Chapter 14
Discretization of the Source Term, Relaxation, and Other Details

Abstract This chapter is devoted to a number of "small" numerical details that may have "big" effects on the solution behavior. First the treatment of the source term in the general case when it is solution dependent (i.e., when $Q^{\phi} = Q^{\phi}(\phi)$) is examined. The source is linearized in terms of the dependent variable and split into two parts, one treated explicitly and the second treated implicitly. This is followed by a discussion of explicit and implicit techniques for under-relaxing the algebraic equations. Several implicit under-relaxing approaches are presented, starting with the well known implicit under relaxation method of Patankar (Numerical heat transfer and fluid flow, 1980) [1], the E-factor method of van Doormaal and Raithby (Numerical Heat Transfer 7:147–163, 1984) [2], and the false transient approach of Mallinson and de Vahl Davis (Journal of Computational Physics 12 (4):435–461, 1973) [3]. Then the residual form of the discretized algebraic equation is introduced. The chapter ends with the presentation of convergence indicators used to evaluate the solution convergence status.

14.1 Source Term Discretization

Source terms (sink and source) appear in the governing equations of many flow and transport phenomena problems. Examples include the equations of turbulence models, chemical reactions, radiation heat transfer, mass transfer, and multiphase flows, to cite a few. These source terms affect not only the physics of the problem, but also the numerical stability of computations. However, if properly handled, source terms may yield improvement in robustness. A general recommendation is to treat negative values (sinks) implicitly, while positive values (sources) should be evaluated explicitly.

The treatment of source terms can be clarified by considering the discretized form of the general conservation equation for the variable ϕ over the element of centroid C and volume V_C with a source explicitly displayed (Fig. 14.1). This equation is given by

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_14

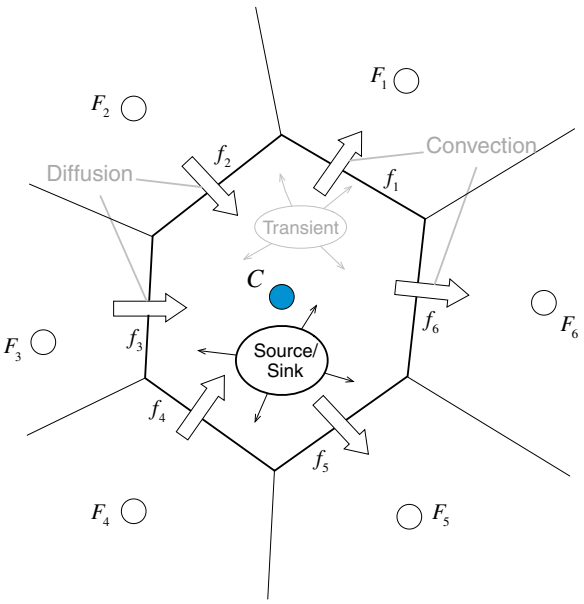

Fig. 14.1 An element ***C*** with a source term Q^{ϕ}

$$a_C\phi_C + \sum_{F\sim NB(C)} a_F\phi_F = Q_C^{\phi}V_C \tag{14.1}$$

where $Q_C^{\phi}V_C$ represents the source term integrated over the element C.

In general the source term is a function of the dependent variable ϕ with its functional relationship expressed as

$$Q_C^{\phi} = Q(\phi_C) \tag{14.2}$$

In this form the source can be explicitly calculated based on the available ϕ values, which in an iterative process represent values from the previous iteration. Whereas this approach is acceptable if the value of Q_C^{ϕ} is constant or relatively small, when the variation in Q_C^{ϕ} is large in comparison with other terms in the equation the rate of convergence can be negatively affected. In such situations, the rate of convergence may be improved by linearizing Q_C^{ϕ} using a Taylor-like series expansion. Denoting values at the previous iteration with a superscript *, the value of the source term Q_C^{ϕ} at the current iteration can be expressed as

$$\begin{aligned} Q(\phi_C) &= Q(\phi_C^*) + \left(\frac{\partial Q}{\partial \phi_C}\right)^* (\phi_C - \phi_C^*) \\ &= \underbrace{\left(\frac{\partial Q}{\partial \phi_C}\right)^* \phi_C}_{\textit{Implicit part}} + \underbrace{Q(\phi_C^*) - \left(\frac{\partial Q}{\partial \phi_C}\right)^* \phi_C^*}_{\textit{Explicit part calculated based on values from previous iteration}} \end{aligned} \tag{14.3}$$

In a control volume context, the right hand side of Eq. (14.1) can be written as

$$\begin{aligned} Q_C^\phi V_C &= \iint_{V_C} Q^\phi dV \\ &= \iint_{V_C} \left(\frac{\partial Q_C^*}{\partial \phi_C}\phi_C\right) dV + \iint_{V_C} \left(Q_C^* - \frac{\partial Q_C^*}{\partial \phi_C}\phi_C^*\right) dV \\ &= \left(\frac{\partial Q_C^*}{\partial \phi_C} V_C\right)\phi_C + \left(Q_C^* - \frac{\partial Q_C^*}{\partial \phi_C}\phi_C^*\right) V_C \\ &= FluxC_C\ \phi_C + FluxV_C \end{aligned} \tag{14.4}$$

Substituting back in Eq. (14.1), the algebraic equation becomes

$$[a_C - FluxC_C]\phi_C + \sum_{F \sim NB(C)} a_F \phi_F = FluxV_C \tag{14.5}$$

In this formulation the implicit part of the source, $FluxC_C$ [defined in Eq. (14.3)], is required to be negative to guarantee diagonal dominance or else the Scarborough criterion may not be fulfilled causing divergence. In addition, for the case when the variable ϕ is positive-definite, the explicit part, $FluxV_C$ [Eq. (14.3)], must be positive to ensure positive ϕ predictions.

