

The structure of this chapter is as follows. In Section 5.2 basic DD methods are discussed briefly, followed by DD for the stationary Navier-Stokes equations in Section 5.3. In Section 5.4 the connection with X-stream is given.

5.2 Basic domain decomposition methods

A survey on DD methods can be found in for example Smith [23], Chan [5] or Saad [22]. Basically two types of DD methods can be distinguished: overlapping and non-overlapping methods. Overlapping DD methods are known as Schwarz alternating methods, which are iterative methods. Non-overlaping methods can be divided into so-called substructuring methods, which are direct methods, and so-called Schur-complement methods, which are iterative methods. Non-overlapping methods differ from overlapping methods by solving an additional interface equation.

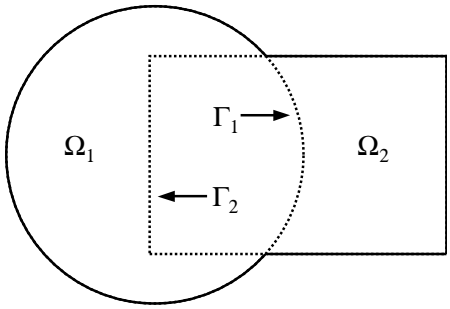

Figure 5.1: An overlapping decomposition of the domain.

In this section we restrict ourselves entirely to one-level DD mehods. In Subsection 5.2.1 the basic Schwarz method is discussed, followed by Schur complement methods in Subsection 5.2.2. In Subsection 5.2.3 convergence properties of the Schwarz method are treated, without going into details.

5.2.1 Alternating Schwarz methods

The simplest DD method is the alternating Schwarz method, which dates from 1870 and was originally intended as a analytical method. The Schwarz method in its original form is known as the multiplicative Schwarz method, which will be described next.

Multiplicative Schwarz method

DD aims to solve the differential equation

$$\mathcal{L}u = f \quad \text{in } \Omega \ , \tag{5.1}$$

with suitable BCs on $\partial\Omega$. The domain Ω is decomposed into subdomains $\bar{\Omega} = \bar{\Omega}_1 \cup \cdots \cup \bar{\Omega}_k$, where Ω is open, and Ω_i are open subsets of Ω. The simplest form of $\mathcal{L}$ is minus the Laplacian $-\Delta$, resulting in the Poisson equation. For simplicity, we restrict ourselves to Dirichlet BCs $u = g$ on $\partial\Omega$. Other BCs can be treated with ease. We only consider two subdomains $\bar{\Omega} = \bar{\Omega}_1 \cup \bar{\Omega}_2$.

Overlapping Schwarz DD methods uses an overlapping decomposition of the domain Ω into subdomains such that $\Omega_1 \cap \Omega_2 \neq \emptyset$. The part of the boundary of Ω_i which is located in the interior of Ω_j $(j \neq i)$ is denoted by Γ_i, see Figure 5.1.

The alternating Schwarz method begins by selecting an initial guess $u_2^{(0)}$ for the values in Ω_2. Then iteratively for $k = 1, \ldots$ one solves the subproblem,

$$\begin{cases} \mathcal{L}u_1^{(k+1)} = f & \text{on } \Omega_1 \ , \\ u_1^{(k+1)} = u^{(k)} & \text{on } \Gamma_1 \ , \\ u_1^{(k+1)} = g & \text{on } \partial\Omega_1 \setminus \Gamma_1 \ , \end{cases} \tag{5.2}$$

for $u_1^{(k)}$, followed by the solution of the subproblem,

$$\begin{cases} \mathcal{L}u_2^{(k+1)} = f & \text{in } \Omega_2 \ , \\ u_2^{(k+1)} = u^{(k+1)} & \text{on } \Gamma_2 \ , \\ u_2^{(k+1)} = g & \text{on } \partial\Omega_2 \setminus \Gamma_2 \ . \end{cases} \tag{5.3}$$

The k-th iterate is then defined by

$$u^{(k)}(x,y) = \begin{cases} u_1^{(k)}(x,y) & \text{if } (x,y) \in \Omega \setminus \Omega_2 \ , \\ u_2^{(k)}(x,y) & \text{if } (x,y) \in \Omega_2 \ . \end{cases} \tag{5.4}$$

It can be shown (see Chan [5] for some references) that for self-adjoint elliptic operator $\mathcal{L}$, the iterates $\{u^{(k)}\}$ converge linearly to the true solution u on Ω, that is

$$\|u - u^{(k)}\|_{\mathcal{L}} \leq \rho^k \|u - u^{(0)}\|_{\mathcal{L}} \ ,$$

where $\rho < 1$ depends on the choice of Ω_1 and Ω_2.

The discrete form of (5.1) is denoted by

$$A\boldsymbol{u} = \boldsymbol{f} \ , \tag{5.5}$$

where A represents the discretization of the continuous operator $\mathcal{L}$ and BCs on the global domain. We restrict ourselves to the case that the grids of the subdomains coincide in the overlap area.

The algebraic Schwarz algorithm will be described in matrix notation. Let I_1 and I_1 the index sets of the unknowns in the interior of Ω_1 and Ω_2 respectively. The total number of unknowns is $n = |I|$ and n_i denotes the number of unknowns in subdomain Ω_i. For the case of (generous) overlap $I_1 \cap I_2 \neq \emptyset$.

Denote by R_i^T a trival extension matrix of dimension $n \times n_i$, defined as

$$(R_i^T u_i)_k = \begin{cases} (u_i)_k & \text{if } k \in I \ , \\ 0 & \text{else} \ , \end{cases}$$

for $u_i \in \mathbb{R}^{n_i}$. The entries of the matrix R_i^T consist of ones and zeroes with at most one '1' in each row. The transpose R_i is a trivial restriction matrix which restricts a full length vector of size n to a subdomain vector of size n_i, by selecting the components of the vector corresponding to I_i. Note that $R_i R_i^T = I_{n_i}$, while $R_i^T R_i \neq I_n$. Also note that the matrices R_i are never formed in practice.

Now the local subdomain matrices can be written in terms of the global matrix A and the restriction matrices R_i as

$$A_{11} = R_1 A R_1^T \ , \quad A_{22} = R_1 A R_1^T \ .$$

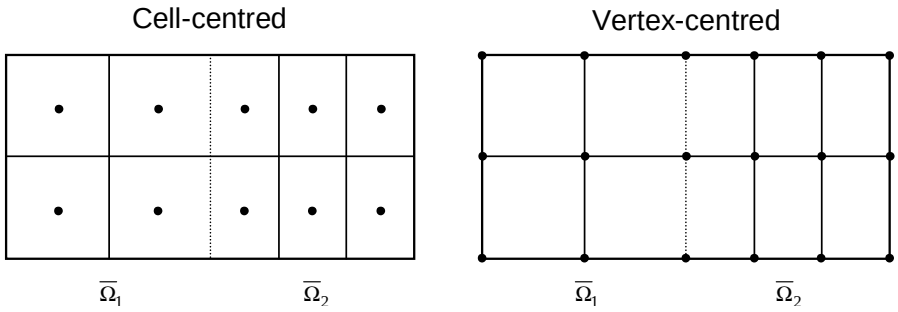

Figure 5.2: Cell-centred and vertex-centred discretization in the case of two subdomains. (• grid points; — finite volume boundaries; - - common grid line.)

The algebraic Schwarz iteration starts with an initial guess $\boldsymbol{u}^{(0)}$ and constructs a sequence of approximations as follows,

$$\boldsymbol{u}^{(k+\frac{1}{2})} = \boldsymbol{u}^{(k)} + R_1^T A_{11}^{-1} R_1(\boldsymbol{f} - A\boldsymbol{u}^{(k)}) \ , \tag{5.6}$$

$$\boldsymbol{u}^{(k+1)} = \boldsymbol{u}^{(k+\frac{1}{2})} + R_2^T A_{22}^{-1} R_2(\boldsymbol{f} - A\boldsymbol{u}^{(k+\frac{1}{2})}) \ . \tag{5.7}$$

For the cell-centred case that the two subdomains have a single grid line in common (see Figure 5.2), $I_1 \cap I_2 = \emptyset$, this iteration is a classical block Gauß-Seidel iteration. For $I_1 \cap I_2 \neq \emptyset$ this iteration corresponds to a block Gauß-Seidel iteration with overlapping blocks.

Define the matrix

$$P_i = R_i^T A_i^{-1} R_i A, \quad i = 1, 2 \ ,$$

then the global error $\boldsymbol{e}^{(k)}$ for (5.6)–(5.7) can be written as

$$\boldsymbol{e}^{(k+\frac{1}{2})} = (I - P_1)\boldsymbol{e}^{(k)} \ , \tag{5.8}$$

$$\boldsymbol{e}^{(k+1)} = (I - P_2)\boldsymbol{e}^{(k+\frac{1}{2})} \ . \tag{5.9}$$

The matrices P_i represent projection operators ($P_i^2 = P_i$). Combining (5.8) and (5.9) yields

$$\boldsymbol{e}^{(k+1)} = (I - P_1)(I - P_2)\boldsymbol{e}^{(k)} \ ,$$

thus the iteration matrix is $(I-P_1)(I-P_2)$, which is why the algoritihm is called multiplicative.

The iteration process (5.6)–(5.7) can be written as a preconditioned Richardson's iteration,

$$\boldsymbol{u}^{(k+1)} = \boldsymbol{u}^{(k)} + M_{\text{gs}}^{-1}(\boldsymbol{f} - A\boldsymbol{u}^{(k)}) \ ,$$

with $M_{\text{gs}}^{-1}A \equiv I - (I - P_2)(I - P_1)$. For the case $I_1 \cap I_2 = \emptyset$, the preconditioner M_{gs} is a lower block triangular matrix of A. Note that the multiplicative Schwarz method does not lend itself for parallellization, because of the preconditioner.

Additive Schwarz method

The additive Schwarz method is governed by computing the residual for $\boldsymbol{u}^{(k)}$ in (5.7),

$$\boldsymbol{u}^{(k+\frac{1}{2})} = \boldsymbol{u}^{(k)} + R_1^T A^{-1} R_1(\boldsymbol{f} - A\boldsymbol{u}^{(k)}) \ , \tag{5.10}$$

$$\boldsymbol{u}^{(k+1)} = \boldsymbol{u}^{(k+\frac{1}{2})} + R_2^T A^{-1} R_2(\boldsymbol{f} - A\boldsymbol{u}^{(k)}) \ . \tag{5.11}$$

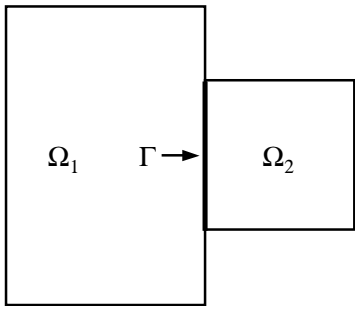

Figure 5.3: A nonoverlapping decomposition of the domain.

This iteration corresponds to a block Jacobi iteration (with overlapping blocks). Writing this in terms of the error $\boldsymbol{e}^{(k)}$ gives

$$\boldsymbol{e}^{(k+1)} = (I - P_1 - P_2)\boldsymbol{e}^{(k)} \ ,$$

which explains why the algorithm is called additive. Rewriting (5.10)–(5.11) as a preconditioned Richardson's iteration results in

$$\boldsymbol{u}^{(k+1)} = \boldsymbol{u}^{(k)} + M_{\text{jac}}^{-1}(\boldsymbol{f} - A\boldsymbol{u}^{(k)}) \ ,$$

with $M_{\text{jac}}^{-1}A \equiv P_1 + P_2$. Note that the additive Schwarz method lends itself well for parallellization, compared to the multiplicative Schwarz algorithm. However, concerning convergence it can be observed that the multiplicative Schwarz method converges faster that the additive Schwarz method.

