Chapter 15
Fluid Flow Computation: Incompressible Flows

Abstract In previous chapters the procedure for discretizing and solving the general transport equation for the variable ϕ in the presence of a known velocity field was formulated. In general, the velocity field is not known and has to be computed by solving the set of Navier-Stokes equations. For incompressible flows this task is complicated by the strong coupling that exist between pressure and velocity and by the fact that pressure does not appear as a primary variable in either the momentum or continuity equations. The focus of this chapter is on presenting a method that addresses these two issues, and computes the flow field for incompressible fluid flows. This is accomplished initially on a one dimensional staggered grid, then on a collocated one dimensional grid and finally on a collocated three dimensional unstructured grid. In addition to fully deriving the SIMPLE, SIMPLEC, PRIME and PISO algorithms, the Rhie-Chow interpolation and its extension to transient, relaxation and body force terms are clearly formulated. Finally, the implementation details for a number of frequently encountered boundary conditions are presented.

15.1 The Main Difficulty

The general conservation equation dealt with in previous chapters can be reformed into an equation similar to the continuity and momentum equations. Yet the numerical techniques presented up till now are not enough to allow for the resolution of the Navier-Stokes equations. Solving general fluid flows requires an algorithm [1] that can deal with the pressure velocity coupling. To understand this issue, the continuity and momentum equations are reproduced below.

F. Moukalled et al., *The Finite Volume Method in Computational Fluid Dynamics*,
Fluid Mechanics and Its Applications 113, DOI 10.1007/978-3-319-16874-6_15

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0 \tag{15.1}$$

$$\frac{\partial}{\partial t}[\rho \mathbf{v}] + \nabla \cdot \{\rho \mathbf{v}\mathbf{v}\} = -\nabla p + \nabla \cdot \left\{ \mu \left[\nabla \mathbf{v} + (\nabla \mathbf{v})^{\mathrm{T}} \right] \right\} + \mathbf{f}_b \tag{15.2}$$

That Eqs. (15.1) and (15.2) are nonlinear is not by itself an unsurmountable difficulty, since such a problem is usually handled by adopting an iterative approach. Moreover, Eq. (15.2) is a vector equation, which when written in terms of its components results in a system of scalar equations that can be solved sequentially. Furthermore, the stress tensor can be reformulated into a diffusion-like term and treated implicitly, with its second part (i.e., the transpose of the velocity gradient) evaluated explicitly based on previous iteration values and added to the source. The main issue that cannot be addressed directly with the numerics of the general scalar equation, is the unavailability of an explicit equation for computing the pressure field that appears in the momentum equation.

A review of Eqs. (15.1) and (15.2) reveals that while the velocity field can be computed using the momentum equation, the pressure field appearing in the momentum equation cannot be computed directly from the continuity equation. This strong yet implicit coupling can be made more evident by rewriting the set of equations in a matrix form as

$$\mathbf{A}\mathbf{u} = \begin{pmatrix} \mathbf{F} & \mathbf{B}^{\mathrm{T}} \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ \mathbf{P} \end{pmatrix} = \begin{pmatrix} \mathbf{f}_b \\ \mathbf{0} \end{pmatrix}. \tag{15.3}$$

In this form, Eq. (15.3) shows a zero diagonal block in the system, which is a characteristic of saddle point problems, indicating that it cannot sustain the solution of the pressure and velocity fields by any iterative mean. Consequently, an equation for pressure is required and should be derived.

One approach is to simply reformulate the system of momentum and continuity equations by decomposing matrix $\mathbf{A}$ into a lower ($\mathbf{L}$) and an upper ($\mathbf{U}$) triangular matrices as

$$\mathbf{A} = \begin{pmatrix} \mathbf{F} & \mathbf{B}^{\mathrm{T}} \\ \mathbf{B} & \mathbf{0} \end{pmatrix} = \begin{pmatrix} \mathbf{F} & 0 \\ \mathbf{B} & -\mathbf{B}\mathbf{F}^{-1}\mathbf{B}^{\mathrm{T}} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{F}^{-1}\mathbf{B}^{\mathrm{T}} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} = \mathbf{L}\mathbf{U} \tag{15.4}$$

where the term $-\mathbf{B}\mathbf{F}^{-1}\mathbf{B}^{\mathrm{T}}$ is the Schur complement matrix.

This is in essence the approach that needs to be followed in order to iteratively solve the Navier-Stokes equations. This technique is embodied in the classical segregated SIMPLE (Semi Implicit Method for Pressure Linked Equations) algorithm of Patankar and Spalding [1–3].

The solution procedure is based on reformulating the Navier-Stokes equations in terms of a momentum and a pressure equation, which are then discretized and solved sequentially. The pressure equation is constructed by combining the semi-discretized momentum and continuity equations (approximation of the Schur complement matrix).

The algorithm is driven by a Picard type iterative procedure during which the momentum equation is solved using the pressure field of the previous iteration. The resulting velocity field conserves momentum but not necessarily mass. This velocity field is then used to construct the pressure equation whose solution is used to correct both the pressure and velocity fields so as to enforce mass conservation. A new iteration is then started and the sequence is repeated until the velocity and pressure fields satisfy both mass and momentum conservation.

This algorithm can be described in matrix form as

$$\begin{pmatrix} \mathbf{I} & \mathbf{D}^{-1}\mathbf{B}^{\mathrm{T}} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ \mathbf{p} \end{pmatrix} = \begin{pmatrix} \mathbf{v}^{*} \\ \mathbf{p}^{*} \end{pmatrix} \tag{15.5}$$

followed by an update to the velocity field using

$$\begin{pmatrix} \mathbf{F} & \mathbf{0} \\ \mathbf{B} & -\mathbf{B}\mathbf{D}^{-1}\mathbf{B}^{\mathrm{T}} \end{pmatrix} \begin{pmatrix} \mathbf{v}^{*} \\ \mathbf{p}^{*} \end{pmatrix} = \begin{pmatrix} \mathbf{f}_b \\ \mathbf{0} \end{pmatrix} \tag{15.6}$$

where in Eqs. (15.5) and (15.6) $\mathbf{F}^{-1}$ is approximated by its inverse diagonal, $\mathbf{D}^{-1}$, and the superscript $(^{*})$ refers to intermediate values at the current iteration. The steps required are summarized as follows:

- Solve: $\mathbf{F}\mathbf{v}^{*} = \mathbf{f}_b$
- Solve: $-\mathbf{B}\mathbf{D}^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{p}^{*} = -\mathbf{B}\mathbf{v}^{*}$
- Update: $\mathbf{v} = \mathbf{v}^{*} - \mathbf{D}^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{p}^{*}$
- Update: $\mathbf{p} = \mathbf{p}^{*}$

This kind of splitting is similar to that used in the SIMPLE family of algorithms, which is the subject of this chapter.

15.2 A Preliminary Derivation

The difficulties faced in developing a solution algorithm for incompressible flow problems will be highlighted by performing the discretization in a one dimensional space over the uniform grid displayed in Fig. 15.1. For simplicity, the flow is assumed to be steady. The simplified continuity and momentum equations (written in conservative form) are given by

$$\frac{\partial(\rho u)}{\partial x} = 0 \tag{15.7}$$

$$\frac{\partial(\rho u u)}{\partial x} = \frac{\partial}{\partial x}\left(\mu \frac{\partial u}{\partial x}\right) - \frac{\partial p}{\partial x} \tag{15.8}$$

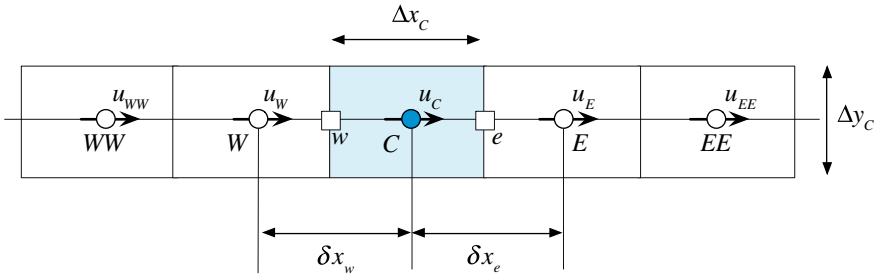

Fig. 15.1 One dimensional domain

15.2.1 Discretization of the Momentum Equation

The discretization of the momentum equation starts by integrating Eq. (15.8) over element C shown in Fig. 15.1 to yield

$$\int_{V_C} \frac{\partial(\rho uu)}{\partial x} dV = \int_{V_C} \frac{\partial}{\partial x}\left(\mu \frac{\partial u}{\partial x}\right) dV - \int_{V_C} \frac{\partial p}{\partial x} dV \tag{15.9}$$

The volume integrals of the convection and diffusion terms in Eq. (15.9) are then transformed into surface integrals by invoking the divergence theorem to give

$$\int_{\partial V_C} (\rho uu) dy = \int_{\partial V_C} \mu \frac{\partial u}{\partial x} dy - \int_{V_C} \frac{\partial p}{\partial x} dV \tag{15.10}$$

Representing the surface integrals by summation of fluxes over the faces of the element, and using a single Gaussian point for the face integrals, the semi-discretized forms of the left and right hand sides of Eq. (15.8) become

$$\underbrace{(\rho u \Delta y)_e}_{\dot{m}_e} u_e + \underbrace{-(\rho u \Delta y)_w}_{\dot{m}_w} u_w = \left(\mu \frac{\partial u}{\partial x} \Delta y\right)_e - \left(\mu \frac{\partial u}{\partial x} \Delta y\right)_w - \int_{V_C} \frac{\partial p}{\partial x} dV \tag{15.11}$$

which can be rewritten as

$$\underbrace{\dot{m}_e u_e + \dot{m}_w u_w}_{Convection} - \underbrace{\left[\left(\mu \frac{\partial u}{\partial x} \Delta y\right)_e - \left(\mu \frac{\partial u}{\partial x} \Delta y\right)_w\right]}_{Diffusion} = - \int_{V_C} \frac{\partial p}{\partial x} dV \tag{15.12}$$

The convection and diffusion terms can be discretized using any of the techniques described in previous chapters to yield an algebraic equation of the form

$$a_C^u u_C + \sum_{F\sim NB(C)} \left(a_F^u u_F\right) = b_C^u - \int_{V_C} \frac{\partial p}{\partial x} dV \tag{15.13}$$

The discretization of the pressure term is deferred till after the discretization of the continuity equation.

15.2.2 Discretization of the Continuity Equation

The discretized form of the continuity equation is obtained by integrating Eq. (15.7) over element C displayed in Fig. 15.1 to give

$$\int_{V_C} \frac{\partial(\rho u)}{\partial x} dV = 0 \tag{15.14}$$

Again making use of the divergence theorem to transform the volume integral into a surface integral and then into summation of fluxes over the faces of the element, the discrete form of the continuity equation is obtained as

$$\sum_{f\sim nb(C)} (\rho u \Delta y)_f = (\rho u \Delta y)_e - (\rho u \Delta y)_w = 0 \tag{15.15}$$

or

$$\sum_{f\sim nb(C)} \dot{m}_f = \dot{m}_e + \dot{m}_w = 0 \tag{15.16}$$

15.2.3 The Checkerboard Problem

The discretization of the pressure term may be accomplished by adopting either of the following two approaches. In the first approach, the volume integral is computed via a single Gaussian integration point resulting in

$$\int_{V_C} \frac{\partial p}{\partial x} dV = \left(\frac{\partial p}{\partial x}\right)_C V_C \tag{15.17}$$

Using a central difference scheme, the discretized form of Eq. (15.17) is obtained as

$$\int_{V_C} \frac{\partial p}{\partial x} dV = \frac{p_E - p_W}{2\Delta x} V_C \tag{15.18}$$

In the second approach, the volume integral of the pressure gradient term is transformed into a surface integral such that

$$\int_{V_C} \frac{\partial p}{\partial x} dV = \int_{\partial V_C} p \, dy \tag{15.19}$$

Rewriting the surface integral as a summation of fluxes over the faces of the element, Eq. (15.19) becomes

$$\int_{V_C} \frac{\partial p}{\partial x} dV = \int_{\partial V_C} p \, dy = p_e \Delta y_e - p_w \Delta y_w = (p_e - p_w)\Delta y = (p_e - p_w) \frac{V_C}{\Delta x} \tag{15.20}$$

Selecting a linear interpolation profile for the variation of pressure, the pressure gradient term can be rewritten as a function of pressure values at the main grid points as

$$\int_{V_C} \frac{\partial p}{\partial x} dV = \left[\frac{1}{2}(p_E + p_C) - \frac{1}{2}(p_C + p_W) \right] \frac{V_C}{\Delta x} = \frac{p_E - p_W}{2\Delta x} V_C \tag{15.21}$$

Thus either approach leads to the same expression involving the pressure difference between the alternating points E and W.

In a similar way, using a linear interpolation profile and noticing that the density is constant and $(\Delta y)_e = (\Delta y)_w = (\Delta y)_C$, the continuity equation can be expressed as

$$u_E - u_W = 0 \tag{15.22}$$

which also relates the velocity at two alternating grid points.

In Eq. (15.21) the pressure gradient term in element C depends on the values of pressure at the two alternating, not consecutive, grid points straddling the element. The same is true for the continuity equation, which enforces conservation only for alternating velocity elements. This implies that non-physical zigzag (or checkerboard) pressure and velocity fields, like the ones shown in Fig. 15.2, will be sensed as uniform fields by the numerical scheme.

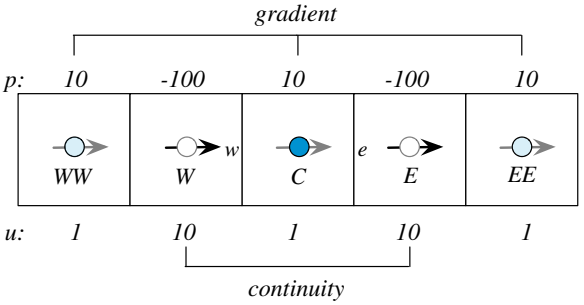

Fig. 15.2 A checkerboard pressure and velocity fields

For the pressure and velocity values shown in Fig. 15.2, the pressure gradient at points W, C, and E are found to be

$$\int_{V_W} \frac{\partial p}{\partial x} dV = (p_C - p_{WW}) \frac{V_W}{2\Delta x_W} = (10 - 10) \frac{V_W}{2\Delta x_W} = 0$$

$$\int_{V_C} \frac{\partial p}{\partial x} dV = (p_E - p_W) \frac{V_C}{2\Delta x_C} = (-100 + 100) \frac{V_C}{2\Delta x_C} = 0$$

$$\int_{V_E} \frac{\partial p}{\partial x} dV = (p_{EE} - p_C) \frac{V_E}{2\Delta x_E} = (10 - 10) \frac{V_E}{2\Delta x_E} = 0$$

and the continuity equation seems to be enforced for each element since

$$\int_{V_W} \frac{\partial u}{\partial x} dV = (u_C - u_{WW}) \frac{V_W}{2\Delta x_W} = (1 - 1) \frac{V_W}{2\Delta x_W} = 0$$

$$\int_{V_C} \frac{\partial u}{\partial x} dV = (u_E - u_W) \frac{V_C}{2\Delta x_C} = (10 - 10) \frac{V_C}{2\Delta x_C} = 0$$

$$\int_{V_E} \frac{\partial u}{\partial x} dV = (u_{EE} - u_C) \frac{V_E}{2\Delta x_E} = (1 - 1) \frac{V_E}{2\Delta x_E} = 0$$

In multi dimensional situations a similar non-physical behavior can arise even if it is harder to visualize. This sets the ground for the next step that presents one approach to resolve this problem.

15.2.4 The Staggered Grid

The culprit in the previous formulation is the uncoupling between the pressure and velocity fields. Coupling can be enforced if the different variables are stored at

staggered locations such that no interpolation is needed to calculate the pressure gradient in the momentum equation and the velocity field in the continuity equation. Such a staggered grid is shown in Fig. 15.3a, b. In the staggered grid the velocity field is stored at cell faces (Fig. 15.3a), while pressure and all other variables are stored at cell centroids (Fig. 15.3b).

With this formulation, the discretized continuity equation for element C becomes

$$\sum_{f\sim nb(C)} \dot{m}_f = \dot{m}_e + \dot{m}_w = 0 \quad \text{or} \quad u_e - u_w = 0 \tag{15.23}$$

with no need for interpolation as the velocity values are available at the e and w locations. Moreover, the momentum equation is integrated over elements similar to element e resulting in the following discretized momentum equation:

$$a_e^u u_e + \sum_{f\sim NB(e)} a_f^u u_f = b_e^u - V_e(\nabla p)_e = b_e^u - V_e \frac{p_E - p_C}{\delta x_e} \tag{15.24}$$

The pressure gradient is related to values at the consecutive grid points straddling the element face with no interpolation needed. Therefore checkerboard pressure and velocity field solutions are inadmissible as they will be easily detected and eliminated by the numerical method.

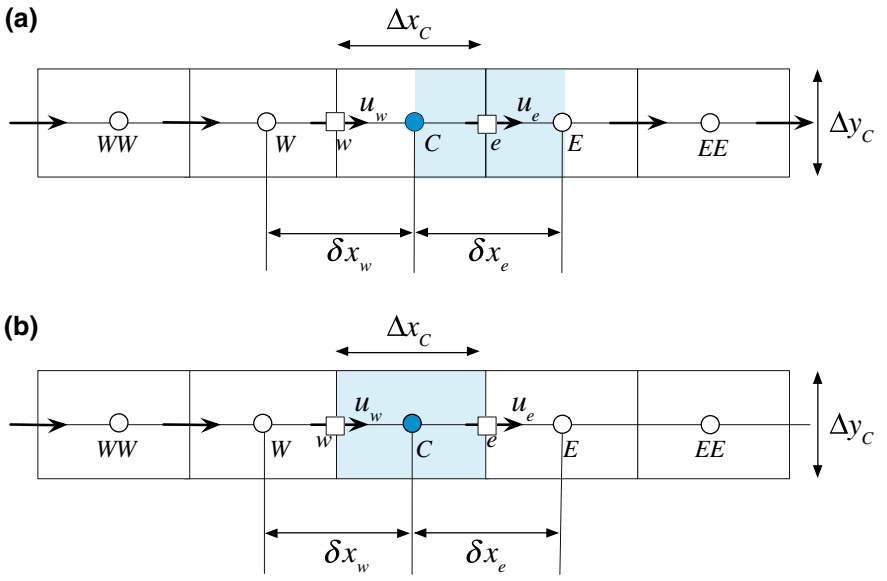

Fig. 15.3 An element for **a** the momentum equation and **b** the continuity equation in a one dimensional staggered grid arrangement

15.2.5 The Pressure Correction Equation

The derivations presented next are based on the work of Patankar and Spalding [2, 3], who developed the initial implementation of the SIMPLE (Semi Implicit Method for Pressure Linked Equations) algorithm.

Starting with the continuity and momentum equations given respectively by (Fig. 15.3)

$$\sum_{f\sim nb(C)} \dot{m}_f = 0 \tag{15.25}$$

$$a_e^u u_e + \sum_{f\sim NB(e)} a_f^u u_f = b_e^u - V_e \left(\frac{\partial p}{\partial x}\right)_e \tag{15.26}$$

the solution proceeds by providing an initial guess for the velocity and pressure fields. Denoting the initial guess or the solution at the starts of any iteration with a superscript (n), then the tentative velocity and pressure fields are given by $u^{(n)}$ and $p^{(n)}$. At any iteration, solving the momentum equation first for the velocity field, the solution obtained is denoted by a superscript * as it is not the final solution at the current iteration. Thus, the momentum equation satisfies

$$a_e^u u_e^* + \sum_{f\sim NB(e)} a_f^u u_f^* = b_e^u - V_e \left(\frac{\partial p^{(n)}}{\partial x}\right)_e \tag{15.27}$$

where the pressure field is still based on values from the previous iteration. The computed velocity field u^* satisfies the momentum equation but not necessarily the continuity equation, since the pressure field is not exact. Therefore a correction is sought to ensure that the velocity (or the mass flow rate) and pressure fields satisfy the continuity equation.

Denoting the correction fields with a superscript prime, i.e., (u', p'), then the sought after velocity and pressure are given by

$$\begin{aligned} u &= u^* + u' \\ p &= p^* + p' \end{aligned} \tag{15.28}$$

Note that the mass flow rate at cell faces will also be corrected according to

$$\begin{aligned} \dot{m}_f &= \dot{m}_f^* + \rho u' S_f^x \\ &= \dot{m}_f^* + m_f' \end{aligned} \tag{15.29}$$

such that the exact mass flow rate satisfies the continuity equation, i.e.,

$$\dot{m}_e + \dot{m}_w = \dot{m}_e^* + \dot{m}_e' + \dot{m}_w^* + \dot{m}_w' = 0 \tag{15.30}$$

which can be rewritten as

$$\dot{m}_e' + \dot{m}_w' = -\dot{m}_e^* - \dot{m}_w^* \tag{15.31}$$

This is an interesting form of the continuity equation showing that once the computed mass flow rate reaches the exact solution and satisfies the continuity equation, then the RHS becomes zero leading to a zero correction field. Thus it is the mass conservation error of the current fields that drives the correction field. The mass flow rates and mass flow rate corrections at an element faces are given by

$$\begin{aligned} \dot{m}_e &= \rho \mathbf{v}_e^* \cdot \mathbf{S}_e = \rho u_e^* S_e^x = \rho u_e^* \Delta y_e \\ \dot{m}_w &= \rho \mathbf{v}_w^* \cdot \mathbf{S}_w = \rho u_w^* S_w^x = -\rho u_w^* \Delta y_w \end{aligned} \tag{15.32}$$

and

$$\begin{aligned} \dot{m}_e' &= \rho \mathbf{v}_e' \cdot \mathbf{S}_e = \rho u_e' S_e^x = \rho u_e' \Delta y_e \\ \dot{m}_w' &= \rho \mathbf{v}_w' \cdot \mathbf{S}_w = \rho u_w' S_w^x = -\rho u_w' \Delta y_w \end{aligned} \tag{15.33}$$

where in Eqs. (15.32) and (15.33) the fact that $S_e^x = \Delta y_e$ and $S_w^x = -\Delta y_w$ has been used.

The pressure field does not appear in Eq. (15.31) and to bring it into the equation, the discrete form of the momentum equation is used. The process starts by rewriting Eq. (15.26) in a more compact form as

$$u_e + H_e(u) = B_e^u - D_e^u \left(\frac{\partial p}{\partial x} \right)_e \tag{15.34}$$

where

$$H_e(u) = \sum_{f \sim NB(e)} \frac{a_f^u}{a_e^u} u_f \quad B_e^u = \frac{b_e^u}{a_e^u} \quad \text{and} \quad D_e^u = \frac{V_e}{a_e^u} \tag{15.35}$$

For the case of the computed velocity field, the above equation is written as

$$u_e^* + H_e(u^*) = B_e^u - D_e^u \left(\frac{\partial p^{(n)}}{\partial x} \right)_e \tag{15.36}$$

Subtracting the computed momentum equation, Eq. (15.36), from the exact one, Eq. (15.34), an equation for the correction field is obtained as

$$u'_e + \underline{H_e(u')} = -D^u_e \left(\frac{\partial p'}{\partial x}\right)_e \tag{15.37}$$

A similar approach is used for the w face yielding

$$u'_w + \underline{H_w(u')} = -D^u_w \left(\frac{\partial p'}{\partial x}\right)_w \tag{15.38}$$

Substituting Eq. (15.33) into the continuity equation, Eq. (15.31), its expanded form becomes

$$\rho_e u'_e \Delta y_e + \left(-\rho_w u'_w \Delta y_w\right) = -(\dot{m}^*_e + \dot{m}^*_w). \tag{15.39}$$

Then replacing the discrete forms of u'_e and u'_w computed from Eqs. (15.37) and (15.38), respectively, in Eq. (15.39), an equation involving pressure correction is obtained and is given by

$$\begin{aligned} &\rho_e \left[-H_e(u') - D^u_e \left(\frac{\partial p'}{\partial x}\right)_e\right] \Delta y_e \\ &\quad - \rho_w \left[-H_w(u') - D^u_w \left(\frac{\partial p'}{\partial x}\right)_w\right] \Delta y_w = -\left(\dot{m}^*_e + \dot{m}^*_w\right) \end{aligned} \tag{15.40}$$

In this equation the pressure field appears in a diffusion like form, which after discretization becomes

$$\begin{aligned} &\rho_e \left[-H_e(u') - D^u_e \left(\frac{p'_E - p'_C}{\Delta x}\right)_e\right] \Delta y_e \\ &\quad + \rho_w \left[-H_w(u') - D^u_w \left(\frac{p'_C - p'_W}{\Delta x}\right)_w\right] (-\Delta y_w) = -\left(\dot{m}^*_e + \dot{m}^*_w\right) \end{aligned} \tag{15.41}$$

or

$$\begin{aligned} &- \rho_e D^u_e \left(\frac{\Delta y_w}{\Delta x_w}\right) \left(p'_E - p'_C\right) - \rho_w D^u_w \left(-\frac{\Delta y_w}{\Delta x_w}\right) \left(p'_C - p'_W\right) \\ &\quad = -\left(\dot{m}^*_e + \dot{m}^*_w\right) + \left(\rho_e H_e(u') \Delta y_e + \rho_w H_w(u')(-\Delta y_w)\right) \end{aligned} \tag{15.42}$$

Rearranging, the pressure correction equation is formulated as

$$a^{p'}_C p'_C + a^{p'}_E p'_E + a^{p'}_W p'_W = b^{p'}_C \tag{15.43}$$

where

$$\begin{aligned} a_E^{p'} &= -\frac{\rho_e D_e^u \Delta y_e}{\delta x_e} \\ a_W^{p'} &= -\frac{\rho_w D_w^u \Delta y_w}{\delta x_w} \\ a_C^{p'} &= -\left(a_E^{p'} + a_W^{p'}\right) \\ b_C^{p'} &= -\left(\dot{m}_e^* + \dot{m}_w^*\right) + \underline{\left[\rho_e \Delta y_e H_e(u') - \rho_w \Delta y_w H_w(u')\right]} \end{aligned} \tag{15.44}$$

The underlined terms in Eqs. (15.37), (15.38), and (15.44) involve corrections which become zero at the state of convergence. Therefore they have no effect on the final solution. Different approximations to these terms result in different algorithms as will be explained later. In the original SIMPLE algorithm these terms are simply neglected. Moreover for one dimensional constant area situations Δy may be set to 1 and dropped from the equations.

15.2.6 The SIMPLE Algorithm on Staggered Grid

Using the momentum and pressure correction equations, a solution to the flow problem can be obtained. In the SIMPLE algorithm this solution is found iteratively by generating pressure and velocity fields that consecutively satisfy the momentum and continuity equations, while approaching the final solution (which satisfies both equations) at every iteration [4–6]. This sequential, rather than simultaneous, solution of the equations is denoted in the literature by the segregated approach. The sequence of events in the segregated SIMPLE algorithm can be summarized as follows:

1. Start with a guessed pressure and velocity fields $p^{(n)}$ and $u^{(n)}$, respectively.
2. Solve the momentum equation given by Eq. (15.27) to obtain a new velocity field u_f^*.
3. Update the mass flow rates using the momentum satisfying velocity field to obtain the $\dot{m}_f^*$ field.
4. Using the new mass flow rates solve the pressure correction equation to obtain a pressure correction field p'.
5. Update the pressure and velocity fields to obtain continuity-satisfying fields using the following equations:

$$
\begin{aligned}
u_f^{**} &= u_f^* + u_f' \quad u_f' = -D_f^u \left(\frac{\partial p'}{\partial x}\right)_f \\
p_C^* &= p_C^{(n)} + p_C' \\
\dot{m}_f^{**} &= \dot{m}_f^* + \dot{m}_f' \quad \dot{m}_f' = -\rho_f D_f^u \Delta y_f \left(\frac{\partial p'}{\partial x}\right)_f
\end{aligned}
\tag{15.45}
$$

6. set $u^{(n)} = u^{**}$ and $p^{(n)} = p^*$
7. Go back to step 2 and repeat until convergence.

The SIMPLE algorithm is best illustrated via the example presented next.