Example 1

In problems involving heat transfer by radiation, the source term in the energy equation takes the form

$$Q^T = A\left(T_\infty^4 - T^4\right)$$

where A is a constant, T is the temperature at any grid point, and T_∞ represents the non-varying ambient temperature. Integrate this source term over an element of centroid C and volume V_C, then linearize it using different alternatives and explain their consequences on convergence.

Solution

$$\int_{V_C} Q^T dV = Q_C^T V_C = FluxV_C + FluxC_C T_C$$

Several arbitrary choices can be selected to linearize Q^T.

Option 1

$$FluxC_C = 0$$
$$FluxV_C = A\left(T_\infty^4 - T_C^4\right)V_C$$

Adopting this alternative may cause the solution to diverge as it results in negative values for $FluxV_C$, whenever $T_C > T_\infty$, leading to possible unphysical negative absolute temperature values during the iterative process.

Option 2
Linearizing with respect to the value of temperature of the previous iteration T_C^*, the expanded form of the source term is obtained as

$$Q_C^T = \left(Q_C^T\right)^* + \left(\frac{dQ_C^T}{dT_C}\right)^* \left(T_C - T_C^*\right)$$
$$= A\left(T_\infty^{*4} - T_C^{*4}\right) - 4AT_C^{*3}\left(T_C - T_C^*\right)$$

Comparing with Eq. (14.9), $FluxC_C$ and $FluxV_C$ are found to be

$$FluxC_C = -4AT_C^{*3}V_C$$
$$FluxV_C = A\left(T_\infty^{*4} + 3T_C^{*4}\right)V_C$$

This is the ideal approach resulting in a positive $FluxV_C$ and a negative $FluxC_C$ and giving the best rate of convergence as the introduced implicitness in the solution is the optimum one.

14.2 Under-Relaxation of the Algebraic Equations

As described in previous chapters, the end product of the discretization process is a set of algebraic equations of the form given by Eq. (14.1), in which a_F refers to a neighboring coefficient (Fig. 14.2) representing the effect of the neighboring variable ϕ_F on the cell variable ϕ_C, b_C is the right hand side of the equation that usually includes the source terms and the effects of other variables, while a_C is the main

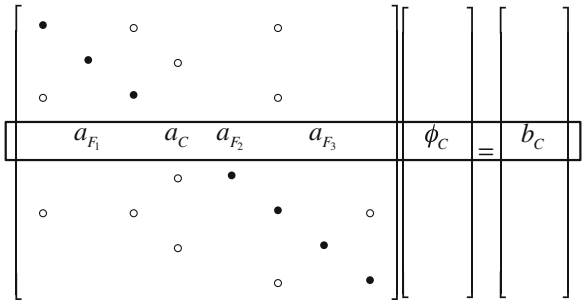

Fig. 14.2 Matrix representation of Eq. (14.1)

coefficient of the algebraic equation and contains the effects of various influences, including the spatial discretization effects, the transient effects, etc. The set of equations represented by Eq. (14.1) is usually diagonally dominant.

In the iterative solution of the system of algebraic equations it is often desirable to slow down the changes in the values of the dependent variable from iteration to iteration. This is needed to improve the convergence of non-linear problems but also to avoid divergence when starting with a guessed initial field that could be far from the solution. The non-linearities can arise because of the non-orthogonality of the grid system, the presence of source terms, the non-linear nature of the modeled equations, etc. One method commonly used to promote convergence by "slowing down" ("relaxing") the (sometimes excessive) changes made to the values of the variable during solution is the relaxation method. The standard relaxation method used in many CFD codes is the implicit under relaxation method of Patankar [1], which was briefly presented in Chap. 8. Other under-relaxation methods have been presented in the literature such as the E-Factor [2] relaxation method and the False transient method [3]. Van Doormaal and Raithby [2] have shown that these different relaxation methods are somewhat related and that the under-relaxation in any of the methods can be related to the under-relaxation in the other methods, as they all basically retard the effect of neighboring elements and sources on the under-relaxed element value. In other words, under-relaxation affects equally the source term in the concerned element and its spatial coefficients. Some of these relaxation methods are presented next.

14.2.1 Under-Relaxation Methods

Solution relaxation may be performed either explicitly after the solution at any iteration is obtained or implicitly by incorporating its effect into the equation before the solution is obtained. Both methods are outlined below.