Krylov subspace acceleration

From the alternating Schwarz method we derived the preconditioned system

$$M^{-1}A\boldsymbol{u} = M^{-1}\boldsymbol{f} \ ,$$

with $M^{-1} = M_{\text{jac}}^{-1}$ or $M^{-1} = M_{\text{gs}}^{-1}$. This preconditioned system can be accelerated by for example a Krylov subspace method.

5.2.2 Schur complement methods

Nonoverlapping DD is based on a decomposition of Ω such that $\bar{\Omega} = \bar{\Omega}_1 \cup \bar{\Omega}_1$ and $\Omega \cap \Omega = \emptyset$, see Figure 5.3.

Now briefly the basic idea of Schur complement methods will be discussed. We restrict ourselves to the Neumann-Dirichlet DD method. It can be shown that for second order elliptic operators $\mathcal{L} = -\nabla \cdot a(x,y)\nabla u$ Equation (5.1) is satisfied on the whole domain if (5.2) holds for the subdomains and if both u and its normal derivative $a\partial u/\partial n$ are continuous across the interface Γ, where n is the outward normal of subdomain Ω_1 on Γ. The alternating Neumann-Dirichlet method begins by selecting an initial guess $u_2^{(0)}$ for the values in Ω_2. Then iteratively

for $k = 1, \ldots$ one solves the subproblem

$$\begin{cases} \mathcal{L}u_1^{(k+1)} = f & \text{on } \Omega_1 \ , \\ a_{\text{left}} \frac{\partial u_1^{(k+1)}}{\partial n} = a_{\text{right}} \frac{\partial u_2^{(k)}}{\partial n} & \text{on } \Gamma \ , \\ u_1^{(k)} = g & \text{on } \partial\Omega_1 \setminus \Gamma \ , \end{cases}$$

for $u_1^{(k+1)}$, followed by the solution of the subproblem,

$$\begin{cases} \mathcal{L}u_2^{(k+1)} = f & \text{in } \Omega_2 \ , \\ u_2^{(k+1)} = u_1^{(k+1)} & \text{on } \Gamma \ , \\ u_2^{(k+1)} = g & \text{on } \partial\Omega_2 \setminus \Gamma \ . \end{cases}$$

The $(k+1)$-th iterate is defined as (5.4). Now the algebraic framework of Schur complements methods will be discussed briefly. Take a vertex-centred discretization (see Figure 5.2), so that there are unknowns on the interface Γ. If we order the unknowns based on the index sets $\bar{I}_1 = I_1 \setminus I_2$, $\bar{I}_2 = I_2 \setminus I_1$ and $\bar{I}_3 = I_1 \cap I_2$ and assume for example a five point discretization stencil, (5.5) has the block structure

$$\begin{bmatrix} K_{11} & \emptyset & K_{13} \\ \emptyset & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \boldsymbol{u} = \boldsymbol{b} \ , \quad \text{with } \boldsymbol{u} = \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \boldsymbol{u}_2 \end{bmatrix} \ .$$

By block-Gauß elimination of unknowns in $\bar{I}_1$ and $\bar{I}_2$, we obtain the Schur complement system for the variables in $\bar{I}_3$ which are located on the interface,

$$(K_{33} - K_{31}K_{11}^{-1}K_{13} - K_{32}K_{22}^{-1}K_{23})\boldsymbol{u}_3 = \bar{\boldsymbol{b}}_3 \ ,$$

with $\bar{\boldsymbol{b}}_3 \equiv \boldsymbol{b}_3 - K_{31}K_{11}^{-1}\boldsymbol{b}_1 - K_{32}K_{22}^{-1}\boldsymbol{b}_2$, and the matrix of this system is called the Schur complement matrix. Schur complement methods aim to solve this system of equations.

It can be shown that the Schwarz method with minimal overlap is a Neumann-Dirichlet method for a suitable preconditioner.

From now on we will restrict ourselves only to the Schwarz method.

5.2.3 Convergence properties

Local coupling between subdomains

The local coupling between the unknowns of the subdomains of the Schwarz DD method presented in the previous section can be increased. Because the Schwarz method can be seen as just a block Gauß-Seidel (multiplicative Schwarz) or block Jacobi (additive Schwarz) BIM, one obvious way is to accelerate the Schwarz method by a Krylov subspace or multigrid acceleration.

Inaccurate subdomain solution on subdomains

In every Schwarz iteration a linear system has to be solved corresponding to a subdomain problem. The Schwarz method is originally based on solving the subdomains problems accurately. However, solving the subdomain problems inaccurately can reduce the computing work drasticly, see for example Brakkee [3] and Verweij [27]. This is because it turns out that decreasing the number of subdomain iterations does not significantly increase the number of Schwarz iterations. In other words, the number of subdomain iterations and the Schwarz iterations can be practically made independent of each other, resulting in a decrease of computing work. This 'dependence' dependents on the topology of the domain and the acceleration technique used for the Schwarz method.

It turned out in Brakkee [3] that reduction in computing work is most favourable when the Schwarz method is accelerated by the GCR Krylov subspace method. This method has good scalability properties: increasing the number of subdomains does not significantly affect the dependence of the Schwarz and the subdomain iterations. This compared to the the unaccelerated Schwarz method, for which the decrease in computing work is less for increasing numbers of subdomains. Note that the accuracy of the subdomain solution cannot be made arbitrary small because from a certain accuracy the Schwarz method will diverge.

Global coupling between subdomains

For matrices resulting from discretisation of elliptic PDEs, global communication between the subdomains or global coupling is important. Elliptic problems have the property that a modification in the data at a certain area or point (for example a BC) influences the solution on the entire domain. In terms of the Schwarz method this means that new information calculated at a subdomain has to be transferred to all other subdomains in a fast way. Because information of a subdomain A can only travel through one interface in each Schwarz iteration to a subdomain B, the convergence decreases when the number of subdomains between A and B increases. So for an increasing number of subdomains the global coupling of the subdomains decreases.

In general there are two techniques for increasing global coupling: increasing subdomain overlap or using a so-called coarse grid correction. Increasing subdomain overlap makes the DD method more or less independent of the subdomain grid size for a constant decomposition. An important drawback of this method is that the amount of computing work increases proportionally to the size of overlap. A better option is to apply a coarse grid correction, which can be distinguished in two methods: multigrid coarse grid correction and deflation coarse grid correction.

Multigrid coarse grid correction, or simply coarse grid correction for a DD results in a two-level algorithm. The basic idea is to construct a preconditioner that smooths both low and high frequency error components. For the additive Schwarz method the block Jacobi preconditioner is used as smoother, see for more details Smith [23]. Other references are Nabben [16], Jenssen & Weinerfelt [10] and Padiy *et al.* [19]. A drawback of the multigrid coarse grid correction is that the geometry of the coarser grid has to be explicitly known in order to construct a preconditioner by interpolation. This can bring difficulties with implementation.

Another coarse grid correction approach is to use a coarse grid correction based on deflation, as described in Section 4.2.3. It turns out that deflation in combination with DD gives

better results compared to coarse grid correction and is easier to implement.

Deflation and domain decomposition

Various publications can be found on deflation in combination with domain decomposition and parallel computing. Some useful references are Frank & Vuik [8], Vuik & Frank [32], Vuik & Frank [33] and Vermolen & Vuik [26]. Other references are Mansfield [14], Mansfield [15] and Keyes [11].

Choosing the the subspaces Z and Y, as described in Section 4.2.3, is usually done at subdomain level. In general $Y = Z$ is taken. The choice of Z that has proven to be successful for non-overlapping Schwarz methods is,

$$\begin{aligned} \boldsymbol{z}_m(i) &= 1 \ , \quad \boldsymbol{x}_i \in \Omega_m \ , \\ \boldsymbol{z}_m(i) &= 0 \ , \quad \boldsymbol{x}_i \notin \Omega_m \ . \end{aligned}$$

This is referred to as subdomain deflation. In this stage of the research we will not go into further details.

5.3 Domain decomposition for the incompressible Navier-Stokes equations

In this section the additive Schwarz DD method is combined with the SIMPLE method (Algorithm 4.7) for solving the incompressible Navier-Stokes equations. The system to be solved in one Picard iteration is given by (4.21). Compared to the single-domain case, the matrices L, G, D and C contain block matrices corresponding to couplings between subdomains. For the two-dimensional Navier-Stokes equations ($d = 2$) and for the case of two (non-overlapping) subdomains ($nblock = 2$) the system explicitly becomes

$$A = \begin{bmatrix} L_1 & 0 & G_1 \\ 0 & L_2 & G_2 \\ D_1 & D_2 & C \end{bmatrix} = \begin{bmatrix} (L_{11})_1 & (L_{12})_1 & 0 & 0 & (G_{11})_1 & (G_{12})_1 \\ (L_{21})_1 & (L_{22})_1 & 0 & 0 & (G_{21})_1 & (G_{22})_1 \\ 0 & 0 & (L_{11})_2 & (L_{12})_2 & (G_{11})_2 & (G_{12})_2 \\ 0 & 0 & (L_{21})_2 & (L_{22})_2 & (G_{21})_2 & (G_{22})_2 \\ (D_{11})_1 & (D_{12})_1 & (D_{11})_2 & (D_{12})_2 & C_{11} & C_{12} \\ (D_{21})_1 & (D_{22})_1 & (D_{21})_2 & (D_{22})_2 & C_{21} & C_{22} \end{bmatrix} , \tag{5.12}$$

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{v} \\ \boldsymbol{p} \end{bmatrix} = \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \\ \boldsymbol{p}_1 \\ \boldsymbol{p}_2 \end{bmatrix} , \quad \boldsymbol{b} = \begin{bmatrix} \boldsymbol{b_1} \\ \boldsymbol{b_2} \\ \boldsymbol{b_3} \end{bmatrix} = \begin{bmatrix} (\boldsymbol{b}_1)_1 \\ (\boldsymbol{b}_1)_2 \\ (\boldsymbol{b}_2)_1 \\ (\boldsymbol{b}_2)_2 \\ (\boldsymbol{b}_3)_1 \\ (\boldsymbol{b}_3)_2 \end{bmatrix} . \tag{5.13}$$

For reasons given in the previous sections, we accelerate the additive Schwarz method by the PGCR method (Algorithm 4.1) or with the DPGCR method (Algorithm 4.3) when a deflation coarse grid correction is necessary. The resulting methods are referred to as the PGCR-Schwarz method and the DPGCR-Schwarz method, respectively. The SIMPLE method is accelerated by the PGCR method, resulting in the PGCR-SIMPLE.

First we will have to choose if we apply DD in SIMPLE, or SIMPLE in DD. Only literature could be found on DD in SIMPLE, for example Teigland [24]. Although the second option seems interesting we drop this method and continue with DD in SIMPLE. For more details on the PGCR-SIMPLE in the (D)PGCR-Schwarz method, see Appendix B.