Example 1

Flow in a Pipe Network

A portion of a water pipe system is shown in Fig. 15.4. *The momentum equation for the flow in the pipes can be written as*

$$\dot{m} = \rho u A = -D\Delta P$$

where $D_A = 0.5$, $D_B = D_F = 0.4$, $D_C = D_E = 0.3$, $D_D = 0.19$, $D_G = 0.1875$, *and* $D_H = 0.35$. *Using the SIMPLE algorithm, calculate the unknown mass flow rates and pressures in the system.*

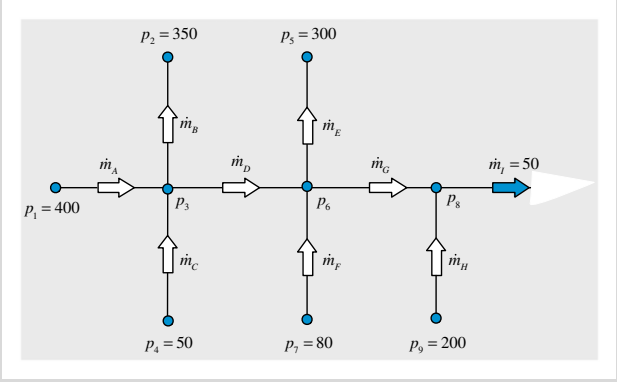

Fig. 15.4 A portion of a water pipe system

Solution

In this system, the mass flow rate field is used as a variable instead of the velocity field. This is not problematic since the momentum equation has been simplified by dropping the convection and diffusion terms as their values are negligible compared to the pressure head.

The solution using the SIMPLE algorithm starts by first computing an initial velocity field using the assumed pressure field, and then predicting a pressure field that enforces continuity on the just computed velocity field.

This procedure is summarized as

1. Start with a guessed pressure field.
2. Compute the mass flow rates using the given momentum equation.
3. Construct a pressure correction equation that enforces continuity (mass conservation) and use it to correct the pressure and velocity fields.

No iterations will be needed since there are no non-linear effects induced by a convection term.

step 1

Start by assigning guessed values to the pressure at the locations where solutions are to be found. Thus assume the following:

$p_3^{(n)} = 300 \quad p_6^{(n)} = 200 \quad p_8^{(n)} = 120$ (other values could have been used)

step 2

Based on the assumed pressure values calculate the various mass flow rates using the momentum equations according to

$$\dot{m}_A^* = D_A\,(p_1 - p_3^{(n)}) = 0.5 * (400 - 300) = 50$$
$$\dot{m}_B^* = D_B\,(p_3^{(n)} - p_2) = 0.4 * (300 - 350) = -20$$
$$\dot{m}_C^* = D_C\,(p_4 - p_3^{(n)}) = 0.3 * (50 - 300) = -75$$
$$\dot{m}_D^* = D_D\,(p_3^{(n)} - p_6^{(n)}) = 0.19 * (300 - 200) = 19$$
$$\dot{m}_E^* = D_E\,(p_6^{(n)} - p_5) = 0.3 * (200 - 300) = -30$$
$$\dot{m}_F^* = D_F\,(p_7 - p_6^{(n)}) = 0.4 * (80 - 200) = -48$$
$$\dot{m}_G^* = D_G\,(p_6^{(n)} - p_8^{(n)}) = 0.1875 * (200 - 120) = 15$$
$$\dot{m}_H^* = D_H\,(p_9 - p_8^{(n)}) = 0.35 * (200 - 120) = 28$$

step 3

Check whether the mass flow rates satisfy continuity by computing $\sum_{\sim k} \dot{m}_k^*$ at all interior points, i.e.,

$$\textit{Node } 3\text{:}\ \sum_{\sim k} \left(\dot{m}_k^*\right) = \dot{m}_B^* + \dot{m}_D^* - \dot{m}_A^* - \dot{m}_C^* = -20 + 19 - 50 + 75 = 24$$

$$\textit{Node } 6\text{:}\ \sum_{\sim k} \left(\dot{m}_k^*\right) = \dot{m}_G^* + \dot{m}_E^* - \dot{m}_D^* - \dot{m}_F^* = 15 - 30 - 19 + 48 = 14$$

$$\textit{Node } 8\text{:}\ \sum_{\sim k} \left(\dot{m}_k^*\right) = \dot{m}_I^* - \dot{m}_G^* - \dot{m}_H^* = 50 - 15 - 28 = 7$$

Since mass conservation is not satisfied, correction fields are needed and pressure correction equations are derived as follows:

$$\sum_{\sim k}\left(\dot{m}_k^* + \dot{m}_k'\right) = 0 \Rightarrow \sum_{\sim k}\left(\dot{m}_k'\right) = -\sum_{\sim k}\left(\dot{m}_k^*\right)$$

In term of pressure corrections, mass flow rate corrections can be expressed as

$$\begin{aligned}
\dot{m}_A' &= D_A\left(-p_3'\right)\\
\dot{m}_B' &= D_B\left(p_3'\right)\\
\dot{m}_C' &= D_C\left(-p_3'\right)\\
\dot{m}_D' &= D_D\left(p_3' - p_6'\right)\\
\dot{m}_E' &= D_E\left(p_6'\right)\\
\dot{m}_F' &= D_F\left(-p_6'\right)\\
\dot{m}_G' &= D_G\left(p_6' - p_8'\right)\\
\dot{m}_H' &= D_H\left(-p_8'\right)
\end{aligned}$$

Note that p_1', p_2', p_4', p_5' and p_7' are set to zero since the corresponding pressure values are known and hence represent the exact values.

The flow field at nodes 3, 6, and 8 are respectively given by

$$\begin{aligned}
\dot{m}_B' + \dot{m}_D' - \dot{m}_A' - \dot{m}_C' &= -\left(\dot{m}_B^* + \dot{m}_D^* - \dot{m}_A^* - \dot{m}_C^*\right)\\
\dot{m}_G' + \dot{m}_E' - \dot{m}_D' - \dot{m}_F' &= -\left(\dot{m}_G^* + \dot{m}_E^* - \dot{m}_D^* - \dot{m}_F^*\right)\\
-\dot{m}_G' - \dot{m}_H' &= -\left(\dot{m}_I^* - \dot{m}_G^* - \dot{m}_H^*\right)
\end{aligned}$$

and using pressure corrections as

$$\begin{aligned}
&D_B\left(p_3'\right) + D_D\left(p_3' - p_6'\right) - D_A\left(-p_3'\right) - D_C\left(-p_3'\right) = -\left(\dot{m}_B^* + \dot{m}_D^* - \dot{m}_A^* - \dot{m}_C^*\right)\\
&D_G\left(p_6' - p_8'\right) + D_E\left(p_6'\right) - D_D\left(p_3' - p_6'\right) - D_F\left(-p_6'\right) = -\left(\dot{m}_G^* + \dot{m}_E^* - \dot{m}_D^* - \dot{m}_F^*\right)\\
&-D_G\left(p_6' - p_8'\right) - D_H\left(-p_8'\right) = -\left(\dot{m}_I^* - \dot{m}_G^* - \dot{m}_H^*\right)
\end{aligned}$$

After simplification the above equations become

$$\begin{aligned}
1.39\left(p_3'\right) - 0.19\left(p_6'\right) &= -\left(\dot{m}_B^* + \dot{m}_D^* - \dot{m}_A^* - \dot{m}_C^*\right)\\
1.0775\left(p_6'\right) - 0.1875\left(p_8'\right) - 0.19\left(p_3'\right) &= -\left(\dot{m}_G^* + \dot{m}_E^* - \dot{m}_D^* - \dot{m}_F^*\right)\\
-0.1875\left(p_6'\right) + 0.5375\left(p_8'\right) &= -\left(\dot{m}_I^* - \dot{m}_G^* - \dot{m}_H^*\right)
\end{aligned}$$

Substituting the tentative mass flow rates, the various correction fields satisfy

$$\begin{aligned} &1.39\left(p_3'\right) - 0.19\left(p_6'\right) = -24 \\ &1.0775\left(p_6'\right) - 0.1875\left(p_8'\right) - 0.19\left(p_3'\right) = -14 \\ &-0.1875\left(p_6'\right) + 0.5375\left(p_8'\right) = -7 \end{aligned}$$

Solving the system of pressure correction equations yields

$$\begin{cases} p_3' = -20 \\ p_6' = -20 \\ p_8' = -20 \end{cases}$$

With the pressure correction computed, the velocity and pressure fields can now be updated to produce a mass conserving velocity field. The mass flow rates are computed as

$$\begin{aligned} \dot{m}_A^{**} &= \dot{m}_A^* + \dot{m}_A' = \dot{m}_A^* - 0.5p_3' = 50 - 0.5(-20) = 60 \\ \dot{m}_B^{**} &= \dot{m}_B^* + \dot{m}_B' = \dot{m}_B^* + 0.4p_3' = -20 + 0.4(-20) = -28 \\ \dot{m}_C^{**} &= \dot{m}_C^* + \dot{m}_C' = \dot{m}_C^* - 0.3p_3' = -75 - 0.3(-20) = -69 \\ \dot{m}_D^{**} &= \dot{m}_D^* + \dot{m}_D' = \dot{m}_D^* + 0.19\left(p_3' - p_6'\right) = 19 + 0.19(-20 + 20) = 19 \\ \dot{m}_E^{**} &= \dot{m}_E^* + \dot{m}_E' = \dot{m}_E^* + 0.3p_6' = -30 + 0.3(-20) = -36 \\ \dot{m}_F^{**} &= \dot{m}_F^* + \dot{m}_F' = \dot{m}_F^* - 0.4p_6' = -48 - 0.4(-20) = -40 \\ \dot{m}_G^{**} &= \dot{m}_G^* + \dot{m}_G' = \dot{m}_G^* + 0.1875\left(p_6' - p_8'\right) = 15 + 0.1875(-20 + 20) = 15 \\ \dot{m}_H^{**} &= \dot{m}_H^* + \dot{m}_H' = \dot{m}_H^* - 0.35p_8' = 28 - 0.35(-20) = 35 \end{aligned}$$

while the pressure is updated using

$$\begin{aligned} p_3^* &= p_3^{(n)} + p_3' = 300 - 20 = 280 \\ p_6^* &= p_6^{(n)} + p_6' = 200 - 20 = 180 \\ p_8^* &= p_8^{(n)} + p_8' = 120 - 20 = 100 \end{aligned}$$

Treat the corrected values as a new guess and repeat. Better estimate for the mass flow rates are computed using the momentum equations as

$$\dot{m}_A^* = D_A\left(p_1 - p_3^{(n)}\right) = 0.5 * (400 - 280) = 60$$
$$\dot{m}_B^* = D_B\left(p_3^{(n)} - p_2\right) = 0.4 * (280 - 350) = -28$$
$$\dot{m}_C^* = D_C\left(p_4 - p_3^{(n)}\right) = 0.3 * (50 - 280) = -69$$
$$\dot{m}_D^* = D_D\left(p_3^{(n)} - p_6^{(n)}\right) = 0.19 * (280 - 180) = 19$$
$$\dot{m}_E^* = D_E\left(p_6^{(n)} - p_5\right) = 0.3 * (180 - 300) = -36$$
$$\dot{m}_F^* = D_F\left(p_7 - p_6^{(n)}\right) = 0.4 * (80 - 180) = -40$$
$$\dot{m}_G^* = D_G\left(p_6^{(n)} - p_8^{(n)}\right) = 0.1875 * (180 - 100) = 15$$
$$\dot{m}_H^* = D_H\left(p_9 - p_8^{(n)}\right) = 0.35 * (200 - 100) = 35$$

The imbalance in the mass flow rate at nodes 3, 6, and 8 are computed as

$$\sum_{\sim k}\left(\dot{m}_k^*\right) = \dot{m}_B^* + \dot{m}_D^* - \dot{m}_A^* - \dot{m}_C^* = -28 + 19 - 60 + 69 = 0$$
$$\sum_{\sim k}\left(\dot{m}_k^*\right) = \dot{m}_G^* + \dot{m}_E^* - \dot{m}_D^* - \dot{m}_F^* = 15 - 36 - 19 + 40 = 0$$
$$\sum_{\sim k}\left(\dot{m}_k^*\right) = \dot{m}_I^* - \dot{m}_G^* - \dot{m}_H^* = 50 - 15 - 35 = 0$$

The pressure correction equations become

$$\begin{aligned} &1.39\left(p_3'\right) - 0.19\left(p_6'\right) = 0 \\ &1.0775\left(p_6'\right) - 0.1875\left(p_8'\right) - 0.19\left(p_3'\right) = 0 \\ &-0.1875\left(p_6'\right) + 0.5375\left(p_8'\right) = 0 \end{aligned}$$

The solution to the pressure correction field is found to be

$$\begin{cases} p_3' = 0 \\ p_6' = 0 \\ p_8' = 0 \end{cases}$$

Thus the solution is obtained in one iteration.

15.2.7 Pressure Correction Equation in Two Dimensional Staggered Cartesian Grids

In a two dimensional Cartesian grid, three grid systems are used. One for the u-velocity component, a second one for the v-velocity component, and a third grid system for the pressure and other variables as illustrated in Fig. 15.5.

The derivations presented above for the pressure correction equation in one dimensional domains can be easily extended into multi dimensional situations. For element C shown in Fig. 15.6, the pressure correction equation is obtained as

$$a_C^{p'} p'_C + a_E^{p'} p'_E + a_W^{p'} p'_W + a_N^{p'} p'_N + a_S^{p'} p'_S = b_C^{p'} \tag{15.46}$$

where

$$\begin{aligned} a_E^{p'} &= -\frac{\rho_e D_e^u \Delta y_C}{\delta x_e} \quad a_W^{p'} = -\frac{\rho_w D_w^u \Delta y_C}{\delta x_w} \\ a_N^{p'} &= -\frac{\rho_n D_n^v \Delta x_C}{\delta y_n} \quad a_S^{p'} = -\frac{\rho_s D_s^v \Delta x_C}{\delta y_s} \\ a_C^{p'} &= -\left(a_E^{p'} + a_W^{p'} + a_N^{p'} + a_S^{p'}\right) \\ b_C^{p'} &= -\left(\dot{m}_e^* + \dot{m}_w^* + \dot{m}_n^* + \dot{m}_s^*\right) \end{aligned} \tag{15.47}$$

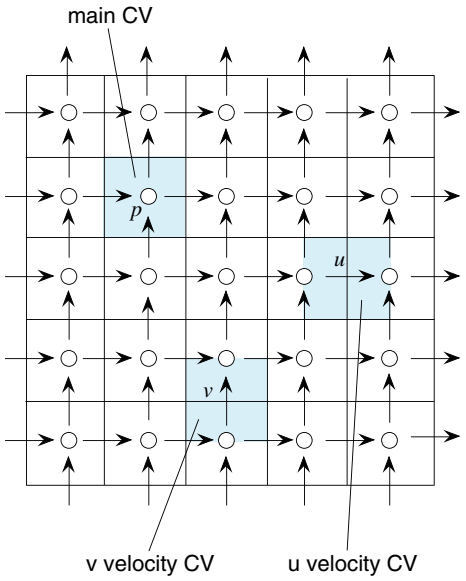

Fig. 15.5 u, v, and p elements in a two dimensional Cartesian staggered grid

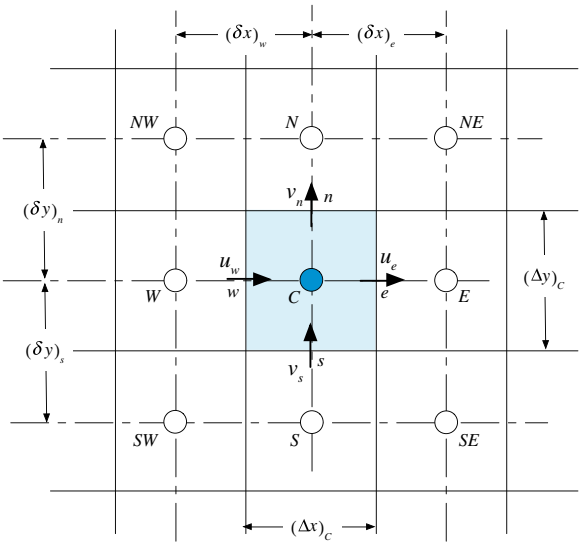

Fig. 15.6 A two dimensional Cartesian element for the derivation of the pressure correction equation

Example 2

In the two dimensional problem shown in Fig. 15.7*, the following quantities are given* $u_w = 50$, $v_s = 20$, $p_N = 0$ *and* $p_E = 10$.

The flow is steady and the density is uniform and equal to 1. *The momentum equations for* u_e *and* v_n *are given by*

$$u_e = -d_e(p_E - p_C)$$
$$v_n = -d_n(p_N - p_C)$$

where the constants $d_e = 1$ *and* $d_n = 0.25$. *The element shown has* $\Delta x = \Delta y = 1$. *Use the SIMPLE algorithm to compute the values of* u_e, v_n, *and* p_C.

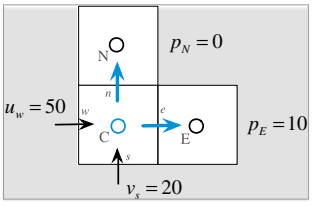

Fig. 15.7 The two dimensional domain used in Example 2

Solution

Start by assigning a guessed value to the pressure at the C location where solution is to be found. Thus assume the following:

$p_C^{(n)} = 100$ (other values could have been guessed)

Based on the assumed pressure value calculate the various velocities using the momentum equation according to

$$u_e^* = -d_e\left(p_E - p_C^{(n)}\right) = -1 * (10 - 100) = 90$$
$$v_n^* = -d_n\left(p_N - p_C^{(n)}\right) = -0.25 * (0 - 100) = 25$$

Since the density is uniform and equal to 1 and $\Delta x = \Delta y = 1$, then

$$\dot{m}_e^* = u_e^* = 90$$
$$\dot{m}_n^* = v_n^* = 25$$

Check whether the mass flow rates satisfy continuity over element C by computing $\sum_{\sim f} \dot{m}_f^*$ at the element faces.

$$\sum\nolimits_{\sim f}\left(\dot{m}_f^*\right) = \dot{m}_e^* - \dot{m}_w^* + \dot{m}_n^* - \dot{m}_s^* = 90 - 50 + 25 - 20 = 45$$

In the above mass conservation equation the negative sign for $\dot{m}_w^*$ and $\dot{m}_s^*$ is explicitly used. Since mass conservation is not satisfied, correction fields are needed and a pressure correction equation is derived as follows:

$$\sum_{\sim f}\left(\dot{m}_f^* + \dot{m}_f'\right) = 0 \Rightarrow \sum_{\sim f}\left(\dot{m}_f'\right) = -\sum_{\sim f}\left(\dot{m}_f^*\right)$$

In term of pressure corrections, mass flow rate corrections can be expressed as

$$\dot{m}_e' = u_e' = p_C' \quad \dot{m}_n' = v_n' = 0.25\, p_C'$$

The correction equation becomes

$$\dot{m}_e' + \dot{m}_n' = -\sum_{\sim f}\dot{m}_f^* \Rightarrow 1.25 p_C' = -45 \Rightarrow p_C' = -36$$

Applying the correction to the mass flow rate and pressure fields, continuity satisfying fields are obtained as

$$\dot{m}_e^{**} = \dot{m}_e^* + \dot{m}_e' = \dot{m}_e^* + p_C' = 90 - 36 = 54$$
$$\dot{m}_n^{**} = \dot{m}_n^* + \dot{m}_n' = \dot{m}_n^* + 0.25p_C' = 25 - 0.25(36) = 16$$
$$p_C^{**} = p_C^* + p_C' = 100 - 36 = 64$$

Treat the corrected values as a new guess and repeat.

$$\dot{m}_e^* = -d_e\left(p_E - p_C^*\right) = -1 * (10 - 64) = 54$$
$$\dot{m}_n^* = -d_n\left(p_N - p_C^*\right) = -0.25 * (0 - 64) = 16$$

The imbalance in the mass flow rate is computed as

$$\sum_{\sim f}\left(\dot{m}_f^*\right) = \dot{m}_e^* - \dot{m}_w^* + \dot{m}_n^* - \dot{m}_s^* = 54 - 50 + 16 - 20 = 0$$

The pressure correction equations become

$$1.25\left(p_c'\right) = 0$$

The solution to the pressure correction field is found to be

$$p_C' = 0$$

Thus the solution is obtained in one iteration as

$$u_e = 54$$
$$v_n = 16$$
$$p_C = 64$$

15.2.8 Pressure Correction Equation in Three Dimensional Staggered Cartesian Grid

Without going into details and for completeness of presentation, the pressure correction equation over the three dimensional staggered Cartesian grid shown in Fig. 15.8, where the u, v, and w velocity components are stored at the (e, w), (n, s), and (t, b) element faces, respectively, is given by

$$a_C^{p'}p_C' + a_E^{p'}p_E' + a_W^{p'}p_W' + a_N^{p'}p_N' + a_S^{p'}p_S' + a_T^{p'}p_T' + a_B^{p'}p_B' = b_C^{p'} \tag{15.48}$$

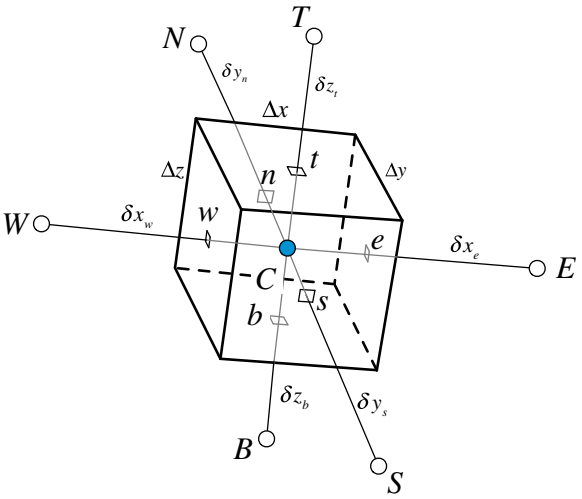

Fig. 15.8 A three dimensional Cartesian element for the derivation of the pressure correction equation

where

$$
\begin{aligned}
a_E^{p'} &= -\frac{\rho_e D_e^u \Delta y_e \Delta z_e}{\delta x_e} \quad a_W^{p'} = -\frac{\rho_w D_w^u \Delta y_w \Delta z_w}{\delta x_w} \\
a_N^{p'} &= -\frac{\rho_n D_n^v \Delta x_n \Delta z_n}{\delta y_n} \quad a_S^{p'} = -\frac{\rho_s D_s^v \Delta x_s \Delta z_s}{\delta y_s} \\
a_T^{p'} &= -\frac{\rho_t D_t^w \Delta x_t \Delta y_t}{\delta z_t} \quad a_B^{p'} = -\frac{\rho_b D_b^w \Delta x_b \Delta y_b}{\delta z_b} \\
a_C^{p'} &= -\left(a_E^{p'} + a_W^{p'} + a_N^{p'} + a_S^{p'} + a_T^{p'} + a_B^{p'}\right) \\
b_C^{p'} &= -\left(\dot{m}_e^* + \dot{m}_w^* + \dot{m}_n^* + \dot{m}_s^* + \dot{m}_t^* + \dot{m}_b^*\right)
\end{aligned}
\tag{15.49}
$$

15.3 Disadvantages of the Staggered Grid

The use of staggered grids was critical to the development of the SIMPLE algorithm. Nevertheless adopting a staggered grid arrangement has its disadvantages. As mentioned above, in two and three dimensions, three and four staggered grid systems, respectively, are required with the velocity components integrated over different elements, as shown in Fig. 15.5 for a two dimensional situation.

Besides the memory requirement to store a grid system for every velocity component and a grid system for pressure and other variables, the staggering procedure itself becomes an issue for non-Cartesian grids and more so for unstructured grids.

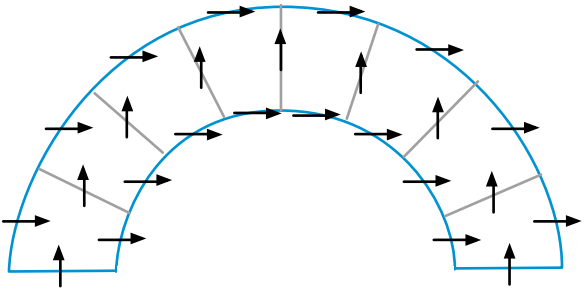

Fig. 15.9 Staggered Cartesian velocity components in a curvilinear grid system

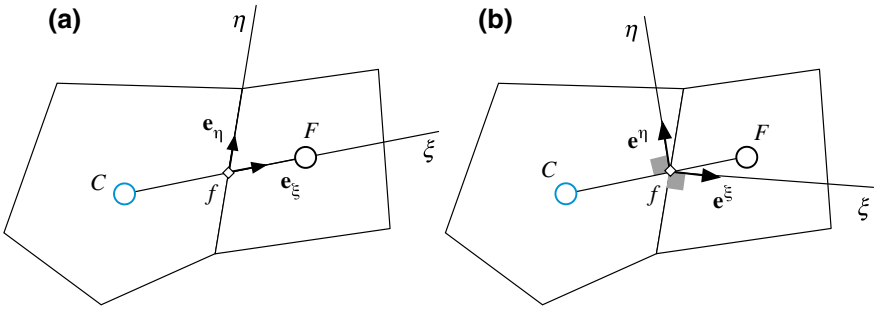

Fig. 15.10 Curvilinear coordinates **a** Covariant component and **b** contra-variant components

In curvilinear grids, the use of Cartesian velocity components can lead to problems when one or more of the surfaces become aligned with the staggered velocity component as shown in Fig. 15.9.

Therefore a better alternative in this case is to use either covariant or contra-variant curvilinear velocity components, as shown in Fig. 15.10a, b, respectively.

An example of staggering using contra-variant velocity components is shown in Fig. (15.11).

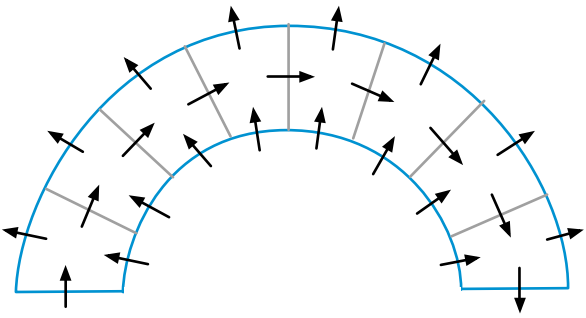

Fig. 15.11 Curvilinear velocities: staggering using contra-variant velocity components

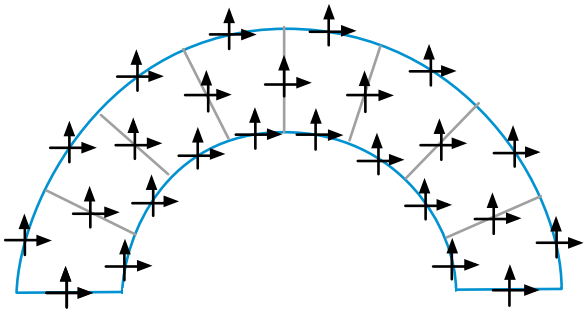

Fig. 15.12 Cartesian components defined at each face

Unfortunately complications arise when discretizing the momentum equations in curvilinear coordinates [1, 3, 4, 7], due to the increased complexity in the treatment of the diffusion term, and because the equations gain non-conservative terms.

Another option shown in Fig. 15.12 is to stagger all Cartesian velocity components in all directions so as to have all velocity components at all faces. This would double (in two dimensions) or triple (in three dimensions) the number of momentum equations to be solved. The problem is further complicated in the case of an unstructured grid. In this case there is no obvious staggering direction, and the only way for a staggering concept to apply is by changing the size of the cell elements used for the pressure and velocity components, or by resorting to staggering all velocity components along all the faces, again dramatically increasing the number of variables to be solved. Finally, the geometric information stored is more than doubled, as a new unstructured grid need to be used for the velocity components.

It turns out that the use of a cell-centered collocated grid system (Fig. 15.13), where all variables are stored at the same location (the cell centroid), is a more attractive solution. It is worth noting that while the velocity components are stored at the centroids of the elements as is the case for pressure or any other variable, the mass flux, a scalar value, in a collocated grid is stored at the element faces. The mass flux can actually be viewed as a contra-variant component, except that in this case it is computed using a custom interpolation of the discrete momentum equation, known as the Rhie-Chow interpolation, which is the subject of the next section.

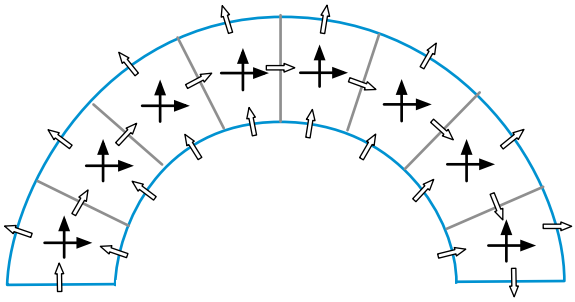

Fig. 15.13 Collocated grid arrangement

15.4 The Rhie-Chow Interpolation

The deficiency in the original collocated formulation presented earlier was in the linear interpolation used to calculate the velocities at the element faces. This interpolation resulted in decoupling the pressure and velocity values at the cell level giving rise to the checkerboard problem. In 1983, Rhie and Chow [8] reported on an interpolation procedure that allowed the formulation of the SIMPLE algorithm on a collocated grid [9–16]. In their method a dissipation term, representing the difference between two estimates of the cell face pressure gradient, is added to the linearly interpolated cell face velocity. As shown in Fig. 15.14 the two pressure gradient estimates are based on different grid stencils.

