

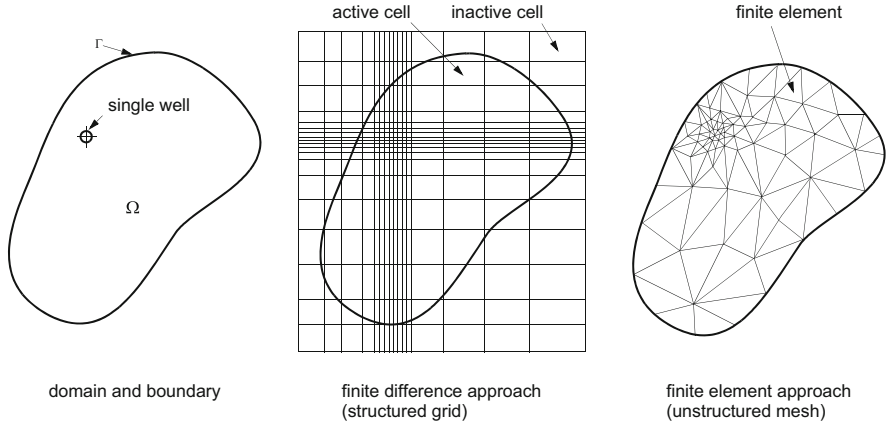
# Chapter 8

## Fundamental Concepts of Finite Element Method (FEM)

### 8.1 Introduction

In the previous Chaps. 3–5 the governing continuum balance equations in form of partial differential equations (PDE's) have been derived for a wide range of flow, mass and heat transport processes in porous and fractured media. Their solution under given IC's and BC's, such as described in Chap. 6, requires appropriate and efficient mathematical methods, which can be firstly grouped into *analytical* and *numerical* methods. There is a family of powerful analytical methods (e.g., Fourier and Laplace transformation, complex variable techniques, Green's functions, perturbation methods, power series), which are capable of solving a certain number of problems in an exact way. However, exact analytical solutions are often only attainable for elementary linear (or quasi-linear) problems on simple (regular) geometries. Very few analytical solutions exist for nonlinear problems with regions of regular geometry, however, these are usually approximate solutions in terms of an infinite series or some transcendental functions that can be evaluated only approximately. If exact analytical solutions are available on idealized problems they are often advantageous in comparison to numerical results for purposes of verification and estimation of errors arising in the alternative numerical methods.

Problems involving irregular geometry, materials with variation in properties, nonlinear relationships and/or complex BC's are intractable by analytical methods and numerical methods must be used in general. They allow the solution for a broad range of problems. The key feature of any numerical method is in the *approximate* solution of the basic PDE's via spatial and temporal discretizations, in which the solution variables, which are basically continuous functions of space and time, are obtained by *discrete* values, defined at specific points in space and time (Fig. 8.1). In doing this approximation, the governing PDE's are replaced by a number (often, a very large number) of linear (or linearized) *algebraic* equations, which can be easily solved via computers. As a consequence of the numerical approximation, *errors* are naturally inherent in the solution and the big challenge of numerical methods is to minimize these numerical errors and find best accurate, convergent and stable



**Fig. 8.1** Example of 2D domain discretized by finite differences and finite elements

solutions by using efficient, general and robust strategies of approximation. It is important to ensure that the approximation satisfies certain important properties of the exact solution, e.g., conservativity, boundedness and consistency (see Sect. 1.2.2 for further discussion).

We can classify the numerical methods as follows:

- Finite difference method (FDM)
- Method of characteristics (MOC)
- Finite element method (FEM)
- Finite volume method (FVM)
- Boundary element method (BEM)
- Meshless method (MLM)
- Spectral element method (SEM)

These methods are closely related. The FDM is the classic numerical approach, e.g., [168]. It is conceptually straightforward and had a high popularity in past. FDM approximates the differential form of the basic PDE in a difference form and is usually restricted to simple (rectangular) geometries and BC's. The specific advantage of FDM lies in the use of regular grids on which the approximation can be most efficiently performed. The development of finite-difference approximations is commonly done by either Taylor series expansion or curve-fitting technique.