First PGCR-SIMPLE will be discussed. For $\hat{L}_i^{-1}$ in the matrices $\bar{B}$ and $\bar{M}$ we take for $\hat{L}_i$

$$\hat{L}_i = \mathrm{diag}(L_i),\ i = 1, \ldots, d\ ,$$

resulting for the case of $d = 2$ and $nblock = 2$ in

$$\bar{B} = \begin{bmatrix} I & 0 & -\hat{L}_1^{-1}G_1 \\ 0 & I & -\hat{L}_2^{-1}G_2 \\ 0 & 0 & I \end{bmatrix} \tag{5.14}$$

and

$$\bar{M} = \begin{bmatrix} L_1 & 0 & 0 \\ 0 & L_2 & 0 \\ D_1 & D_2 & C - \sum_{i=1}^{2} D_i \hat{L}_i^{-1} G_i \end{bmatrix}\ . \tag{5.15}$$

The matrix $\bar{M}B^{-1}$ is taken as a preconditioner in the PGCR method (Algorithm 4.1), resulting in the PGCR-SIMPLE algorithm. Within each PGCR-SIMPLE iteration $k = 1, \ldots, ngcr1$ the search directions $\boldsymbol{s}^{(k)} = \bar{B}\bar{M}^{-1}\boldsymbol{r}^{(k-1)}$ has to be computed. The vector $\boldsymbol{s}^{(k)}$ can be computed by applying the following distributive step

Solve $\bar{M}\boldsymbol{q} = \boldsymbol{r}^{(k-1)}$;
$\boldsymbol{s}^{(k)} = \bar{B}\boldsymbol{q}$;

where $\boldsymbol{q}$ is an auxiliary variable. Substitution of $\bar{M}$ (5.15) and $\bar{B}$ (5.14) yields

1. Solve $L_i\boldsymbol{q}_i = \boldsymbol{r}_i^{(k-1)},\ i = 1, \ldots, d$;
2. Solve $(C - \sum_{i=1}^{d} D_i\hat{L}_i^{-1}G_i)\boldsymbol{q}_{d+1} = \boldsymbol{r}_{d+1} - \sum_{i=1}^{d} D_i\boldsymbol{q}_i$;
3. $\boldsymbol{s}^{(k)} = \bar{B}\boldsymbol{q}$;

Now DD is applied to the systems in steps 1 and 2: for the d systems in (1) the PGCR algorithm is used with the block Jacobi preconditioner blockdiag(L_i), for the system in (2) the DPGCR algorithm is used with the block Jacobi preconditioner blockdiag($C - \sum_{i=1}^{d} D_i\hat{L}_i^{-1}G_i$). This is done because the matrix in the system of step 2 can been seen as a discrete Laplacian operator.

Writing out the distributive steps to avoid computations of inverses leads to the following algorithm. Note that deflation is only written out till a certain level; systems involving the matrix Z are not explicitly written out.

Algorithm 5.1 (DPGCR-Schwarz in PGCR-SIMPLE). Consider the system $A\boldsymbol{y} = \boldsymbol{b}$ resulting from the discretisation of the Navier-Stokes equations in one Picard iteration. Let $\boldsymbol{y}^{(0)}$ be an initial guess of this system, $Y = Z$ and Z a certain subspace. Then this algorithm describes in pseudo code how the deflated PGCR Krylov subspace accelerated additive Schwarz DD method is applied within the PGCR accelerated SIMPLE.

$\boldsymbol{r}_i^{(0)} = \boldsymbol{b}_i - L_i\boldsymbol{y}_i^{(0)} - G_i\boldsymbol{y}_{d+1}^{(0)},\ i = 1, \ldots, d$;
$\boldsymbol{r}_{d+1}^{(0)} = \boldsymbol{b}_{d+1} - \sum_{i=1}^{d} D_i\boldsymbol{y}_i^{(0)} - C\boldsymbol{y}_{d+1}^{(0)}$;

for $k = 1, \ldots, ngcr1$ **do**
$\boldsymbol{q}_i = \mathbf{pgcrschw}[L_i, \boldsymbol{r}_i^{(k-1)}, \hat{\boldsymbol{q}}_i],\ i = 1, \ldots, d;$
$\boldsymbol{q}_{d+1} = \mathbf{dpgcrschw}[C - \sum_{i=1}^{d} D_i \hat{L}_i^{-1} G_i, \boldsymbol{r}_{d+1} - \sum_{i=1}^{d} D_i \boldsymbol{q}_i, \hat{\boldsymbol{q}}_{d+1}];$
$\boldsymbol{s}_i^{(k)} = \alpha_i(\boldsymbol{q}_i - \hat{L}_i^{-1} G_i \boldsymbol{q}_{d+1}),\ i = 1, \ldots, d;$
$\boldsymbol{s}_{d+1}^{(k)} = \alpha_{d+1} \boldsymbol{q}_{d+1};$
$\boldsymbol{v}_i^{(k)} = L_i \boldsymbol{s}_i^{(k)} + G_i \boldsymbol{s}_{d+1}^{(k)},\ i = 1, \ldots, d;$
$\boldsymbol{v}_{d+1}^{(k)} = \sum_{i=1}^{d} D_i \boldsymbol{s}_i^{(k)} + C \boldsymbol{s}_{d+1}^{(k)};$
$[\boldsymbol{v}^{(k)}, \boldsymbol{s}^{(k)}] := \mathbf{orthonorm1}[\boldsymbol{v}^{(k)}, \boldsymbol{s}^{(k)}, \{\boldsymbol{v}^{(1)}, \ldots, \boldsymbol{v}^{(k-1)}\}, \{\boldsymbol{s}^{(1)}, \ldots, \boldsymbol{s}^{(k-1)}\}];$
$\beta = (\boldsymbol{r}^{(k-1)}, \boldsymbol{v}^{(k)});$
$\boldsymbol{y}^{(k)} = \boldsymbol{y}^{(k-1)} + \beta \boldsymbol{s}^{(k)};$
$\boldsymbol{r}^{(k)} = \boldsymbol{r}^{(k-1)} - \beta \boldsymbol{v}^{(k)};$
end for
$\boldsymbol{y} \approx \boldsymbol{y}^{(k-1)};$

function $\bar{\boldsymbol{y}} = \mathbf{pgcrschw}[\bar{A}, \bar{\boldsymbol{b}}, \bar{\boldsymbol{y}}^{(0)}]$
$\bar{\boldsymbol{r}}^{(0)} = \bar{\boldsymbol{b}} - \bar{A} \bar{\boldsymbol{y}}^{(0)};$
for $l = 1, \ldots, ngcr2$ **do**
Solve $\bar{A}_{mm} \bar{\boldsymbol{s}}_m^{(l)} = \bar{\boldsymbol{r}}_m^{(l-1)}$ (inaccurately), $m = 1, \ldots, nblock;$
$\bar{\boldsymbol{v}}^{(l)} = \bar{A} \bar{\boldsymbol{s}}^{(l)};$
$[\bar{\boldsymbol{v}}^{(l)}, \bar{\boldsymbol{s}}^{(l)}] := \mathbf{orthonorm2}\ [\bar{\boldsymbol{v}}^{(l)}, \bar{\boldsymbol{s}}^{(l)}, \{\bar{\boldsymbol{v}}^{(1)}, \ldots, \bar{\boldsymbol{v}}^{(l-1)}\}, \{\bar{\boldsymbol{s}}^{(1)}, \ldots, \bar{\boldsymbol{s}}^{(l-1)}\}];$
$\bar{\beta} = (\bar{\boldsymbol{r}}^{(l-1)}, \bar{\boldsymbol{v}}^{(l)});$
$\bar{\boldsymbol{y}}^{(l)} = \bar{\boldsymbol{y}}^{(l-1)} + \bar{\beta} \bar{\boldsymbol{s}}^{(l)};$
$\bar{\boldsymbol{r}}^{(l)} = \bar{\boldsymbol{r}}^{(l-1)} - \bar{\beta} \bar{\boldsymbol{v}}^{(l)};$
end for
$\bar{\boldsymbol{y}} \approx \bar{\boldsymbol{y}}^{(l-1)};$

function $\bar{\boldsymbol{y}} = \mathbf{dpgcrschw}[\bar{A}, \bar{\boldsymbol{b}}, \bar{\boldsymbol{y}}^{(0)}]$
$\tilde{\boldsymbol{y}}^{(0)} = \mathbf{0};$
$\bar{\boldsymbol{r}}^{(0)} = \bar{\boldsymbol{b}} - \bar{A} \bar{\boldsymbol{y}}^{(0)};$
Solve $Z^T \bar{A} Z \bar{\boldsymbol{q}}_1 = Z^T \bar{\boldsymbol{r}}^{(0)};$
$\tilde{\boldsymbol{r}}^{(0)} = \bar{\boldsymbol{r}}^{(0)} - AZ \boldsymbol{q}_1;$
for $l = 1, \ldots, ngcr2$ **do**
Solve $\bar{A}_{mm} \bar{\boldsymbol{s}}_m^{(l)} = \tilde{\boldsymbol{r}}^{(l-1)}$ (inaccurately), $m = 1, \ldots, nblock;$
Solve $Z^T \bar{A} Z \bar{\boldsymbol{q}}_2 = Z^T \bar{A} \bar{\boldsymbol{s}}^{(l)};$
$\bar{\boldsymbol{v}}^{(l)} = \bar{A}(\bar{\boldsymbol{s}}^{(l)} - Z \bar{\boldsymbol{q}}_2);$
$[\bar{\boldsymbol{v}}^{(l)}, \bar{\boldsymbol{s}}^{(l)}] := \mathbf{orthonorm3}\ [\bar{\boldsymbol{v}}^{(l)}, \bar{\boldsymbol{s}}^{(l)}, \{\bar{\boldsymbol{v}}^{(1)}, \ldots, \bar{\boldsymbol{v}}^{(l-1)}\}, \{\bar{\boldsymbol{s}}^{(1)}, \ldots, \bar{\boldsymbol{s}}^{(l-1)}\}];$
$\bar{\beta} = (\tilde{\boldsymbol{r}}^{(l-1)}, \bar{\boldsymbol{v}}^{(l)});$
$\tilde{\boldsymbol{y}}^{(l)} = \tilde{\boldsymbol{y}}^{(l-1)} + \bar{\beta} \bar{\boldsymbol{s}}^{(l)};$
$\tilde{\boldsymbol{r}}^{(l)} = \tilde{\boldsymbol{r}}^{(l-1)} - \bar{\beta} \bar{\boldsymbol{v}}^{(l)};$
end for
Solve $Z^T \bar{A} Z \bar{\boldsymbol{q}}_3 = Z^T \bar{A} \tilde{\boldsymbol{y}}^{(l-1)};$
$\bar{\boldsymbol{y}} \approx Z(\bar{\boldsymbol{q}}_1 - \bar{\boldsymbol{q}}_3) + \tilde{\boldsymbol{y}}^{(l-1)} + \bar{\boldsymbol{y}}^{(0)};$

In the above algorithm the routines **dpcgrschw** and **pgcrschw** are called when PGCR-Schwarz DD is applied with respective deflation and without deflation. In the third argument

of this routines an initial guess is provided. The routines **orthonorm1**, **orthonorm2** and **orthonorm3** refer to the orthonormalization process used, see Section 4.2.2.

The relaxation parameters $\alpha_i,\ i = 1, \ldots, d$ in the PGCR-SIMPLE loop refer to the velocity components, the relaxation parameter α_{d+1} to the pressure. When applying a Krylov subspace acceleration it is expected that less relaxation is necessary, so it is likely that relaxation parameters closer to 1 can be chosen.

The solution procedure for solving the incompressible Navier-Stokes is given by the following algorithm.

Algorithm 5.2 (Picard iteration for DPGCR-Schwarz in PGCR-SIMPLE). Let $\boldsymbol{y}^{(0)}$ be an initial guess. Then this algorithm solves the non-linear system (4.20) for the stationary incompressible Navier-Stokes equations.

for $k = 1, \ldots,$ convergence **do**
  Solve $A(\boldsymbol{y}^{(k-1)})\boldsymbol{y}^{(k)} = \boldsymbol{b}$ with DPGCR-Schwarz in PGCR-SIMPLE (Algorithm 5.1);
end do
$\boldsymbol{y} \approx \boldsymbol{y}^{(k-1)}$;

Before solving the new system in Algorithm 5.2 it can be necessary to perform a diagonal scaling. This can improve performance of the PGCR-SIMPLE algorithm, see for example Vuik & Saghir [35] for details.

Some aspects on parallel computing

In parallel computing several aspects are of importance, for example: speedup, efficiency, scalability, communication and load balance. We will not go in details on parallel computing, but refer to a standard textbook, for example Kumar [13].