This procedure will be shown to be equivalent to constructing a pseudo-momentum equation at the element face with its coefficients linearly interpolated from the coefficients of the momentum equations at the centroids of the elements straddling the face and its pressure gradient computed using a small grid stencil. In that respect, the Rhie-Chow interpolation simply mimics the small stencil pressure-velocity coupling of the staggered grid arrangement.

Starting with the discretized x-momentum equations for cells C and F, which are

$$\begin{aligned} u_C + H_C[u] &= B_C^u - D_C^u \left(\frac{\partial p}{\partial x}\right)_C \\ u_F + H_F[u] &= B_F^u - D_F^u \left(\frac{\partial p}{\partial x}\right)_F \end{aligned} \tag{15.50}$$

A u_f velocity equation similar to that of Eq. (15.50), with the pressure gradient linked to the local neighboring pressure values, as illustrated in Fig. 15.14, will have the following form:

$$u_f + H_f[u] = B_f^u - D_f^u \left(\frac{\partial p}{\partial x}\right)_f . \tag{15.51}$$

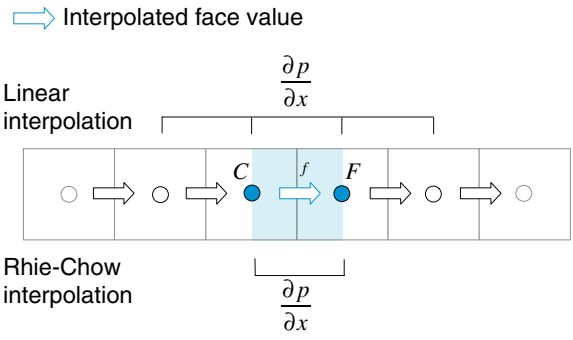

Fig. 15.14 The two pressure gradient estimates in the Rhie-Chow interpolation technique

Since in a collocated grid, the coefficients of this equation cannot be directly computed, they are approximated by interpolation from the coefficients of the neighboring nodes. Using a linear interpolation profile, these coefficients are computed as

$$\begin{aligned} H_f[u] &= \frac{1}{2}\left(H_C[u] + H_F[u]\right) = \overline{H_f}[u] \\ B_f^u &= \frac{1}{2}\left(B_C^u + B_F^u\right) = \overline{B_f^u} \\ D_f^u &= \frac{1}{2}\left(D_C^u + D_F^u\right) = \overline{D_f^u} \end{aligned} \tag{15.52}$$

Employing the values given in Eq. (15.52), the pseudo-momentum equation at the element face becomes

$$u_f + \overline{H_f}[u] = \overline{B_f^u} - \overline{D_f^u}\left(\frac{\partial p}{\partial x}\right)_f \tag{15.53}$$

This is in all practical sense the momentum equation on a "staggered" grid, which is reconstructed using the collocated grid momentum coefficients.

In all above equations and for later use, values with an over bar are obtained by linear interpolation between the values at points C and F according to

$$\overline{\square}_f = g_C \square_C + g_F \square_F \tag{15.54}$$

where g_C and g_F are geometric interpolation factors related to the position of the element face f with respect to the nodes C and F, as explained in previous chapters.

Using Eq. (15.50), $\overline{H_f}$ is rewritten as

$$\begin{aligned} \overline{H_f}[u] &= \frac{1}{2}\left(-u_C + B_C^u - D_C^u\left(\frac{\partial p}{\partial x}\right)_C - u_F + B_F^u - D_F^u\left(\frac{\partial p}{\partial x}\right)_F\right) \\ &= -\overline{u_f} - \overline{D_f^u}\overline{\left(\frac{\partial p}{\partial x}\right)_f} + \overline{B_f^u} \end{aligned} \tag{15.55}$$

where the coefficient approximation can be shown to be second order accurate, i.e.,

$$\begin{aligned} \overline{D_f^u\left(\frac{\partial p}{\partial x}\right)_f} - \overline{D_f^u}\overline{\left(\frac{\partial p}{\partial x}\right)_f} &= \frac{1}{2}\left(D_C^u\left(\frac{\partial p}{\partial x}\right)_C + D_F^u\left(\frac{\partial p}{\partial x}\right)_F\right) \\ &\quad - \frac{1}{2}\left(D_C^u + D_F^u\right) \times \frac{1}{2}\left(\left(\frac{\partial p}{\partial x}\right)_C + \left(\frac{\partial p}{\partial x}\right)_F\right) \\ &= \frac{1}{4}D_C^u\left(\left(\frac{\partial p}{\partial x}\right)_C - \left(\frac{\partial p}{\partial x}\right)_F\right) + \frac{1}{4}D_F^u\left(\left(\frac{\partial p}{\partial x}\right)_F - \left(\frac{\partial p}{\partial x}\right)_C\right) \\ &\approx O(\Delta x^2) \end{aligned} \tag{15.56}$$

Substituting Eq. (15.55) into Eq. (15.53), the velocity at the element face using the Rhie-Chow interpolation method is obtained as

$$u_f = -\overline{H_f}[u] + \overline{B_f^u} - \overline{D_f^u}\left(\frac{\partial p}{\partial x}\right)_f = \underbrace{\overline{u_f}}_{\substack{\text{average}\\ \text{velocity}}} \underbrace{- \overline{D_f^u}\left(\left(\frac{\partial p}{\partial x}\right)_f - \overline{\left(\frac{\partial p}{\partial x}\right)_f}\right)}_{\text{correction term}} \tag{15.57}$$

For a multi dimensional situation, similar interpolation formulae can be derived for the y and z velocity components and are given by

$$v_f = \overline{v_f} - \overline{D_f^v}\left(\left(\frac{\partial p}{\partial y}\right)_f - \overline{\left(\frac{\partial p}{\partial y}\right)_f}\right) \tag{15.58}$$

$$w_f = \overline{w_f} - \overline{D_f^w}\left(\left(\frac{\partial p}{\partial z}\right)_f - \overline{\left(\frac{\partial p}{\partial z}\right)_f}\right) \tag{15.59}$$

Equations (15.57)–(15.59) can be written in a vector form, more suitable for deriving the multi-dimensional pressure correction equation, as

$$\mathbf{v}_f = \overline{\mathbf{v}_f} - \overline{\mathbf{D}_f^{\mathbf{v}}}\left(\nabla p_f - \overline{\nabla p_f}\right) \tag{15.60}$$

where

$$\overline{\mathbf{D}_f^{\mathbf{v}}} = \begin{bmatrix} \overline{D_f^u} & 0 & 0 \\ 0 & \overline{D_f^v} & 0 \\ 0 & 0 & \overline{D_f^w} \end{bmatrix} \tag{15.61}$$

and where ∇p_f is computed as per Sect. 9.4 using

$$\nabla p_f = \overline{\nabla p_f} + \underbrace{\left[\frac{p_F - p_C}{d_{CF}} - \left(\overline{\nabla p_f} \cdot \mathbf{e}_{CF}\right)\right]\mathbf{e}_{CF}}_{\text{Correction to interpolated face gradient}} \tag{15.62}$$

and yielding a stencil in the **CF** direction formed only from the adjacent cell values p_F and p_C as

$$\begin{aligned} \nabla p_f \cdot \mathbf{e}_{CF} &= \overline{\nabla p_f} \cdot \mathbf{e}_{CF} + \left[\frac{p_F - p_C}{d_{CF}} - \left(\overline{\nabla p_f} \cdot \mathbf{e}_{CF}\right)\right]\mathbf{e}_{CF} \cdot \mathbf{e}_{CF} \\ &= \frac{p_F - p_C}{d_{CF}} \end{aligned} \tag{15.63}$$

With the face velocities closely linked to the pressure of adjacent cells, checkerboard fields are inadmissible rendering solutions on collocated grids viable.

15.5 General Derivation

Before proceeding with the development of the multidimensional collocated pressure correction equation, the discretized multidimensional momentum equation is first presented.

15.5.1 The Discretized Momentum Equation

The momentum equation given by Eq. (15.2) is slightly modified and written as

$$\underline{\frac{\partial}{\partial t}[\rho \mathbf{v}]} + \underline{\nabla \cdot \{\rho \mathbf{v}\mathbf{v}\}} = -\nabla p + \underline{\nabla \cdot \{\mu \nabla \mathbf{v}\}} + \nabla \cdot \{\mu (\nabla \mathbf{v})^{\mathrm{T}}\} + \mathbf{f}_b \qquad (15.64)$$

The discretized form of Eq. (15.64) in the time interval $[t - \Delta t/2, t + \Delta t/2]$ is sought over element C shown in Fig. 15.15.

In Eq. (15.64), the three underlined expressions represent, from left to right, the unsteady, convection, and diffusion term, respectively. The discretization of these terms proceeds as presented in previous chapters. The remaining terms are evaluated explicitly and treated as sources. The volume integral of the second part of the shear stress term is transformed into a surface integral using the divergence theorem and then into a summation of surface fluxes as

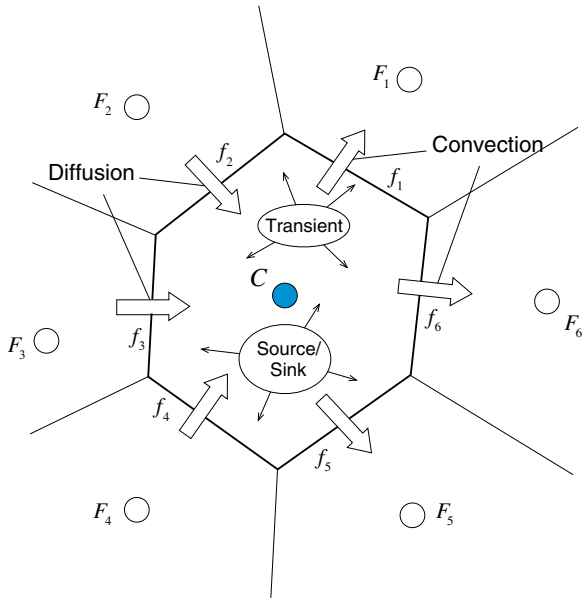

Fig. 15.15 Element C in a general unstructured grid system

$$\int_{V_C} \nabla \cdot \left\{ \mu (\nabla \mathbf{v})^{\mathrm{T}} \right\} dV = \int_{\partial V_C} \left\{ \mu (\nabla \mathbf{v})^{\mathrm{T}} \right\} \cdot d\mathbf{S} = \sum_{f \sim nb(C)} \mu (\nabla \mathbf{v})_f^{\mathrm{T}} \cdot \mathbf{S}_f \tag{15.65}$$

where the expanded form of $(\nabla \mathbf{v})_f^{\mathrm{T}} \cdot \mathbf{S}_f$ in a three dimensional coordinate system is given by

$$(\nabla \mathbf{v})_f^{\mathrm{T}} \cdot \mathbf{S}_f = \begin{bmatrix} \dfrac{\partial u}{\partial x} S_f^x + \dfrac{\partial u}{\partial y} S_f^y + \dfrac{\partial u}{\partial z} S_f^z \\ \dfrac{\partial v}{\partial x} S_f^x + \dfrac{\partial v}{\partial y} S_f^y + \dfrac{\partial v}{\partial z} S_f^z \\ \dfrac{\partial w}{\partial x} S_f^x + \dfrac{\partial w}{\partial y} S_f^y + \dfrac{\partial w}{\partial z} S_f^z \end{bmatrix} \tag{15.66}$$

The volume integral of the pressure gradient is also treated as a source term and evaluated explicitly as

$$\int_{V_C} \nabla p \, dV = (\nabla p)_C V_C \tag{15.67}$$

or transformed into a surface integral according to Eq. (2.85) and computed as

$$\int_{V_C} \nabla p \, dV = \int_{\partial V_C} p \, d\mathbf{S} = \sum_{f \sim nb(C)} p_f \mathbf{S}_f \tag{15.68}$$

The body force term is integrated directly over the control volume to yield

$$\int_{V_C} \mathbf{f}_b dV = (\mathbf{f}_b)_C V_C \tag{15.69}$$

Using a first order Euler scheme for the discretization of the unsteady term, a HR scheme for the convection term implemented via the deferred correction approach, and decomposing the diffusion flux into an implicit part aligned with the grid and an explicit cross diffusion part, the discretized momentum equation is written in vector form as

$$a_C^{\mathbf{v}} \mathbf{v}_C + \sum_{F \sim NB(C)} a_F^{\mathbf{v}} \mathbf{v}_F = \mathbf{b}_C^{\mathbf{v}} \tag{15.70}$$

where the coefficients are given by

$$\begin{aligned} a_C^{\mathbf{v}} &= FluxC_C + \sum_{f\sim nb(C)} (FluxC_f) \\ a_F^{\mathbf{v}} &= FluxF_f \\ \mathbf{b}_C^{\mathbf{v}} &= -FluxV_C - \sum_{f\sim nb(C)} FluxV_f + \sum_{f\sim nb(C)} \mu_f (\nabla \mathbf{v})_f^{\mathrm{T}} \cdot \mathbf{S}_f - (\nabla p)_C V_C \end{aligned} \tag{15.71}$$

with the face fluxes calculated using

$$\begin{aligned} FluxC_f &= \underbrace{\left\| \dot{m}_f, 0 \right\|}_{\substack{convection \\ contribution}} + \underbrace{\mu_f \frac{E_f}{d_{CF}}}_{\substack{diffusion \\ contribution}} \\ FluxF_f &= \underbrace{-\left\| -\dot{m}_f, 0 \right\|}_{\substack{convection \\ contribution}} - \underbrace{\mu_f \frac{E_f}{d_{CF}}}_{\substack{diffusion \\ contribution}} \\ FluxV_f &= -\mu_f (\nabla \mathbf{v})_f \cdot \mathbf{T}_f + \dot{m}_f (\mathbf{v}_f^{HR} - \mathbf{v}_f^U) \end{aligned} \tag{15.72}$$

and the element fluxes computed from

$$\begin{aligned} FluxC_C &= \frac{\rho_C V_C}{\Delta t} \\ FluxV_C &= \underbrace{-\frac{\rho_C^\circ V_C}{\Delta t} \mathbf{v}_C^\circ}_{\substack{transient \\ contribution}} - \underbrace{(\mathbf{f}_b)_C V_C}_{\substack{source\ term \\ contribution}} \end{aligned} \tag{15.73}$$

Even though the algebraic form of the momentum equation, Eq. (15.70), is linear, its coefficients depend on the velocity and pressure fields. This nonlinearity is handled by an iterative process during which the coefficients are calculated at the start of every iteration based on values of the dependent variables obtained in the previous iteration. This change in the values of the coefficients results in large changes in $\mathbf{v}$ and affects the rate of convergence to the degree of even causing divergence. To slow down the changes, under-relaxation can be applied when the transient time steps used are large. Denoting the under relaxation factor by $\lambda^{\mathbf{v}}$ and adopting Patankar's implicit relaxation approach, the under relaxed momentum equation can be written as

$$\frac{a_C^{\mathbf{v}}}{\lambda^{\mathbf{v}}} \mathbf{v}_C + \sum_{F\sim NB(C)} a_F^{\mathbf{v}} \mathbf{v}_F = \mathbf{b}_C^{\mathbf{v}} + \frac{1-\lambda^{\mathbf{v}}}{\lambda^{\mathbf{v}}} a_C^{\mathbf{v}} \mathbf{v}_C^{(n)} \tag{15.74}$$

By redefining $a_C^{\mathbf{v}}$ and $\mathbf{b}_C^{\mathbf{v}}$ such that

$$\begin{aligned} a_C^{\mathbf{v}} &\leftarrow \frac{a_C^{\mathbf{v}}}{\lambda^{\mathbf{v}}} \\ \mathbf{b}_C^{\mathbf{v}} &\leftarrow \mathbf{b}_C^{\mathbf{v}} + \frac{1-\lambda^{\mathbf{v}}}{\lambda^{\mathbf{v}}} a_C^{\mathbf{v}} \mathbf{v}_C^{(n)} \end{aligned} \tag{15.75}$$

the under-relaxed momentum equation can be rewritten as

$$a_C^{\mathbf{v}} \mathbf{v}_C + \sum_{F \sim NB(C)} a_F^{\mathbf{v}} \mathbf{v}_F = \mathbf{b}_C^{\mathbf{v}} \tag{15.76}$$

For the derivation of the collocated pressure correction equation, the pressure gradient is taken out of the $\mathbf{b}_C^{\mathbf{v}}$ source term and displayed explicitly to yield

$$\mathbf{b}_C^{\mathbf{v}} = -V_C (\nabla p)_C + \hat{\mathbf{b}}_C^{\mathbf{v}} \tag{15.77}$$

Substituting back in Eq. (15.76), the momentum equation becomes

$$\mathbf{v}_C + \sum_{F \sim NB(C)} \frac{a_F^{\mathbf{v}}}{a_C^{\mathbf{v}}} \mathbf{v}_F = -\frac{V_C}{a_C^{\mathbf{v}}} (\nabla p)_C + \frac{\hat{\mathbf{b}}_C^{\mathbf{v}}}{a_C^{\mathbf{v}}}. \tag{15.78}$$

Defining the following vector operators:

$$\begin{aligned} \mathbf{H}_C[\mathbf{v}] &= \sum_{f \sim NB(C)} \frac{a_F^{\mathbf{v}}}{a_C^{\mathbf{v}}} \mathbf{v}_F \\ \mathbf{B}_C^{\mathbf{v}} &= \frac{\hat{\mathbf{b}}_C^{\mathbf{v}}}{a_C^{\mathbf{v}}} \\ \mathbf{D}_C^{\mathbf{v}} &= \frac{V_C}{a_C^{\mathbf{v}}} \end{aligned} \tag{15.79}$$

Equation (15.78) is reformulated as

$$\mathbf{v}_C + \mathbf{H}_C[\mathbf{v}] = -\mathbf{D}_C^{\mathbf{v}} (\nabla p)_C + \mathbf{B}_C^{\mathbf{v}}, \tag{15.80}$$

a form that will be useful in later derivations.

15.5.2 The Collocated Pressure Correction Equation

As in the case of a staggered grid, starting with guessed values or values obtained from the previous iteration $\left(\mathbf{v}^{(n)}, \dot{m}^{(n)}, p^{(n)}\right)$, the momentum equation, Eq. (15.80), is first solved to obtain a momentum conserving velocity field $\mathbf{v}^*$. Thus the obtained solution satisfies

$$\mathbf{v}_C^* + \mathbf{H}_C[\mathbf{v}^*] = -\mathbf{D}_C^{\mathbf{v}}\left(\nabla p^{(n)}\right)_C + \mathbf{B}_C^{\mathbf{v}} \tag{15.81}$$

while the final solution should satisfy Eq. (15.80). The difference between these two equations is that the velocity field in Eq. (15.80) satisfies both the momentum and continuity equations while the one in Eq. (15.79) does not necessarily satisfy the continuity equation because of the linearization in which pressure and velocity are based on the previous iteration values. Therefore corrections to the velocity, mass flow rate, and pressure fields are needed to enforce mass conservation. Denoting these corrections by $(\mathbf{v}', p', \dot{m}')$ the relations between the exact and computed fields can be written as

$$\begin{aligned} \mathbf{v} &= \mathbf{v}^* + \mathbf{v}' \\ p &= p^{(n)} + p' \\ m &= m^* + m' \end{aligned} \tag{15.82}$$

Substituting the mass flow rate given by Eq. (15.82) into Eq. (15.25), the continuity equation becomes

$$\sum_{f\sim nb(C)} \dot{m}_f' = -\sum_{f\sim nb(C)} \dot{m}_f^* \quad \text{where } \dot{m}_f^* = \rho_f \mathbf{v}_f^* \cdot \mathbf{S}_f \tag{15.83}$$

with the face velocity computed using the Rhie-Chow interpolation as

$$\mathbf{v}_f^* = \overline{\mathbf{v}_f^*} - \overline{\mathbf{D}_f^{\mathbf{v}}}\left(\nabla p_f^{(n)} - \overline{\nabla p_f^{(n)}}\right). \tag{15.84}$$

When the computed mass flow rate field is conservative, the RHS of Eq. (15.83) is zero yielding a zero correction field. On the other hand, an incorrect velocity field leads to an imbalance in mass and a nonzero value of the RHS of Eq. (15.83) implying the need for a correction field for conservation to be enforced.

Mass flow rate corrections can be written in terms of velocity corrections, which can be derived by subtracting Eq. (15.81) from Eq. (15.80) to yield

$$\mathbf{v}'_C + \mathbf{H}_C[\mathbf{v}'] = -\mathbf{D}_C^{\mathbf{v}}(\nabla p')_C \tag{15.85}$$

A similar equation holds for element F and is given by

$$\mathbf{v}'_F + \mathbf{H}_F[\mathbf{v}'] = -\mathbf{D}_F^{\mathbf{v}}(\nabla p')_F \tag{15.86}$$

The mass flow rate correction at a cell face can be expressed as

$$\dot{m}'_f = \rho_f \mathbf{v}'_f \cdot \mathbf{S}_f \tag{15.87}$$

where the face velocity correction is obtained by subtracting Eq. (15.84) from Eq. (15.60) to give

$$\mathbf{v}'_f = \overline{\mathbf{v}'_f} - \overline{\mathbf{D}_f^{\mathbf{v}}}(\nabla p'_f - \overline{\nabla p'_f}) \tag{15.88}$$

Substitution of Eqs. (15.87) and (15.88) in Eq. (15.83), leads to the following form of the pressure correction equation:

$$\underline{\sum_{f\sim nb(C)} (\rho_f \overline{\mathbf{v}'_f} \cdot \mathbf{S}_f) + \sum_{f\sim nb(C)} \left(\rho_f \overline{\mathbf{D}_f^{\mathbf{v}} \nabla p'_f} \cdot \mathbf{S}_f\right)} - \sum_{f\sim nb(C)} \left(\rho_f \overline{\mathbf{D}_f^{\mathbf{v}}} (\nabla p')_f \cdot \mathbf{S}_f\right) = -\sum_{f\sim nb(C)} \dot{m}_f^* \tag{15.89}$$

In this equation the underlined part represents the effects of the neighboring velocity corrections on the velocity correction of the element under consideration. This influence becomes clearer by interpolating Eqs. (15.85) and (15.86) to the face yielding the following equivalent expression for the underline terms:

$$\overline{\mathbf{v}'_f} + \overline{\mathbf{H}_f}[\mathbf{v}'] = -\overline{\mathbf{D}_f^{\mathbf{v}}}(\overline{\nabla p'})_f \Rightarrow \overline{\mathbf{v}'_f} + \overline{\mathbf{D}_f^{\mathbf{v}}}(\overline{\nabla p'})_f = -\overline{\mathbf{H}_f}[\mathbf{v}']. \tag{15.90}$$

Substituting Eq. (15.90) in Eq. (15.89), the pressure correction equation is rewritten as

$$\sum_{f\sim nb(C)} \left(-\rho_f \overline{\mathbf{D}_f^{\mathbf{v}}} (\nabla p')_f \cdot \mathbf{S}_f\right) = -\sum_{f\sim nb(C)} \dot{m}_f^* + \underline{\sum_{f\sim nb(C)} (\rho_f \overline{\mathbf{H}_f}[\mathbf{v}'] \cdot \mathbf{S}_f)} \tag{15.91}$$

or more explicitly in the form

$$\sum_{f\sim nb(C)} \left(-\rho_f \overline{\mathbf{D}_f^{\mathbf{v}}} (\nabla p')_f \cdot \mathbf{S}_f\right) = -\sum_{f\sim nb(C)} \dot{m}_f^* + \underline{\sum_{f\sim nb(C)} \left(\rho_f \left(\overline{\sum_{F\sim NB(C)} \frac{a_F^{\mathbf{v}}}{a_C^{\mathbf{v}}} \mathbf{v}'_F}\right) \cdot \mathbf{S}_f\right)}. \tag{15.92}$$

In Eq. (15.91) or (15.92) the treatment of the underlined term is critical to rendering the equation solvable. In the original SIMPLE algorithm it is neglected,

thus linking the velocity correction at a point directly to pressure corrections. Because this is a correction equation the modification or dropping of the term will not affect the final solution, since at convergence the corrections become zero. However it will affect the convergence rate in that the larger is the neglected term the higher will be the error present in the approximation at each iteration.

The remaining terms in Eq. (15.91) or (15.92) can be easily treated. The coefficients of the pressure correction equation are obtained as per the discretization of the diffusion term in Chap. 8, specifically the treatment of anisotropic diffusion.

Thus the term on the LHS is modified into a gradient dot product of the form

$$\begin{aligned}\left(\overline{\mathbf{D}_f^{\mathbf{v}}}(\nabla p')_f\right)\cdot\mathbf{S}_f &= \left((\nabla p')_f\overline{\mathbf{D}_f^{\mathbf{v}}}^T\right)\cdot\mathbf{S}_f \\ &= (\nabla p')_f\cdot\left(\overline{\mathbf{D}_f^{\mathbf{v}}}^T\cdot\mathbf{S}_f\right) \\ &= (\nabla p')_f\cdot\mathbf{S}'_f\end{aligned} \tag{15.93}$$

The expanded expression of $\mathbf{S}'_f$ is given by

$$\mathbf{S}'_f = \overline{\mathbf{D}_f^{\mathbf{v}}}^T\cdot\mathbf{S}_f = \begin{bmatrix}\overline{D_f^u} & 0 & 0\\ 0 & \overline{D_f^v} & 0\\ 0 & 0 & \overline{D_f^w}\end{bmatrix}\begin{bmatrix}S_f^x\\ S_f^y\\ S_f^z\end{bmatrix} = \begin{bmatrix}\overline{D_f^u}S_f^x\\ \overline{D_f^v}S_f^y\\ \overline{D_f^w}S_f^z\end{bmatrix} \tag{15.94}$$

Working with $\mathbf{S}'_f$, the discretization of the pressure correction gradient term proceeds as usual resulting in

$$\begin{aligned}(\nabla p')_f\cdot\mathbf{S}'_f &= (\nabla p')_f\cdot\mathbf{E}_f + (\nabla p')_f\cdot\mathbf{T}_f \\ &= \frac{E_f}{d_{CF}}\left(p'_F - p'_C\right) + \underline{(\nabla p')_f\cdot\mathbf{T}_f}\end{aligned} \tag{15.95}$$

where the following decomposition of $\mathbf{S}'_f$ was used:

$$\mathbf{S}'_f = \mathbf{E}_f + \mathbf{T}_f. \tag{15.96}$$

The type of decomposition could be any of those reviewed in Chap. 8, as will be detailed later. The underlined term, arising due to grid non-orthogonality, can either be neglected or retained. If neglected, it will not affect the final solution as it is a correction term. If retained, then it will be treated explicitly with an internal loop (non-orthogonal loop in OpenFOAM®). As the solution starts with a zero pressure correction field at every iteration, the term has to be updated iteratively while solving the equation.

Dropping the non-orthogonal contribution, the linearized term of the pressure correction equation becomes

$$
\begin{aligned}
(\nabla p')_f \cdot \mathbf{S}'_f &= \frac{E_f}{d_{CF}}\left(p'_F - p'_C\right) \\
&= \mathcal{D}_f\left(p'_F - p'_C\right)
\end{aligned}
\tag{15.97}
$$

Substituting back in Eq. (15.91) the algebraic form of the pressure correction equation is obtained as

$$
a_C^{p'} p'_C + \sum_{F \sim NB(C)} a_F^{p'} p'_F = b_C^{p'} \tag{15.98}
$$

with the coefficients given by

$$
\begin{aligned}
a_F^{p'} &= FluxF_f = -\rho_f \mathcal{D}_f \\
a_C^{p'} &= -\sum_{f \sim nb(C)} FluxF_f = -\sum_{F \sim NB(C)} a_F^{p'} \\
b_C^{p'} &= -\sum_{f \sim nb(C)} FluxV_f + \underline{\sum_{f \sim nb(C)} \left(\rho_f \overline{\mathbf{H}_f}[\mathbf{v}'] \cdot \mathbf{S}_f\right)} \\
&= -\sum_{f \sim nb(C)} \dot{m}_f^* + \underline{\sum_{f \sim nb(C)} \left(\rho_f \overline{\mathbf{H}_f}[\mathbf{v}'] \cdot \mathbf{S}_f\right)}
\end{aligned}
\tag{15.99}
$$

Note that different approximations to the underlined terms in Eq. (15.99) result in different algorithms. In the original SIMPLE algorithm these terms are simply neglected.