The MOC as a traditional solution method [165, 184] is only applicable to PDE of hyperbolic type, i.e., for advection-dominated transport processes. It is based on the concept of trajectories (or characteristics) on which a large number mathematical particles are tracked. While mainly 1D and partly 2D unsteady flow processes could be successfully modeled, the method is rather cumbersome when extended to multidimensional problems, dealing with complex BC's and nonlinearities.

The basic ideas underlying the FEM have a long history. Ritz [445] and Galerkin [181] presented variational integral formulations of a PDE and

approximate solutions based on their minimization. Pioneering work of FEM in the modern form that we know today dates back to the early 1940s given by Hrenikoff [264] and Courant [104]. First applications were done for aero structures in the late 1950s [525]. Clough [88] coined the term *finite element method* at that time. The power of FEM was quickly recognized and the first textbook on FEM appeared in the mid-1960s by Zienkiewicz and Cheung [589], which boosted the development of FEM in many fields of sciences and engineering lasting up to now. First applications of FEM for porous-media problems were given by Zienkiewicz and Cheung [589], Pinder and Gray [421] and Huyakorn and Pinder [280]. Since then, the FEM has become one of the basic tools for numerical analysis in structural mechanics, fluid dynamics, heat transfer and numerical mathematics (for literature review see Sect. 1.3 with Table 1.3).

Today, the FEM represents a collection of theory-rich techniques and is based on the weak (or variational) formulation of the governing initial-boundary-value problem. This theoretical foundation on weak formulation is quite distinct from FDM. The weak formulation is an integral approach, which is a natural and an adequate approach of a continuum balance statement. FEM subdivides the continuum in a finite number of elements, for which the balance statements are discretely applied. The resultant algorithm of the FEM can be universally expressed as a matrix statement with all formation processes on a *generic* master element. The generic master element statement is then *assembled* into a global matrix statement. BC's can be brought directly into the generic master element providing accurate expressions of surface integrals for the PDE global domain boundary on which any flux-type BC is applicable. The FEM is essentially geometry-free. In principle, FEM can be applied to domains of arbitrary shape and with quite arbitrary BC's. FEM by its nature leads to unstructured meshes (Fig. 8.1). Most complex types of geometries can be simply handled. These features make the FEM a general, systematic, very powerful and highly flexible numerical method, which is superior to the other numerical methods.

There is a wide variety of methods called finite volume methods (FVM's), e.g., [83, 162]. Sometimes they are termed as control volume methods or previously, integrated FDM. FVM is usually also based on weak formulations of the basic problem similar to FEM, however, the approximation of the balance terms relies on evaluation of surface integrals, where boundary fluxes are developed via finite differences. In this process, the conservation is enforced across the surfaces of the adjoining control volume. It allows the construction of cost-effective schemes for both structured and unstructured grids. It has been demonstrated [209] that the FVM is inherently a FEM if using low-order elements (basically linear). It can be shown [83, 165] that FVM can be formulated from either FDM or FEM. Identical discrete schemes result for FVM and FEM [284] if using low-order approximations and equivalent meshing via control volumes and elements, respectively. However, serious problems with FVM can arise when cross-derivatives (such as anisotropic problems, e.g., associated with the hydrodynamic dispersion tensor  $D_k$ , (3.184)) appear in the governing PDE. Commonly, diffusive/conductive gradient terms are approximated in FVM by using a two-point flux approximation (TPFA) scheme

applied to two adjacent cell values. But, TPFA is insufficient to express diffusive fluxes, where off-diagonal values in an anisotropic diffusion/conduction tensor exist (cf. Chap. 7). To circumvent this drawback multi-point flux approximation (MPFA) can be used [412], which, however, requires a nonlinear evaluation making FVM rather cumbersome and potentially less accurate. Moreover, higher-order approximations and complex geometries on arbitrarily unstructured meshing can lead to further difficulties in FVM.

The BEM is based on boundary integral equations in which only the boundaries of a domain are used to obtain approximate solutions, e.g., [54, 350]. It reduces the solution of the problem to one dimension less than the original problem (e.g., a 3D problem is solved by a 2D approximation), however, the resulting matrix systems are full, whereas the other numerical methods generally result in sparse matrices. The most serious aspect with BEM is that a fundamental solution (free space Green's function) of the PDE must be available, which commonly requires linear equations with constant coefficients (i.e., homogeneous materials). Thus, the application of BEM is limited to special problems.