As mentioned before, GCR acceleration and deflation lead to additional work and memory storage. When Algorithm 5.1 is parallellized one has to take that into account. Memory storage is generally of less importance than additional work because computers contain more and more memory. Additional work that requires global or local (nearest neighbor) communication is of more importance in parallel computing. In Algorithm 5.1 global communication especially occurs in the GCR algorithms during the orthonormalization process when inner products has to be calculated. Local communication occurs in all the loops in the algorithm where matrix vector multiplications are required.

In general one wants to minimize global communication, because global communication is most expensive. Because there are different orthonormalization algorithms it is therefore obvious to choose the most efficient one, see Frank & Vuik [7], Vuik & Frank [29] or Vuik e.a [34]. It is advisable to use the RCGS or MGS orthonormalization process, depending on the parallel computer and the number of unknowns per processor. However, one has to keep in mind that a decrease of communication cost does not always guarantees an increase of the overall parallel performance, because the time won could be lost again by the additional computing work or by load imbalance.

5.4 Connection with X-stream

In X-stream a Schwarz DD method with the following properties is implemented:

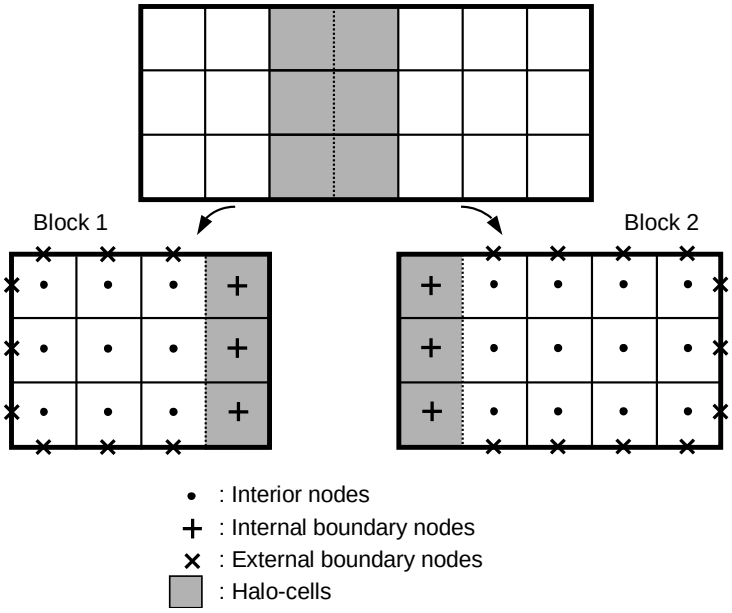

Figure 5.4: Decomposition of the domain in two blocks with the halo-cells (shaded). The dashed line denotes the block interface.

- one-level
- additive
- minimal overlap
- unaccelerated
- cell-centred
- block-structured
- inaccurate subdomain with SIP or TDMA
- local grid refinement at block level
- application of different models on different blocks
- parallellized with MPI (Message Passing Interface)

For more details we refer to Verweij [27] or Twerda [25]. We will only go into more detail on the decomposition used, as depicted in Figure 5.4 for the case of two blocks. The shaded cells denote virtual cells called halo-cells. The Schwarz method goes as follows. First the internal boundary nodes are updated for block 1, then values for the interior nodes of block 1 are computed. After this, the values of the internal boundary nodes of block 2 are updated by this values, *i.e.* copied to the halo-cells of block 2. Then the values of the internal nodes of block 2 are computed, and the process repeats. In X-stream this process can be executed in parallel.

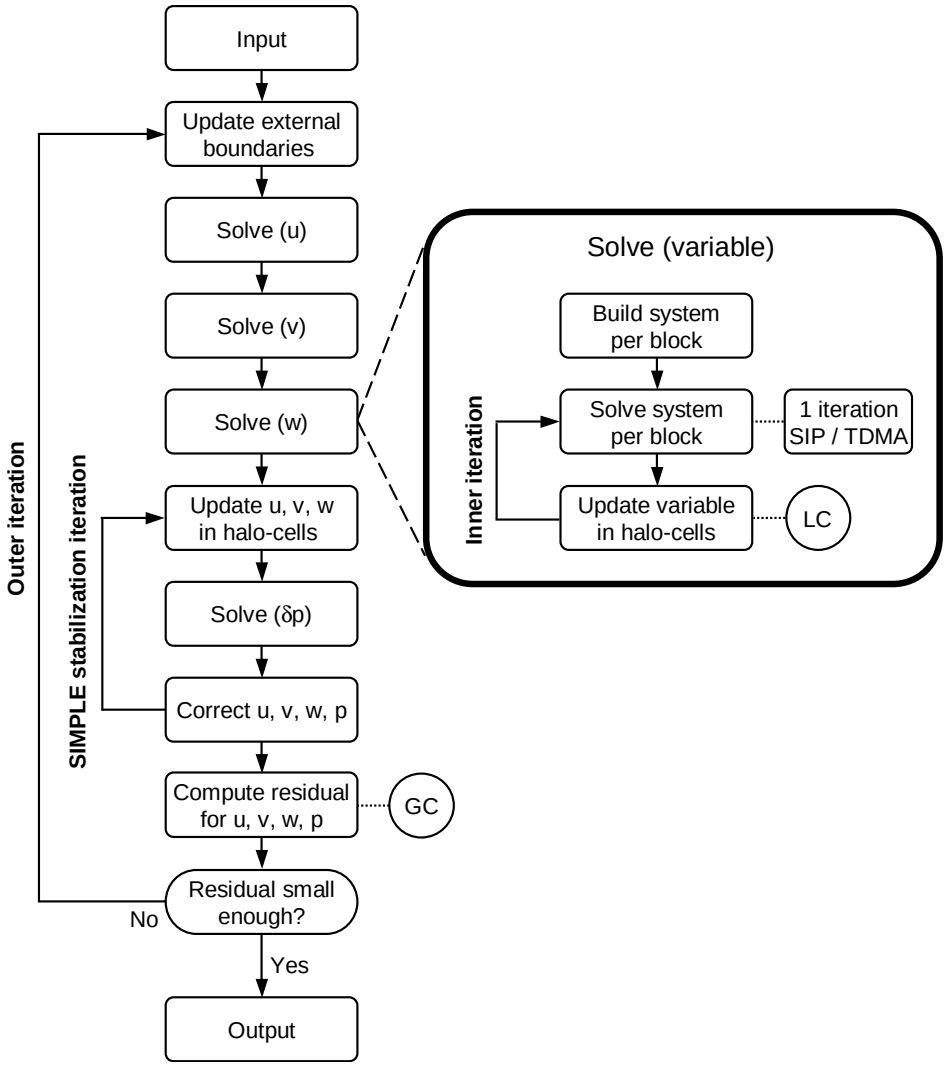

Figure 5.5: Flowchart of the solution procedure in X-stream for solving the stationary incompressible Navier-Stokes equations. (LC: local communication; GC: global communication.)

In X-stream a set of subdomains (blocks) for which a same model is applied is referred to as 'domain'. For the solution of the Navier-Stokes equations a Schwarz iteration iteration is reffered to as an 'inner iteration' and a SIMPLE iteration as an 'outer iteration'.

A global flowchart of the current solution procedure for solving the Navier-Stokes equations in X-stream is given by Figure 5.5. The 'SIMPLE stabilization iteration' refers to a method for improving convergence of the pressure system. The abbreviations LC and GC denote respectively local and global communication between the processors.

In Figure 5.6 a possible flowchart for the DPGCR-Schwarz method in the PGCR-SIMPLE (Algorithm 5.1) is given. Note that in only the routine **pgcrschwarz** is drawn, the routine **dpgcrschwarz** goes in a similar way.

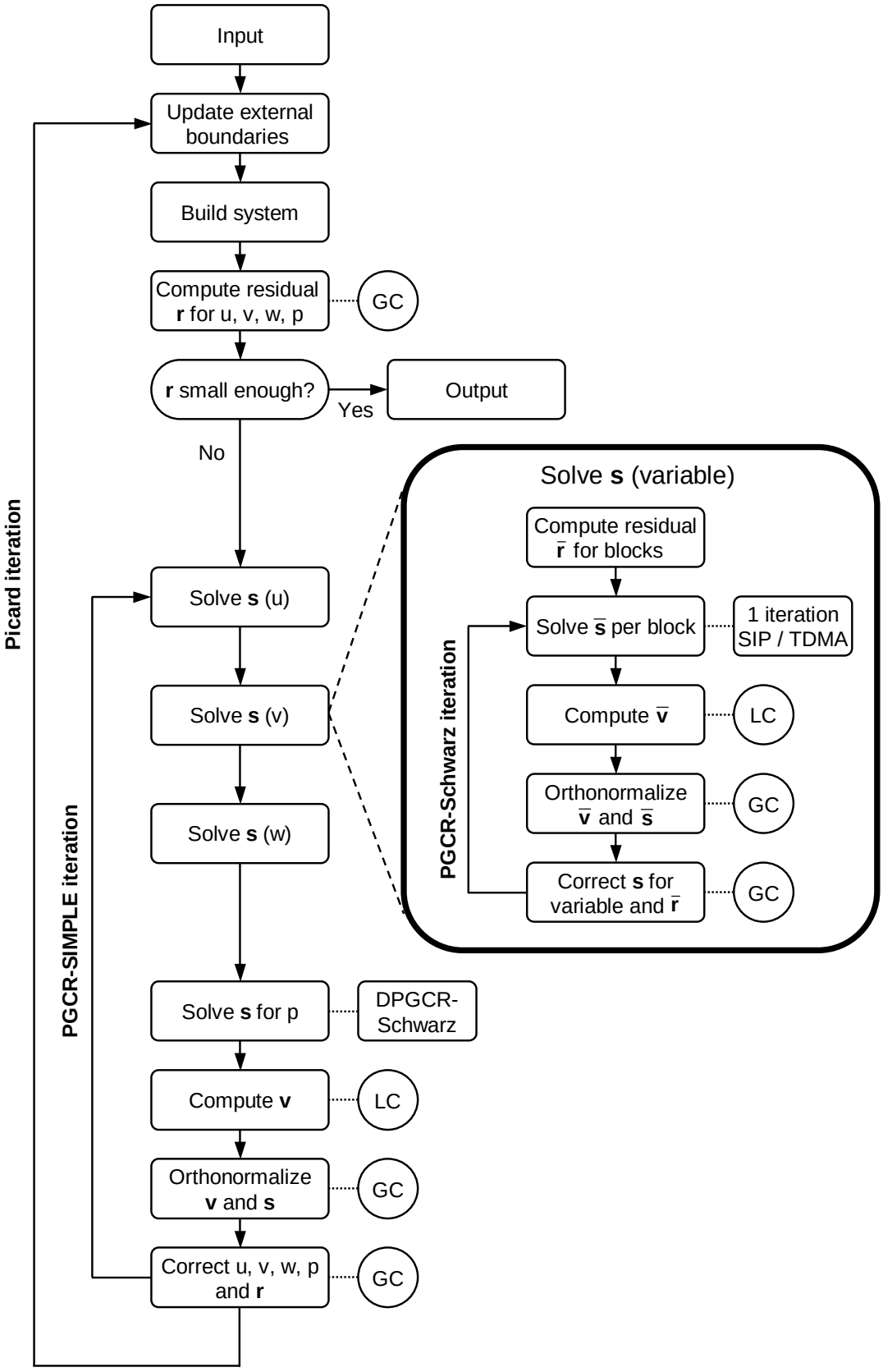

Figure 5.6: Possible flowchart of the improved solution procedure in X-stream for solving the stationary incompressible Navier-Stokes equations. (LC: local communication; GC: global communication.)

6

Test cases in X-stream

In this chapter three testcases in X-stream will be described briefly. The most simple testcase is given in Section 6.1. In Section 6.2 a more complicated testcase is described concerning the flow in a cavity, followed by a glass tank simulation in Section 6.3.