Finally the mass flow rate $\dot{m}_f^*$ in Eq. (15.99), is the one computed after solving the momentum equation using as usual the Rhie-Chow interpolation technique with the latest velocity field, i.e.,

$$
\dot{m}_f^* = \rho \mathbf{v}_f^* \cdot \mathbf{S}_f = \rho \overline{\mathbf{v}_f^*} \cdot \mathbf{S}_f - \overline{\mathbf{D}_f^{\mathbf{v}}}\left(\nabla p_f^{(n)} - \overline{\nabla p_f^{(n)}}\right) \cdot \mathbf{S}_f. \tag{15.100}
$$

Following the calculation of the pressure correction field, the pressure and velocity at the element centroids and the mass flow rate at the element faces are all corrected. As mentioned above, the underlined term in Eq. (15.99) is neglected in the SIMPLE algorithm resulting in large pressure correction values that may slow the rate of convergence or cause divergence. To increase robustness and improve the convergence behavior, pressure correction values obtained from Eq. (15.98) are explicitly under relaxed. No under relaxation is used when updating the velocity and mass flow rate fields since the pressure correction will ensure mass conservation for these fields. Denoting the under relaxation factor by λ^p, the following correction equations are used:

$$\begin{aligned}
\mathbf{v}_C^{**} &= \mathbf{v}_C^{*} + \mathbf{v}'_C \quad \mathbf{v}'_C = -\mathbf{D}_C^{\mathbf{v}}(\nabla p')_C \\
\dot{m}_f^{**} &= \dot{m}_f^{*} + \dot{m}'_f \quad \dot{m}'_f = -\rho_f \overline{\mathbf{D}_f^{\mathbf{v}}} \nabla p'_f \cdot \mathbf{S}_f \\
p_C^{*} &= p_C^{(n)} + \lambda^p p'_C
\end{aligned} \tag{15.101}$$

15.5.3 Calculation of the $\mathcal{D}_f$ Term

The type of decomposition suggested in Eq. (15.96) could be any of those reviewed in Chap. 8 with different approaches leading to different expressions for $\mathcal{D}_f$ as derived below.

15.5.3.1 Minimum Correction Approach

For this approach $\mathbf{E}_f$ is obtained by substituting $\mathbf{S}'_f$ for $\mathbf{S}_f$ in Eq. (8.68) leading to

$$\mathbf{E}_f = \left(\mathbf{e}_{CF} \cdot \mathbf{S}'_f\right)\mathbf{e}_{CF} \tag{15.102}$$

where $\mathbf{e}_{CF}$ is a unit vector in the CF direction. Combining Eqs. (15.94), (15.102), and (8.64), $\mathbf{E}_f$ becomes

$$\mathbf{E}_f = \frac{d_{CF}^x \overline{D_f^u} S_f^x + d_{CF}^y \overline{D_f^v} S_f^y + d_{CF}^z \overline{D_f^w} S_f^z}{d_{CF}^2} \mathbf{d}_{CF} \tag{15.103}$$

Using Eq. (15.103), the following expression for $\mathcal{D}_f$ is derived:

$$\mathcal{D}_f = \frac{E_f}{d_{CF}} = \frac{d_{CF}^x \overline{D_f^u} S_f^x + d_{CF}^y \overline{D_f^v} S_f^y + d_{CF}^z \overline{D_f^w} S_f^z}{\left(d_{CF}^x\right)^2 + \left(d_{CF}^y\right)^2 + \left(d_{CF}^z\right)^2} \tag{15.104}$$

15.5.3.2 Orthogonal Correction Approach

The definition of $\mathbf{E}_f$ in this case is obtained from Eq. (8.69) and written as

$$\mathbf{E}_f = S'_f \mathbf{e}_{CF} \tag{15.105}$$

Combining Eqs. (15.105), (15.94), and (8.64), $\mathcal{D}_f$ is found to be

$$\mathcal{D}_f = \frac{E_f}{d_{CF}} = \sqrt{\frac{\left(\overline{D_f^u} S_f^x\right)^2 + \left(\overline{D_f^v} S_f^y\right)^2 + \left(\overline{D_f^w} S_f^z\right)^2}{\left(d_{CF}^x\right)^2 + \left(d_{CF}^y\right)^2 + \left(d_{CF}^z\right)^2}} \tag{15.106}$$

15.5.3.3 Over-Relaxed Approach

In this method, $\mathbf{E}_f$ is computed from Eqs. (8.64) and (8.70) as

$$\mathbf{E}_f = \frac{\mathbf{S}'_f \cdot \mathbf{S}'_f}{\mathbf{d}_{CF} \cdot \mathbf{S}'_f} \mathbf{d}_{CF} \tag{15.107}$$

Combining Eq. (15.107) with Eq. (15.94), $\mathcal{D}_f$ is found to be

$$\mathcal{D}_f = \frac{\left(\overline{D_f^u} S_f^x\right)^2 + \left(\overline{D_f^v} S_f^y\right)^2 + \left(\overline{D_f^w} S_f^z\right)^2}{d_{CF}^x \overline{D_f^u} S_f^x + d_{CF}^y \overline{D_f^v} S_f^y + d_{CF}^z \overline{D_f^w} S_f^z} \tag{15.108}$$

Any of the above approaches can be adopted to calculate the value of $\mathcal{D}_f$.

15.5.4 The Collocated SIMPLE Algorithm

The sequence of events in the collocated SIMPLE algorithm is displayed in Fig. 15.16 and can be summarized as follows:

1. To compute the solution at iteration $n + 1$, start with the solution at time t for pressure, velocity, and mass flow rate fields, i.e., $p^{(n)}$, $u^{(n)}$, and $\dot{m}^{(n)}$, respectively, as the initial guess.
2. Solve the momentum equation given by Eq. (15.70) to obtain a new velocity field $\mathbf{v}^*$.
3. Update the mass flow rate at the element faces using the Rhie-Chow interpolation (Eq. 15.100) to compute a momentum satisfying mass flow rate field $\dot{m}^*$.
4. Using the new mass flow rates assemble the pressure correction equation (Eq. 15.98) and solve it to obtain a pressure correction field p'.
5. With the pressure correction field update the pressure and velocity fields at the element centroids and the mass flow rate at the element faces to obtain continuity-satisfying fields using Eq. (15.101). These resulting fields are denoted by u^{**}, $\dot{m}^{**}$, and p^*, respectively.
6. Treat the obtained solution as a new initial guess, return to step 2 and repeat until convergence.
7. Set the solution at time $n + 1$ (i.e., $t + \Delta t$) to be equal to the converged solution and set the current time $n + 1$ (i.e., $t + \Delta t$) to be n (i.e., t).
8. Advance to the next time step.
9. Return to step 1 and repeat until the last time step is reached.

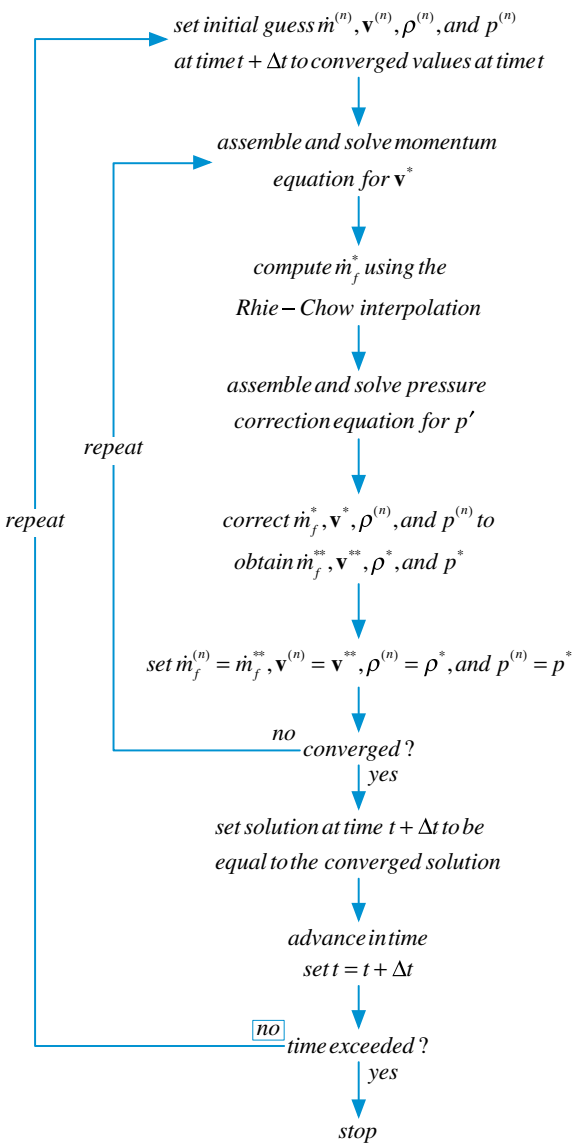

Fig. 15.16 A flow chart of the SIMPLE algorithm

Example 3

In the two dimensional problem shown below, the following quantities are given $p_W = 100$, $p_N = 20$ *and* $p_E = 50$ *and an inlet condition at face s with* $v_s = 20$ *and zero pressure gradient.*

The flow is steady state and the density is uniform of value 1. The momentum equations for u_e *and* v_n *are*

$$u_C = -d_x(p_e - p_w)$$

and

$$v_C = -d_y(p_n - p_s)$$

where the constants $d_x = 1$ *and* $d_y = 0.25$. *The element shown has* $\Delta x = \Delta y = 1$. *Use the collocated SIMPLE algorithm to derive the pressure correction equation and solve it to find the pressure for element C. Take* $p_C^{(n)} = 70$ *as an initial guess for pressure at C* (Fig. 15.17).

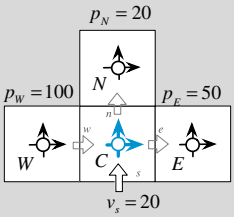

Fig. 15.17 The two dimensional domain used in Example 3

Solution

At the inlet the zero gradient condition can be used to compute

$$p_s = p_C = p_C^{(n)} = 70$$

now compute $u_C^{(n)}, v_C^{(n)}$ using

$$\begin{aligned} u_C^* &= -d_x\left(p_e^{(n)} - p_w^{(n)}\right) \\ &= -1(60 - 85) \\ &= 25 \end{aligned}$$

and

$$\begin{aligned} v_C^* &= -d_y\left(p_n^{(n)} - p_s^{(n)}\right) \\ &= -0.25(45 - 70) \\ &= 6.25 \end{aligned}$$

since $p_e^{(n)} = 0.5\left(p_E^{(n)} + p_C^{(n)}\right)$, $p_n^{(n)} = 0.5\left(p_E^{(n)} + p_N^{(n)}\right)$ and $p_w^{(n)} = 0.5\left(p_E^{(n)} + p_W^{(n)}\right)$

Now the pressure correction equation is constructed by substituting the mass flux and its correction into the mass conservation equation. Using the Rhie-Chow interpolation the face fluxes are computed as

$$\dot{m}_e^* = u_e^* \Delta y = \overline{u_e^*} - d_x \left[\underbrace{\left(p_E^{(n)} - p_C^{(n)}\right)}_{\substack{\text{pressure difference} \\ \text{across face}}} - \underbrace{0.5\left(\left(p_e^{(n)} - p_w^{(n)}\right) + \left(p_{ee}^{(n)} - p_e^{(n)}\right)\right)}_{\substack{\text{average pressure difference} \\ \text{across face}}} \right]$$

$$= \underbrace{0.5\left[d_x\left(p_e^{(n)} - p_w^{(n)}\right) + d_x\left(p_{ee}^{(n)} - p_{ww}^{(n)}\right)\right]}_{\overline{u_e^*}} - d_x \left[\underbrace{\left(p_E^{(n)} - p_C^{(n)}\right)}_{\substack{\text{pressure difference} \\ \text{across face}}} - \underbrace{0.5\left(\left(p_e^{(n)} - p_w^{(n)}\right) + \left(p_{ee}^{(n)} - p_e^{(n)}\right)\right)}_{\substack{\text{average pressure difference} \\ \text{across face}}} \right]$$

$$= -d_x\left(p_E^{(n)} - p_C^{(n)}\right)$$

$$= -1(50 - 70)$$

$$= 20$$

similarly for the n and w face

$$\dot{m}_n^* = v_n^* \Delta x = \overline{v_n^*} - d_y \left[\underbrace{\left(p_N^{(n)} - p_C^{(n)}\right)}_{\substack{\text{pressure difference} \\ \text{across face}}} - \underbrace{0.5\left(\left(p_n^{(n)} - p_s^{(n)}\right) + \left(p_{nn}^{(n)} - p_n^{(n)}\right)\right)}_{\substack{\text{average pressure difference} \\ \text{across face}}} \right]$$

$$= -d_y\left(p_N^{(n)} - p_C^{(n)}\right)$$

$$= -0.25(20 - 70)$$

$$= 12.5$$

$$\dot{m}_w^* = -u_w^* \Delta y = -\overline{u_w^*} + d_x \left[\underbrace{\left(p_C^{(n)} - p_W^{(n)}\right)}_{\substack{\text{pressure difference} \\ \text{across face}}} - \underbrace{0.5\left(\left(p_e^{(n)} - p_w^{(n)}\right) + \left(p_w^{(n)} - p_{ww}^{(n)}\right)\right)}_{\substack{\text{average pressure difference} \\ \text{across face}}} \right]$$

$$= d_x\left(p_C^{(n)} - p_W^{(n)}\right)$$

$$= 1(70 - 100)$$

$$= -30$$

with $\dot{m}_s = -20\Delta x = -20$ the pressure correction equation is constructed from

$$\left(\dot{m}_e^* + \dot{m}_e'\right) + \left(\dot{m}_n^* + \dot{m}_n'\right) + \left(\dot{m}_w^* + \dot{m}_w'\right) + \dot{m}_s = 0$$

and

$$\begin{aligned}\dot{m}'_e + \dot{m}'_n + \dot{m}'_w &= -\left(\dot{m}^*_e + \dot{m}^*_n + \dot{m}^*_w + \dot{m}_s\right)\\ &= -(20 + 12.5 - 30 - 20)\\ &= 17.5\end{aligned}$$

now

$$\begin{aligned}\dot{m}'_e &= -d_x\left(p'_E - p'_C\right)\\ \dot{m}'_n &= -d_y\left(p'_N - p'_C\right)\\ \dot{m}'_w &= d_x\left(p'_C - p'_W\right)\end{aligned}$$

thus we have

$$-\left(p'_E - p'_C\right) - 0.25\left(p'_N - p'_C\right) + \left(p'_C - p'_W\right) = 17.5$$

or

$$2.25p'_C - p'_E - 0.25p'_N - p'_W = 17.5$$

if the E, N and W values were exact then we would have

$$2.25p'_C = 17.5$$

or

$$p'_C = 7.78$$

then we would proceed with correcting the pressure and velocity components at C to yield

$$\begin{aligned}p^*_C = p^{(n)}_C + p'_C &= 77.78\\ u'_C &= 0 \Rightarrow u^{**}_C = 25\\ v'_C &= -d_y\left(p'_n - p'_s\right)\\ &= -0.25\left(0.5p'_C - p'_C\right)\\ &= -0.25(0.5 \times 7.78 - 7.78)\\ &= 0.9725\end{aligned}$$

so

$$\begin{aligned}v^{**}_C &= 6.25 + 0.9725\\ &= 7.2225\end{aligned}$$

15.6 Boundary Conditions

A **boundary element** has at least one face located at a boundary patch, which is denoted as a **boundary face** (Fig. 15.18). The treatment of boundary conditions at a boundary face is critical to the accuracy and robustness of any CFD code. Thus for any pressure-based code to succeed, it is imperative for the boundary conditions of both momentum and pressure-correction equations to be properly implemented. This section describes in detail the implementation of a variety of boundary conditions.

A first note of interest is the expression of the Rhie-Chow interpolation at a boundary face, which has to be modified since the averaging cannot be performed in a fashion similar to an interior face. The average at a boundary face is written in terms of the element value as

$$\overline{\square}_b = \square_C \tag{15.109}$$

where b refers to the boundary face centroid and C to the element centroid. Adopting this practice, the averages in the Rhie-Chow interpolation, the velocity, and the mass flow rate at a boundary face become

$$\begin{aligned} \overline{\mathbf{v}_b^*} &= \mathbf{v}_C^* \\ \overline{\nabla p_b^{(n)}} &= \nabla p_C^{(n)} \\ \overline{\mathbf{D}_b^{\mathbf{v}}} &= \mathbf{D}_C^{\mathbf{v}} \end{aligned} \tag{15.110}$$

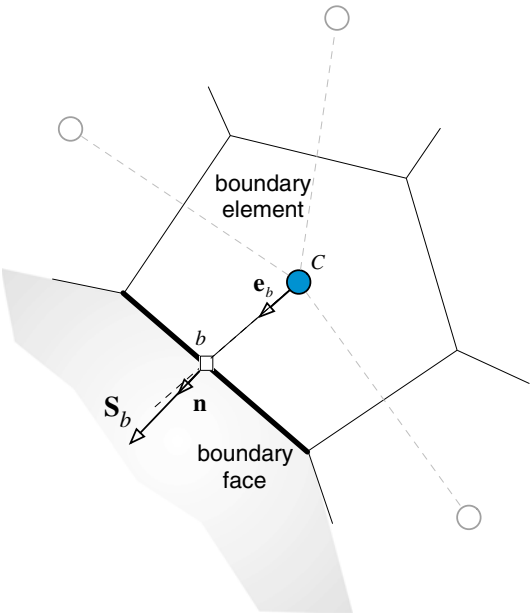

Fig. 15.18 An example of a boundary element

$$\begin{aligned}\underbrace{\mathbf{v}_b^*}_{\text{boundary face}} &= \underbrace{\overline{\mathbf{v}_b^*} - \mathbf{D}_C^{\mathbf{v}}\left(\nabla p_b^{(n)} - \overline{\nabla p_b^{(n)}}\right)}_{\text{standard Rhie-Chow}} \\ &= \underbrace{\mathbf{v}_C^* - \mathbf{D}_C^{\mathbf{v}}\left(\nabla p_b^{(n)} - \nabla p_C^{(n)}\right)}_{\text{boundary Rhie-Chow}}\end{aligned} \tag{15.111}$$

$$\begin{aligned}\dot{m}_b^* &= \rho_b \mathbf{v}_b^* \cdot \mathbf{S}_b \\ &= \rho_b \left[\mathbf{v}_C^* - \mathbf{D}_C^{\mathbf{v}}\left(\nabla p_b^{(n)} - \nabla p_C^{(n)}\right)\right] \cdot \mathbf{S}_b \\ &= \rho_b \mathbf{v}_C^* \cdot \mathbf{S}_b - \rho_b \mathcal{D}_C\left(p_b^{(n)} - p_C^{(n)}\right) - \rho_b\left(\mathbf{D}_C^{\mathbf{v}} \nabla p_b^{(n)} \cdot \mathbf{T}_b - \mathbf{D}_C^{\mathbf{v}} \nabla p_C^{(n)} \cdot \mathbf{S}_b\right)\end{aligned} \tag{15.112}$$

In what follows the implementation of boundary conditions is first presented for the momentum equation, followed by the implementation of the boundary conditions for the pressure (correction) equation. For the cases when the boundary conditions for the momentum and pressure correction equations are co-dependent, their full treatment is detailed in the pressure correction equation section.

15.6.1 Boundary Conditions for the Momentum Equation

The semi-discretized form of the momentum equation can be expressed as

$$\underbrace{\frac{(\rho\mathbf{v})_C - (\rho\mathbf{v})_C^{\circ}}{\Delta t} V_C}_{\text{element discretization}} + \underbrace{\sum_{f\sim nb(C)} \left(\dot{m}_f \mathbf{v}_f\right)}_{\text{face discretization}} = -\underbrace{\sum_{f\sim nb(C)} \left(p_f \mathbf{S}_f\right)}_{\text{face discretization}} + \underbrace{\sum_{f\sim nb(C)} \left(\tau_f \cdot \mathbf{S}_f\right)}_{\text{face discretization}} + \underbrace{\mathbf{B}}_{\substack{\text{element} \\ \text{discretization}}} \tag{15.113}$$

where the discretization type of the various terms is explicitly stated. Terms that are evaluated at the element faces should be modified along a boundary face in accordance with the type of boundary condition used. Therefore, for boundary elements, the terms evaluated at the element faces are written as

$$\underbrace{\sum_{f\sim nb(C)} \left(\dot{m}_f \mathbf{v}_f\right)}_{\text{face discretization}} = \sum_{f\sim \text{interior } nb(C)} \left(\dot{m}_f \mathbf{v}_f\right) + \underbrace{\dot{m}_b \mathbf{v}_b}_{\text{boundary face}} \tag{15.114}$$

$$\underbrace{\sum_{f\sim nb(C)}\left(\tau_f\cdot\mathbf{S}_f\right)}_{\text{face discretization}}=\sum_{f\sim \text{interior } nb(C)}\left(\tau_f\cdot\mathbf{S}_f\right)+\underbrace{\tau_b\cdot\mathbf{S}_b}_{\text{boundary face}}$$
$$=\sum_{f\sim \text{interior } nb(C)}\left(\tau_f\cdot\mathbf{S}_f\right)+\underbrace{\mathbf{F}_b}_{\substack{\text{boundary}\\ \text{face}}} \tag{15.115}$$

$$\underbrace{\sum_{f\sim nb(C)}\left(p_f\mathbf{S}_f\right)}_{\text{face discretization}}=\sum_{f\sim \text{interior } nb(C)}\left(p_f\mathbf{S}_f\right)+\underbrace{p_b\mathbf{S}_b}_{\text{boundary face}} \tag{15.116}$$

where subscript b refers to a value at the boundary. As previously stated, the pressure term may be discretized following either an element or a face discretization. In either case the expanded form is the same since $V_C(\nabla p)_C$ is calculated as $\sum_{f\sim nb(C)} p_f\mathbf{S}_f$ implying that the value at the boundary is always required. To show the way boundary pressure affects solution, the expanded form of the pressure gradient (i.e., face discretization) is adopted in the implementation of boundary conditions.

15.6.1.1 Wall Boundary Conditions

Generally a no-slip or a slip boundary condition may be applied to a moving or a stationary wall. The implementation involves computing and linearizing the shear stress at the wall. This is different from the Dirichlet condition, though for a cartesian grid the two conditions may be shown to be identical.

No-Slip Wall Boundary $(p_b = ?; \dot{m}_b = 0; \mathbf{v}_b = \mathbf{v}_{wall})$

A no slip boundary condition implies that the velocity of the fluid at the wall $\mathbf{v}_b$ is equal to the velocity of the wall $\mathbf{v}_{wall}$. For a stationary wall, the boundary velocity $\mathbf{v}_b$ is zero. The known velocity at the wall could be wrongly viewed as a Dirichlet boundary condition. However this is not really the case, since what is needed is to have a zero normal boundary flux while also accounting for the shear stress. This cannot be satisfied by simply setting $\mathbf{v}_b = \mathbf{v}_{wall}$. Figure 15.19 shows that this can be guaranteed by ensuring a shear stress that is tangential to the wall in addition to a boundary velocity equation that is equal to the wall velocity. The force $\mathbf{F}_b$ exerted by the wall on the fluid can be written as

$$\mathbf{F}_b = \mathbf{F}_\perp + \mathbf{F}_\parallel \tag{15.117}$$

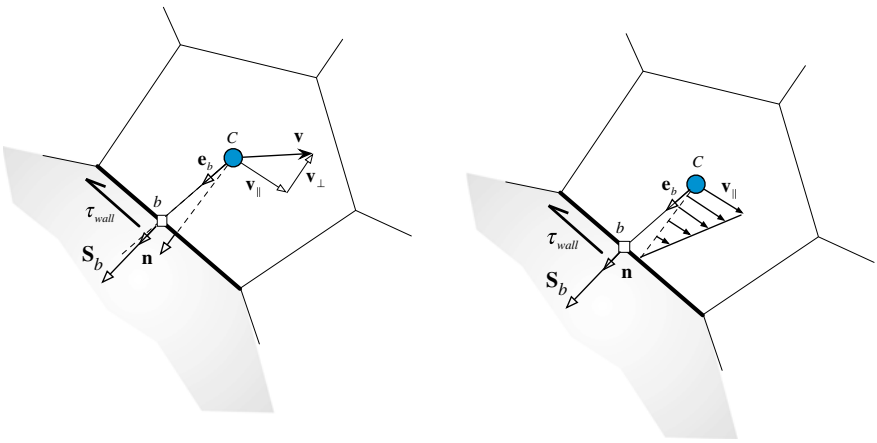

Fig. 15.19 A schematic of a no-slip wall boundary condition

where $\mathbf{F}_{\parallel}$ is in the tangential direction to the wall and $\mathbf{F}_{\perp}$ is in the normal direction, which is supposed to be zero. Thus

$$\mathbf{F}_b = \mathbf{F}_{\parallel} = \tau_{wall} S_b \tag{15.118}$$

where τ_{wall} is the shear stress exerted by the wall on the fluid given by

$$\tau_{wall} = -\mu \frac{\partial \mathbf{v}_{\parallel}}{\partial d_{\perp}}. \tag{15.119}$$

In Eq. (15.119) $\mathbf{v}_{\parallel}$ is the velocity vector in the direction parallel to the wall and $d_{\perp}$ is the normal distance from the centroid of the boundary element to the wall computed as

$$\begin{aligned} \mathbf{n} &= \frac{\mathbf{S}_b}{S_b} = n_x \mathbf{i} + n_y \mathbf{j} + n_z \mathbf{k} \\ d_{\perp} &= \mathbf{d}_{Cb} \cdot \mathbf{n} = \frac{\mathbf{d}_{Cb} \cdot \mathbf{S}_b}{S_b} \end{aligned} \tag{15.120}$$

and $\mathbf{n}$ the wall normal unit vector. The velocity vector $\mathbf{v}_{\parallel}$ can be written as the difference between $\mathbf{v}$ and its normal component as

$$\mathbf{v}_{\parallel} = \mathbf{v} - (\mathbf{v} \cdot \mathbf{n})\mathbf{n} = \left\{ \begin{array}{c} u - \left(u n_x + v n_y + w n_z\right) n_x \\ v - \left(u n_x + v n_y + w n_z\right) n_y \\ w - \left(u n_x + v n_y + w n_z\right) n_z \end{array} \right\} \tag{15.121}$$

From Eq. (15.119), the wall shear stress can be approximated using

$$\begin{aligned}\tau_{wall} &\approx -\mu_b \frac{(\mathbf{v}_C - \mathbf{v}_b)_{||}}{d_\perp} = -\mu_b \frac{(\mathbf{v}_C - \mathbf{v}_b) - [(\mathbf{v}_C - \mathbf{v}_b)\cdot\mathbf{n}]\mathbf{n}}{d_\perp} \\ &= -\frac{\mu_b}{d_\perp}\left\{\begin{array}{l}(u_C - u_b) - \big[(u_C - u_b)n_x + (v_C - v_b)n_y + (w_C - w_b)n_z\big]n_x \\ (v_C - v_b) - \big[(u_C - u_b)n_x + (v_C - v_b)n_y + (w_C - w_b)n_z\big]n_y \\ (w_C - w_b) - \big[(u_C - u_b)n_x + (v_C - v_b)n_y + (w_C - w_b)n_z\big]n_z\end{array}\right\}\end{aligned} \tag{15.122}$$

from which the boundary force for a laminar flow can be obtained as

$$\mathbf{F}_b = -\frac{\mu_b S_b}{d_\perp}\left\{\begin{array}{l}(u_C - u_b) - \big[(u_C - u_b)n_x + (v_C - v_b)n_y + (w_C - w_b)n_z\big]n_x \\ (v_C - v_b) - \big[(u_C - u_b)n_x + (v_C - v_b)n_y + (w_C - w_b)n_z\big]n_y \\ (w_C - w_b) - \big[(u_C - u_b)n_x + (v_C - v_b)n_y + (w_C - w_b)n_z\big]n_z\end{array}\right\} \tag{15.123}$$