For all numerical methods mentioned so far mesh configurations are required consisting of elements, cells or control volumes formed by connecting nodal points in a predefined manner. Unlikely, various methods have been developed which depend on finite number of points rather than meshes. They are called meshless methods (MLM's), finite point methods (FPM's) or element free Galerkin (EFG) methods, e.g., [44, 352]. Although most of the meshless methods have high computational cost as compared to FEM, they provide advantages for a certain class of problems such as moving boundaries, phase transformation, crack propagation and large deformation in solids as well as modeling of multiscale phenomena. The major advantage of MLM is the elimination of the need for mesh generation, which can be itself a difficult task. However, MLM's are not (yet) sophisticated enough for application in a general context. They often require background cells to improve numerical stability and accuracy so that in current practice, MLM's have shown not to be truly mesh-free.

The idea of MLM's has been adopted and modified in the so-called eXtended FEM (XFEM), e.g., [172]. It tries to combine the advantages of FEM and MLM, while alleviating existing drawbacks of MLM. In the XFEM singularities, material discontinuities, high gradients and other non-smooth properties can be described by an extended set of discontinuous basis functions without the need of local remeshing or alignment of the discontinuity (e.g., fractures) to edges or faces of a finite element mesh as usually necessary in standard FEM. However, XFEM is commonly prone to ill-conditioning of the resulting matrix systems, often in a drastic extent, so that standard solution techniques (e.g., preconditioned iterative solvers) are most likely to fail. It is an active field of research to improve the XFEM (e.g., using stable XFEM [17]) for finding more tractable approaches in practical applications.

The SEM represents a combination of the classic spectral method and FEM, e.g., [151, 196, 334]. It can generate solutions of very high accuracy with relatively few terms in the approximate solution, provided that the exact solution is sufficiently smooth (but possibly steep). In contrast to the standard FEM, the unknown

coefficients in the approximate solution must not be identified with nodal unknowns. Instead, in SEM formulations the approximate functions are built by Fourier series, Legendre polynomials or Chebyshev polynomials. The main advantage of SEM relies on the exponential convergence property as soon as smooth solutions are involved. For instance, doubling the mesh resolution reduces the numerical error by two orders of magnitude, not by a mere factor of 4 as in standard numerical methods (FEM, FDM, FVM) with second-order algebraic convergence. But, the main drawback of SEM is its inability to handle complex geometries and material discontinuities (even though effort is current to overcome these difficulties, e.g., [413]). It significantly limits the applicability of SEM. Furthermore, SEM has shown insufficiently effective for solving linear problems [83].

## 8.2 Basic Model Equations and Prototypical PDE's

The basic continuum equations of the variable-density flow, mass and heat transport in porous and fractured media have been developed and fully expressed in Chaps. 3 and 4. They have been summarized in Table 3.7 for general variably saturated porous media, in Table 3.9 for fully saturated porous media (groundwater), in Table 3.10 for 2D unconfined aquifers and in Table 3.11 for 2D confined aquifers. Additionally, Tables 4.5–4.7 list the equations for variable-density flow, mass and heat transport of discrete features. A typical set of these coupled governing PDE's can be expressed in the following compact form:

$$\mathcal{L}(\phi) = \begin{cases} \mathbf{m} \cdot \frac{\partial(\mathbf{c} \cdot \phi)}{\partial t} + \nabla \cdot (\mathbf{f}^a - \mathbf{f}^d) - \mathbf{b} & = \mathbf{0} \quad \text{divergence form} \\ \mathbf{n} \cdot \frac{\partial \phi}{\partial t} + \mathbf{a} \cdot (\mathbf{q} \cdot \nabla \phi) - \nabla \cdot \mathbf{f}^d - \mathbf{b} + \mathbf{d} = \mathbf{0} & \text{convective form} \end{cases} \quad (8.1)$$