6.1 Constant density flow in an unit cube

In this testcase (X-stream reference XTC-35) the stationary incompressible Navier-Stokes equations are solved for an unit cube. The flow has a constant density, is laminar, and no buoyancy is considered.

The geometry is three-dimensional and is given by in respectively x, y and z-direction 1 [m] $\times$ 1 [m] $\times$ 1 [m]. The inlet is located at $z = 0$, the outlet at $z = 1$. The four remaining boundaries are all walls.

For the velocity components uniform Dirichlet BCs are taken at the inlet and homogeneous Neumann BCs at the outlet and at the walls.

Figure 6.1 shows the geometry, as well as the resulting velocity field when two blocks are taken.

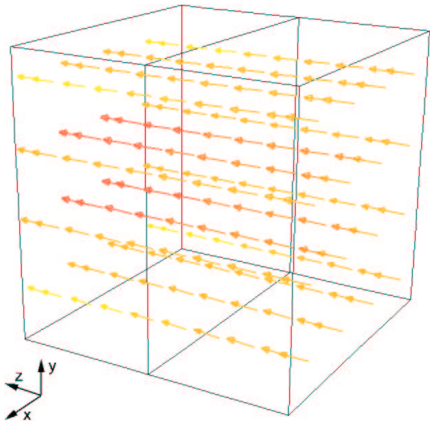

Figure 6.1: Geometry and velocity field for a constant density flow in an unit cube.

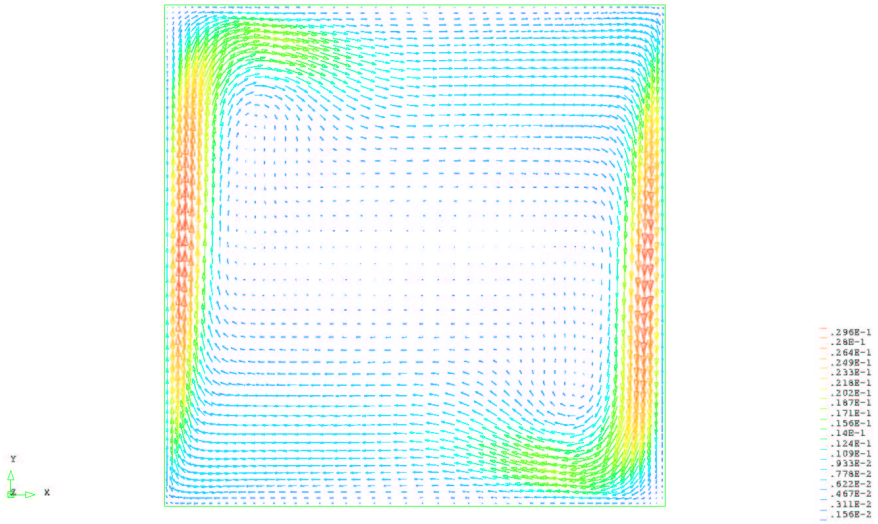

Figure 6.2: The resulting velocity field (arrows) and temperatures (color) for the square cavity.

6.2 Variable density buoyant flow in a square cavity

The equations to be solved in this testcase (X-stream reference XTC-33) are the stationary incompressible Navier-Stokes equations and the stationary energy equation. The flow considered has a variable density depending on the temperature, is laminar, and buoyancy effects are taken into account.

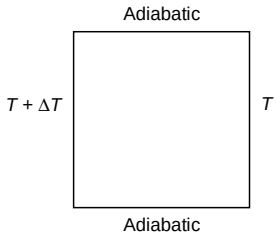

Figure 6.3: Temperature BCs for the cavity. (left wall: hot; right wall: cold.)

The geometry is a three-dimensional square plate in width of one cell. BCs for the temperature are depicted in Figure 6.3. For the hot and cold wall Dirichlet conditions are taken, for the adiabatic walls homogeneous Neumann conditions. For the velocities at all walls no-slip Dirichlet conditions are taken. Because of the temperature difference of the cold and hot wall natural convection occurs resulting in a circulation flow.

The resulting velocity field and the temperature distribution is given in Figure 6.2. The computations are done only for one block.

6.3 Simulation of a flow in a glass tank

In this testcase (X-stream reference XTC-07) the stationary incompressible Navier-Stokes equations and the stationary energy equation are solved. The flow considered is a buoyant

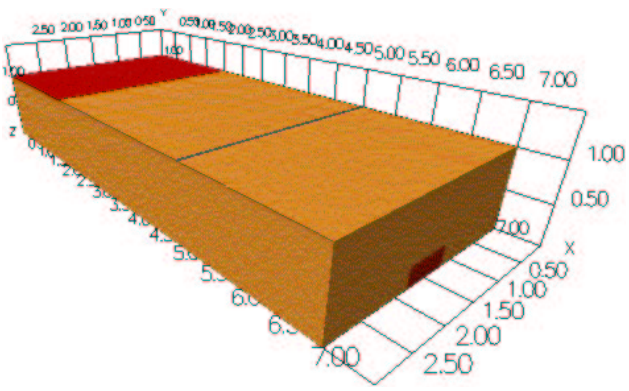

Figure 6.4: Geometry of the glass tank.

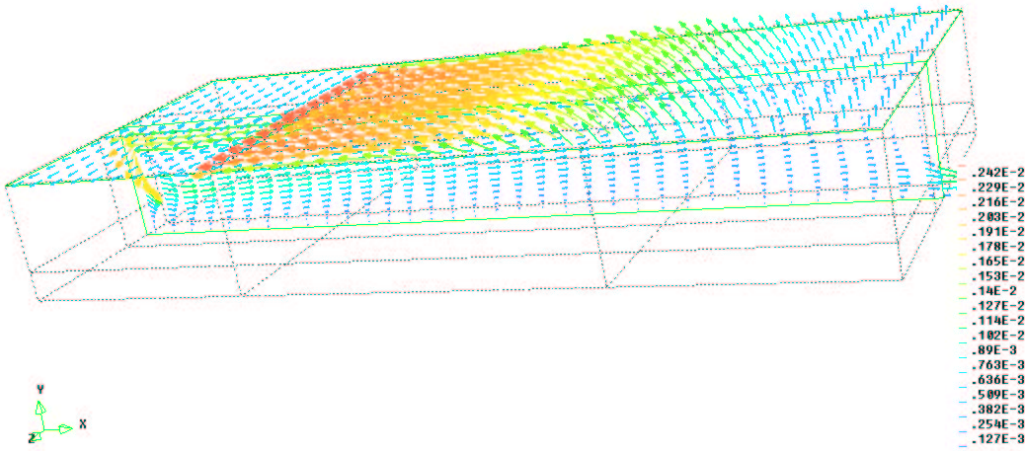

Figure 6.5: The resulting velocities (arrows) and temperatures (color) in the glass tank.

laminar flow. The density as well as the dynamic viscosity depend on the temperature.

The geometry of the glass tank is as in Figure 6.4. The overall dimensions in respectively x, y and z-direction is 7 [m] $\times$ 1 [m] $\times$ 3 [m]. The red areas correspond to the inlet of the glass fractions and the outlet of the melted glass. The glass inlet runs from $x = 0$ [m] to $x = 1$ [m] over the entire width of the tank. The glass outlet is located at $x = 7$ [m] and is centred in z-direction with a width of 0.6 [m] and height of 0.25 [m].

At the inlet homogeneous Dirichlet BCs are taken for the velocities and a linear profile Dirichlet condition for the temperature in the x-direction. At the glass outlet a homogeneous Neumann condition is taken. At the glass surface homogeneous Dirichlet conditions are taken for the velocity components and a piecewise linear profile Dirichlet condition in the x-direction. For the walls no-slip Dirichlet conditions for the velocities are taken and homogeneous Neumann conditions for the temperature.

Figure 6.5 shows the velocity field and the temperatures for the case eighteen blocks are used.

7

Continuation of the Master's project

In the following six months of the Master's project the current X-stream DD algorithm is targeted to be accelerated. This will be primary done for the stationary incompressible Navier-Stokes equations. Roughly the plan for the continuation of the Master's project consists of the following three steps.

Step 1: Implementation of DPGCR-Schwarz (MGS and RCGS) for the pressure

In this step the Master's project will be mainly focussed. This step is divided into the following substeps.

1. Implementation and testing the DPGCR-Schwarz algorithm in Matlab.
2. Implementation of the PGCR-Schwarz algorithm (MGS and RCGS) in X-stream.
3. Testing the PGCR-Schwarz algorithm.
4. Implementation of the DPGCR-Schwarz algorithm.
5. Testing the DPGCR-Schwarz algorithm.
6. Analysis of parameter variation on the total number of SIMPLE iterations.
7. Parallellization.
8. Testing parallellization.
9. Analysis of parameter variation on the total wall-clock time.

The implementation and testing step in Matlab is only done to check the DPGCR-Schwarz algorithm for possible errors. In Matlab the DPGCR-Schwarz algorithm will be implemented for the two-dimensional Poisson problem on a small number of subdomains. Note that the implementation and testing of the DPGCR-Schwarz algorithm is done in two substeps: first PGCR-Schwarz is implemented and tested, second deflation is added and tested. The main reason for this is that one wants to keep the code structured as much as possible.

After testing DPGCR-Schwarz with the the most simple testcase (see Section 6.1), a parameter variation analysis is carried out for all the three testcases in order to analyse the effect of changing parameters on the total number of SIMPLE iterations. The parameters to be varied are: number of subdomains, subdomain topology, grid point distribution, deflation

vectors, number of PGCR-Schwarz iterations and pressure relaxation parameter. For a large number of DPGCR-Schwarz iterations also the restarting / truncation parameters has to be considered.

The next substep is to parallellize the code followed by some testing and a final parameter variation analysis in order to analyse the effect of the parameters on the total wall-clock time. In this analysis also differences between the MGS and RCGS will be analysed.

Step 2: Implementation of PGCR-SIMPLE (MGS and RCGS)

The second step is to implement the PGCR-SIMPLE in X-stream according to the following substeps

1. Implementation of the PGCR-SIMPLE.
2. Testing the PGCR-SIMPLE.
3. Analysis of parameter variation on the total number of Picard iterations.
4. Parallellization.
5. Testing paralellization.
6. Analysis of parameter variation on the total wall-clock time.

After implementation and testing PGCR-SIMPLE in X-stream an analysis of parameter variation is being performed. The same parameters as in step 1 are being varied, in addition with a new parameter: the number of PGCR-SIMPLE iterations. Note that no restarting / truncation has to be carried out, because the total number of PGCR-SIMPLE iterations are kept small. After parallellization and testing the final substep is to perform a parameter variation analysis in order to analyse the effect on the parameters on the wall-clock time. Just like in step 2 also differences between MGS and RCGS will be analysed. Note that the parallel performance differences between MGS and RCGS are likely to be small because generally only a small number of PGCR-SIMPLE iterations are required.

Step 3: Implementation of PGCR-Schwarz (MGS and RCGS) for the velocities

The last step is to to implement the PGCR-Schwarz algorithm for the velocities. This step is expected to give the least increament of the multi-block performance. The following steps will be carried out

1. Implementation of the PGCR-Schwarz algorithm.
2. Testing the PGCR-Schwarz algorithm.
3. Analysis of parameter variation on the total number of Picard iterations.
4. Parallellization.
5. Testing parallellization.
6. Analysis of parameter variation on the total wall-clock time.

The analysis of parameter variation after the implementation and testing PGCR-Schwarz introduces some more parameters to vary: number of PGCR-Schwarz iterations for each velocity component, velocity relaxation parameter for each velocity. The parameter variation analysis after parallellization also introduces some new parameters if the number of PGCR-Schwarz iterations is large: restarting / truncation parameter, MGS or RCGS.