Using Eq. (15.123) the coefficients of the boundary elements for the momentum equation in the x, y and z directions are modified as follows:

u-component equation

$$\begin{aligned} a_C^u &\leftarrow \underbrace{a_C^u}_{\text{interior faces contribution}} + \underbrace{\frac{\mu_b S_b}{d_\perp}(1 - n_x^2)}_{\text{boundary face contribution}} \\ 0 &\leftarrow a_{F=b}^u \\ b_C^u &\leftarrow \underbrace{b_C^u}_{\text{interior faces contribution}} + \underbrace{\frac{\mu_b S_b}{d_\perp}\big[u_b(1 - n_x^2) + (v_C - v_b)n_y n_x - (w_C - w_b)n_z n_x\big] - p_b S_b^x}_{\text{boundary face contribution}} \end{aligned} \tag{15.124}$$

v-component equation

$$\begin{aligned} a_C^v &\leftarrow \underbrace{a_C^v}_{\text{interior faces contribution}} + \underbrace{\frac{\mu_b S_b}{d_\perp}\left(1 - n_y^2\right)}_{\text{boundary face contribution}} \\ 0 &\leftarrow a_{F=b}^v \\ b_C^v &\leftarrow \underbrace{b_C^v}_{\text{interior faces contribution}} + \underbrace{\frac{\mu_b S_b}{d_\perp}\left[(u_C - u_b)n_x n_y + v_b\left(1 - n_y^2\right) + (w_C - w_b)n_z n_y\right] - p_b S_b^y}_{\text{boundary face contribution}} \end{aligned} \tag{15.125}$$

w-component equation

$$
\begin{aligned}
a_C^w &\leftarrow \underbrace{a_C^w}_{\text{interior faces contribution}} + \underbrace{\frac{\mu_b S_b}{d_\perp}\left(1-n_z^2\right)}_{\text{boundary face contribution}} \\
0 &\leftarrow a_{F=b}^w \\
b_C^w &\leftarrow \underbrace{b_C^w}_{\text{interior faces contribution}} + \underbrace{\frac{\mu_b S_b}{d_\perp}\left[(u_C-u_b)n_x n_z + (v_C-v_b)n_y n_z + w_b\left(1-n_z^2\right)\right] - p_b S_b^z}_{\text{boundary face contribution}}
\end{aligned}
\tag{15.126}
$$

The unknown boundary pressure p_b is extrapolated from the interior solution using either a truncated Taylor series expansion around point C such that pressure is found from

$$p_b = p_C + \nabla p_C^{(n)} \cdot \mathbf{d}_{Cb} \tag{15.127}$$

or the mass flow rate expressed via the Rhie-Chow interpolation as

$$\dot{m}_b^* = \rho_b \mathbf{v}_b^* \cdot \mathbf{S}_b - \rho_b \mathbf{D}_C^{\mathbf{v}} \left(\nabla p_b^{(n)} - \nabla p_C^{(n)}\right) \cdot \mathbf{S}_b \tag{15.128}$$

Since the mass flow rate and velocity at the wall boundary are zero, the above equation reduces to

$$0 = 0 - \rho_b \mathbf{D}_C^{\mathbf{v}} \left(\nabla p_b^{(n)} - \nabla p_C^{(n)}\right) \cdot \mathbf{S}_b \tag{15.129}$$

which can be modified into

$$
\begin{aligned}
\mathbf{D}_C^{\mathbf{v}} \nabla p_b^{(n)} \cdot \mathbf{S}_b &= \nabla p_b^{(n)} \cdot \mathbf{S}'_b \\
&= \nabla p_C^{(n)} \cdot \mathbf{S}'_b
\end{aligned}
\tag{15.130}
$$

Expressing $\mathbf{S}'_b$ as the sum of the two vector $\mathbf{E}_b$ and $\mathbf{T}_b$, Eq. (15.130) becomes

$$\nabla p_b^{(n)} \cdot (\mathbf{E}_b + \mathbf{T}_b) = \nabla p_C^{(n)} \cdot \mathbf{S}'_b \tag{15.131}$$

Since $\mathbf{E}_b$ is in the direction of $\mathbf{Cb}$, the above equation can be modified to

$$\mathcal{D}_C (p_b - p_C) = \left(\nabla p_C^{(n)} \cdot \mathbf{S}'_b - \nabla p_b^{(n)} \cdot \mathbf{T}_b\right) \tag{15.132}$$

from which the boundary pressure is obtained as

$$p_b = p_C + \frac{\left(\nabla p_C^{(n)} \cdot \mathbf{S}'_b - \nabla p_b^{(n)} \cdot \mathbf{T}_b\right)}{\mathcal{D}_C} \tag{15.133}$$

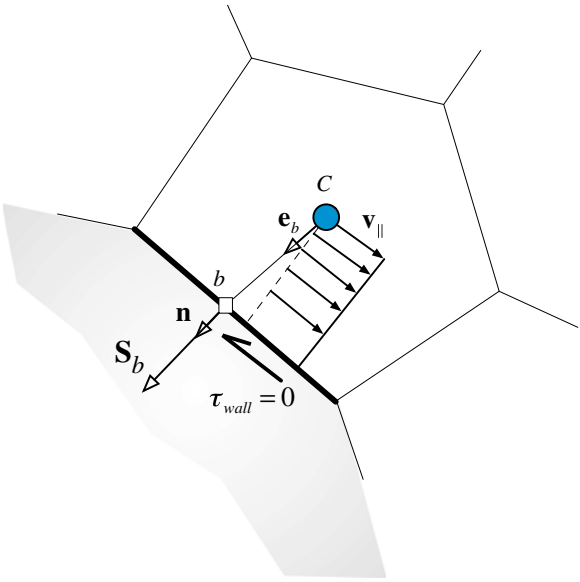

Fig. 15.20 A schematic of a slip wall boundary condition

Slip Wall Boundary $(p_b = ?; \dot{m}_b = 0; \mathbf{F}_b = 0)$

For this boundary condition, the wall shear stress is zero (Fig. 15.20). Therefore the boundary force is zero. The boundary pressure is computed as for the no-slip wall boundary case using Eq. (15.127) or Eq. (15.133). The coefficients of the momentum equation (in vector form) are modified as follows:

$$\begin{aligned} a_C^{\mathbf{v}} &\leftarrow \underbrace{a_C^{\mathbf{v}}}_{\text{interior faces contribution}} \\ 0 &\leftarrow a_{F=b}^{\mathbf{v}} \\ \mathbf{b}_C^{\mathbf{v}} &\leftarrow \underbrace{\mathbf{b}_C^{\mathbf{v}}}_{\text{interior faces contribution}} - \underbrace{p_b \mathbf{S}_b}_{\text{boundary face contribution}} \end{aligned} \tag{15.134}$$

15.6.1.2 Inlet Boundary Conditions

Three types of inlet boundary conditions are considered. (i) specified velocity; (ii) specified static pressure and velocity direction; and (iii) specified total pressure and velocity direction. All treatments of the pressure boundary conditions will be detailed in the *pressure correction boundary conditions* section.

Specified Velocity ($p_b = ?; \dot{m}_b$ *specified*; $\mathbf{v}_b$ *specified*)

For a specified velocity boundary condition at inlet (Fig. 15.21) the convection ($\dot{m}_b \mathbf{v}_b$) and diffusion ($\mathbf{F}_b = \tau_b \cdot \mathbf{S}_b$) terms at the boundary face are calculated using the known values of velocity $\mathbf{v}_b$ and mass flow rate $\dot{m}_b$. The pressure at the boundary is extrapolated from the boundary element centroid as

$$p_b = p_C + \nabla p_C^{(n)} \cdot \mathbf{d}_{Cb} \tag{15.135}$$

The terms involving the boundary velocity are treated explicitly by adding them to the source term and setting the coefficient $a^{\mathbf{v}}_{F=b}$ to zero while adding its value to the $a^{\mathbf{v}}_C$ coefficient.

The coefficients of the boundary element are modified according to

$$\begin{aligned} a^{\mathbf{v}}_C &\leftarrow a^{\mathbf{v}}_C \\ b^{\mathbf{v}}_C &\leftarrow b^{\mathbf{v}}_C - a^{\mathbf{v}}_{F=b}\mathbf{v}_b \\ 0 &\leftarrow a^{\mathbf{v}}_{F=b} \end{aligned} \tag{15.136}$$

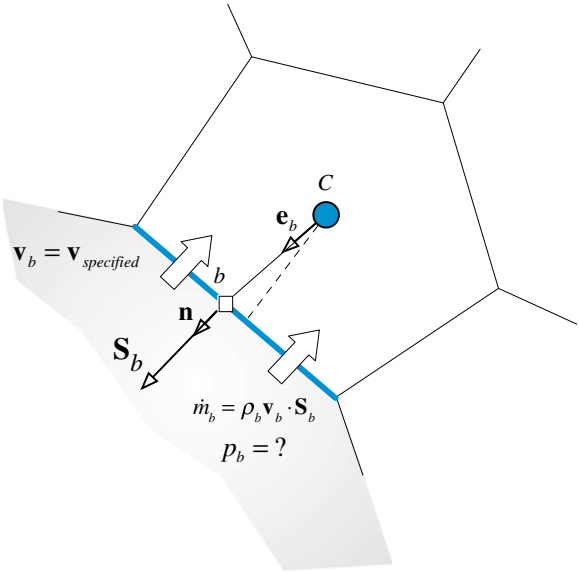

Fig. 15.21 A schematic of specified velocity boundary condition at inlet

Specified Pressure and Velocity Direction $(p_b = p_{specified};\ \dot{m}_b?;\ \mathbf{e_v}\ specified;\ \mathbf{v}_b?)$

In the case of a specified static pressure at inlet (Fig. 15.22), p_b is known. The velocity being unknown, has to be computed from the pressure gradient at the boundary. To this end, a velocity direction should be specified as part of the boundary condition.

As the boundary pressure p_b is known, its value is directly used in calculating the pressure gradient in the boundary element without any special treatment. Therefore p_b is used in calculating ∇p_C.

The mass flow rate at the boundary is computed from the continuity equation (see next section). Then, for a specified velocity direction given by the unit vector $\mathbf{e_v}$, the velocity for a specified pressure boundary condition at inlet is obtained as

$$\dot{m}_b^{**} = \rho_b \mathbf{v}_b^{**} \cdot \mathbf{S}_b = \rho_b \left\|\mathbf{v}_b^{**}\right\| \mathbf{e_v} \cdot \mathbf{S}_b \Rightarrow \left\|\mathbf{v}_b^{**}\right\| = \frac{\dot{m}_b^{**}}{\rho_b (\mathbf{e_v} \cdot \mathbf{S}_b)} \Rightarrow \mathbf{v}_b^{**} = \left\|\mathbf{v}_b^{**}\right\| \mathbf{e_v} \tag{15.137}$$

The velocity at the boundary is computed at every iteration and the problem is solved as in the case of a specified velocity with the coefficients in the momentum equation modified according to Eq. (15.136).

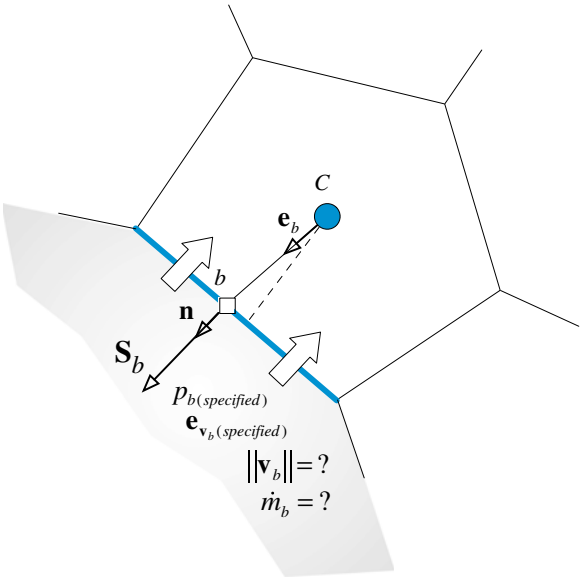

Fig. 15.22 A schematic of specified pressure and velocity direction boundary condition at inlet

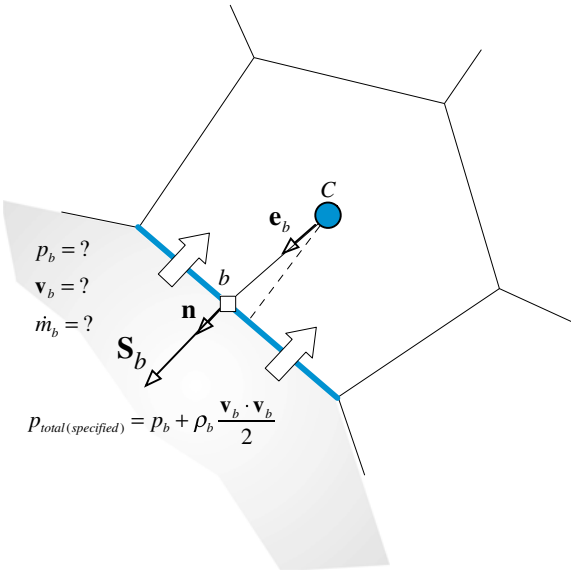

Fig. 15.23 A schematic of specified total pressure and velocity direction boundary condition at inlet

Specified Total Pressure and Velocity Direction $(p_{o,b} = p_{o,specified}; \dot{m}_b?; \mathbf{e}_\mathbf{v}\ specified; \mathbf{v}_b?)$

In the case of a specified total pressure at inlet (Fig. 15.23) the velocity direction should also be specified. However, the magnitude of the velocity and the pressure at the boundary are unknown though related using the total pressure definition given by

$$p_0 = \underbrace{p}_{\text{static pressure}} + \underbrace{\frac{1}{2}\rho \mathbf{v} \cdot \mathbf{v}}_{\text{dynamic pressure}} \tag{15.138}$$

where p_0 is the total pressure, p the static pressure, ρ the density, and $\mathbf{v}$ the velocity vector. The mass flow rate at the boundary is computed from the continuity equation (see next section). Knowing the mass flow rate, the velocity is computed in the same manner as for the specified pressure case using Eq. (15.137). The velocity is thus treated as a known velocity condition (i.e., a Dirichlet boundary condition) with the coefficients in the momentum equation modified according to Eq. (15.136).

15.6.1.3 Outlet Boundary Conditions

Three types of boundary conditions at an outlet are considered: (i) a specified static pressure, (ii) a specified mass flow rate, and (iii) a fully developed flow.

Specified Static Pressure $\left(p_b = p_{specified}; \dot{m}_b?; \mathbf{v}_b?\right)$

For the momentum equation, fully developed conditions are assumed at a specified pressure outlet (Fig. 15.24) implying a zero velocity gradient along the direction of the surface vector at outlet. This is also equivalent to assuming the velocity at the outlet to be equal to that of the boundary element. Thus it is similar to a zero flux boundary condition whose implementation is rather simple.

The modifications to the coefficients are given by

$$\begin{aligned} a_C^{\mathbf{v}} &\leftarrow \underbrace{a_C^{\mathbf{v}}}_{\text{interior faces contribution}} + \underbrace{\dot{m}_b}_{\text{boundary face contribution}} \\ 0 &\leftarrow a_{F=b}^{\mathbf{v}} \\ \mathbf{b}_C^{\mathbf{v}} &\leftarrow \underbrace{\mathbf{b}_C^{\mathbf{v}}}_{\text{interior faces contribution}} - \underbrace{p_b \mathbf{S}_b}_{\text{boundary face contribution}} \end{aligned} \tag{15.139}$$

This sets the contribution of the outlet velocity to zero and uses the specified pressure boundary value in the computation of the pressure gradient.

However to ensure that the flux is zeroed in the outflow surface vector direction only, the velocity is usually extrapolated to the outlet using the boundary flux, which is computed from the boundary element flux as

$$\nabla \mathbf{v}_b = \nabla \mathbf{v}_C - (\nabla \mathbf{v}_C \cdot \mathbf{e}_b)\mathbf{e}_b \tag{15.140}$$

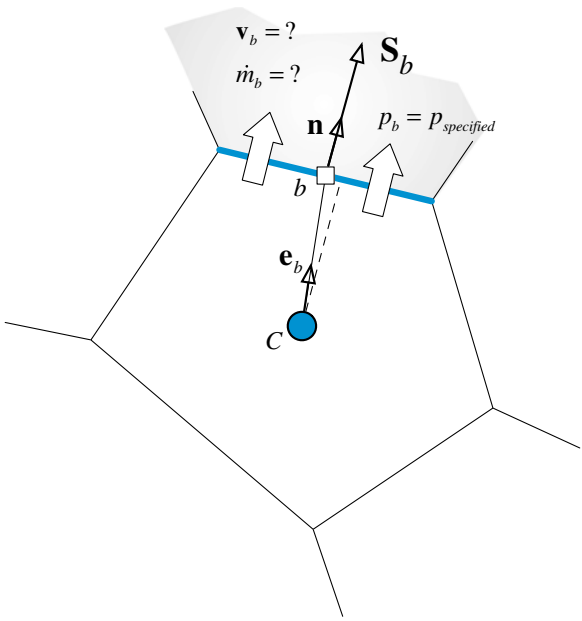

Fig. 15.24 A schematic of specified static pressure boundary condition at outlet

This ensures that the gradient along the boundary surface vector is zero. Then, using a Taylor series expansion, the velocity at the boundary is computed as

$$\mathbf{v}_b = \mathbf{v}_C + \nabla \mathbf{v}_b \cdot \mathbf{d}_{Cb} \tag{15.141}$$

where $\nabla \mathbf{v}_b$ is used instead of $\nabla \mathbf{v}_C$. Therefore an additional correction is now added to the source term and the modifications to the coefficients become

$$\begin{aligned} a_C^{\mathbf{v}} &\leftarrow \underbrace{a_C^{\mathbf{v}}}_{\text{interior faces contribution}} + \underbrace{\dot{m}_b}_{\text{boundary face contribution}} \\ 0 &\leftarrow a_{F=b}^{\mathbf{v}} \\ \mathbf{b}_C^{\mathbf{v}} &\leftarrow \underbrace{\mathbf{b}_C^{\mathbf{v}}}_{\text{interior faces contribution}} \underbrace{-\dot{m}_b(\nabla \mathbf{v}_b \cdot \mathbf{d}_{Cb}) - p_b \mathbf{S}_b}_{\text{boundary face contribution}} \end{aligned} \tag{15.142}$$

Specified Mass Flow Rate $\left(\dot{m}_b = \dot{m}_{specified}; p_b ? \mathbf{v}_b ?\right)$

Since the flow is incompressible, a specified mass flow rate boundary condition (Fig. 15.25) is equivalent to specifying the normal component of velocity at the boundary. The velocity is calculated by assuming its direction to be the same as that at the main grid point, i.e., $(\mathbf{e}_\mathbf{v})_b = (\mathbf{e}_\mathbf{v})_C$. Thus, the velocity is expressed as

$$\mathbf{v}_b = |\mathbf{v}_b|(\mathbf{e}_\mathbf{v})_C \tag{15.143}$$

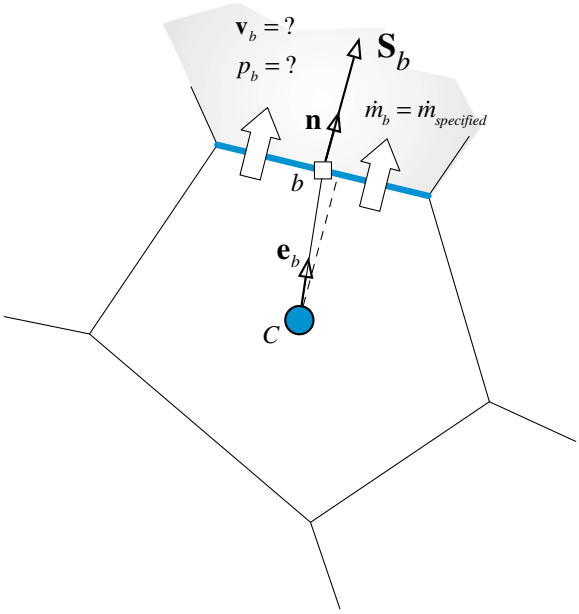

Fig. 15.25 A schematic of specified mass flow rate boundary condition at outlet

with $|\mathbf{v}_b|$ obtained from

$$\dot{m}_b = \rho_b \mathbf{v}_b \cdot \mathbf{S}_b = \rho_b |\mathbf{v}_b| (\mathbf{e}_\mathbf{v})_C \cdot \mathbf{S}_b \Rightarrow |\mathbf{v}_b| = \frac{\dot{m}_b}{\rho_b (\mathbf{e}_\mathbf{v})_C \cdot \mathbf{S}_b} \tag{15.144}$$

allowing $\mathbf{v}_b$ to be calculated. Thus for momentum, a specified velocity boundary condition is applied. The coefficients of the boundary elements are modified according to Eq. (15.136).

Fully Developed Outlet Flow

For a fully developed flow, the velocity gradient normal to the outlet surface is assumed to be zero. Hence the velocity at the outlet is assumed to be known and computed from the zero normal gradient using Eqs. (15.140) and (15.141). As for the pressure at the boundary, it can be extrapolated from the interior pressure field using

$$p_b = p_C + \nabla p_C \cdot \mathbf{d}_{Cb} \tag{15.145}$$

The velocity is treated as known and the coefficients of the momentum equation are modified according to Eq. (15.142).

15.6.1.4 Symmetry Boundary Condition

A scalar is reflected across a symmetry boundary. Thus, a symmetry boundary condition for a scalar variable is imposed by setting its normal gradient to zero, as with an insulated wall boundary condition. For the velocity vector, the symmetry condition shown in Fig. 15.26 also implies that it is reflected about the symmetry boundary with its parallel component (i.e., parallel to the symmetry boundary) retaining magnitude and direction, while its normal component becoming zero. This results in a zero shear stress but a non-zero normal stress along the symmetry boundary. Thus, the same boundary condition can be used to impose a slip wall boundary condition for viscous flows.

The unit vector in the direction normal to the boundary $\mathbf{n}$ and the normal distance to the boundary $d_\perp$ are given by Eq. (15.120). Therefore the velocity components normal and parallel to a symmetry boundary satisfy

$$\begin{aligned} \mathbf{v}_\perp &= 0 \\ \frac{\partial \mathbf{v}_{||}}{\partial n} &= 0 \end{aligned} \tag{15.146}$$

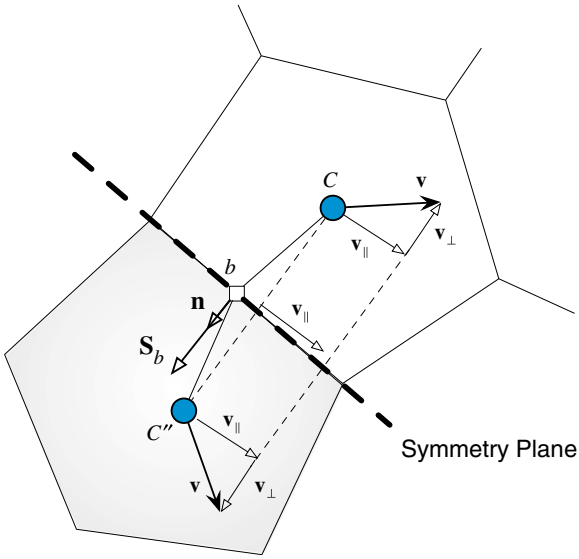

Fig. 15.26 A schematic of a symmetry boundary condition

The normal component of velocity can be written as

$$\mathbf{v}_\perp = (\mathbf{v}\cdot\mathbf{n})\mathbf{n} = \left\{ \begin{array}{l} \left(u_C n_x + v_C n_y + w_C n_z\right) n_x \\ \left(u_C n_x + v_C n_y + w_C n_z\right) n_y \\ \left(u_C n_x + v_C n_y + w_C n_z\right) n_z \end{array} \right\} \tag{15.147}$$

while the parallel component is given by Eq. (15.121). The boundary force $\mathbf{F}_b$ can be decomposed into a normal component $\mathbf{F}_\perp$ and a parallel component $\mathbf{F}_{||}$. As the shear stress along a symmetry boundary is zero, the parallel component of $\mathbf{F}_b$ is zero. Denoting the normal stress by $\sigma_\perp$, the force $\mathbf{F}_b$ is given by

$$\mathbf{F}_b = \sigma_\perp S_b \tag{15.148}$$

The normal stress component can be approximated as

$$\sigma_\perp \simeq -2\mu_b \frac{\mathbf{v}_\perp}{d_\perp} = -2\frac{\mu_b}{d_\perp} \left\{ \begin{array}{l} \left(u_C n_x + v_C n_y + w_C n_z\right) n_x \\ \left(u_C n_x + v_C n_y + w_C n_z\right) n_y \\ \left(u_C n_x + v_C n_y + w_C n_z\right) n_z \end{array} \right\} \tag{15.149}$$

from which the boundary force is found to be

$$\mathbf{F}_b = \mathbf{F}_n = -2\frac{\mu_b S_b}{d_\perp} \left\{ \begin{array}{l} \left(u_C n_x + v_C n_y + w_C n_z\right) n_x \\ \left(u_C n_x + v_C n_y + w_C n_z\right) n_y \\ \left(u_C n_x + v_C n_y + w_C n_z\right) n_z \end{array} \right\} \tag{15.150}$$

The pressure gradient in the direction normal to a symmetry boundary is zero. Mathematically this is written as

$$\nabla p_b \cdot \mathbf{n} = 0 \tag{15.151}$$

The pressure at a symmetry boundary should be extrapolated from the interior of the domain. Therefore to ensure a zero normal gradient, the pressure gradient at the symmetry boundary is computed as

$$\nabla p_b = \nabla p_C - (\nabla p_C \cdot \mathbf{n})\mathbf{n} \tag{15.152}$$

Thus, the pressure is obtained from

$$p_b = p_C + \nabla p_b \cdot \mathbf{d}_{Cb} \tag{15.153}$$

Using the above equations, the coefficients of the boundary elements for the momentum equation in the x, y and z directions are modified as follows:

u-component equation

$$\begin{aligned} a_C^u &\leftarrow \underbrace{a_C^u}_{\text{interior faces contribution}} + \underbrace{\frac{2\mu_b S_b}{d_\perp} n_x^2}_{\text{boundary face contribution}} \\ 0 &\leftarrow a_{F=b}^u \\ b_C^u &\leftarrow \underbrace{b_C^u}_{\text{interior faces contribution}} \underbrace{-\frac{2\mu_b S_b}{d_\perp}\left[v_C n_y + w_C n_z\right] n_x - p_b S_b^x}_{\text{boundary face contribution}} \end{aligned} \tag{15.154}$$

v-component equation

$$\begin{aligned} a_C^v &\leftarrow \underbrace{a_C^v}_{\text{interior faces contribution}} + \underbrace{\frac{2\mu_b S_b}{d_\perp} n_y^2}_{\text{boundary face contribution}} \\ 0 &\leftarrow a_{F=b}^v \\ b_C^v &\leftarrow \underbrace{b_C^v}_{\text{interior faces contribution}} \underbrace{-\frac{2\mu_b S_b}{d_\perp}\left[u_C n_x + w_C n_z\right] n_y - p_b S_b^y}_{\text{boundary face contribution}} \end{aligned} \tag{15.155}$$

w-component equation

$$\begin{aligned} a_C^w &\leftarrow \underbrace{a_C^w}_{\text{interior faces contribution}} + \underbrace{\frac{2\mu_b S_b}{d_\perp} n_z^2}_{\text{boundary face contribution}} \\ 0 &\leftarrow a_{F=b}^w \\ b_C^w &\leftarrow \underbrace{b_C^w}_{\text{interior faces contribution}} \underbrace{-\frac{2\mu_b S_b}{d_\perp}\left[u_C n_x + v_C n_y\right] n_z - p_b S_b^z}_{\text{boundary face contribution}} \end{aligned} \tag{15.156}$$

While this is not a comprehensive list of momentum boundary conditions, it does cover the most common types.

15.6.2 Boundary Conditions for the Pressure Correction Equation

The continuity equation for a boundary cell can be written as

$$\sum_{f\sim nb(C)} \dot{m}_f + \underbrace{\dot{m}_b}_{\text{boundary face}} = 0 \tag{15.157}$$

or

$$\sum_{f\sim nb(C)} (\dot{m}_f^* + \dot{m}_f') + \underbrace{(\dot{m}_b^* + \dot{m}_b')}_{\text{boundary face}} = 0 \tag{15.158}$$

where $\dot{m}_b^*$ is the boundary mass flux and $\dot{m}_b'$ is its correction. While for an internal face the mass flux and its correction are defined by Eqs. (15.100) and (15.101), for a boundary face the definition is slightly different. Since at a boundary face only the boundary cell contributes to the average quantities, the use of Eq. (15.109) in combination with Eqs. (15.100) and (15.101) gives

$$\begin{aligned} \dot{m}_b^* &= \rho_b \mathbf{v}_C^* \cdot \mathbf{S}_b - \rho_b \mathbf{D}_C^{\mathbf{v}} (\nabla p_b^{(n)} - \nabla p_C^{(n)}) \cdot \mathbf{S}_b \\ \dot{m}_b' &= -\rho_b \mathcal{D}_C (p_b' - p_C') \end{aligned} \tag{15.159}$$

In implementing boundary conditions the values of $\dot{m}_b^*$, $\dot{m}_b'$, p_b, and p_b' must be calculated. Based on the discussions related to the momentum equation, three types of boundary conditions can be inferred. The first type is designated by "specified mass flow rate" (e.g., walls or velocity specified at inlets). For this category $\dot{m}_b' = 0$, which is similar to a zero scalar flux boundary condition, and no modification to the pressure-correction equation is needed. The pressure however has to be computed at

the boundary from the interior field. The second type of boundary conditions is termed "pressure specified" where $p'_b = 0$ and for which a Dirichlet-like condition has to be enforced for the pressure-correction equation. For this condition, $\dot{m}^*_b$ is computed from the boundary and interior pressure field. In the third type, an implicit relation exists between the pressure and the mass flow rate, as in a specified total pressure boundary condition. In this case, an explicit equation is extracted from the implicit relation and substituted into the pressure-correction equation.