14.2.2 Explicit Under-Relaxation

In the explicit under-relaxation method, at the end of every iteration after a new solution is obtained, all cells in the computational domain are visited and the predicted value $\left(\phi_C^{new,\,predicted}\right)$ in any cell C is modified according to

$$\phi_C^{new,\,used} = \phi_C^{old} + \lambda^{\phi}\left(\phi_C^{new,\,predicted} - \phi_C^{old}\right) \tag{14.6}$$

where λ^{ϕ} is the relaxation factor, which for both explicit and implicit relaxation can be interpreted according to its assigned value as follows:

1. A value of $\lambda^{\phi} < 1$ results in under-relaxation. This may slow down the speed of convergence but increases the stability of the calculation, i.e., it decreases the possibility of divergence or oscillations in the solution.
2. A value of $\lambda^{\phi} = 1$ corresponds to no relaxation. If applied, then the predicted values during an iteration are the ones used at the next iteration.
3. A value of $\lambda^{\phi} > 1$ leads to over-relaxation. It can sometimes be used to accelerate convergence but usually decreases the stability of the calculations.

Explicit under-relaxation is used to under-relax pressure in the SIMPLE algorithm, which will be introduced in the next chapter. Further, in problems where the fluid properties depend on the solution and are iteratively updated, explicit under-relaxation may be necessary to promote convergence. Examples include, but are not limited to, the turbulent viscosity in turbulent flows, the density in compressible flows, and computed interface values using HR schemes. In addition, it may be used to under-relax individual terms in the conservation equation such as the source term, and in some cases gradients of solution variables.

14.2.3 Implicit Under-Relaxation Methods

Several approaches in this category have been developed. The standard method that was presented in Chap. 8 is Patankar's approach [1], a summary of which is given here for completeness. Other methods include the E-factor approach and the false transient technique, which are also discussed.

14.2.3.1 Patankar's Under-Relaxation

As mentioned above, the iterative solution of a system of equations can be under-relaxed by introducing a relaxation factor λ^{ϕ} and expressed via Eq. (14.6). To simplify the notation used with implicit under-relaxation, Eq. (14.6) is modified to

$$\phi_C = \phi_C^* + \lambda^\phi \left(\phi_C^{new\ iteration} - \phi_C^*\right) \tag{14.7}$$

where ϕ_C^* is the value of ϕ_C from the previous iteration. In Patankar's relaxation approach, $\phi_C^{new\ iteration}$ in Eq. (14.7) is replaced by its equivalent expression from Eq. (14.1) to yield

$$\phi_C = \phi_C^* + \lambda^\phi \left(\left(\frac{-\sum\limits_{F \sim NB(C)} a_F \phi_F + b_C}{a_C} \right) - \phi_C^* \right) \tag{14.8}$$

re-arranging, the equation becomes

$$\frac{a_C}{\lambda^\phi}\phi_C + \sum_{F \sim NB(C)} a_F \phi_F = b_C + \frac{\left(1 - \lambda^\phi\right)}{\lambda^\phi} a_C \phi_C^* \tag{14.9}$$

In Eq. (14.9) the relaxation factor λ^ϕ modifies the diagonal coefficient and the right hand side without modifying the equation mathematically. Since $\lambda^\phi < 1$, under relaxation increases the diagonal dominance of the algebraic system and enhances the stability of the iterative linear solver. This is an important advantage when compared to the explicit approach.

However it is worth noting that the implicit relaxation applies a relation that is proportional to the diagonal coefficient. Thus the relaxation will be larger for a larger diagonal coefficient, which translates into a larger relaxation of higher importance for smaller control volumes. This is demonstrated in the next section.

14.2.3.2 E-Factor Relaxation

The E-Factor method [2] is a reformulation of Patankar's method. It is derived by rewriting Eq. (14.1) in the following form:

$$a_C \phi_C = b_C - \sum_{F \sim NB(C)} a_F \phi_F \tag{14.10}$$

under-relaxing, the right hand side of Eq. (14.10) is transformed to

$$a_C \phi_C = \lambda^\phi \left(b_C - \sum_{F \sim NB(C)} a_F \phi_F \right) + \left(1 - \lambda^\phi\right) a_C \phi_C^* \tag{14.11}$$

Replacing the under-relaxation factor with $\frac{E^\phi}{1+E^\phi}$, Eq. (14.11) becomes

$$a_C\phi_C = \frac{E^\phi}{1+E^\phi}\left(b_C - \sum_{F\sim NB(C)} a_F\phi_F\right) + \left(1 - \frac{E^\phi}{1+E^\phi}\right)a_C\phi_C^* \tag{14.12}$$

which can be reformulated as

$$a_C\left(1+\frac{1}{E^\phi}\right)\phi_C + \sum_{F\sim NB(C)} a_F\phi_F = b_C + \frac{1}{E^\phi}a_C\phi_C^* \tag{14.13}$$

With this formulation the under-relaxation effect can be readily interpreted in term of some artificial transient time scale that advances ϕ_C at each solver iteration. The time step Δt can be shown to be proportional to the characteristic time interval Δt^* according to

$$\Delta t = E^\phi \Delta t^* \tag{14.14}$$

where

$$\Delta t^* = \frac{\rho_C V_C}{a_C} \tag{14.15}$$

In Eq. (14.15) ρ_C is the density of the fluid in cell C of volume V_C. The characteristic time interval is related to the time required to diffuse and convect a change of ϕ_C across the element. Thus the E-factor is equivalent to an element CFL number.