From the above steps it should be clear that parameter variation plays an important role in the accelerating a multi-block algorithm. Even for a simple multi-block algorithm for solving the Navier-Stokes equations it is not always obvious what parameters to choose and most of them are chosen by means of trial-and-error. At each point one accelerates a multi-block algorithm more parameters are introduced resulting in a more complex algorithm. The problem is that most of the parameters are problem dependent and choosing the parameters becomes more difficult.

A

PWI method

A.1 Derivation of the PWI method

In this section the PWI method given by Equation (3.8) is derived. The basic idea of PWI is to approximate the cell face velocity components by a central discretization and adding and subtracting an extra term involving the pressure,

$$u^{\alpha}_{\boldsymbol{j}+\boldsymbol{e}_\alpha} \approx \frac{1}{2}(u^{\alpha}_{\boldsymbol{j}} + u^{\alpha}_{\boldsymbol{j}+2\boldsymbol{e}_\alpha}) + (\frac{h_1 h_2}{a^\alpha} p_{,\alpha})_{\boldsymbol{j}+\boldsymbol{e}_\alpha} - (\frac{h_1 h_2}{a^\alpha} p_{,\alpha})_{\boldsymbol{j}+\boldsymbol{e}_\alpha} \ .$$

The second and third term in the above equation are approximated differently. The second term is approximated by

$$\begin{aligned}
(\frac{h_1 h_2}{a^\alpha} p_{,\alpha})_{\boldsymbol{j}+\boldsymbol{e}_\alpha} &\approx \tfrac{1}{2}[(\frac{h_1 h_2}{a^\alpha} p_{,\alpha})_{\boldsymbol{j}+2\boldsymbol{e}_\alpha} + (\frac{h_1 h_2}{a^\alpha} p_{,\alpha})_{\boldsymbol{j}}] \ , \\
&= \tfrac{1}{2}[(\frac{h_1 h_2}{a^\alpha})_{\boldsymbol{j}+2\boldsymbol{e}_\alpha} (p_{,\alpha})_{\boldsymbol{j}+2\boldsymbol{e}_\alpha} + (\frac{h_1 h_2}{a^\alpha})_{\boldsymbol{j}} (p_{,\alpha})_{\boldsymbol{j}}] \ , \\
&= \tfrac{1}{2}[(\frac{h_1 h_2}{a^\alpha})_{\boldsymbol{j}+2\boldsymbol{e}_\alpha} \frac{p_{\boldsymbol{j}+4\boldsymbol{e}_\alpha} - p_{\boldsymbol{j}}}{2h_\alpha} + (\frac{h_1 h_2}{a^\alpha})_{\boldsymbol{j}} \frac{p_{\boldsymbol{j}+2\boldsymbol{e}_\alpha} - p_{\boldsymbol{j}-2\boldsymbol{e}_\alpha}}{2h_\alpha}] \ , \\
&= \tfrac{1}{2}[(\frac{h_\beta}{2a^\alpha})_{\boldsymbol{j}+2\boldsymbol{e}_\alpha} (p_{\boldsymbol{j}+4\boldsymbol{e}_\alpha} - p_{\boldsymbol{j}}) + (\frac{h_\beta}{2a^\alpha})_{\boldsymbol{j}} (p_{\boldsymbol{j}+2\boldsymbol{e}_\alpha} - p_{\boldsymbol{j}-2\boldsymbol{e}_\alpha})] \ ,
\end{aligned} \tag{A.1}$$

and the third term by

$$\begin{aligned}
(\frac{h_1 h_2}{a^\alpha} p_{,\alpha})_{\boldsymbol{j}+\boldsymbol{e}_\alpha} &\approx (\frac{h_1 h_2}{a^\alpha})_{\boldsymbol{j}+\boldsymbol{e}_\alpha} (p_{,\alpha})_{\boldsymbol{j}+\boldsymbol{e}_\alpha} \ , \\
&= \tfrac{1}{2}[(\frac{h_1 h_2}{a^\alpha})_{\boldsymbol{j}+2\boldsymbol{e}_\alpha} + (\frac{h_1 h_2}{a^\alpha})_{\boldsymbol{j}}] \frac{p_{\boldsymbol{j}+2\boldsymbol{e}_\alpha} - p_{\boldsymbol{j}}}{h_\alpha} \ , \\
&= \tfrac{1}{2}[(\frac{h_\beta}{2a^\alpha})_{\boldsymbol{j}+2\boldsymbol{e}_\alpha} (2p_{\boldsymbol{j}+2\boldsymbol{e}_\alpha} - 2p_{\boldsymbol{j}}) + (\frac{h_\beta}{2a^\alpha})_{\boldsymbol{j}} (2p_{\boldsymbol{j}+2\boldsymbol{e}_\alpha} - 2p_{\boldsymbol{j}})] \ .
\end{aligned} \tag{A.2}$$

Subtracting expression (A.2) from (A.1) results in

$$\begin{aligned}
&\tfrac{1}{2}[(\frac{h_\beta}{2a^\alpha})_{\boldsymbol{j}+2\boldsymbol{e}_\alpha} (p_{\boldsymbol{j}+4\boldsymbol{e}_\alpha} - 2p_{\boldsymbol{j}+2\boldsymbol{e}_\alpha} + p_{\boldsymbol{j}}) + (\frac{h_\beta}{2a^\alpha})_{\boldsymbol{j}} (-p_{\boldsymbol{j}+2\boldsymbol{e}_\alpha} - p_{\boldsymbol{j}-2\boldsymbol{e}_\alpha} + 2p_{\boldsymbol{j}})] \\
&\qquad = (\frac{h_\beta}{4a^\alpha} \Delta^\alpha p)|_{\boldsymbol{j}}^{\boldsymbol{j}+2\boldsymbol{e}_\alpha} \ ,
\end{aligned}$$

and by this Equation (3.8) is derived.

A.2 Discretization of the continuity equation with the PWI method

In this section the discretised coninuity equation with the PWI method given by Equation (3.9) is derived. Writing out Equation (3.6) yields

$$h_2(u^1_{\boldsymbol{j}+\boldsymbol{e}_1} - u^1_{\boldsymbol{j}-\boldsymbol{e}_1}) + h_1(u^2_{\boldsymbol{j}+\boldsymbol{e}_2} - u^2_{\boldsymbol{j}-\boldsymbol{e}_2}) = 0 \ . \tag{A.3}$$

Approximation of the first term with the PWI method (3.8) gives

$$\begin{aligned} h_2(u^1_{\boldsymbol{j}+\boldsymbol{e}_1} - u^1_{\boldsymbol{j}-\boldsymbol{e}_1}) &= h_2([\tfrac{1}{2}(u^1_{\boldsymbol{j}} + u^1_{\boldsymbol{j}+2\boldsymbol{e}_1}) + (\frac{h_2}{4a^1}\Delta^1 p)|^{\boldsymbol{j}+2\boldsymbol{e}_1}_{\boldsymbol{j}}] \\ &\quad -[\tfrac{1}{2}(u^1_{\boldsymbol{j}-2\boldsymbol{e}_1} + u^1_{\boldsymbol{j}}) + (\frac{h_2}{4a^1}\Delta^1 p)|^{\boldsymbol{j}}_{\boldsymbol{j}-2\boldsymbol{e}_1}] \ ,) \\ &= \tfrac{1}{2}h_2 u^1|^{\boldsymbol{j}+2\boldsymbol{e}_1}_{\boldsymbol{j}-2\boldsymbol{e}_1} \\ &\quad +h_2^2[(\frac{1}{4a^1}\Delta^1 p)_{\boldsymbol{j}+2\boldsymbol{e}_1} - (\frac{1}{2a^1}\Delta^1 p)_{\boldsymbol{j}} + (\frac{1}{4a^1}\Delta^1 p)_{\boldsymbol{j}-2\boldsymbol{e}_1}] \ , \end{aligned} \tag{A.4}$$

and the second term in Equation (A.3) can be analogous be written as

$$h_1(u^2_{\boldsymbol{j}+\boldsymbol{e}_2} - u^2_{\boldsymbol{j}-\boldsymbol{e}_2}) = \frac{1}{2}h_1 u^2|^{\boldsymbol{j}+2\boldsymbol{e}_2}_{\boldsymbol{j}-2\boldsymbol{e}_2} + h_1^2[(\frac{1}{4a^2}\Delta^2 p)_{\boldsymbol{j}+2\boldsymbol{e}_2} - (\frac{1}{2a^2}\Delta^2 p)_{\boldsymbol{j}} + (\frac{1}{4a^2}\Delta^2 p)_{\boldsymbol{j}-2\boldsymbol{e}_2}] \ . \tag{A.5}$$

Substitution of (A.4) and (A.5) in Equation (A.3) results in

$$\begin{aligned} &\tfrac{1}{2}h_2 u^1|^{\boldsymbol{j}+2\boldsymbol{e}_1}_{\boldsymbol{j}-2\boldsymbol{e}_1} + \tfrac{1}{2}h_1 u^2|^{\boldsymbol{j}+2\boldsymbol{e}_2}_{\boldsymbol{j}-2\boldsymbol{e}_2} \\ &+ h_2^2[(\frac{1}{4a^1}\Delta^1 p)_{\boldsymbol{j}+2\boldsymbol{e}_1} - (\frac{1}{2a^1}\Delta^1 p)_{\boldsymbol{j}} + (\frac{1}{4a^1}\Delta^1 p)_{\boldsymbol{j}-2\boldsymbol{e}_1}] \\ &+ h_1^2[(\frac{1}{4a^2}\Delta^2 p)_{\boldsymbol{j}+2\boldsymbol{e}_2} - (\frac{1}{2a^2}\Delta^2 p)_{\boldsymbol{j}} + (\frac{1}{4a^2}\Delta^2 p)_{\boldsymbol{j}-2\boldsymbol{e}_2}] = 0 \ . \end{aligned}$$

When this equation is multiplied by a factor 2, this results in (3.9).

B

PGCR-SIMPLE in DPGCR-Schwarz

The system formed by (5.12) and (5.13) can be rearranged into the mathematically equivalent system

$$A = \begin{bmatrix} (L_{11})_1 & 0 & (G_{11})_1 & (L_{12})_1 & 0 & (G_{12})_1 \\ 0 & (L_{11})_2 & (G_{11})_2 & 0 & (L_{12})_2 & (G_{12})_2 \\ (D_{11})_1 & (D_{11})_2 & C_{11} & (D_{12})_1 & (D_{12})_2 & C_{12} \\ (L_{21})_1 & 0 & (G_{21})_1 & (L_{22})_1 & 0 & (G_{22})_1 \\ 0 & (L_{21})_2 & (G_{21})_2 & 0 & (L_{22})_2 & (G_{22})_2 \\ (D_{21})_1 & (D_{21})_2 & C_{21} & (D_{22})_1 & (D_{22})_2 & C_{22} \end{bmatrix}, \boldsymbol{y} = \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{v}_1 \\ \boldsymbol{p}_1 \\ \boldsymbol{u}_2 \\ \boldsymbol{v}_2 \\ \boldsymbol{p}_2 \end{bmatrix}, \boldsymbol{b} = \begin{bmatrix} (\boldsymbol{b_1})_1 \\ (\boldsymbol{b_2})_1 \\ (\boldsymbol{b_3})_1 \\ (\boldsymbol{b_1})_2 \\ (\boldsymbol{b_2})_2 \\ (\boldsymbol{b_3})_2 \end{bmatrix} .$$

This matrix is of the form

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} ,$$

where A_{12} and A_{21} represent coupling between the two subdomains. The idea is now to apply the PGCR accelerated additive Schwarz DD method with a block Jacobi preconditioner to this system in combination with a deflation coarse grid correction. In each Schwarz iteration systems of the form

$$A_{mm}\bar{\boldsymbol{y}}_m = \bar{\boldsymbol{b}}_m, \ m = 1, \ldots, M ,$$

has to be solved. Seen the structure of A_{mm} an obvious choice is to do this with the PGCR accelerated SIMPLE. In PGCR-SIMPLE search directions of the form

$$\bar{\boldsymbol{s}}^{(l)} = \bar{B}\bar{M}^{-1}\bar{\boldsymbol{r}}^{(l-1)}$$

has to be computed, where the matrices $\bar{B}$ and $\bar{M}$ for a matrix A_{mm} for the case of $d = 2$ and $nblock = 2$ are given by

$$B = \begin{bmatrix} I & 0 & -(\hat{L}_{mm})_1^{-1}(G_{mm})_1 \\ 0 & I & -(\hat{L}_{mm})_2^{-1}(G_{mm})_2 \\ 0 & 0 & I \end{bmatrix} ,$$

and

$$M = \begin{bmatrix} (L_{mm})_1 & 0 & 0 \\ 0 & (L_{mm})_2 & 0 \\ (D_{mm})_1 & (D_{mm})_2 & C_{mm} - \sum_{k=1}^{2}(D_{mm})_k(\hat{L}_{mm})_k^{-1}(G_{mm})_k \end{bmatrix} ,$$

where

$$(\hat{L}_{mm})_i = \mathrm{diag}[(L_{mm})_i], \ i = 1, \ldots, d .$$

The search directions $\bar{\boldsymbol{s}}^{(l)}$ are computed by performing the distributive step

Solve $M\bar{\boldsymbol{q}} = \bar{\boldsymbol{r}}^{(l-1)}$;
$\bar{\boldsymbol{s}}^{(l)} = B\bar{\boldsymbol{q}}$;

where $\bar{\boldsymbol{q}}$ is an auxiliary variable. The following algorithm describes in pseudo code the resulting PGCR-SIMPLE in the DPGCR-Schwarz algorithm.