Details regarding the various types of boundary conditions and their implementation are now given.

15.6.2.1 Wall Boundary Condition

$$(p_b = ?; \dot{m}_b = 0; \mathbf{v}_b = \mathbf{v}_{wall}) \quad \text{or} \quad (p_b = ?; \dot{m}_b = 0; \mathbf{F}_b = 0)$$

Whether it is a slip (Fig. 15.20) or no-slip (Fig. 15.19) wall boundary condition the mass flow rate is zero. Therefore $\dot{m}'_b = 0$, which is equivalent to a specified zero flux and implying that no modification is needed for the pressure-correction equation. However the pressure at the wall is required and is computed using Eq. (15.127) or Eq. (15.133) or a low order extrapolation profile, as shown below.

$$p_b = \begin{cases} p_C + \nabla p_C^{(n)} \cdot \mathbf{d}_{Cb} & Eq.\ (15.127) \\ p_C + \dfrac{\left(\nabla p_C^{(n)} \cdot \mathbf{S}'_b - \nabla p_b^{(n)} \cdot \mathbf{T}_b\right)}{\mathcal{D}_C} & Eq.\ (15.133) \\ p_C & loworder\ extrapolation \end{cases} \tag{15.160}$$

15.6.2.2 Inlet Boundary Conditions

Specified Velocity $(p_b = ?; \dot{m}_b\ specified; \mathbf{v}_b\ specified)$

For a specified velocity at inlet (Fig. 15.21), the mass flux is known and its correction is set to zero, i.e., $\dot{m}'_b = 0$. Thus, similar to a wall boundary condition, the term is simply dropped from the pressure-correction equation. The pressure at the boundary is extrapolated from the internal pressure field using Eq. (15.127) or Eq. (15.133) or a low order extrapolation profile as summarized in Eq. (15.160).

Specified Pressure and Velocity Direction $(p_b = p_{specified}; \dot{m}_b?; \mathbf{e}_\mathbf{v}\ specified; \mathbf{v}_b?)$

In the case of a specified static pressure at inlet (Fig. 15.22), p_b is known and thus p'_b is set to zero but $\dot{m}'_b \neq 0$. The inlet is treated as a Dirichlet boundary condition for the pressure-correction equation. The coefficient of the p' equation becomes

$$a_C^{p'} = \underbrace{\sum_{f\sim nb(C)} \rho_f \mathcal{D}_f}_{\text{interior faces contribution}} + \underbrace{\rho_b \mathcal{D}_C}_{\text{boundary face contribution}} \tag{15.161}$$

Specified Total Pressure and Velocity Direction $\left(p_{o,b} = p_{o,specified};\ \dot{m}_b?;\ \mathbf{e_v}\ specified;\ \mathbf{v}_b?\right)$

As mentioned earlier, for a specified total pressure (Fig. 15.23), the velocity direction should also be specified. The total pressure relation given by Eq. (15.138) is first rewritten as a function of the mass flow rate and pressure by replacing the velocity magnitude by the mass flux. Thus,

$$\begin{aligned} \dot{m}_b = \rho \mathbf{v}_b \cdot \mathbf{S}_b = \rho |\mathbf{v}_b| \mathbf{e_v} \cdot \mathbf{S}_b \Rightarrow \rho |\mathbf{v}_b| = \frac{\dot{m}_b}{\mathbf{e_v} \cdot \mathbf{S}_b} \\ \Rightarrow p_{o,b} = p_b + \frac{1}{2\rho_b} \frac{\dot{m}_b^2}{\left(\mathbf{e_v} \cdot \mathbf{S}_b\right)^2} \end{aligned} \tag{15.162}$$

Using a Taylor expansion about p_b, p_b' is obtained as

$$p_b + p_b' = p_b + \frac{\partial p_b}{\partial \dot{m}_b}\left(\dot{m}_b'\right) \Rightarrow p_b' = \frac{\partial p_b}{\partial \dot{m}_b} \dot{m}_b' \tag{15.163}$$

Differentiating Eq. (15.162) with respect to $\dot{m}_b$ and substituting into Eq. (15.163), the final form of p_b' is found to be

$$p_b' = -\frac{\dot{m}_b^*}{\rho_b \left(\mathbf{e_v} \cdot \mathbf{S}_b\right)^2} \dot{m}_b' = -\frac{\rho_b \mathbf{v}_b^* \cdot \mathbf{v}_b^*}{\dot{m}_b^*} \dot{m}_b' \tag{15.164}$$

Substituting Eq. (15.164) in Eq. (15.159), the mass flux correction is expressed as

$$\dot{m}_b' = -\rho_b \mathcal{D}_C \left(p_b' - p_C'\right) \Rightarrow \dot{m}_b' = \frac{\dot{m}_b^* \rho_b \mathcal{D}_C}{\dot{m}_b^* - \mathcal{D}_C \left(\rho_b \mathbf{v}_b^* \cdot \rho_b \mathbf{v}_b^*\right)} p_C' \tag{15.165}$$

Replacing $\dot{m}_b'$ in the expanded continuity equation (Eq. 15.158) by its expression from Eq. (15.165), the modified $a_C^{p'}$ coefficient for the boundary cell becomes

$$a_C^{p'} = \underbrace{\sum_{f\sim nb(C)} \rho_f \mathcal{D}_f}_{\text{interior faces contribution}} + \underbrace{\frac{\rho_b \dot{m}_b^* \mathcal{D}_C}{\dot{m}_b^* - \mathcal{D}_C \left(\rho_b \mathbf{v}_b^* \cdot \rho_b \mathbf{v}_b^*\right)}}_{\text{boundary face contribution}} \tag{15.166}$$

15.6.2.3 Outlet Boundary Conditions

Specified Pressure $\left(p_b = p_{specified}; \dot{m}_b?;\ \mathbf{v}_b?\right)$

For a specified pressure at outlet (Fig. 15.24) p'_b is set to zero. On the other hand, $\dot{m}'_b$ is computed as

$$\dot{m}'_b = -\rho_b \mathcal{D}_C (p'_b - p'_C) \tag{15.167}$$

The velocity direction being needed, it is customary to take the direction of $\mathbf{v}_b$ to be that of the upwind velocity $\mathbf{v}_C$. The expression of the $a_C^{p'}$ coefficient in the pressure-correction equation becomes

$$a_C^{p'} = \underbrace{\sum_{f \sim nb(C)} \rho_f \mathcal{D}_f}_{\text{interior faces contribution}} + \underbrace{\rho_b \mathcal{D}_C}_{\text{boundary face contribution}} \tag{15.168}$$

Specified Mass Flow Rate $\left(\dot{m}_b = \dot{m}_{specified}; p_b? \mathbf{v}_b?\right)$

For a specified mass flow rate at outlet (Fig. 15.25), $\dot{m}'_b$ is zero and is simply dropped from the pressure correction equation with no modifications required for the coefficients of the boundary elements. By setting $\dot{m}'_b$ to zero in Eq. (15.159), the pressure correction (or pressure) at the boundary is set equal to the pressure correction (or pressure) at the boundary cell centroid.

Fully Developed Outlet Flow

For a fully developed flow, the velocity at the outlet is assumed to be known and computed from the zero normal gradient. This means that $\dot{m}_b$ at the outlet is known. Therefore no correction is needed and $\dot{m}'_b$ is set to zero. However, as the boundary pressure is unknown, it is extrapolated from the interior pressure field. Since the velocity is iteratively updated, the above treatment does not guarantee overall conservation except at convergence. It is customary with incompressible flows to overcome this issue and to enforce global mass conservation at any iteration by modifying $\dot{m}_b$ at the boundary to satisfy overall mass conservation. This is done by calculating the total mass flow rate entering the domain $\sum \dot{m}_{in}$. Then based on the calculated mass flow rates at outlet boundaries, the total mass flow rate leaving

the domain $\sum \dot{m}_{out}$ is computed. The mass flow rate at an outlet is adjusted according to

$$\dot{m}_{out} \leftarrow \dot{m}_{out} \frac{\sum \dot{m}_{in}}{\sum \dot{m}_{out}} \tag{15.169}$$

To be able to apply the above treatment, the outlet should be placed far away from any recirculation zone.

15.6.2.4 Symmetry Boundary Condition

The mass flow rate across a symmetry line (Fig. 15.26) is zero and as such its correction is set to zero, i.e., $\dot{m}'_b = 0$. Thus, similar to a wall boundary condition, the mass flow rate correction term is simply dropped from the pressure-correction equation. The pressure at the boundary is extrapolated from the internal pressure field using Eq. (15.127) or Eq. (15.133) or a low order extrapolation profile as summarized in Eq. (15.160).

15.6.2.5 The Relative Nature of Pressure

For incompressible flow problems in which the normal velocities are prescribed on all boundaries, a difficulty arises due to the relative nature of pressure. In such cases, since only the pressure gradient appears in the momentum equation, there is no means for determining the absolute level of pressure, and only pressure differences have physical meaning. This indeterminacy of the pressure level leads to a singular coefficient matrix $\mathbf{A}$ and the direct solution of the system $\mathbf{A}\phi = \mathbf{b}$ fails. The singularity is easily removed by simply setting the pressure at one point in the domain to a prescribed value. The remaining pressures are then calculated relative to this value.

15.7 The SIMPLE Family of Algorithms

In the SIMPLE algorithm [13], velocity and pressure are treated in a segregated (sequential) manner, with the pressure field computed by deriving a pressure correction equation that exploits the discrete momentum equation to replace the velocity field in the continuity equation with a pressure term. In the derivation, a velocity correction term, $\overline{\mathbf{H}_f}[\mathbf{v}']$, was neglected as retaining it renders the equation unmanageable.

Discarding this term does not affect the final solution since its value is zero at convergence. Rather, it affects the path to convergence because during the initial

iterations its value can be significant. This large value may either cause divergence or slow down the rate of convergence as a result of an overestimated pressure correction field. To counterbalance that, in SIMPLE the pressure is under relaxed by computing its value using $p = p^* + \lambda^p p'$, where λ^p is the pressure under relaxation factor. For optimum convergence, λ^p is usually set equal to $(1-\lambda^{\mathbf{v}})$, where $\lambda^{\mathbf{v}}$ is the momentum under relaxation factor, with more information on this provided later.

Despite the use of under relaxation, the rate of convergence of the SIMPLE algorithm remains problem dependent and researchers sought alternatives for further improvements. Their effort culminated in the development of a SIMPLE-like family of algorithms such as SIMPLEC [17], SIMPLER [3], PISO [18], SIMPLEX [5], PRIME [19], SIMPLEM [20], and SIMPLEST [21]. Moukalled and Darwish [22] unified the formulation of these algorithms for both incompressible and compressible flows while Darwish et al. [23] and Jang et al. [24] assessed their performance. It is not the intention here to give a full account of these algorithms, rather, attention will be focussed on the two most popular variants, which are the SIMPLEC (SIMPLE Consistent) algorithm of Van Doormal and Raithby and the PISO (Pressure-Implicit Split Operator) algorithm of Issa. These two algorithms present two different approaches for dealing with the $\overline{\mathbf{H}_f}[\mathbf{v}']$ term. In SIMPLEC the velocity correction at the main grid point is approximated as the weighted average of the velocity corrections at the neighboring locations altering the term $\overline{\mathbf{H}_f}[\mathbf{v}']$ into a modified one, $\tilde{\mathbf{H}}_f[\mathbf{v}']$, of smaller magnitude, which is then neglected. In the PISO algorithm, the $\overline{\mathbf{H}_f}[\mathbf{v}']$ term is accounted for as part of the split operator approach. In all other algorithms, the $\overline{\mathbf{H}_f}[\mathbf{v}']$ term is neglected as in SIMPLE and modifications are introduced either to the momentum equations or the $\mathbf{D}^{\mathbf{v}}$ operator. Because the PISO algorithm is equivalent to one step of the SIMPLE algorithm and one or more steps of the PRIME algorithm, the latter is also detailed.

In the PRIME [19] algorithm the momentum equation is solved explicitly. This explicit treatment of the momentum equation is justified by its small contribution to the convergence of the entire flow field. On the other hand, finding the correct solution for the pressure field represents the most important factor impacting the overall convergence.

In SIMPLEST [21], the coefficients in the momentum equation are separated into their convection and diffusion parts with the convection terms treated explicitly and the diffusion terms implicitly, thus affecting $\mathbf{D}^{\mathbf{v}}$ and $\mathbf{H}$. The justification for the explicit treatment of convection is based on the similarity between the propagation of disturbances at a finite speed without any change in magnitude in a pure convection situation, and the propagation of error, from a particular point to the neighboring grid points, in a single iteration of explicit iterative methods. While the implicit treatment of diffusion is argued based on the similarity between the propagation of disturbances in a pure diffusion situation instantaneously in all directions with rapid decay in their amplitude and the reduction of errors throughout the entire solution domain, in a single iteration, by implicit solution methods.

In SIMPLEM (SIMPLE-Modified) [20], the pressure-correction equation is solved before the momentum equation implying that the pressure field is computed using the old velocity field. This results in better pressure corrections than velocity corrections and interchange the disadvantages and advantages of the SIMPLE algorithm.

In SIMPLER (SIMPLE-Revised) [3], an additional equation is developed from which the pressure is directly calculated while the SIMPLE-like pressure-correction equation is used to update the velocity field. The reason for a separate pressure equation being that, once the velocity field is updated using the predicted pressure correction field, it no longer satisfies the momentum equations. Therefore, the pressure should be calculated from another equation to match the velocities, so that the momentum equations are also satisfied.

The SIMPLEX algorithm [5] was developed with the aim of ensuring that the rate of convergence will not degrade with grid refinement. It differs from SIMPLE in the way the $\mathbf{D}^{\mathbf{v}}$ field is computed. This is done by using an additional set of equations, which is developed and solved based on the assumption that the influence of the spatial distribution of pressure difference changes little with grid refinement. Therefore, if the pressure difference influence is restricted to a cell, it would be appropriate to assume that, by extrapolation, the pressure difference at the main grid point adequately represents the pressure differences at the element faces.

Though all of the above algorithms were originally derived for a segregated grid, they are applicable within a collocated grid framework.

15.7.1 The SIMPLEC Algorithm

The SIMPLEC (SIMPLE-Consistent) [17] algorithm is a modified version of the SIMPLE algorithm derived by simply assuming that the velocity correction at point C is the weighted average of the corrections at the neighboring grid points. Mathematically this is expressed by

$$\mathbf{v}'_C \approx \frac{\sum_{F\sim NB(C)} a_F^{\mathbf{v}} \mathbf{v}'_F}{\sum_{F\sim NB(C)} a_F^{\mathbf{v}}} \Rightarrow \sum_{F\sim NB(C)} a_F^{\mathbf{v}} \mathbf{v}'_F \approx \mathbf{v}'_C \sum_{F\sim NB(C)} a_F^{\mathbf{v}} \tag{15.170}$$

and using the $\mathbf{H}$ operator, Eq. (15.170) can be written as

$$\sum_{F\sim NB(C)} \frac{a_F^{\mathbf{v}} \mathbf{v}'_F}{a_C^{\mathbf{v}}} \approx \mathbf{v}'_C \sum_{F\sim NB(C)} \frac{a_F^{\mathbf{v}}}{a_C^{\mathbf{v}}} \Rightarrow \mathbf{H}_C[\mathbf{v}'] \approx \mathbf{v}'_C \mathbf{H}_C[1] \tag{15.171}$$

Therefore instead of neglecting the $\overline{\mathbf{H}_C}[\mathbf{v}']$ term as in SIMPLE, it is replaced by the approximate value given by the above equation. With this approximation the velocity correction given by Eq. (15.85) becomes

$$(1 + \mathbf{H}_C[1])\mathbf{v}'_C = -\mathbf{D}^{\mathbf{v}}_C(\nabla p')_C \Rightarrow \mathbf{v}'_C = -\tilde{\mathbf{D}}^{\mathbf{v}}_C(\nabla p')_C \tag{15.172}$$

Equation (15.172) can then be used to derive the pressure correction equation.

The same result can be achieved by adding and subtracting the term $\sum_{F \sim NB(C)} a^{\mathbf{v}}_F \mathbf{v}_C$ from the momentum equation obtained by combining Eqs. (15.76) and (15.77), leading to the following modified equation:

$$\left(a^{\mathbf{v}}_C + \sum_{F \sim NB(C)} a^{\mathbf{v}}_F\right)\mathbf{v}_C + \sum_{F \sim NB(C)} a^{\mathbf{v}}_F(\mathbf{v}_F - \mathbf{v}_C) = -V_C(\nabla p)_C + \hat{\mathbf{b}}^{\mathbf{v}}_C \tag{15.173}$$

which, in turn, can be written as

$$\mathbf{v}_C + \tilde{\mathbf{H}}_C[\mathbf{v} - \mathbf{v}_C] = -\tilde{\mathbf{D}}^{\mathbf{v}}_C(\nabla p)_C + \tilde{\mathbf{B}}^{\mathbf{v}}_C \tag{15.174}$$

By using Eq. (15.174), the velocity correction equation becomes

$$\mathbf{v}'_C = -\tilde{\mathbf{H}}_C[\mathbf{v}' - \mathbf{v}'_C] - \tilde{\mathbf{D}}^{\mathbf{v}}_C(\nabla p')_C \tag{15.175}$$

Then the term $\tilde{\mathbf{H}}_C[\mathbf{v}' - \mathbf{v}'_C]$ is dropped, which is equivalent to the approximation given by Eq. (15.171), and the modified velocity correction is used in deriving the pressure correction equation.

Due to a better estimate in SIMPLEC (i.e., a smaller term is dropped), the relaxation of pressure becomes unnecessary and as compared to SIMPLE, the resulting velocity corrections will satisfy better the momentum equations. Consequently, a higher rate of convergence is obtained. Thus, with the exceptions of dropping $\tilde{\mathbf{H}}_C[\mathbf{v}' - \mathbf{v}'_C]$ rather than $\mathbf{H}_C[\mathbf{v}']$ and replacing $\mathbf{D}^{\mathbf{v}}_C$ by $\tilde{\mathbf{D}}^{\mathbf{v}}_C$, the steps involved in the SIMPLEC algorithm are similar to those of the SIMPLE algorithm.

15.7.2 The PRIME Algorithm

In the PRIME (PRessure Implicit Momentum Explicit) [19] algorithm, the momentum equation is solved explicitly. This explicit treatment is justified by the small contribution to the convergence of the entire flow field by the iterative sweeps of the momentum equation. On the other hand, finding the correct solution for the pressure field represents the most important factor in the overall convergence. Based on this argument, the PRIME algorithm can be summarized as follows:

The momentum equation is solved explicitly to obtain a new velocity field $\mathbf{v}^*$ using

$$\mathbf{v}_C^* = -\mathbf{H}_C\left[\mathbf{v}^{(n)}\right] - \mathbf{D}_C^{\mathbf{v}}\left(\nabla p^{(n)}\right)_C + \mathbf{B}_C^{\mathbf{v}} \tag{15.176}$$

This velocity field is employed to derive the pressure correction equation. Thus defining the correction fields such that

$$\mathbf{v}_C^{**} = \mathbf{v}_C^* + \mathbf{v}'_C \quad p_C^* = p_C^{(n)} + p'_C \tag{15.177}$$

the corrected field would satisfy

$$\mathbf{v}_C^{**} = -\mathbf{H}_C[\mathbf{v}^{**}] - \mathbf{D}_C^{\mathbf{v}}(\nabla p^*)_C + \mathbf{B}_C^{\mathbf{v}} = -\mathbf{H}_C[\mathbf{v}^* + \mathbf{v}'] - \mathbf{D}_C^{\mathbf{v}}\left[\nabla\left(p^{(n)} + p'\right)\right]_C + \mathbf{B}_C^{\mathbf{v}} \tag{15.178}$$

leading to the following expression relating velocity and pressure correction:

$$\mathbf{v}'_C = \underline{-\left(\mathbf{H}_C\left[\mathbf{v}^* - \mathbf{v}^{(n)}\right] + \mathbf{H}_C[\mathbf{v}']\right)} - \mathbf{D}_C^{\mathbf{v}}\nabla p'_C \tag{15.179}$$

Substituting Eq. (15.179) and its correction into the continuity equation yields

$$-\sum_{f\sim nb(C)} \rho_f \overline{\mathbf{D}_f}\nabla p'_f \cdot \mathbf{S}_f = -\sum_{f\sim nb(C)} \dot{m}_f^* + \underline{\sum_{f\sim nb(C)} \left[\rho_f\left(\overline{\mathbf{H}_f}\left[\mathbf{v}^* - \mathbf{v}^{(n)}\right] + \overline{\mathbf{H}_f}[\mathbf{v}']\right)\cdot \mathbf{S}_f\right]} \tag{15.180}$$

where the underlined terms in Eqs. (15.179) and (15.180) are neglected.

The terms neglected in PRIME $\left(\mathbf{H}_C\left[\mathbf{v}^* - \mathbf{v}^{(n)}\right] + \mathbf{H}_C[\mathbf{v}']\right)$ can become smaller than the term neglected in SIMPLE $(\mathbf{H}_C[\mathbf{v}'])$ if $\mathbf{H}_C[\mathbf{v}']$ and $\mathbf{H}_C\left[\mathbf{v}^* - \mathbf{v}^{(n)}\right]$ are of opposite signs. It is worth noting that $\mathbf{H}_C[\mathbf{v}'] = \mathbf{H}_C[\mathbf{v}^{**} - \mathbf{v}^*]$ is a correction to satisfy continuity, while $\mathbf{H}_C\left[\mathbf{v}^* - \mathbf{v}^{(n)}\right]$ is a correction to satisfy momentum. Usually the corrector added to satisfy momentum is opposite to that added to satisfy continuity and hence, the neglected term $\left(\mathbf{H}_C\left[\mathbf{v}^* - \mathbf{v}^{(n)}\right] + \mathbf{H}_C[\mathbf{v}']\right)$ is smaller. Moreover, since the momentum equations are explicitly solved, no under-relaxation is required. This has the advantage of increasing the stability of the algorithm.

15.7.3 The PISO Algorithm

In the PISO algorithm [18, 25], the $\mathbf{H}_C[\mathbf{v}']$ term is accounted for as part of a correction procedure composed of two or more steps. The first step is similar to the

SIMPLE algorithm where $\mathbf{v}'$ is computed from Eq. (15.83) while neglecting $\mathbf{H}_C[\mathbf{v}']$. The continuity satisfying velocity $\mathbf{v}^{**}$ and pressure p^* fields are used to recalculate the coefficients of the momentum equation and then to solve it explicitly. The new velocity field $\mathbf{v}^{***}$ is used to calculate the mass flow rate field $\dot{m}^{***}$ at the element faces using the Rhie-Chow interpolation. Then, $\mathbf{H}_C[\mathbf{v}']$ is partially recovered in a second corrector step where the velocity correction is written as

$$\begin{aligned}
\mathbf{v}_C^{****} &= \mathbf{v}_C^{***} + \mathbf{v}''_C \\
&= -\mathbf{H}_C^{**}[\mathbf{v}^{**}] - (\mathbf{D}_C^{\mathbf{v}})^{**}(\nabla p^*)_C + \mathbf{v}''_C \\
&= -\mathbf{H}_C^{**}[\mathbf{v}^* + \mathbf{v}'] - (\mathbf{D}_C^{\mathbf{v}})^{**}(\nabla p^*)_C + \mathbf{v}''_C \\
&= -\mathbf{H}_C^{**}[\mathbf{v}^*] - \mathbf{H}_C^{**}[\mathbf{v}'] - (\mathbf{D}_C^{\mathbf{v}})^{**}(\nabla p^*)_C + \mathbf{v}''_C \\
&= \underbrace{-\mathbf{H}_C^{**}[\mathbf{v}^*] - (\mathbf{D}_C^{\mathbf{v}})^{**}(\nabla p^*)_C}_{\approx \mathbf{v}_C^{**}} - \mathbf{H}_C^{**}\left[-\mathbf{D}_C^{\mathbf{v}}(\nabla p')_C\right] + \mathbf{v}''_C \\
&\approx \mathbf{v}_C^{**} + \underline{\mathbf{v}''_C - \mathbf{H}_C^{**}\left[\mathbf{D}_C^{\mathbf{v}}(\nabla p')_C\right]}
\end{aligned} \tag{15.181}$$

In Eq. (15.181) the underlined term represents the portion of the $\mathbf{H}_C[\mathbf{v}']$ that is recovered by the second corrector step. The second velocity correction satisfies

$$\mathbf{v}''_C = -\underline{\mathbf{H}_C^{**}[\mathbf{v}'']} - (\mathbf{D}_C^{\mathbf{v}})^{**}(\nabla p'')_C \tag{15.182}$$

Using Eq. (15.181) with the Rhie-Chow interpolation between points C and F, a new pressure-correction field is obtained as

$$-\sum_{f\sim nb(C)} \rho_f \overline{\mathbf{D}_f} \nabla p''_f \cdot \mathbf{S}_f = -\sum_{f\sim nb(C)} \dot{m}_f^* + \underline{\sum_{f\sim nb(C)} \left(\rho_f \overline{\mathbf{H}_f}[\mathbf{v}''] \cdot \mathbf{S}_f\right)} \tag{15.183}$$

where the underlined terms in Eqs. (15.182) and (15.183) are again neglected. This corrector step may be repeated as many times as desired, each time recovering a new additional portion of $\mathbf{H}_C[\mathbf{v}']$.

By following the sequence of events, it can be easily seen that PISO may be considered as a combination of one SIMPLE step and one or more PRIME steps, hence combining the implicitness of the SIMPLE algorithm with the stability of the PRIME algorithm. The sequence of events in the collocated PISO algorithm can be summarized as follows:

1. To compute the solution at time $t + \Delta t$, use as an initial guess the solution at time t for pressure, velocity, and mass flow rate fields $p^{(n)}$, $u^{(n)}$, and $\dot{m}^{(n)}$, respectively.

SIMPLE Step

2. Solve implicitly the momentum equation given by Eq. (15.70) to obtain a new velocity field $\mathbf{v}^*$.
3. Update the mass flow rate at the cell faces using the Rhie-Chow interpolation technique (Eq. 15.100) to obtain a momentum satisfying mass flow rate field $\dot{m}_f^*$.

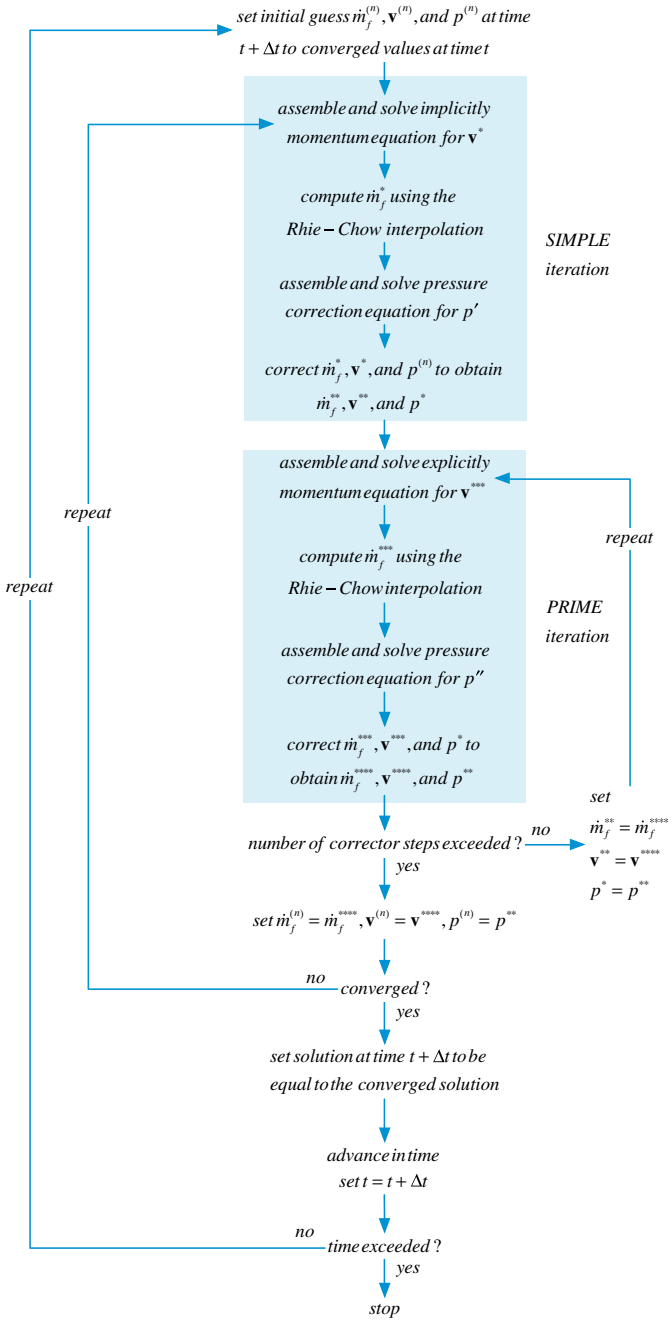

Fig. 15.27 A flow chart of the PISO algorithm

4. Using the new mass flow rates, assemble the pressure correction equation (Eq. 15.98) and solve it to obtain a pressure correction field p'.
5. Update the pressure and velocity fields at the cell centroids and the mass flow rate at the cell faces to obtain continuity-satisfying fields using Eq. (15.101).