in  $\Omega \subset \mathfrak{N}^D, t \geq t_0$

where  $\mathcal{L}(\phi)$  denotes the PDE system written in terms of the *state variable*  $\phi = \phi(\mathbf{x}, t)$ . It is expressed on the (physical) domain  $\Omega$ , with the bounding closure  $\Gamma$ , lying on  $D$ -dimensional Euclidean space  $\mathfrak{N}^D$ , and for time  $t$  starting at, and proceeding from some initial time  $t_0$ . For the solution, appropriate BC's are required on the entirety of  $\Gamma$  and IC's on  $\Omega \cup \Gamma$  are necessary as described in Chap. 6. In (8.1) the following vector and matrix definitions are used:

$$\phi = \begin{pmatrix} h \\ C_k \\ T \end{pmatrix}, \mathbf{f}^a = \begin{pmatrix} \mathbf{0} \\ \mathbf{q} C_k \\ \rho c \mathbf{q} T \end{pmatrix}, \mathbf{f}^d = \begin{pmatrix} k_r \mathbf{K} f_{\mu} \cdot (\nabla h + \chi \mathbf{e}) \\ \mathbf{D}_k \cdot \nabla C_k \\ \Lambda \cdot \nabla T \end{pmatrix}, \mathbf{a} = [0, 1, \rho c],$$

$$\mathbf{b} = \begin{pmatrix} Q + Q_{\text{EOB}} \\ \tilde{R}_k - \varepsilon s \partial_k \mathfrak{R}_k C_k \\ H_e \end{pmatrix}, \mathbf{d} = \begin{pmatrix} 0 \\ C_k Q \\ \rho c (T - T_0) Q \end{pmatrix}, \mathbf{m} = \left[ s S_o + \varepsilon \frac{\partial s}{\partial h}, 1, 1 \right],$$

$$\mathbf{n} = \left[ s S_o + \varepsilon \frac{\partial s}{\partial h}, \quad \varepsilon s \mathfrak{H}_k, \quad \varepsilon s \rho c + (1 - \varepsilon) \rho^s c^s \right],$$

$$\mathbf{c} = [1, \quad \varepsilon s \mathfrak{H}_k, \quad \varepsilon s \rho c + (1 - \varepsilon) \rho^s c^s] \quad (8.2)$$

where  $\mathbf{f}^a$  and  $\mathbf{f}^d$  are the advective and dispersive (diffusive) flux tensors, respectively, which are expressed in terms of functions derived from the state variable  $\phi$ . Note that in (8.2)  $[\dots]$  symbolizes diagonal matrices, cf. (2.22).

The coupled system of PDE's (8.1) has to be solved for  $\phi$  via FEM. Due to its nonlinearity and complexity specific treatments are necessary in dependence on the underlying problem class, e.g., variable-density flow, unsaturated porous media, chemical reaction systems, fracture modeling, heat exchange. It is useful to discuss the FEM solutions for each problem class in a separate manner. However, for introducing the FEM and explaining the principal solution steps it is convenient to start with a simpler PDE written for a scalar state variable  $\phi$ , which is representative for all of the flow and transport processes under consideration. An appropriate prototypical PDE is the following *advection (convection)-dispersion equation* (ADE), incorporating effects of advection, dispersion (diffusion), retardation and decay (as illustrated in Fig. 8.2), written in its *divergence form* as

$$\mathcal{L}(\phi) = \frac{\partial(\mathcal{R}\phi)}{\partial t} + \nabla \cdot (\mathbf{q}\phi) - \nabla \cdot (\mathbf{D} \cdot \nabla \phi) + \vartheta \phi - H - Q_{\phi w} = 0 \quad (8.3)$$

$$\text{in } \Omega \subset \mathfrak{H}^D, \quad t \geq t_0$$

to be solved for  $\phi$  subject to a set of BC's of Dirichlet, Neumann and Cauchy type as well as well-type SPC (see Chap. 6), which typically are

$$\begin{aligned} \phi &= \phi_D && \text{on } \Gamma_D \times t[t_0, \infty) \\ (\phi \mathbf{q} - \mathbf{D} \cdot \nabla \phi) \cdot \mathbf{n} &= q_N^\dagger && \text{on } \Gamma_N \times t[t_0, \infty) \\ (\phi \mathbf{q} - \mathbf{D} \cdot \nabla \phi) \cdot \mathbf{n} &= -\Phi^\dagger(\phi_C - \phi) && \text{on } \Gamma_C \times t[t_0, \infty) \\ Q_{\phi w} &= -\sum_w \phi_w Q_w(t) \delta(\mathbf{x} - \mathbf{x}_w) && \text{on } \mathbf{x}_w \in \Omega \times t[t_0, \infty) \end{aligned} \quad (8.4)$$