It is clear from Eq. (14.15) that the time step advancement of the E-Factor relaxation is dependent on the cell volume, with the solution in a smaller element advancing more slowly than in a coarser element. This can be detrimental to the convergence rate for a steady state solution since it is very common to use highly stretched elements with small volumes near boundaries, thus forcing a critical region in the computational domain to advance at a very small time step compared to the remainder of the domain. This is also a characteristic of the Patankar relaxation method.

The relation between E^ϕ and λ^ϕ can be shown to be

$$E^\phi = \frac{1}{1-\lambda^\phi}. \tag{14.16}$$

In general values of E^ϕ are chosen in the range of 4–10, corresponding to values between 0.75 and 0.9 for λ^ϕ.

Example 2

In the figure below an illustrative boundary mesh is shown. Elements A and D represent stretched elements near a wall boundary, with the volume ratio of about $V_{C_A}/V_{C_D} \approx 0.1$. *The value of the diagonal coefficients for such meshes are usually dominated by the diffusion coefficient and in this case would be around* $a_{C_A}/a_{C_D} \approx 2$ *since element A has a boundary face.*

Compute the relative pseudo transient time for elements A and D if an under-relaxation factor of 0.8 is applied (Fig. 14.3).

Fig. 14.3 Schematic of a boundary mesh with stretched elements

Solution

Solution starts by computing the equivalent E factor for the applied relaxation factor

$$E = \frac{1}{1-\lambda} = \frac{1}{1-0.8} = 5$$

thus the pseudo time step for each of the elements is

$$\begin{aligned} \Delta t &= E^{\phi}\Delta t^{*} \\ &= 5\frac{\rho_C V_C}{a_C} \end{aligned}$$

Therefore the relative pseudo time steps for elements A and D can be computed as

$$\frac{\Delta t_A}{\Delta t_B} = \left(\frac{V_{C_A}}{a_{C_A}}\right) / \left(\frac{V_{C_D}}{a_{C_D}}\right) = \left(\frac{V_{C_A}}{V_{C_D}}\right)\left(\frac{a_{C_D}}{a_{C_A}}\right) \approx 0.1\left(\frac{1}{2}\right) = 0.05$$

implying that the pseudo time step for element D is nearly 20 times that of element A.

14.2.3.3 False Transient Relaxation

The false transient relaxation method [3] is a modification of the Euler first order implicit transient method, wherein the previous iteration values are used instead of the old time step values. As in the Euler method the diagonal dominance of the algebraic equation is increased through the addition of the pseudo-transient term, $a_C^o \phi_C$, to the diagonal coefficient and the pseudo old time step term, $a_C^o \phi_C^*$, to the right hand side. With these modifications the equation becomes

$$\left(a_C + a_C^o\right)\phi_C + \sum_{F \sim NB(C)} a_F \phi_F = b_C + a_C^o \phi_C^* \tag{14.17}$$

where the coefficient a_C^o is computed as

$$a_C^o = \frac{\rho_C V_C}{\Delta t} \tag{14.18}$$

and is basically equal to the transient coefficient obtained from the first order implicit Euler discretization of the transient term, ρ_C is the density, V_C the element volume, and Δt a user-defined false time step. For large values of Δt, the added term is negligible, and under-relaxation effects are negligible. Therefore the solution of the equation is the same as the original unrelaxed one. For very small values of Δt, the value of a_C^o becomes large dominating other terms. The solution is heavily under-relaxed leading to minute change in the value of ϕ_C (i.e., $\phi_C \approx \phi_C^*$).

In addition to allowing the solution to advance consistently over the entire computational domain, the false transient method ensures the addition of a non-zero contribution to the diagonal coefficient even in extreme cases when the diagonal coefficient is zero.

There is no general rule for assigning optimum under-relaxation factors as values used in one case may not work properly for another case. In addition, different equations may be assigned different under-relaxation factors. Further, it is not necessary to use the same under-relaxation value throughout the computational domain. Furthermore, under-relaxation values may vary from iteration to iteration. For the SIMPLE algorithm to be described in Chaps. 15 and 16, Raithby and Schneider [4] derived an optimum relation between the under-relaxation factors for the velocity and pressure fields, which will be presented in the next chapter.

14.3 Residual Form of the Equation

The discretized algebraic equation has been written so far in its "direct" or "standard" form as

$$a_C\phi_C + \sum_{F\sim NB(C)} a_F\phi_F = b_C, \tag{14.19}$$

which is the form used in OpenFOAM®.

Equation (14.19) can also be written in "correction" or "residual" form by rearranging its terms so as to solve for the correction needed to satisfy the equation. Thus if ϕ_C^* and ϕ_C' are the previous iteration value of ϕ_C and the correction needed to satisfy Eq. (14.19), respectively, then the solution is given by

$$\phi_C = \phi_C^* + \phi_C' \tag{14.20}$$

and Eq. (14.19) can be rewritten as

$$a_C\left(\phi_C^* + \phi_C'\right) + \sum_{F\sim NB(C)} a_F\left(\phi_F^* + \phi_F'\right) = b_C \tag{14.21}$$

or

$$a_C\phi_C' + \sum_{F\sim NB(C)} a_F\phi_F' = b_C - \left(a_C\phi_C^* + \sum_{F\sim NB(C)} a_F\phi_F^*\right) \tag{14.22}$$

Note that the right hand side of Eq. (14.22) represents the residual error of the equation for the field ϕ_C^*. Denoting this residual over element C by Res_C^ϕ, Eq. (14.22) becomes

$$a_C\phi_C' + \sum_{F\sim NB(C)} a_F\phi_F' = Res_C^\phi \tag{14.23}$$

It is worth noting that for the exact field Res_C^ϕ would be zero.