PRIME Step(s)

6. Using the latest available velocity and pressure fields, calculate the coefficients of the momentum equation and solve it explicitly.
7. Update the mass flow rate at the cell faces using the Rhie-Chow interpolation technique.
8. Using the new mass flow rates, assemble the pressure correction equation (Eq. 15.183) and solve it to obtain a pressure correction field.
9. Update pressure, velocity, and mass flow rate fields using expressions similar to the ones given in Eq. (15.101).
10. Go to step 6 and repeat based on the desired number of corrector steps.
11. Set the initial guess for velocity, mass flow rate, and pressure as u^{**}, $\dot{m}^{**}$, and p^{*}.
12. Go back to step 2 and repeat until convergence.
13. Set the solution at time $t + \Delta t$ to be equal to the converged solution and set the current time $t + \Delta t$ to be t.
14. Advance to the next time step.
15. Go back to step 1 and repeat until the last time step is reached.

A flowchart of the PISO algorithm is presented in Fig. 15.27.

15.8 Optimum Under-Relaxation Factor Values for v and p'

To promote convergence in the SIMPLE algorithm the momentum and continuity equations are under relaxed using the under relaxation factors $\lambda^{\mathbf{v}}$ and λ^{p}, respectively. An important task is to find the under relaxation values that will result in the optimum convergence rate. Recalling that the velocity correction is obtained without any under relaxation from

$$\mathbf{v}'_C = -\mathbf{D}_C(\nabla p')_C \tag{15.184}$$

Moreover, in calculating the pressure field, the pressure correction is under relaxed in order for the velocity correction field given by Eq. (15.184) to satisfy the exact velocity correction equation given by

$$\mathbf{v}'_C = -\mathbf{H}_C[\mathbf{v}'] - \lambda^{p}\mathbf{D}_C(\nabla p')_C \tag{15.185}$$

Equating Eqs. (15.184) and (15.185), an expression for λ^p is obtained as

$$-\mathbf{D}_C(\nabla p')_C = -\mathbf{H}_C[\mathbf{v}'] - \lambda^p \mathbf{D}_C(\nabla p')_C \Rightarrow \lambda^p = 1 + \frac{\mathbf{H}_C[\mathbf{v}']}{\mathbf{v}'_C} = 1 + \frac{\sum_{F\sim NB(C)} a_F^{\mathbf{v}} \mathbf{v}'_F}{a_C^{\mathbf{v}} \mathbf{v}'_C} \tag{15.186}$$

The SIMPLEC algorithm eliminated the need to under relax pressure correction and resulted in the optimum acceleration rate. Therefore, using the approximation introduced in SIMPLEC, the velocity correction at C can be written as the weighted average of the velocity corrections at the neighboring grid points such that

$$\mathbf{v}'_C \approx \frac{\sum_{F\sim NB(C)} a_F^{\mathbf{v}} \mathbf{v}'_F}{\sum_{F\sim NB(C)} a_F^{\mathbf{v}}} \tag{15.187}$$

From Eqs. (15.70)–(15.73) the coefficient $a_C^{\mathbf{v}}$ can be expressed as

$$a_C^{\mathbf{v}} = \frac{1}{\lambda^{\mathbf{v}}} \left(a_C^{\mathbf{v}} - \sum_{F\sim NB(C)} a_F^{\mathbf{v}} + \sum_{f\sim nb(C)} \dot{m}_f \right) \tag{15.188}$$

which in the limit of a steady state solution (the case for which under relaxation is used, since for an unsteady situation the time step plays the role of the under relaxation factor) reduces to

$$a_C^{\mathbf{v}} = -\frac{1}{\lambda^{\mathbf{v}}} \sum_{F\sim NB(C)} a_F^{\mathbf{v}} \tag{15.189}$$

Substituting Eq. (15.189) in Eq. (15.187), the velocity correction is approximated as

$$\mathbf{v}'_C \approx -\frac{\sum_{F\sim NB(C)} a_F^{\mathbf{v}} \mathbf{v}'_F}{\lambda^{\mathbf{v}} a_C^{\mathbf{v}}} \Rightarrow a_C^{\mathbf{v}} \mathbf{v}'_C \approx -\frac{\sum_{F\sim NB(C)} a_F^{\mathbf{v}} \mathbf{v}'_F}{\lambda^{\mathbf{v}}} \tag{15.190}$$

substituting Eq. (15.190) in Eq. (15.186), an expression relating $\lambda^{\mathbf{v}}$ and λ^p is obtained as

$$\lambda^p \approx 1 - \lambda^{\mathbf{v}} \tag{15.191}$$

Experience has shown that the performance of the SIMPLE algorithm with its under relaxation factors satisfying Eq. (15.191) is similar to that of the SIMPLEC algorithm.

15.9 Treatment of Various Terms with the Rhie-Chow Interpolation

15.9.1 Treatment of the Under-Relaxation Term

The use of the collocated grid method with the Rhie-Chow interpolation resulted in solutions that are dependent on the value of under relaxation factor in the momentum equation. To eliminate this dependence, a modification to the Rhie-Chow interpolation is required. The under relaxed momentum equation is written as

$$\frac{1}{\lambda^{\mathbf{v}}} a_C^{\mathbf{v}} \mathbf{v}_C = -\sum_{F \sim NB(C)} a_F^{\mathbf{v}} \mathbf{v}_F + \mathbf{b}_C^{\mathbf{v}} - V_C \nabla p_C + \left(\frac{1-\lambda^{\mathbf{v}}}{\lambda^{\mathbf{v}}}\right) a_C^{\mathbf{v}} \mathbf{v}_C^{(n)} \tag{15.192}$$

where $\mathbf{b}_C^{\mathbf{v}}$ is the source term of the momentum equation from which the pressure and under relaxation source terms are extracted and $\mathbf{v}_C^{(n)}$ is the previous iteration value of velocity at cell centroid C. The corresponding under relaxed momentum equation using a staggered grid formulation can be expressed as

$$\frac{1}{\lambda^{\mathbf{v}}} a_f^{\mathbf{v}} \mathbf{v}_f = -\sum_{nb \sim NB(f)} a_{nb}^{\mathbf{v}} \mathbf{v}_{nb} + \mathbf{b}_f^{\mathbf{v}} - V_f \nabla p_f + \left(\frac{1-\lambda^{\mathbf{v}}}{\lambda^{\mathbf{v}}}\right) a_f^{\mathbf{v}} \mathbf{v}_f^{(n)} \tag{15.193}$$

The Rhie-Chow interpolation method mimics the staggered grid formulation by forming a pseudo-momentum equation at the cell face. It is because of this behavioral imitation that the Rhie-Chow interpolation is successful. Therefore as a guiding principle, the yardstick to any modification to the Rhie-Chow interpolation should be whether the modified formulation is similar to the staggered grid formulation. Therefore, the form of the under relaxed equation using the Rhie-Chow interpolation should be given by

$$\frac{1}{\lambda^{\mathbf{v}}} \overline{a_f^{\mathbf{v}}} \mathbf{v}_f = -\overline{\sum_{nb \sim NB(f)} a_{nb}^{\mathbf{v}} \mathbf{v}_{nb} + \mathbf{b}_f^{\mathbf{v}}} - \overline{V_f} \nabla p_f + \left(\frac{1-\lambda^{\mathbf{v}}}{\lambda^{\mathbf{v}}}\right) \overline{a_f^{\mathbf{v}}} \mathbf{v}_f^{(n)} \tag{15.194}$$

The average of the first term on the right hand side is obtained as

$$\begin{aligned}
-\overline{\sum_{nb \sim NB(f)} a_{nb}^{\mathbf{v}} \mathbf{v}_{nb} + \mathbf{b}_f^{\mathbf{v}}} &= -g_C \left(\sum_{F \sim NB(C)} \left(a_F^{\mathbf{v}} \mathbf{v}_F \right) + \mathbf{b}_C^{\mathbf{v}} \right) - g_F \left(\sum_{N \sim NB(F)} \left(a_N^{\mathbf{v}} \mathbf{v}_N \right) + \mathbf{b}_F^{\mathbf{v}} \right) \\
&= g_C \left[\frac{1}{\lambda^{\mathbf{v}}} a_C^{\mathbf{v}} \mathbf{v}_C + V_C \nabla p_C - \left(\frac{1-\lambda^{\mathbf{v}}}{\lambda^{\mathbf{v}}} \right) a_C^{\mathbf{v}} \mathbf{v}_C^{(n)} \right] \\
&\quad + g_F \left[\frac{1}{\lambda^{\mathbf{v}}} a_F^{\mathbf{v}} \mathbf{v}_F + V_F \nabla p_F - \left(\frac{1-\lambda^{\mathbf{v}}}{\lambda^{\mathbf{v}}} \right) a_F^{\mathbf{v}} \mathbf{v}_F^{(n)} \right] \\
&= \frac{1}{\lambda^{\mathbf{v}}} \overline{a_f^{\mathbf{v}} \mathbf{v}_f} + \overline{V_f \nabla p_f} - \left(\frac{1-\lambda^{\mathbf{v}}}{\lambda^{\mathbf{v}}} \right) \overline{a_f^{\mathbf{v}} \mathbf{v}_f^{(n)}}
\end{aligned} \tag{15.195}$$

Substituting Eq. (15.195) into Eq. (15.194), the extended Rhie-Chow interpolated cell face velocity $\mathbf{v}_f$ is obtained as

$$\mathbf{v}_f = \overline{\mathbf{v}_f} - \overline{\mathbf{D}_f^{\mathbf{v}}}\left(\nabla p_f - \overline{\nabla p_f}\right) + (1-\lambda^{\mathbf{v}})\left(\mathbf{v}_f^{(n)} - \overline{\mathbf{v}_f^{(n)}}\right) \tag{15.196}$$

Not accounting for the effect of under-relaxation on the face velocity results in solutions that depend on the under relaxation factor.

15.9.2 Treatment of the Transient Term

When solving a transient problem with a backward Euler transient scheme the discretized momentum equation can be written as

$$a_C^{\mathbf{v}}\mathbf{v}_C = -\sum_{F\sim NB(C)}\left(a_F^{\mathbf{v}}\mathbf{v}_F\right) + \mathbf{b}_C^{\mathbf{v}} - V_C\nabla p_C + a_C^{\circ}\mathbf{v}_C^{\circ} \tag{15.197}$$

where $\mathbf{b}_C^{\mathbf{v}}$ is the source term of the momentum equation from which the pressure and transient source terms are extracted. The equivalent equation for the staggered grid variable arrangement has a similar form given by

$$a_f^{\mathbf{v}}\mathbf{v}_f = -\sum_{nb\sim NB(f)}\left(a_{nb}^{\mathbf{v}}\mathbf{v}_{nb}\right) + \mathbf{b}_f^{\mathbf{v}} - V_f\nabla p_f + a_f^{\circ}\mathbf{v}_f^{\circ} \tag{15.198}$$

Using the Rhie-Chow interpolation method, a pseudo cell-face equation will be constructed as

$$\overline{a_f^{\mathbf{v}}}\mathbf{v}_f = -\overline{\sum_{nb\sim NB(f)} a_{nb}^{\mathbf{v}}\mathbf{v}_{nb} + \mathbf{b}_f^{\mathbf{v}}} - \overline{V_f}\nabla p_f + \overline{a_f^{\circ}}\mathbf{v}_f^{\circ} \tag{15.199}$$

The average of the first term on the right hand side is obtained as

$$\begin{aligned}
-\overline{\sum_{nb\sim NB(f)} a_{nb}^{\mathbf{v}}\mathbf{v}_{nb} + \mathbf{b}_f^{\mathbf{v}}} &= -g_C\left(\sum_{F\sim NB(C)}\left(a_F^{\mathbf{v}}\mathbf{v}_F\right) + \mathbf{b}_C^{\mathbf{v}}\right) \\
&\quad - g_F\left(\sum_{N\sim NB(F)}\left(a_N^{\mathbf{v}}\mathbf{v}_N\right) + \mathbf{b}_F^{\mathbf{v}}\right) \\
&= g_C\left[a_C^{\mathbf{v}}\mathbf{v}_C + V_C\nabla p_C - a_C^{\circ}\mathbf{v}_C^{\circ}\right] \\
&\quad + g_F\left[a_F^{\mathbf{v}}\mathbf{v}_F + V_F\nabla p_F - a_F^{\circ}\mathbf{v}_F^{\circ}\right] \\
&= \overline{a_f^{\mathbf{v}}\mathbf{v}_f} + \overline{V_f\nabla p_f} - \overline{a_f^{\circ}\mathbf{v}_f^{\circ}}
\end{aligned} \tag{15.200}$$

Substituting into Eq. (15.84), the extended Rhie-Chow interpolated cell face velocity $\mathbf{v}_f$ is obtained

$$\mathbf{v}_f = \overline{\mathbf{v}_f} - \overline{\mathbf{D}_f^{\mathbf{v}}}\left(\nabla p_f - \overline{\nabla p_f}\right) + \frac{\overline{a_f^{\circ}\mathbf{D}_f^{\mathbf{v}}}}{V_f}\left(\mathbf{v}_f^{\circ} - \overline{\mathbf{v}_f^{\circ}}\right) \tag{15.201}$$

Not accounting for the effect of the unsteady term on the face velocity results in solutions that are time step dependent and have an oscillatory behavior for small time step. This correction is valid only for the first order Euler discretization. In case of more accurate time discretization schemes similar corrections can be performed following the same principles.

15.9.3 Treatment of the Body Force Term

When treating body forces in the staggered grid arrangement, the stencil of the body force term is exactly that of the pressure gradient term. In the case of a collocated grid arrangement, the body force, velocity, and momentum variables are calculated at the same location. Thus, in order to have a discretization of the body force that retains a similar stencil as the pressure, a redistribution of the body force term is needed. The discretized momentum equation is written as

$$a_C^{\mathbf{v}}\mathbf{v}_C = -\sum_{F\sim NB(C)} a_F^{\mathbf{v}}\mathbf{v}_F + \mathbf{b}_C^{\mathbf{v}} - V_C(\nabla p)_C + V_C\overline{\overline{\mathbf{B}_C^{\mathbf{v}}}} \tag{15.202}$$

where the double bar indicates two averaging steps. The first step is to compute $\overline{\mathbf{B}_C^{\mathbf{v}}}$ (Fig. 15.28) at the cell face as

$$\overline{\mathbf{B}_f^{\mathbf{v}}} = g_C\mathbf{B}_C^{\mathbf{v}} + (1 - g_C)\mathbf{B}_F^{\mathbf{v}} \tag{15.203}$$

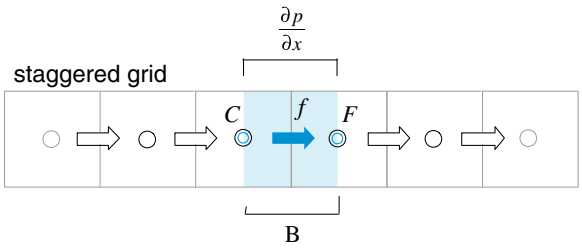

Fig. 15.28 Treatment of body force and pressure gradient on a staggered grid

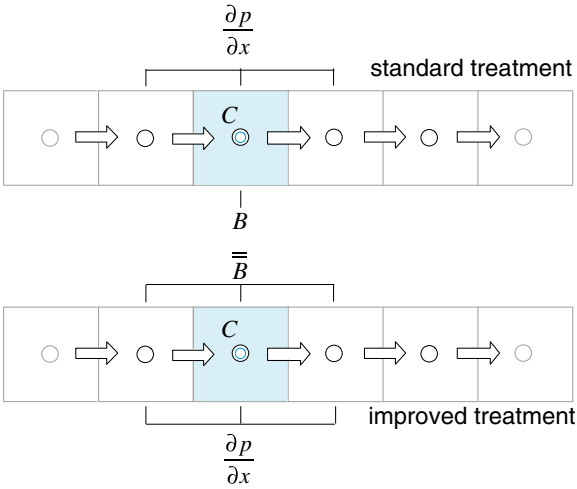

Fig. 15.29 Standard and improved Rhie-Chow treatment of body forces

while the second (Fig. 15.29) is to get an average of these face values at the cell centre.

The average values at cell center can best be derived [26] by considering the one dimensional situation depicted in Fig. 15.30.

For the case of a stationary fluid, the pressure gradient should be in equilibrium with the body forces leading to

$$0 = -\nabla p_f + \mathbf{B}_f^{\mathbf{v}} \tag{15.204}$$

Expanding the above equation, a relation between the pressures at C and F can be written as

$$p_C = p_F + B_f \delta y \tag{15.205}$$

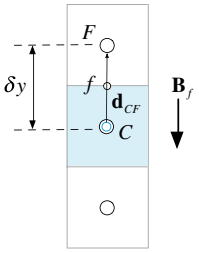

Fig. 15.30 One dimensional stationary fluid

or more generally as

$$p_C = p_F + \mathbf{B}_f \cdot \mathbf{d}_{CF} \tag{15.206}$$

where B_f is the magnitude of $\mathbf{B}_f$ given by

$$B_f = \rho_f g \tag{15.207}$$

For incompressible flows, the variation with temperature of the density appearing in the body force term is modeled using the Boussinesq approximation as given by Eq. (3.101).

Again for cell C the pressure gradient should be in equilibrium with the body forces, resulting in

$$0 = -\nabla p_C + \mathbf{B}_C^{\mathbf{v}} \Rightarrow \nabla p_C = \mathbf{B}_C^{\mathbf{v}} \tag{15.208}$$

However the pressure gradient for cell C is computed as

$$\begin{aligned} \nabla p_C &= \frac{\sum_f p_f \mathbf{S}_f}{V_C} \\ &= \frac{\sum_f (g_C p_C + (1 - g_C) p_F) \mathbf{S}_f}{V_C} \end{aligned} \tag{15.209}$$

substituting from Eq. (15.206) gives

$$\begin{aligned} \nabla p_C &= \frac{\sum_f \left(g_C \left(p_F + \overline{\mathbf{B}_f^{\mathbf{v}}} \cdot \mathbf{d}_{CF} \right) + (1 - g_C) p_F \right) \mathbf{S}_f}{V_C} \\ &= \frac{\sum_f p_F \mathbf{S}_f}{V_C} + \frac{\sum_f g_C \left(\overline{\mathbf{B}_f^{\mathbf{v}}} \cdot \mathbf{d}_{CF} \right) \mathbf{S}_f}{V_C} \\ &= p_F \underbrace{\frac{\sum_f \mathbf{S}_f}{V_C}}_{0} + \frac{\sum_f g_C \left(\overline{\mathbf{B}_f^{\mathbf{v}}} \cdot \mathbf{d}_f \right) \mathbf{S}_f}{V_C} \\ &= \frac{\sum_f g_C \left(\overline{\mathbf{B}_f^{\mathbf{v}}} \cdot \mathbf{d}_f \right) \mathbf{S}_f}{V_C} \\ &= \overline{\overline{\mathbf{B}_C^{\mathbf{v}}}} \end{aligned} \tag{15.210}$$

which implies that

$$\overline{\overline{\mathbf{B}_C^{\mathbf{v}}}} = \frac{\sum_f g_C \left(\overline{\mathbf{B}_f^{\mathbf{v}}} \cdot \mathbf{d}_f \right) \mathbf{S}_f}{V_C} \tag{15.211}$$

The second requirement is that the cell-face velocity be similar to that of the staggered arrangement equation:

$$\overline{a_f^{\mathbf{v}}}\mathbf{v}_f = \overline{-\sum_{nb\sim NB(f)} a_{nb}^{\mathbf{v}}\mathbf{v}_{nb} + \mathbf{b}_f^{\mathbf{v}}} - \overline{V_f}\nabla p_f + \overline{V_f\mathbf{B}_f^{\mathbf{v}}} \tag{15.212}$$

where $\mathbf{b}_C^{\mathbf{v}}$ is the source term given in Eq. (15.71) from which the pressure and body force terms are extracted. The averaging of the coefficients yields

$$\begin{aligned}
-\overline{\sum_{nb\sim NB(f)} a_{nb}^{\mathbf{v}}\mathbf{v}_{nb} + \mathbf{b}_f^{\mathbf{v}}} &= -g_C\left(\sum_{F\sim NB(C)}\left(a_F^{\mathbf{v}}\mathbf{v}_F\right) + \mathbf{b}_C^{\mathbf{v}}\right) \\
&\quad - g_F\left(\sum_{N\sim NB(F)}\left(a_N^{\mathbf{v}}\mathbf{v}_N\right) + \mathbf{b}_F^{\mathbf{v}}\right) \\
&= g_C\left[a_C^{\mathbf{v}}\mathbf{v}_C + V_C\nabla p_C - V_C\overline{\overline{\mathbf{B}_C^{\mathbf{v}}}}\right] \\
&\quad + g_F\left[a_F^{\mathbf{v}}\mathbf{v}_F + V_F\nabla p_F - V_F\overline{\overline{\mathbf{B}_F^{\mathbf{v}}}}\right] \\
&= \overline{a_f^{\mathbf{v}}\mathbf{v}_f} + \overline{V_f\nabla p_f} - \overline{V_f\overline{\overline{\mathbf{B}_f^{\mathbf{v}}}}}
\end{aligned} \tag{15.213}$$

and substituting into Eq. (15.179), the extended Rhie-Chow interpolated cell face velocity $\mathbf{v}_f$ is obtained as

$$\mathbf{v}_f = \overline{\mathbf{v}_f} - \overline{\mathbf{D}_f^{\mathbf{v}}}\left(\nabla p_f - \overline{\nabla p_f}\right) + \overline{\mathbf{D}_f^{\mathbf{v}}}\left(\overline{\mathbf{B}_f^{\mathbf{v}}} - \overline{\overline{\mathbf{B}_f^{\mathbf{v}}}}\right) \tag{15.214}$$

where $\overline{\overline{\mathbf{B}_f^{\mathbf{v}}}}$ is calculated as

$$\overline{\overline{\mathbf{B}_f^{\mathbf{v}}}} = g_C\overline{\mathbf{B}_C^{\mathbf{v}}} + (1 - g_C)\overline{\mathbf{B}_F^{\mathbf{v}}} \tag{15.215}$$

The above additional treatment of the cell face velocity increases the overall robustness of the solution procedure for situations where variations in body forces are important (e.g., free-surface flows).

15.9.4 Combined Treatment of Under-Relaxation, Transient, and Body Force Terms

In general all three terms described above should be dealt with together. This necessitates modifying the Rhie-Chow interpolation to account for all three effects. Fortunately this can easily be derived by using the principle of superposition leading to the following interface velocity:

$$\begin{aligned}\mathbf{v}_f = \overline{\mathbf{v}_f} - \overline{\mathbf{D}_f^{\mathbf{v}}}\left(\nabla p_f - \overline{\nabla p_f}\right) + \overline{\mathbf{D}_f^{\mathbf{v}}}\left(\overline{\mathbf{B}_f^{\mathbf{v}}} - \overline{\overline{\mathbf{B}_f^{\mathbf{v}}}}\right) \\ + \frac{\overline{a_f^{\circ}\mathbf{D}_f^{\mathbf{v}}}}{V_f}\left(\mathbf{v}_f^{\circ} - \overline{\mathbf{v}_f^{\circ}}\right) + (1-\lambda^{\mathbf{v}})\left(\mathbf{v}_f^{(n)} - \overline{\mathbf{v}_f^{(n)}}\right)\end{aligned} \tag{15.216}$$

where in calculating $\overline{\mathbf{D}_f^{\mathbf{v}}}$ the under relaxed value of the $a_C^{\mathbf{v}}$ coefficient is used.

15.10 Computational Pointers

15.10.1 uFVM

In uFVM, the pressure correction equation is implemented in one script file denoted by cfdAssembleMdotTerm. Listing 15.1 shows the core of the algorithm whereby the coefficients of the pressure correction equation are assembled by linearizing the fluxes at each of the interior faces. In addition, the mdot field (i.e., the mass flow rate at the faces) is calculated based on Eq. (15.216), which is subdivided into 9 terms (i.e., terms I trough IX) and assembled step by step to produce the cell face velocity. The terms into which the velocity at the face is decomposed are as follows:

term I: the interpolated velocity field $\overline{\mathbf{v}_f}$,
terms II and III: the face and average pressure gradients $-\overline{\mathbf{D}_f^{\mathbf{v}}}\left(\nabla p_f - \overline{\nabla p_f}\right)$,
terms IV and V: the average and redistributed body forces $\overline{\mathbf{D}_f^{\mathbf{v}}}\left(\overline{\mathbf{B}_f^{\mathbf{v}}} - \overline{\overline{\mathbf{B}_f^{\mathbf{v}}}}\right)$,
terms VI and VII: the transient fluxes $\frac{\overline{a_f^{\circ}\mathbf{D}_f^{\mathbf{v}}}}{V_f}\left(\mathbf{v}_f^{\circ} - \overline{\mathbf{v}_f^{\circ}}\right)$, and
terms VIII and IX: the relaxation correction term $(1-\lambda^{V})(\mathbf{v}_f^{(n)} - \overline{\mathbf{v}_f^{(n)}})$.

```
    %
    % assemble term I
    %     density_f [v]_f.Sf
    %
    U_bar_f = (dot(vel_bar_f(:,:)',Sf(:,:)'))';
    local_FLUXVf =local_FLUXVf+ density_f.*U_bar_f;
    %
    % Assemble term II and linearize it
    %      - density_f ([DPVOL]_f.P_grad_f).Sf
    %
    DUSf = [DU1_f.*Sf(:,1),DU2_f.*Sf(:,2),DU3_f.*Sf(:,3)];
    geoDiff =( dot(Sf(:,:)',DUSf') ./ dot(CN(:,:)',Sf(:,:)'))';
    local_FLUXCf1 = local_FLUXCf1 + density_f.*geoDiff;
    local_FLUXCf2 = local_FLUXCf2 - density_f.*geoDiff;
    local_FLUXVf  = local_FLUXVf  - density_f.*dot(p_grad_f',T')';
    %
    %  assemble term III
    %    density_f ([P_grad]_f.([DPVOL]_f.Sf))
    %
    local_FLUXVf = local_FLUXVf +
density_f.*dot(p_grad_bar_f(iFaces,:)',DUSf(iFaces,:)')';
    %
    % assemble terms IV and V
    %    density_f [DBVOL]_f.([B]_f -[[B]]_f).S_f
    %
    local_FLUXVf = local_FLUXVf +
density_f[iFace]*FVVectorDotProduct(FVTensorVectorDotProduct(DB_f,FVVe
ctorSubstract(FVMakeVector(bf1_bar_f[iFace],bf2_bar_f[iFace]),FVMakeVe
ctor(bf1_redistributed_f[iFace],bf2_redistributed_f[iFace]))),S_f) ;
    %
    % assemble terms VI and VII
    %    [Dt]_f (U_Old_f -[v_old]_f.S_f)
    %
    U_bar_old_f = [velx_old_bar_f[iFace] vely_old_bar_f[iFace])] *
S_f'
    local_FLUXVf = local_FLUXVf + DT_f*(mdot_old_f[iFace] -
density_old_f[iFace]*U_bar_old_f);
    %
    % assemble terms VIII and IX
    %     (1-URF)(U_f -[v]_f.S_f)
    %
    local_FLUXVf = local_FLUXVf + (1.0-Mdot_URF)*(mdot_previous_f -
density_f.*U_bar_f);
    %
    % assemble the flow term dot for the face
    %
    local_mdot_f = local_FLUXCf_1*(pressure[iElement1]+ Pref) +
local_FLUXCf_2*(pressure[iElement2]+ Pref) + local_FLUXVf;
    %


    %
    % Assemble in Global Fluxes
    %
```

Listing 15.1 Script used for the calculation of the mass flow rates and coefficients of the pressure correction equation in uFVM

```
    theFluxes.FLUXC1f(iFaces,1) = local_FLUXCf1;
    theFluxes.FLUXC2f(iFaces,1) = local_FLUXCf2;
    theFluxes.FLUXVf(iFaces,1)  = local_FLUXVf;
    %
    theFluxes.FLUXTf(iFaces,1)  = theFluxes.FLUXC1f.*pressureC(iFaces)
+ theFluxes.FLUXC2f.*pressureN(iFaces) + theFluxes.FLUXVf(iFaces);
    %
    mdot_f = theFluxes.FLUXTf(iFaces);
```

Listing 15.1 (continued)

15.10.2 OpenFOAM®

The numerical techniques introduced so far are used in what follows to develop OpenFOAM® [27] applications for solving the incompressible Navier-Stokes equations.