Algorithm B.1 (PGCR-SIMPLE in DPGCR-Schwarz). Consider the rearranged system $A\boldsymbol{y} = \boldsymbol{b}$ resulting from the discretization of the Navier-Stokes equations in one Picard iteration. Let $\boldsymbol{y}^{(0)}$ be an initial guess of this system, $Y = Z$ and Z a certain subspace. Then this algorithm describes in pseudo code how the PGCR Krylov subspace accelerated SIMPLE is applied in the deflated PGCR accelerated additive Schwarz method.

$\tilde{\boldsymbol{y}}^{(0)} = \boldsymbol{0}$;
$\boldsymbol{r}^{(0)} = \boldsymbol{b} - A\boldsymbol{y}^{(0)}$;
Solve $Z^T AZ\boldsymbol{q}_1 = Z^T\boldsymbol{r}^{(0)}$;
$\tilde{\boldsymbol{r}}^{(0)} = \boldsymbol{r}^{(0)} - AZ\boldsymbol{q}_1$;
for $k = 1, \ldots, ngcr1$ **do**
  $\boldsymbol{s}_m^{(k)} = \textbf{pgcrsimple}[A_{mm}, \tilde{\boldsymbol{r}}_m^{(k-1)}, \hat{\boldsymbol{s}}_m^{(k)}],\ m = 1, \ldots, nblock$;
  Solve $Z^T AZ\boldsymbol{q}_2 = Z^T A\boldsymbol{s}^{(k)}$;
  $\boldsymbol{v}^{(k)} = A(\boldsymbol{s}^{(k)} - Z\boldsymbol{q}_2)$;
  $[\boldsymbol{v}^{(k)}, \boldsymbol{s}^{(k)}] = \textbf{orthonorm1}[\boldsymbol{v}^{(k)}, \boldsymbol{s}^{(k)}, \{\boldsymbol{v}^{(1)}, \ldots, \boldsymbol{v}^{(k-1)}\}, \{\boldsymbol{s}^{(1)}, \ldots, \boldsymbol{s}^{(k-1)}\}]$;
  $\beta = (\tilde{\boldsymbol{r}}^{(k-1)}, \boldsymbol{v}^{(k)})$;
  $\tilde{\boldsymbol{y}}^{(k)} = \tilde{\boldsymbol{y}}^{(k-1)} + \beta\boldsymbol{s}^{(k)}$;
  $\tilde{\boldsymbol{r}}^{(k)} = \tilde{\boldsymbol{r}}^{(k-1)} - \beta\boldsymbol{v}^{(k)}$;
end for
Solve $Z^T AZ\boldsymbol{q}_3 = Z^T A\tilde{\boldsymbol{y}}^{(k-1)}$;
$\boldsymbol{y} = Z(\boldsymbol{q}_1 - \boldsymbol{q}_3) + \tilde{\boldsymbol{y}}^{(k-1)} + \boldsymbol{y}^{(0)}$;

$\bar{\boldsymbol{y}} =$ function $\textbf{pgcrsimple}[\bar{A}, \bar{\boldsymbol{b}}, \bar{\boldsymbol{y}}^{(0)}]$
  $\bar{\boldsymbol{r}}^{(0)} = \bar{\boldsymbol{b}} - \bar{A}\bar{\boldsymbol{y}}^{(0)}$;
  for $l = 1, \ldots, ngcr2$ **do**
    Solve $\bar{L}_i\bar{\boldsymbol{q}}_i = \bar{\boldsymbol{r}}_i^{(l-1)},\ i = 1, \ldots, d$;
    Solve $(C - \sum_{i=1}^d \bar{D}_i\hat{\bar{L}}_i^{-1}\bar{G}_i)\bar{\boldsymbol{q}}_{d+1} = \bar{\boldsymbol{r}}_{d+1}^{(l-1)} - \sum_{i=1}^d \bar{D}_i\bar{\boldsymbol{q}}_i$;
    $\bar{\boldsymbol{s}}_i^{(l)} = \alpha_i(\bar{\boldsymbol{q}}_i - \hat{\bar{L}}_i^{-1}\bar{G}_i\bar{\boldsymbol{q}}_{d+1}),\ i = 1, \ldots, d$;
    $\bar{\boldsymbol{s}}_{d+1}^{(l)} = \alpha_{d+1}\bar{\boldsymbol{q}}_{d+1}$;
    $\bar{\boldsymbol{v}}^{(l)} = A\bar{\boldsymbol{s}}^{(l)}$;
    $[\bar{\boldsymbol{v}}^{(l)}, \bar{\boldsymbol{s}}^{(l)}] := \textbf{orthonorm2}\ [\bar{\boldsymbol{v}}^{(l)}, \bar{\boldsymbol{s}}^{(l)}, \{\bar{\boldsymbol{v}}^{(1)}, \ldots, \bar{\boldsymbol{v}}^{(l-1)}\}, \{\bar{\boldsymbol{s}}^{(1)}, \ldots, \bar{\boldsymbol{s}}^{(l-1)}\}]$;
    $\bar{\beta} = (\bar{\boldsymbol{r}}^{(l-1)}, \bar{\boldsymbol{v}}^{(l)})$;
    $\bar{\boldsymbol{y}}^{(l)} = \bar{\boldsymbol{y}}^{(l-1)} + \bar{\beta}\bar{\boldsymbol{s}}^{(l)}$;
    $\bar{\boldsymbol{r}}^{(l)} = \bar{\boldsymbol{r}}^{(l-1)} - \bar{\beta}\bar{\boldsymbol{v}}^{(l)}$;
  end for
  $\bar{\boldsymbol{y}} = \bar{\boldsymbol{y}}^{(l-1)}$;

In the third argument of the routine **pgcrsimple** an initial guess is provided. The routines **orthonorm1** and **orthonorm2** refer to the orthonormalization process used, see Section 4.2.2.

The following algorithm describes the solution procedure for the Navier-Stokes equations.

Algorithm B.2 (Picard iteration for PGCR-SIMPLE in DPGCR-Schwarz). Let $\boldsymbol{y}^{(0)}$ be an initial guess. Then this algorithm solves the non-linear system (4.20) for the stationary incompressible Navier-Stokes equations.

for $k = 1, \ldots,$ convergence **do**
    Solve $A(\boldsymbol{y}^{(k-1)})\boldsymbol{y}^{(k)} = \boldsymbol{b}$ with PGCR-SIMPLE in DPGCR-Schwarz (Algorithm B.1);
end for
$\boldsymbol{y} \approx \boldsymbol{y}^{(k-1)}$;

No literature could be found on the above approach for solving the Navier-Stokes equations. It is possible that this approach results in weaker local coupling between the unknowns of the subdomains. Also it is questionable how deflation deals with the elliptic part of the Navier-Stokes equations and increases the global communication between the subdomains. However, this approach could be interesting and perhaps deserves some further investigation.

Bibliography

[1] Batchelor, G.K. (1967). *An Introduction to Fluid Dynamics*. Cambridge: Cambridge University Press.

[2] Bird, R.B., *et al.* (2001). *Transport Phenomena*. New York: Wiley, second edition.

[3] Brakkee, E. (1996). *Domain Decomposition for the Incompressible Navier-Stokes Equations*. PhD thesis. Delft: Delft University Press.

[4] Cai, X. (2002). 'Overlapping Domain Decomposition Methods'. To appear in: *Computational Partial Differential Equations Using Diffpack - Advanced Topics*.

[5] Chan, T.F. and T.P. Mathew (1994). 'Domain decomposition algorithms'. *Acta Numerica*, Vol. 1, pp. 61–141.

[6] Ferziger, J.H. and M. Perić (1999). *Computational Methods for Fluid Dynamics*. Heidelberg: Springer, second edition.

[7] Frank, J. and C. Vuik (1999). 'Parallel implementation of a multiblock method with approximate subdomain solution'. *Appl. Numer. Math.*, Vol. 30, pp. 403–423.

[8] Frank, J. and C. Vuik (2001). 'On the construction of deflation-based preconditioners'. *SIAM J. Sci. Comput.*, Vol. 23, pp. 442–462.

[9] Golub, G.H. and C.F. Van Loan (1996). *Matrix Computations*. Baltimore: The John Hopkins University Press, third edition.

[10] Jenssen, C.B. and P.Å. Weinerfelt (1995). 'A Coarse Grid Correction Scheme for Implicit Multi Block Euler Calculations'. *AIAA J.*, Vol. 33, No. 10, pp. 1816–1821.

[11] Keyes, D.E. (1998). 'How Scalable is Domain Decomposition in Practice?'. *Proc. of the 11th Int. Conf. on Domain Decomposition Methods*, Greenwich, pp. 286–297.

[12] Kim, S. and H-C. Lee (2000). 'On Accuracy of Operator Splitting Techniques for Unsteady Navier-Stokes Equations'. *TEAM seminar*, Department of Mathematics, University of Kentucky.

[13] Kumar, V. *et al.* (1994). *Introduction to Parallel Computing: Design and Analysis of Algorithms*. Redwood City: Benjamin Cummings.

[14] Mansfield, L. (1990). 'On the conjugate gradient solution of the Schur complement system obtained from domain decomposition'. *SIAM J. Numer. Anal.*, Vol. 7, No. 6, pp. 1612–1620.

[15] Mansfield, L. (1991). 'Damped Jacobi Preconditioning And Coarse Grid Deflation for Conjugate Gradient Iteration on Parallel Computers'. *SIAM J. Sc. Stat. Comput.*, Vol. 12, No. 6, pp. 1314–1323.

[16] Nabben, R. (2002). 'Comparisons between multiplicative and additive Schwarz iterations in domain decomposition methods'. To appear in: *Numerische Mathematik*.

[17] Nicolaides, R.A. (1987). 'Deflation of Conjugate Gradients with Applications to Boundary Value Problems'. *SIAM J. Numer. Anal.*, Vol. 24, No. 2, pp. 355–365.

[18] Nieuwstadt, F.T.M. (1998). *Turbulentie: theorie en toepassingen van turbulente stromingen*. Utrecht: Epsilon Uitgaven.