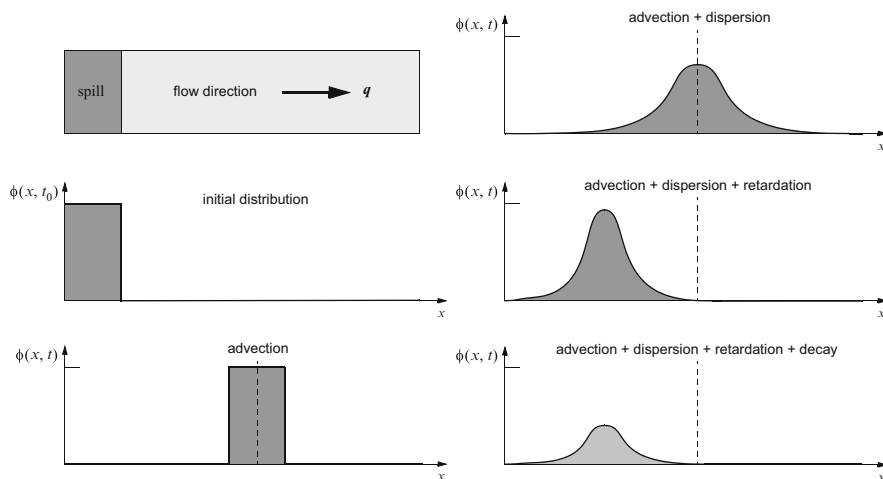
and written in its *convective form* as

$$\mathcal{L}(\phi) = \mathcal{K} \frac{\partial \phi}{\partial t} + \mathbf{q} \cdot \nabla \phi - \nabla \cdot (\mathbf{D} \cdot \nabla \phi) + (\vartheta + Q)\phi - H - Q_{\phi w} = 0 \quad (8.5)$$

$$\text{in } \Omega \subset \mathfrak{H}^D, \quad t \geq t_0$$

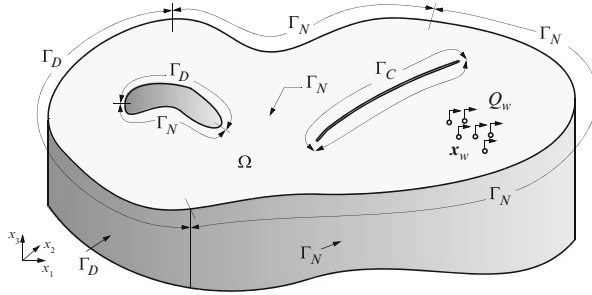
subject to the Dirichlet, Neumann and Cauchy BC's as well as well-type SPC as

$$\begin{aligned} \phi &= \phi_D && \text{on } \Gamma_D \times t[t_0, \infty) \\ -(\mathbf{D} \cdot \nabla \phi) \cdot \mathbf{n} &= q_N && \text{on } \Gamma_N \times t[t_0, \infty) \\ -(\mathbf{D} \cdot \nabla \phi) \cdot \mathbf{n} &= -\Phi(\phi_C - \phi) && \text{on } \Gamma_C \times t[t_0, \infty) \\ Q_{\phi w} &= -\sum_w (\phi_w - \phi(\mathbf{x}_w)) Q_w(t) \delta(\mathbf{x} - \mathbf{x}_w) && \text{on } \mathbf{x}_w \in \Omega \times t[t_0, \infty) \end{aligned} \quad (8.6)$$