While mathematically equivalent to Eq. (14.19), Eq. (14.23) has one numerical advantage. In this form, numerical errors during the solution of the equation are slightly less than those associated with the standard form for cases when small variations are expected for large values of ϕ.

14.3.1 Residual Form of Patankar's Under-Relaxation

The residual form of the implicit Patankar relaxation equation is derived by rewriting Eq. (14.19) in the following form:

$$a_C\left(\phi_C^* + \phi_C'\right) = \lambda^\phi \left(b_C - \sum_{F \sim NB(C)} a_F\left(\phi_F^* + \phi_F'\right) \right) + \left(1 - \lambda^\phi\right) a_C \phi_C^* \quad (14.24)$$

which can be simplified to

$$a_C \phi_C' + \lambda^\phi \sum_{F \sim NB(C)} a_F \phi_F' = \lambda^\phi \left[b_C - \left(a_C \phi_C^* + \sum_{F \sim NB(C)} a_F \phi_F^* \right) \right] \quad (14.25)$$

Noticing that the right hand side of the above equation represents the residual of the original equation, Eq. (14.25) can be written as

$$\frac{a_C}{\lambda^\phi} \phi_C' + \sum_{F \sim NB(C)} a_F \phi_F' = Res_C^\phi \quad (14.26)$$

implying that under-relaxing the equation in residual form necessitates modifying only the diagonal coefficient.

14.4 Residuals and Solution Convergence

In any iterative solution process it is important to be able to determine when the solution can be considered good enough, or when the error can be estimated to be below a certain tolerance, or even to what precision the conservation equations have been satisfied. Having the tools to answer any of the above questions is an important ingredient for any CFD code. This can be rephrased as how to evaluate the degree of convergence of the solution field without knowing the final solution. To this end a number of indicators were proposed over the years, from the simple monitoring of a point at a location over a number of iterations, to the monitoring of an integrated value such as the drag coefficient, the total mass flow, the wall shear stress, and so on, or more commonly the monitoring of some type of equation residual. The challenge being that the method has to be used for a wide range of flow parameters and for a variety of geometries and boundary conditions.

14.4.1 Residuals

As a solution to the discretized system of equations represented by Eq. (14.1) is sought, the error in the balance equation is quantified by defining the element residual error as

$$Res_C^{\phi} = b_C - \left(a_C \phi_C + \sum_{F \sim NB(C)} a_F \phi_F \right). \tag{14.27}$$

It is clear that when the solution is reached and the equation satisfied, the value of Res_C^{ϕ} will be zero. Using Res_C^{ϕ} a number of residual indicators for the whole domain can be derived as detailed next.

14.4.2 Absolute Residual

As defined by Eq. (14.27), the residual may be a positive or a negative quantity. Since the sign is immaterial, the absolute value of the Res_C^{ϕ}, denoted by R_C^{ϕ}, is used to decide whether a solution has converged or not. If R_C^{ϕ} decreases with iterations then the solution will be converging otherwise it will be diverging. The value of R_C^{ϕ} at point C is defined as

$$R_C^{\phi} = \left| b_C - \left(a_C \phi_C + \sum_{F \sim NB(C)} a_F \phi_F \right) \right| \tag{14.28}$$

14.4.3 Maximum Residual

The solution is assumed to have converged when the maximum value of the absolute residuals, defined as,

$$R_{C,\max}^{\phi} = \max_{all\ cells} \left| b_C - \left(a_C \phi_C + \sum_{F \sim NB(C)} a_F \phi_F \right) \right| = \max_{all\ cells} \left(R_C^{\phi} \right) \tag{14.29}$$

over the domain drops below a vanishing quantity ε, i.e.,

$$R_{C,\max}^{\phi} \leq \varepsilon \Rightarrow \text{solution has converged.} \tag{14.30}$$

14.4.4 Root-Mean Square Residual

Another parameter used as a convergence indicator, is the average of the square root of the sum of the squares of the absolute residuals $R_{C,rms}^{\phi}$, mathematically given by

$$R^{\phi}_{C,\,rms} = \sqrt{\frac{\sum\limits_{C\sim all\ elements}\left(b_C - \left(a_C\phi_C + \sum\limits_{F\sim NB(C)} a_F\phi_F\right)\right)^2}{number\ of\ elements}} = \sqrt{\frac{\sum\limits_{C\sim all\ cells}\left(R^{\phi}_{C}\right)^2}{number\ of\ elements}}. \tag{14.31}$$

In this case the convergence criteria is written as

$$R^{\phi}_{C,\,rms} \leq \varepsilon \Rightarrow \text{solution has converged.} \tag{14.32}$$