15.10.2.1 Pressure Correction SIMPLE Solvers

Based on the SIMPLE algorithm, a number of solvers will be constructed. The base solver, simpleFoam, will be presented first. This is followed by a number of versions, with each one adding more capabilities to the base code. These solvers can be summarized as follows:

1. simpleFoam (not the OpenFOAM® built-in solver) is the base code that incorporates the SIMPLE Algorithm in its most basic form.
2. simpleFoamImproved extends the base code to allow for improved treatment of relaxation.
3. simpleFoamTransient adds transient capabilities to the steady-state simpleFoam.
4. simpleFoamBuoyancy adds to the code the body force treatment.

More versions will be covered in the chapters to follow, each one with extended capabilities, added by modifying the base code described in this chapter. A list of the versions that will be covered in the next chapters is given below.

5. simpleFoamCompressible is the compressible version of simpleFoam (Chap. 16)
6. simpleFoamTurbulent includes capabilities for treating turbulent flows (Chap. 17).

simpleFoam

Before reviewing the simpleFoam code, some basic notational issues are addressed. The first step is to define, as shown in Listing 15.2, the geometric fields and parameters that will be initialized and used in the code.

```
#include "fvCFD.H"
#include "orthogonalSnGrad.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

int main(int argc, char *argv[])
{

#   include "setRootCase.H"
#   include "createTime.H"
#   include "createMesh.H"
```

Listing 15.2 The #include macro derivatives used to define the types of objects needed

In Listing 15.2, the #include macro directives outside the *main* function are needed to define the types of objects that are then declared and used in the application. The #include "*fvCFD.H*" contains a list of definitions for classes that are in general necessary to build any application in OpenFOAM®. In the developed application an additional header, not present in the *fvCFD.H* header, necessary for the SIMPLE solver implementation will be added.

The use of the #include statements inside the *main* function is a compacting procedure, with each declared statement representing a piece of the code moved to the corresponding file name. For example, the statement #include "createMesh.H" just represents the code shown in Listing 15.3, which is necessary to instantiate the mesh class.

```
createMesh.H
~~~~~~~~~~~~
Foam::Info
    << "Create mesh for time = "
    << runTime.timeName() << Foam::nl << Foam::endl;


Foam::fvMesh mesh
(
    Foam::IOobject
    (
        Foam::fvMesh::defaultRegion,
        runTime.timeName(),
        runTime,
        Foam::IOobject::MUST_READ
    )
);
```

Listing 15.3 The code representing the #include createMesh.H file necessary to instantiate the mesh class

Once the necessary initialization has been performed, the next step is the definition of the proper fields or variables needed by the solver. These are defined in file "*createFields.H*". The first defined field, shown in Listing 15.4, is the pressure field (p).

```
Info << "Reading field p\n" << endl;
volScalarField p
(
    IOobject
    (
        "p",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);
```

Listing 15.4 Script used to define the pressure field

Since the solution is obtained by solving a pressure correction equation instead of a pressure equation, a pressure correction field (pp) is also defined (Listing 15.5).

```
const volScalarField::GeometricBoundaryField& pbf=p.boundaryField();
wordList pbt = pbf.types();
volScalarField pp
(
    IOobject
    (
        "pp",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("zero", p.dimensions(), 0.0),
    pbt
);
```

Listing 15.5 Script used to define the pressure correction field

It is worth noting that in Listing 15.5 a different constructor is used to define the pressure correction field. Since the pressure corrector represents the pressure itself, the same boundary conditions defined for the real pressure can be used for the pressure correction field without the need to define the same quantity twice.

The list of pressure boundary types displayed in Listing 15.6 are now copied under the pbt variable in Listing 15.5 and directly used in the constructor of pp.

```
// Set pp boundary values
forAll(pp.boundaryField(), patchi)
{
      if (isType<fixedValueFvPatchScalarField>(pp.boundaryField()
[patchi]))
    {
       fixedValueFvPatchScalarField& ppbound =

refCast<fixedValueFvPatchScalarField>(pp.boundaryField()[patchi]);

       ppbound == scalarField(ppbound.size(),0.0);
    }
}
```

Listing 15.6 Script showing the declaration of the different pressure boundary types

The corrector should reset to zero the correction field at every iteration and should also apply a zero value at all boundaries for which a Dirichlet boundary condition is used for the pressure.

The velocity and corresponding mass flux fields must also to be defined. As depicted in Listing 15.7, the velocity field is defined through an input file while the mass flux field can be defined as a derived quantity.

```
Info << "Reading field U\n" << endl;
volVectorField U
(
    IOobject
    (
        "U",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);
surfaceScalarField mDot
(
    IOobject
    (
        "mDot",
        runTime.timeName(),
        mesh,
        IOobject::READ_IF_PRESENT,
        IOobject::AUTO_WRITE
    ),
    linearInterpolate(U) & mesh.Sf()
);
```

Listing 15.7 Script used to define the velocity and mass flux fields

Finally the fluid thermo-physical properties should also be defined. For incompressible laminar flows this involves simply assigning a value to the kinematic viscosity nu, as shown in Listing 15.8.

```
Info<< "Reading transportProperties\n" << endl;
IOdictionary transportProperties
(
    IOobject
    (
        "transportProperties",
        runTime.constant(),
        mesh,
        IOobject::MUST_READ_IF_MODIFIED,
        IOobject::NO_WRITE
    )
);
dimensionedScalar nu
(
    transportProperties.lookup("nu")
);
```

Listing 15.8 Code used to define the fluid thermo-physical properties

After defining all variables the implementation of the SIMPLE algorithm can proceed. A *while* loop can be used for the cases when the stopping criterion is the number of SIMPLE iterations. For each single loop, the momentum and pressure correction equations are solved and updates of the variables are performed. Starting with the momentum equation written as (the form solved in OpenFOAM®),

$$\nabla \cdot \{\mathbf{vv}\} = \nu \nabla^2 \mathbf{v} - \nabla p \tag{15.217}$$

where the kinematic viscosity ν is defined as

$$\nu = \frac{\mu}{\rho} \tag{15.218}$$

and p represents the pressure divided by the density, i.e.,

$$p = \frac{static\,pressure}{\rho} \tag{15.219}$$

its solution is translated into the script shown in Listing 15.9.

```
// Solve the Momentum equation
fvVectorMatrix UEqn
(
        fvm::div(mDot, U)
      - fvm::laplacian(nu, U)
);
UEqn.relax();
solve
(
    UEqn == -fvc::grad(p)
);
```

Listing 15.9 Script to solve the momentum equation

The first instruction in Listing 15.9 defines the finite volume discretization of the momentum equation in vector form with its storage matrix (the three components of the velocity vector are solved in a segregated manner despite its vectorial implementation). The system is then implicitly relaxed and solved via an iterative solver.

Once the momentum equation is solved, a new guess for the velocity field is obtained. This velocity does not satisfy the continuity equation in general and the assembly of the continuity equation in the form of a pressure correction equation is now required to correct the flow field. The pressure correction equation is written as

$$-\nabla \cdot (\bar{\mathbf{D}}_f \nabla p') = -\nabla \cdot (\mathbf{v}) \tag{15.220}$$

where a component of $\bar{\mathbf{D}}$ at an element centroid is computed as

$$D = \frac{V}{a_c^{\mathbf{v}}} \tag{15.221}$$

with values at the faces obtained by interpolation. Its implementation is translated into the following syntax (Listing 15.10):

```
pp = scalar(0.0)*pp;
pp.correctBoundaryConditions();

fvScalarMatrix ppEqn
(
    - fvm::laplacian(DUf, pp, "laplacian(pDiff,pp)")
    + fvc::div(mDot)
);
```

Listing 15.10 Script to implement the pressure correction equation

where div(mDot) is basically $\sum \dot{m}_f$, while pp represents p' and is reset to zero at each iteration. Moreover, the DUf variable is the value of D at the cell face obtained, as shown in Listing 15.11, by linear interpolation using the values at the nodes straddling the face. Further, .A() represents the diagonal terms in the momentum matrix divided by the volume.

```
volScalarField DU = 1.0/UEqn.A();
surfaceScalarField DUf("DUf",linearInterpolate(DU));
```

Listing 15.11 Script to calculate the values of DUf

Listing 15.12 computes the mass flow rate at cell faces (*mDot*) using the Rhie-Chow interpolation where the calculation of ∇p_f and $\overline{\nabla p_f}$ is clearly shown.

```
const surfaceVectorField ed = mesh.delta()()/mag(mesh.delta()());
Foam::fv::orthogonalSnGrad<scalar> faceGradient(mesh);
surfaceVectorField gradp_avg_f = linearInterpolate(fvc::grad(p));
surfaceVectorField gradp_f = gradp_avg_f - (gradp_avg_f & ed)*ed +
(faceGradient.snGrad(p))*ed;

surfaceVectorField U_avg_f = linearInterpolate(U);

// Rhie-Chow interplation
mDot = (U_avg_f & mesh.Sf()) - ( (DUf*( gradp_f - gradp_avg_f)) &
mesh.Sf() );
```

Listing 15.12 Script to compute the mass flow rate at cell faces using the Rhie-Chow interpolation

The pressure correction equation is now fully set and is solved by executing the statement in Listing 15.13.

```
ppEqn.solve();
```

Listing 15.13 Statement used to solve the pressure correction equation

Once the pressure correction equation is solved, the velocity, pressure, and mass flow rate fields are updated using the obtained pressure correction field. Starting with the mass flow rate field (i.e., the *mDot* flux), it is updated by executing the below statement (Listing 15.14).

```
mDot += ppEqn.flux();
```

Listing 15.14 Statement used to update the mass flow rate field to satisfy continuity

The *flux*() function in Listing 15.14 provides the matrix multiplication, the extra diagonal matrix coefficients, and the corresponding solution. Recalling the finite volume discretization of the diffusion term, each extra diagonal of coefficients represents a face of the mesh. Thus the update of the fluxes can be performed in a more consistent way using directly the matrix coefficients and cell values. A simplified version of the *flux*() function is shown in Listing 15.15.

```
for (label face=0; face<lowerAddr.size(); face++)
{
    mDotPrime[face] =
        upperCoeffs[face]*pp[upperAddr[face]]
      - lowerCoeffs[face]*pp[lowerAddr[face]];
}

return mDotPrime;
```

Listing 15.15 A simplified version of the *flux()* function where the flux correction *mDotPrime* is computed

In Listing 15.15 the correction flux *mDotPrime* (Eq. 15.101) is basically evaluated by performing a loop over the faces using the upper and lower coefficients of the matrix and multiplying these coefficients with the corresponding cell values.

Finally the velocity and pressure at cell centroids are updated using the script shown in Listing 15.16,

```
scalar URF = mesh.equationRelaxationFactor("pp");
p += URF*pp;
p.correctBoundaryConditions();
U -= fvc::grad(pp)*DU;
U.correctBoundaryConditions();
```

Listing 15.16 Update of the velocity and pressure fields at cell centroids

where the variable URF is the explicit relaxation factor for pressure update λ^p, necessary for a stable SIMPLE solver.

simpleFoamImproved

In simpleFoamImproved the Rhie-Chow interpolation is extended to account for the relaxation of the velocity field. This is translated into the syntax presented below in Listing 15.17 that expands the generic Rhie-Chow interpolation to include the additional term in Eq. (15.196).

```
// Rhie-Chow interplation
mdotf =   (U_avg_f & mesh.Sf()) - ( (DUf*( gradp_f - gradp_avg_f)) &
mesh.Sf() )
        +(scalar(1) - URFU)*(mdotf.prevIter() - (U_avg_prevIter_f &
mesh.Sf()));
```

Listing 15.17 Improved Rhie-Chow interpolation accounting for under relaxation

Thus the fields mdotf.prevIter() and U.prevIter() need to be defined.

simpleFoamTransient

In simpleFoamTransient the Rhie-Chow interpolation is extended to account for the transient term. Thus the expression for the mass flow rate becomes (Listing 15.18).

```
// Rhie-Chow interplation
mdotf =   (U_avg_f & mesh.Sf()) - ( (DUf*( gradp_f - gradp_avg_f)) &
mesh.Sf() )
        +(scalar(1) - URFU)*(mdotf.prevIter() - (U_avg_prevIter_f &
mesh.Sf()))
        + DTf*( mdotf_old - (U_old_f& mesh.Sf()));
```

Listing 15.18 Rhie-Chow interpolation accounting for the effects of under-relaxation and the unsteady term

Furthermore the main loop is modified to add a transient loop, with the main code becoming as shown in Listing 15.19.

```
        pimpleControl pimple(mesh);
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

    Info<< "\nStarting time loop\n" << endl;
    while (runTime.run())
    {
        #include "readTimeControls.H"
        #include "CourantNo.H"
        #include "setDeltaT.H"
        runTime++;

        Info<< "Time = " << runTime.timeName() << nl << endl;

        scalar iter=0;
        while (pimple.loop())
        {
            iter++;
            Info<< "Iteration = " << iter << nl << endl;
            //
            U.storePrevIter();
            mdotf.storePrevIter();

            // Pressure-velocity SIMPLE corrector

            #include "UEqn.H"
            #include "ppEqn.H"


        }
        runTime.write();

        Info<< "ExecutionTime = " << runTime.elapsedCpuTime() << " s"
        << "  ClockTime = " << runTime.elapsedClockTime() << " s"
        << nl << endl;
    }
```

Listing 15.19 The main loop used for solving unsteady flow problems

simpleFoamBuoyancy

The simpleFoamBuoyancy solver adds to simpleFoamTransient the following capabilities: (i) the solution of the energy equation, (ii) the inclusion of a body source term in the momentum equation, and (iii) an account of the effects of the body force term redistribution in the Rhie-Chow interpolation.

The codes used to introduce these modifications are shown in Listings (15.20), (15.21), and (15.22).

The code needed to solve the energy equation is given in Listing (15.20).

```
// Solve the Energy equation

  fvScalarMatrix TEqn
  (
          fvm::ddt(T)
        + fvm::div(phi, T)
        - fvm::laplacian(K, T)
  );

   TEqn.relax();
   TEqn.solve();
```

Listing 15.20 The script used to solve the energy equation

The source term in the momentum equation is implemented as (Listing 15.21),

```
surfaceVectorField B_f = linearInterpolate(- beta*(T-To)*g);
volVectorField B_reconstructed = fvc::average(B_f);

    solve
    (
        UEqn == -fvc::grad(p) + B_reconstructed
    );
```

Listing 15.21 Accounting for the body force term source in the momentum equation

while the calculation of the mass fluxes using the Rhie-Chow interpolation are as displayed in Listing (15.22).

```
surfaceVectorField B_reconstructed_f =
linearInterpolate(B_reconstructed);
// Rhie-Chow interplation
phi = (U_avg_f & mesh.Sf())
     - DUf*( (gradp_f*mesh.magSf())-(gradp_avg_f&mesh.Sf()))
        + (scalar(1) - URFU)*(phi.prevIter() - (U_avg_prevIter_f &
mesh.Sf()))
+ DTf*(phi_old - (U_old_f& mesh.Sf()))
+ DUf* ( (B_f& mesh.Sf()) - (B_reconstructed_f& mesh.Sf()) )  ;
```

Listing 15.22 The modified Rhie-Chow interpolation accounting for body forces

15.11 Closure

This chapter presented the segregated pressure-based approach for solving incompressible flow problems on collocated grids. It also demonstrated that the success of the Rhie-Chow interpolation on collocated grids is due to its formation of a pseudo-momentum equation at the cell face that has a tight pressure gradient stencil similar to the one resulting from a staggered grid formulation. In addition, the details of implementing the most commonly encountered boundary conditions in the momentum and pressure-correction equations were discussed. The next chapter will extend the pressure based method to predict compressible fluid flow at all speeds.

15.12 Exercises

Exercise 1
A portion of a water-supply system is shown in Fig. 15.31. The flow rate $\dot{m}$ in a pipe section is given by

$$\dot{m} = C\Delta p$$

where Δp is the pressure drop over the length of the pipe section, and C is the hydraulic conductance. The following data is known:

$$\begin{aligned} p_1 &= 400,\ p_2 = 350 \\ \dot{m}_F &= 25 \\ C_A &= 0.4, C_B = 0.2, C_C = 0.1, C_D = 0.3, C_E = 0.2 \end{aligned}$$

Find p_3, p_4, p_5, $\dot{m}_A, \dot{m}_B, \dot{m}_C, \dot{m}_D$ and $\dot{m}_E$ using the following procedure

- Start with a guess for p_3, p_4, and p_5.
- Compute $\dot{m}^*$ values based on the guessed pressures.
- Construct the pressure-correction equations and solve for p'_3, p'_4 and p'_5.
- Update the pressures and the $\dot{m}^*$ values

Do you need to iterate? Why?

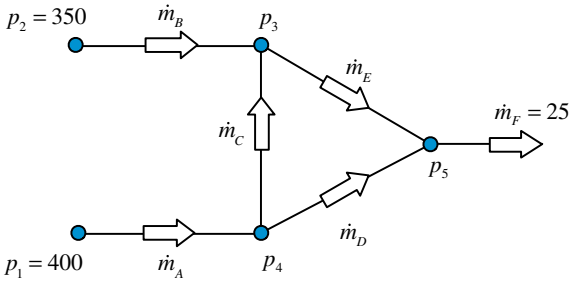

Fig. 15.31 A portion of a water-supply system

Exercise 2

A one dimensional flow through a porous material is governed by

$$c|u|u + \frac{dp}{dx} = 0$$

where c is a constant. The continuity equation is

$$\frac{du}{dx} = 0$$

$$x_2 - x_1 = 1 \quad x_3 - x_2 = 2$$
$$S_A = 3 \qquad S_B = 2$$

Use the SIMPLE procedure for the grid shown in Fig. 15.32 to compute p_2, u_A, and u_B from the following data:

$$c_A = 0.3 \quad c_B = 0.15$$
$$p_1 = 150 \quad p_3 = 18$$

with the size and area at the center of each control volume given by

$$\Delta x_A = 1; A_A = 3$$
$$\Delta x_B = 3; A_B = 2$$

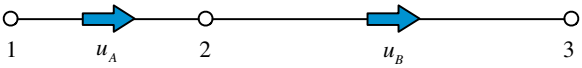

Fig. 15.32 One dimensional flow in a porous material

Exercise 3

In the Steady, one dimensional, constant density situation shown in Fig. 15.33, the velocity u is calculated at locations B and C, while the pressure is calculated at locations 1 and 2. The velocity correction formulae are written as

$$u_B = D_B(p_1 - p_2) \quad \text{and} \quad u_C = D_C(p_2 - p_3)$$

where the values of D_B and D_C are 3 and 4, respectively. The boundary conditions are $u_A = 5$ and $p_3 = 70$.

(a) If at a given stage in the iteration process, the momentum equations give $u_B^* = 4$ and $u_C^* = 6$, calculate the values of p_1 and p_2.
(b) Explain how you could obtain the values of p_1 and p_3 if the right hand boundary condition is given as $u_C = 5$ instead of $p_3 = 70$.

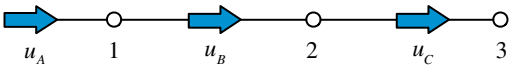

Fig. 15.33 Incompressible flow in a one dimensional domain

Exercise 4

Consider the main control volume shown in Fig. 15.34. A staggered mesh is used with the u and v velocity components stored as shown. The following quantities are given: $u_w = 7$, $v_s = 3$, $p_N = 0$ and $p_E = 50$. The flow is steady and the density is constant. The momentum equations for u_e and v_n are given by:

$$u_e = -D_e(p_E - p_C)$$
$$v_n = -D_n(p_N - p_C)$$

Also given $D_e = 2$, $D_n = 1.6$, and the control volume has $\Delta x = \Delta y = 1$.

(a) Starting with a guessed value of $p_C^{(n)} = 50$, use the SIMPLE algorithm to find u_e and v_n.
(b) Is an iteration loop needed for this problem? Explain.

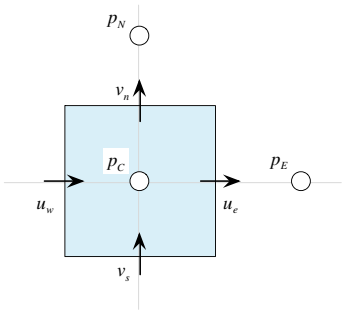

Fig. 15.34 A main control volume in a two-dimensional staggered grid arrangement

Exercise 5

Consider the simplified one-dimensional Forchheimer model for flow in porous media given by

$$bu^2 = -k\frac{dp}{dx}$$

with the continuity equation given by

$$\frac{d(\varepsilon u)}{dx} = 0$$

In the above equations b is a constant and ε is the porosity coefficient that accounts for the effective porous area.

Devise a SIMPLE-like procedure to compute p_C, u_e, and u_w for the following data:

$$\begin{aligned} \Delta x &= 0.1; \quad \Delta y = 1 \\ b_W &= 5; \quad b_C = 4; b_E = 3 \\ \varepsilon_e &= 0.9; \quad \varepsilon_w = 0.6 \\ p_W &= 40; \quad p_E = -200 \end{aligned}$$

Start with the following initial values for velocity and pressure (Fig. 15.35):

$$u_e^* = u_w^* = 3 \text{ and } p_C = -100.$$

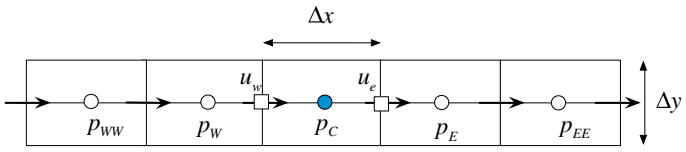

Fig. 15.35 One-dimensional Forchheimer model for flow in porous media

Exercise 6

Compute the interface velocities u_e and u_w using the Rhie-Chow interpolation and compare it to the averaged values $\bar{u}_e$ and $\bar{u}_w$ knowing the following data (Fig. 15.36):

$$\begin{aligned} &p_{WW} = 10; p_W = 12; p_C = 16; p_E = 24; p_{EE} = 40, \text{and} \\ &u_W = 5; u_C = 10; u_E = 40. \end{aligned}$$

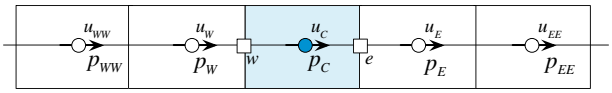

Fig. 15.36 A one dimensional collocated grid

Exercise 7
In OpenFOAM® develop a SIMPLEC pressure correction algorithm by modifying the SIMPLE algorithm described in this chapter. Hint: in order to find the summation of the extra diagonal coefficients use the H1() function of the fvMatrix.

Exercise 8
Check the pisoFoam solver located in $FOAM_SRC/../applications/solvers/incompressible/pisoFoam/pisoFoam.C and compare it with the algorithm described in this chapter, i.e., "The Collocated PISO Algorithm". Find out the inconsistency with the standard OpenFOAM® implementation.

Exercise 9
Develop a pressure correction PISO algorithm for OpenFOAM®.

References

1. Patankar SV (1981) A calculation procedure for two dimensional elliptic situations. Numer Heat Transfer 4(4):409–425
2. Patankar SV (1980) Numerical heat transfer and fluid flow. Hemisphere, NY
3. Patankar SV, Spalding DB (1972) A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. Int J Heat Mass Transf 15(10):1787–1806
4. Harlow FH, Welch JE (1965) Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. Phys Fluids 8(12):2182–2189
5. Van Doormaal JP, Raithby GD (1985) An evaluation of the segregated approach for predicting incompressible fluid flows. ASME Paper 85-HT-9, Presented at the national heat transfer conference, Denver, Colorado
6. Raithby GD, Schneider GE (1979) Numerical solution of problems in incompressible fluid flow: treatment of the velocity-pressure coupling. Numer Heat Transfer, Part A 2(4):417–440
7. Patankar SV (1975) Numerical prediction of three-dimensional flow. In Launder BE (ed) studies in convection: theory, measurement, and application, vol 1. Academic, New York, pp 1–9
8. Rhie CM, Chow WL (1983) Numerical study of the turbulent flow past an airfoil with trailing edge separation. AIAA J 21:1525–1532
9. Rhie CM (1988) A three-dimensional passage flow analysis method aimed at centrifugal impellers. Comput Fluids 13:443–460
10. Majumdar S (1988) Role of under relaxation in momentum interpolation for calculation of flow with nonstaggered grids. Numer Heat Transfer 13:125–132
11. Miller TF, Schmidt FW (1988) Use of a pressure-weighted interpolation method for the solution of incompressible Navier-Stokes equations on a nonstaggered grid system. Numer Heat Transfer 14:213–233
12. Karki KC, Patankar SV (1988) Calculation procedure for viscous incompressible flows in complex geometries. Numer Heat Transfer 14:295–307
13. Choi SK, Nam HY, Cho M (1993) Use of the momentum interpolation method for numerical solution of incompressible flows in complex geometries: choosing cell face velocities. Numer Heat Transfer, Part B 23:21–41
14. Choi SK, Nam HY, Lee YB, Cho M (1993) An efficient three-dimensional calculation procedure for incompressible flows in complex geometries. Numer Heat Transfer, Part B 23:387–400

15. Choi SK, Nam HY, Cho M (1994) Use of staggered and nonstaggered grid arrangements for incompressible flow calculations on nonorthogonal grids. Numer Heat Transfer, Part B 25 (2):193–204
16. Choi SK, Nam HY, Cho M (1994) Systematic comparison of finite-volume calculation methods with staggered and nonstaggered grid arrangements. Numer Heat Transfer, Part B 25 (2):205–221
17. Van Doormaal JP, Raithby GD (1984) Enhancement of the SIMPLE method for predicting incompressible fluid flows. Numer Heat Transfer 7:147–163
18. Issa RI (1982) Solution of the implicit discretized fluid flow equations by operator splitting. Mechanical Engineering Report, FS/82/15, Imperial College, London
19. Maliska CR, Raithby GD (1983) Calculating 3-D fluid flows using non-orthogonal grid. In: Proceedings of the third international conference on numerical methods in laminar and turbulent flows, Seattle, pp 656–666
20. Acharya S, Moukalled F (1989) Improvements to incompressible flow calculation on a non-staggered curvilinear grid. Numer Heat Transfer, Part B 15:131–152
21. Spalding DB (1980) Mathematical modelling of fluid mechanics, heat transfer and mass transfer processes. Mechanical Engineering Department Report HTS/80/1, Imperial College of Science, Technology and Medicine, London
22. Moukalled F, Darwish M (2000) A unified formulation of the segregated class of algorithms for fluid flow at all speeds. Numer Heat Transfer, Part B 37:103–139
23. Darwish M, Asmar D, Moukalled F (2004) A comparative assessment within a multigrid environment of segregated pressure-based algorithms for fluid flow at all speeds. Numer Heat Transfer, Part B 45(1):49–74
24. Jang DS, Jetli R, Acharya S (1986) Comparison of the PISO, SIMPLER and SIMPLEC algorithms for the treatment of the pressure-velocity coupling in steady flow problems. Numer Heat Transfer 10:209–228
25. Yen RH, Liu CH (1993) Enhancement of the SIMPLE algorithm by an additional explicit corrector step. Numer Heat Transfer, Part B 24:127–141
26. Mecinger J (2012) An alternative finite volume discretization of body force field on collocated grids. In: Petrova R (ed) Finite volume method-powerful means of engineering design. ISBN:978-953-51-0445-2
27. OpenFOAM, 2015 Version 2.3.x. http://www.openfoam.org