[19] Padiy, A., O. Axelsson and B. Polman (2000). 'Generalized Augmented Matrix Preconditioning Approach and its Application to Iterative Solution of Ill-Conditioned Algebraic Systems'. *SIAM J. Matrix Anal. Appl.*, Vol. 22, No. 3, pp. 793–818.

[20] Patankar, S.V. (1980). *Numerical Heat Transfer and Fluid Flow*. New York: McGraw-Hill.

[21] Post, L. (1988). *Modelling of flow and combustion in a glass melting furnace*. PhD thesis. Delft: Delft University Press.

[22] Saad, Y. (2000). *Iterative Methods for Sparse Linear Systems*. Boston: PWS Publishing Company, second edition.

[23] Smith, B.F., *et al.* (1996). *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*. New York: Cambridge University Press.

[24] Teigland, R. (1998). 'A Domain Decomposition Strategy for Simulation of Industrial Fluid Flows'. *Proc. 9th Int. Conf. on DDMs*, http:
www.ddm.org.

[25] Twerda, A. (2000). *Advanced computing methods for complex flow simulation*. PhD thesis. Delft: Delft University Press.

[26] Vermolen, F.J. and C. Vuik (2001). *The influence of deflation vectors at interfaces on the deflated conjugate gradient method*. Report of the Department of Applied Mathematical Analysis, ISSN 1389-6520, Delft University of Technology.

[27] Verweij, R.L. (1999). *Parallel Computing for furnace simulations using domain decomposition*. PhD thesis. Delft: Delft University Press.

[28] Vuik, C. (1996). *Voortgezette numerieke lineaire algebra*. Delft University of Technology.

[29] Vuik, C. and J. Frank (1999). 'A Parallel Implementation of the Block Preconditioned GCR Method'. *Proc. 7th int. conf. HPCN*, Berlin: Springer, Lecture Notes in Computer Science 1593, pp. 1052–1060.

[30] Vuik, C., A. Segal *et al.* (1999). 'An efficient preconditioned CG method for the solution of layered problems with extreme contrasts in the coefficients'. *J. Comp. Phys.*, Vol. 152, pp. 385–403.

[31] Vuik, C., A. Saghir *et al.* (2000). 'The Krylov accelerated SIMPLE(R) method for flow problems in industrial furnaces'. *Int. J. Numer. Meth. Fluids*, Vol. 33, pp. 1027–1040.

[32] Vuik, C. and J. Frank (2000). 'Deflated ICCG methods applied to problems with extreme contrasts in the coefficients'. *Proc. 16th IMACS World Congress*, Lausanne.

[33] Vuik, C. and J. Frank (2001). 'Coarse grid acceleration of a parallel block preconditioner'. *Future Generation Computer Systems*, Vol. 17, pp. 933–940.

[34] Vuik, C., J. Frank and A. Segal (2001). 'A parallel block-preconditioned GCR method for incompressible flow problems'. *Future Generation Computer Systems*, Vol. 18, pp. 31–40.

[35] Vuik, C. and A. Saghir (2002). *The Krylov accelerated SIMPLE(R) method for incompressible flow*. Report 02-01 of the Department of Applied Mathematical Analysis, ISSN 1389-6520, Delft University of Technology.

[36] Vuik, C., A. Segal, *et al.* (2002). 'A comparison of various deflation vectors applied to elliptic problems with discontinuous coefficients'. *Appl. Numer. Math.*, Vol. 41, pp. 219–233.

[37] Wesseling, P. (1991). *An Introduction to Multigrid Methods*. Chichester: Wiley.

[38] Wesseling, P. (2001). *Principles of Computational Fluid Dynamics*. Heidelberg: Springer.

[39] Wilcox, D.C. (1993). *Turbulence Modeling for CFD*. La Cañada: DCW Industries.

Nomenclature

Roman symbols

Symbol	*Description*	*Units*	*Chapter*
a	sum of coefficients in the PWI method	[—]	3
A	matrix of the linear system	[—]	4, 5
b	source term in the DAS	[—]	3, 4, 5
B	iteration matrix	[—]	4
B	postconditioning matrix	[—]	4, 5
c_p	specific heat at constant pressure	[$\mathrm{m^2\ s^{-2}\ K^{-1}}$]	2
C	linear algebraic operator in DAS arising from PWI	[—]	3, 4, 5
d	number of space dimensions	[—]	2, 3, 5
D	diffusion coefficient	[$\mathrm{m^2\ s^{-1}}$]	2
D	linear algebraic operator in DAS for the velocities	[—]	3, 4, 5
e	internal energy per unit mass	[$\mathrm{m^2\ s^{-2}}$]	2
e	auxiliary variable for denoting grid positions	[—]	3
e	global truncation error	[—]	3, 4, 5
E	total energy per unit mass	[$\mathrm{m^2\ s^{-2}}$]	2
f	body force	[$\mathrm{kg\ m^{-2}\ s^{-2}}$]	2
f	dimensionless body force	[—]	3
f	inhomogeneous term in differential equation	[—]	5
g	gravitational acceleration	[$\mathrm{m\ s^{-2}}$]	2
g	boundary value	[—]	5
G	linear algebraic operator in DAS	[—]	3, 4, 5
h	enthalpy per unit mass	[$\mathrm{m^2\ s^{-2}}$]	2
h	cell size	[—]	3
I	identity matrix	[—]	4, 5
I	index set	[—]	5
k	turbulent kinetic energy per unit mass	[$\mathrm{m^2\ s^{-2}}$]	2
k	iteration number	[—]	4, 5
l	iteration number	[—]	5
L	length scale	[m]	3
L	discrete operator	[—]	3
L	approximation of the nonlinear operator N	[—]	4, 5
m	block number	[—]	5
M	molar mass	[$\mathrm{kg\ mole^{-1}}$]	2
M	preconditioner	[—]	4, 5

n	unit normal on a surface	[—]	3
n	dimension of the system	[—]	4
$nblock$	total number of blocks	[—]	5
$ngcr1$	total number of PGCR-SIMPLE iterations	[—]	5
$ngcr2$	total number of (D)PGCR-Schwarz iterations	[—]	5
N	number	[—]	2
N	nonlinear algebraic operator in DAS	[—]	3, 4, 5
N	remainder matrix in the splitting	[—]	4
N	number of algebraic equations	[—]	4
p	pressure	[kg m^{-1} s^{-2}]	2
p	dimensionless pressure	[—]	3
p	mesh Péclet number	[—]	3
P	production of energy	[kg m^{-1} s^{-3}]	2
q	energy flux	[kg s^{-4}]	2
q	source term in transport equation		3
q	dimensionless source term in transport equation	[—]	3
q	auxiliary variable	[—]	4, 5
r	residual	[—]	4, 5
R	trivial extension matrix	[—]	5
Re	Reynolds number	[—]	3
R^0	universal gas constant	[m^2 s^{-2} K^{-1}]	2
s	rate-of-strain	[m s^{-2}]	2
s	search direction Krylov method	[—]	4, 5
S	surface	[—]	3
t	time	[s]	2, 3
t	dimensionless time	[—]	3
T	temperature	[K]	2
u	velocity	[m s^{-1}]	2, 3
u	dimensionless velocity	[—]	3
u	unknown	[—]	5
U	diffusion velocity	[m s^{-1}]	2
U	length scale for the velocity	[m s^{-1}]	3
v	velocity	[m s^{-1}]	2
v	dimensionless velocity	[—]	3
v	variable in Krylov method	[—]	4, 5
W	computing work	[—]	4
x	coordinate	[m]	2, 3
x	dimensionless coordinate	[—]	3
x	auxiliary variable	[—]	4
y	initial position of a particle	[—]	3
y	unknown	[—]	4
Y	mass fraction	[—]	2
Y	subspace in deflation method	[—]	4
Z	subspace in deflation method	[—]	4

Greek symbols

Symbol	*Description*	*Units*	*Chapter*
α	number for chemical species	[—]	2
α	relaxation parameter	[—]	4, 5
α	parameter in orthonormalization process	[—]	4
Γ	effective transport coefficient		2, 3
Γ	artificial boundary	[—]	5
δ	small parameter	[—]	3, 4
Δ	operator in the PWI method	[—]	3
ϵ	turbulent dissipation energy per unit mass	[m^2 s^{-3}]	2
ε	inverse of the Péclet number	[—]	3
κ	condition number	[—]	4
λ	thermal conductivity	[kg m s^{-3} K^{-1}]	2
μ	dynamic viscosity	[kg m^{-1} s^{-1}]	2, 3
μ_t	dynamic eddy viscosity	[kg m^{-1} s^{-1}]	2
ρ	density	[kg m^{-3}]	2
ρ	dimensionless density	[—]	3
σ	stress tensor	[kg m^{-1} s^{-2}]	2
σ	dimensionless stress tensor	[—]	3
σ	turbulent Prandtl number	[—]	2
τ	shear-stress tensor	[kg s^{-2}]	2
τ	time step	[—]	3
ϕ	property of a material		2, 3
φ	property of a material		3
φ	dimensionless property of a material	[—]	3
Ω	domain	[—]	3, 5

Subscripts

Subscript	*Description*	*Chapter*
cds	abbreviation for central difference scheme	3
E	abbreviation for east	3
gs	abbreviation for Gauß-Seidel	5
h	reference to enthalpy	2
j	reference to direction	2, 3
j	reference to a grid point	3
jac	abbreviation for Jacobi	5
k	reference to turbulent kinetic energy	2
N	abbreviation for north	3, 4
NW	abbreviation for northeast	4
P	reference to node in cell	4
rad	abbreviation for radiative	2
reac	abbreviation for reaction	2

s	abbreviation for species	2
S	abbreviation for south	3, 4
SE	abbreviation for southeast	4
uds	abbreviation for upwind difference scheme	3
W	abbreviation for west	3
α	reference to a specie α	2
α	reference to a coordinate	3
β	reference to a coordinate	3
ϵ	reference to the turbulent dissipation	2
ϕ	reference to a material property	2
μ	reference to the dynamic viscosity	2

Superscripts

Superscript	*Description*	*Chapter*
b	abbreviation for body force	2, 3
i	reference to position of node in i-direction	4
j	reference to position of node in j-direction	4
T	transpose	4
α	reference to a coordinate	3
β	reference to a coordinate	3

Overlines

Overline	*Description*	*Chapter*
$'$	Reynolds fluctuation	2
$''$	Favre fluctuation	2
$\bar{}$	Reynolds average	2
$\bar{}$	reference to postconditioning	3
$\bar{}$	closure	5
$\bar{}$	reference to Schwarz	5
$\hat{}$	approximation	4, 5
$\tilde{}$	Favre average	2
$\tilde{}$	reference to solution of deflated system	4, 5

Abbreviations

Abbreviation	*Description*
BC	Boundary Condition
BIM	Basic Iterative Method
CG	Conjugate Gradient
CDS	Central Difference Scheme
CFD	Computational Fluid Dynamics

DAS	Differential-Algebraic System
DD	Domain Decomposition
DPGCR	Deflated Preconditioned Generalized Conjugate Residual
FV	Finite Volume
GC	Global Communication
GCR	Generalized Conjugate Residual
HDS	Hybrid Difference Scheme
LC	Local Communication
LHS	Left-Hand Side
MGS	Modified Gram-Schmidt
MPI	Message Passing Interface
PCG	Preconditioned Conjugate Gradient
PDE	Partial Differential Equation
PGCR	Preconditioned Generalized Conjugate Residual
DPCG	Deflated Preconditioned Conjugate Gradient
RCGS	Reorthogonalized Classical Gram-Schmidt
RHS	Right-Hand Side
SIMPLE	Semi-Implicit Method for Pressure-Linked Equations
SIMPLER	Semi-Implicit Method for Pressure-Linked Equations Revised
SIP	Strongly Implicit Procedure
SPD	Symmetric Positive Definite
TDMA	Tri-Diagonal Matrix Algorithm
UDS	Upwind Difference Scheme
XTC	X-stream Test-Case