14.4.5 Normalization of the Residual

The level of the absolute residual is a strong function of the variable ϕ. Therefore different variables result in different levels of R^{ϕ}_{C}. This makes it difficult to discern whether a solution has converged or not. In such cases a better insight can be gained through scaling the different residuals by dividing them by their respective maximum fluxes. Recalling that a_C represents the sum of the fluxes over the element, the residuals are scaled relative to the local value of the property ϕ to obtain a relative error by dividing them by the maximum value of $a_C\phi_C$ over the domain such that

$$R^{\phi}_{C,\,scaled} = \frac{\left|a_C\phi_C + \sum\limits_{F\sim NB(C)} a_F\phi_F - b_C\right|}{\max\limits_{all\ cells}|a_C\phi_C|}. \tag{14.33}$$

The solution is assumed to have converged when the maximum value of the scaled absolute residuals has dropped below a vanishing quantity ε, i.e.,

$$\max_{all\ cells}\left(R^{\phi}_{C,scaled}\right) \leq \varepsilon \Rightarrow \text{solution has converged} \tag{14.34}$$

It is common to require ε for scaled residuals to be of the order of 10^{-3} to 10^{-5} or less for convergence.

In addition to using either the absolute or scaled residuals, it is always insightful to monitor integrated quantities, as mentioned above, before concluding that the solution has converged. Always ensure proper convergence before declaring that a solution has converged as non-converged solutions can be misleading.

14.5 Computational Pointers

In this section, the source term linearization and relaxation techniques used in uFVM and OpenFOAM® are discussed.

14.5.1 uFVM

14.5.1.1 Source Term Linearization

The linearization and assembly of the source term in uFVM is implemented in the function cfdAssembleSourceTerm displayed in Listing 14.1. The function relies on the linearization set by the user, which has to supply the constant part of the source, Sb, and the linearized part, Sc. These terms are then added to FLUXV and FLUXC, as in Eq. (14.4).

It is worth noting that in uFVM the equations are solved in residual form, which explains the implementation of the total source in FLUXTE, rather than its constant part only.

```
theEquationField = cfdGetMeshField(theEquationName);
phi = theEquationField.phi(iElements);
%
Sb = cfdComputeFormulaAtLocale(theTerm.Sb,'Interior Elements')';
Sc = cfdComputeFormulaAtLocale(theTerm.Sc,'Interior Elements')';
%
volume = [theMesh.elements.volume];
%
% Assemble Source Term
%
pos = zeros(1,size(phi));
pos(Sc<0) = 1;
theFluxes.FLUXCE = -pos .* Sc .* volume;
theFluxes.FLUXTE = -(Sb +Sc .*phi) .* volume;
```

Listing 14.1 Linearization and implementation of the source term

14.5.1.2 Under-Relaxation

The implicit under-relaxation method of Patankar is implemented in ufvm. Since the equations are solved in residual form, their under-relaxation (Listing 14.2) requires modifying their diagonal coefficients only, as demonstrated by Eq. (14.26).

```
function cfdApplyURF(theEquationName)
%=====================================================
%  written by the CFD Group @ AUB, Fall 2006
%=====================================================
theEquation = cfdGetModel(theEquationName);
urf = theEquation.urf;
theCoefficients = cfdGetCoefficients;
theCoefficients.ac = theCoefficients.ac/urf;
cfdSetCoefficients(theCoefficients);
```

Listing 14.2 Implementation of Patankar's implicit under-relaxation method

14.5.2 OpenFOAM®

14.5.2.1 Source Term Linearization

In Sect. 14.1, the treatment of the source in the transport equation of a generic variable ϕ was discussed. The suggested linearization (or implicit treatment) can be viewed as imposing an artificial time step onto the matrix of coefficients, thus affecting the characteristic time of advancing the solution. In addition it increases diagonal dominance and enhances the solution robustness of the algebraic system of equations by allowing for an inherent relaxation to come into action when needed. That is, whenever the negative source term changes substantially, the system self-adapt the time step resolution in order to capture the characteristics of the modeled phenomenon. This is in contrast with the non-linearization approach (explicit treatment), which necessitates heavier under-relaxation of the system of equations with the relaxation factor being generally not optimal.

For the discretization of the source term OpenFOAM® [5] uses the implicit fvm:: and explicit fvc: : operators. Specifically the implementation of the fvc:: operator can be found in the directory "$FOAM_SRC/finiteVolume/finiteVolume/fvc/" in the corresponding files fvcSup.H and fvcSup.C. However it is usually the norm to define the explicit source term into the equation without the need for using the fvc:: operator. For example considering the case of a generic scalar transport equation with a source term that does not depend directly on the main variable and thus cannot be linearized, given by

$$\nabla \cdot (\rho U \phi) - \nabla \cdot (k \nabla(\phi)) = aU^2 \tag{14.35}$$

In OpenFOAM® Eq. (14.35) can be implemented, as shown in Listing 14.3,

```
fvMatrix<scalar> phiEqn
(
fvm::div(mDot,phi) - laplacian(k,phi) == a*magSqr(U)
);
```

Listing 14.3 Defining an explicit source term without invoking the **fvc** operator

with the source term not requiring any special wrapping function or operator.

The implementation of the **fvm::** functions can be found in the directory "$FOAM_SRC/finiteVolume/finiteVolume/fvm/" in the corresponding files **fvmSup.H** and **fvmSup.C**. The discretization of the linearized source is set in function **fvm::Sp**. As per Eq. (14.5), the implicit part of the source term is added as a contribution to the main diagonal of the coefficient matrix. For that purpose the function **Sp** is defined in Listing 14.4.

```
template<class Type>
Foam::tmp<Foam::fvMatrix<Type> >
Foam::fvm::Sp
(
    const DimensionedField<scalar, volMesh>& sp,
    const GeometricField<Type, fvPatchField, volMesh>& vf
)
{
    const fvMesh& mesh = vf.mesh();

    tmp<fvMatrix<Type> > tfvm
    (
        new fvMatrix<Type>
        (
            vf,
            dimVol*sp.dimensions()*vf.dimensions()
        )
    );
    fvMatrix<Type>& fvm = tfvm();

    fvm.diag() += mesh.V()*sp.field();

    return tfvm;
}
```

Listing 14.4 Script used for the definition and implementation of the **Sp** function

It is worth mentioning that the function **Sp** treats the source term irrespective of the sign of the slope of its linearized form. This means that for the case when the slope of the linearized term is positive the operation may lead to divergence of the solution algorithm as it destroys the diagonal dominance of the set of algebraic equations. Thus it is always important to ensure that the implicit treatment is used only when it results in a negative slope of the linearized term. For the case when the

slope of the linearized source term can assume, in different regions of the domain, different values (positive and negative), the negative contributions should be treated as implicit and positive contribution as explicit. For that purpose OpenFOAM® provides a special source term function denoted by **fvm::SuSp** in which the implicit/explicit treatment is automatically performed. The script of this function is given in Listing 14.5.

```
template<class Type>
Foam::tmp<Foam::fvMatrix<Type> >
Foam::fvm::SuSp
(
    const DimensionedField<scalar, volMesh>& susp,
    const GeometricField<Type, fvPatchField, volMesh>& vf
)
{
    const fvMesh& mesh = vf.mesh();

    tmp<fvMatrix<Type> > tfvm
    (
        new fvMatrix<Type>
        (
            vf,
            dimVol*susp.dimensions()*vf.dimensions()
        )
    );
    fvMatrix<Type>& fvm = tfvm();

    fvm.diag() += mesh.V()*max(susp.field(), scalar(0));

    fvm.source() -= mesh.V()*min(susp.field(), scalar(0))
        *vf.internalField();

    return tfvm;
}
```

Listing 14.5 Script used for the definition and implementation of the **SuSp** function

In this function both diagonal and source term vectors are filled depending on the local sign of the slope of the linearized source term. In fact the use of the **max/min** function achieves the selective discretization. For instance, if in a generic cell of the domain the slope of the linearized source term assumes a negative value (here taken as positive) the contribution to the source vector is zero (i.e., min(SuSp.field(), scalar(0)) = 0) and the opposite for the diagonal contribution.

14.5.2.2 Under-Relaxation

The under-relaxation methods used in OpenFOAM® include both the implicit technique of Patankar and the explicit variable relaxation. More specifically the

implicit under relaxation is applied only to the fvMatrix object (i.e., the actual finite volume discretization matrix) while the explicit relaxation is defined only for **GeometricField** objects.

The explicit relaxation described by Eq. (14.6) can be found in the file GeometricField.C in the "$FOAM_SRC/OpenFOAM/fields/GeometricFields/GeometricField" directory. The dedicated function (Listing 14.6) inside the GeometricField class for performing this task is given by

```
template<class Type, template<class> class PatchField, class GeoMesh>
void   Foam::GeometricField<Type,   PatchField,   GeoMesh>::relax(const
scalar alpha)
{
    if (debug)
    {
        InfoIn
        (
            "GeometricField<Type, PatchField, GeoMesh>::relax"
            "(const scalar alpha)"
        )  << "Relaxing" << endl << this->info() << " by " << alpha <<
endl;
    }
    operator==(prevIter() + alpha*(*this - prevIter()));
}
```

Listing 14.6 Script of the function in the **GeometricField** class for explicit under relaxation

where the operator "==" defines the new value of the **GeometricField** itself based on the current value and the previous one in accordance with Eq. (14.6).

In general to explicitly relax a variable, first its value is stored in the **`prevIter()`** array, after that calculations are performed to obtain the new predicted value, and finally relaxation is applied. For example, using the pressure "p" as the GeometricField variable the following should be written (Listing 14.7):

```
volScalarField p
(
    IOobject
    (
        "p",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);
… //any preliminary operation
p.storePrevIter();
p=…   //Perform the operation for the new predicted pressure
p.relax();
```

Listing 14.7 Explicit under relaxation of the pressure field

In this case the value of the relaxation factor is directly read from the **fvSolution** dictionary. In case the developer requires a certain constant value, say 0.5, the last line in Listing 14.5 should be replaced by "**`p.relax(0.5);`**".

The Patankar relaxation is applied directly to the matrix of coefficients and in OpenFOAM® it is implemented inside the **fvMatrix** class. The file **fvMatrix.C** in the directory "$FOAM_SRC/finiteVolume/fvMatrices/fvMatrix" contains the definition of the implicit relaxation given in Listing 14.8.

```
template<class Type>
void Foam::fvMatrix<Type>::relax(const scalar alpha)
{
    if (alpha <= 0)
    {
        return;
    }
…
```

Listing 14.8 Script for the definition of implicit relaxation

The function definition is quite long, mainly due to the constraints imposed by the diagonal dominance requirement of the matrix of coefficients, but the relevant code reads (Listing 14.9)

```
    Field<Type>& S = source();
    scalarField& D = diag();

     // Store the current unrelaxed diagonal for use in updating the
source
    scalarField D0(D);
…
    // ... then relax
    D /= alpha;
    // Finally add the relaxation contribution to the source.
    S += (D - D0)*psi_.internalField();
```

Listing 14.9 Excerpts of the script used for implicit under relaxation

In the first part, after checking the sign of the slope of the linearized source, references to the diagonal and source vectors are setup. The original diagonal is stored under the "**D0**" scalar field and then divided by the relaxation factor "**alpha**". An additional contribution is added to the source vector of the matrix where "**psi_**" is the variable associated with the **fvMatrix** class. The source term looks slightly different from the right hand side of Eq. (14.4) but defines exactly the same contribution, as shown in the following equation:

$$\frac{(1-\lambda)}{\lambda} a_C \phi_C^* = \left(\frac{a_C}{\lambda} - a_C\right) \phi_C^* \tag{14.36}$$

14.6 Closure

The chapter dealt with the treatment of the source term in the general conservation equation and discussed several methods used for under-relaxing the algebraic system of equations. The chapter also discussed some of the indicators used for checking convergence. The next chapter is devote for the solution of incompressible flow problems.

14.7 Exercises

Exercise 1
Linearize the following source term where the dependent variable is ϕ:

$$Q_C^{\phi} = A + B|\phi_C|\phi_C$$

Exercise 2
Consider the following equation:

$$\frac{\partial \phi}{\partial t} - \nabla \cdot \Gamma \nabla \phi = -\beta(\phi - 1)^{1/3} - \kappa\left(\phi^4 - 1\right)$$

Derive the algebraic equation for an element C. Use a first order Euler scheme for the transient term and linearize the source terms given that β and κ are both positive.

Exercise 3
Consider the steady convection diffusion equation given by

$$\nabla \cdot (\rho \mathbf{v} \phi) - \nabla \cdot \Gamma \nabla \phi = 2 - \phi^3$$

a. Discretize the above equation using the SMART scheme with a deferred correction approach for the convection term, and linearize the source term.
b. Then apply under-relaxation to the discretized equation using
   (i) Patankar's under-relaxation method
   (ii) The E-Factor relaxation method
   (iii) The false transient method

Exercise 4

In the solution of turbulent flows (which will be the subject of Chap. 17), turbulence models are introduced. One such models is the well-known two-equation $k-\varepsilon$ turbulence model, with k and ε governed by conservation equations that have the form of the general conservation equation. The ε equation in the model has the following source term:

$$Q_C^{\varepsilon} = C_{\varepsilon 1}\frac{\varepsilon}{k}P_k - C_{\varepsilon 2}\rho\frac{\varepsilon^2}{k}$$

where ρ is the fluid density, $C_{\varepsilon 1}$ and $C_{\varepsilon 2}$ are positive constants, and P_k, k, and ε are all positive quantities. Suggest a linearization for Q_C^{ε}.

Exercise 5

In solving the momentum equation in $r\theta Z$ coordinates, the momentum equation in the θ direction has the following source term:

$$Q_C^{v_\theta} = -\rho\frac{v_r v_\theta}{r}$$

Suggest a linearization for $Q_C^{v_\theta}$.

Exercise 6

The Fithigh-Nagumo equations, presented below, model the evolution of animal coat pattern formation. The equations represent concentrations of two chemical substances influencing skin pigmentation whose reaction-diffusion interaction result in the formation of patterns that are reminiscent of the zebra or jaguars skins. The variations in the resulting patterns depend on the constants used in the model. The model equations are given by

$$\frac{\partial\phi}{\partial t} = \nabla\cdot a\nabla\phi + \phi - \phi^3 - \varphi + k$$
$$\tau\frac{\partial\varphi}{\partial t} = \nabla\cdot b\nabla\varphi + \phi - \varphi$$

Solve these equations using uFVM and OpenFOAM® for the following values of the constants:

$$a = 2.8\times 10^{-4}$$
$$b = 5\times 10^{-3}$$
$$\tau = 0.1$$
$$k = -0.005$$

Let the computational domain be a square of dimension [4, 4] discretized with a mesh of size 100×100 elements. As boundary conditions, use a zero gradient over all boundaries with a random initial conditions for the ϕ (values within [0, 1]) and

initial conditions for $\varphi = 1 - \phi$. Solve the problem over 5 s with a time step $\Delta t \leq 10^{-4}$ s, using the first order Euler Implicit Scheme and the second order Crank-Nicholson scheme and compare results. Make sure that the source term is linearized.

References

1. Patankar S (1980) Numerical heat transfer and fluid flow. McGraw Hill, New York
2. Van Doormaal JP, Raithby GD (1984) Enhancement to the SIMPLE method for predicting incompressible fluid flows. Numer Heat Transf 7:147–163
3. Mallinson GD, de Vahl Davis G (1973) The method of the false transient for the solution of coupled elliptic equations. J Comput Phys 12(4):435–461
4. Raithby GD, Schneider GE (1979) Numerical solution of problems in incompressible fluid flow: treatment of the velocity-pressure coupling. Numer Heat Transf 2:417–440
5. OpenFOAM (2015) Version 2.3.x. http://www.openfoam.org