**Fig. 8.2** Effects of advection, dispersion, retardation and decay on a scalar transport quantity  $\phi$ : Advection simply translates the quantity by the advective velocity  $q$ , dispersion spreads the quantity both downstream and upstream and smoothes the fronts, retardation delays the advective transport and reduces effects of dispersion, and decay accounts for disappearance of an amount of the quantity

where the boundary  $\Gamma = \Gamma_D \cup \Gamma_N \cup \Gamma_C$  is composed of the three segments,  $\Gamma_D$ ,  $\Gamma_N$  and  $\Gamma_C$ , which do not overlap each other:  $\Gamma_D \cap \Gamma_N \cap \Gamma_C = \emptyset$ . Usually,  $\Gamma_D \neq \emptyset$  is required for steady-state problems ( $\partial\phi/\partial t = 0$ ), unless  $\Gamma_C \neq \emptyset$ . It is assumed that each of the boundary segments can be further subdivided into different portions of the same BC type, e.g.,  $\Gamma_N = \Gamma_{N_I} \cup \Gamma_{N_O} \cup \dots$ , however, which must not be necessarily connected (Fig. 8.3). The scalar state variable  $\phi$  can stand for the hydraulic head  $h$  (or pressure head  $\psi$ ), for a species concentration  $C_k$  or the temperature  $T$  in accordance with the corresponding problem class to be solved. In the above Eqs. (8.3)–(8.6),  $\mathbf{n}$  is the positive outward-directed unit normal to  $\Gamma$ ,  $\mathbf{q}$  is the (at first assuming known) advective flux,  $\mathcal{R}$  and  $\check{\mathcal{R}}$  are storage (retardation) coefficients, which are prototypical for the coefficients appearing in (8.2) (note that for an ADE applied to a porous medium they include porosity and saturation),  $\mathbf{D}$  is a dispersion (diffusion/conduction) tensor,  $\vartheta$  is a (linear) decay parameter,  $H$  is a general source/sink term,  $Q$  is a flow supply term (without well-type SPC),  $Q_{\phi_w}$  is the singular well sink/source function with given well pumping rate  $Q_w(t)$  and known  $\phi_w$  at well  $w$  of location  $\mathbf{x}_w$ ,  $\phi_D$  is the prescribed value of  $\phi$  on the Dirichlet boundary segment  $\Gamma_D$ ,  $q_N^\dagger$  and  $q_N$  are the prescribed fluxes on the Neumann boundary segment  $\Gamma_N$  for the divergence and the convective form of the ADE, respectively, and  $\phi_C$  is a known value of  $\phi$  on the Cauchy boundary segment  $\Gamma_C$  associated with the transfer coefficients  $\Phi^\dagger$  and  $\Phi$  related to the divergence and convective form of the governing ADE, respectively. Note that  $q_N^\dagger$  and  $q_N$  as well as  $\phi_C$  associated with  $\Phi^\dagger$  or  $\Phi$  have different meaning in the divergence form (8.3) and the convective form (8.6) since in the divergence form the boundary flux consists of



**Fig. 8.3** Domain  $\Omega \subset \mathfrak{R}^3$  and boundary sections of Dirichlet type  $\Gamma_D \subset \Gamma$ , Neumann type  $\Gamma_N \subset \Gamma$  and Cauchy type  $\Gamma_C \subset \Gamma$  as well as SPC's  $Q_w$  for wells at  $x_w \in \Omega$

the total (advective plus dispersive) flux while in the convective form the boundary flux implies only a dispersive flux. However, as has been shown in Sect. 6.3.2 the Neumann-type BC of the divergence form is equivalent to the Cauchy-type BC of (8.6) if using for the convective form, cf. (6.21) and (6.28),

$$-(D \cdot \nabla \phi) \cdot n = q \cdot n(\phi_C - \phi) \quad \text{on } \Gamma_C \times t[t_0, \infty) \quad (8.7)$$

Finally, the statement of the PDE problem (8.3) or (8.5) has to be completed by specifying an IC in the form:

$$\phi(x, t_0) = \phi_0(x) \quad \text{in } \bar{\Omega} \quad (8.8)$$

where  $\phi_0$  is a given function of  $\phi$  at position  $x$  and initial time  $t_0$  with  $\bar{\Omega} = \Omega \cup \Gamma$ .

### 8.3 Mathematical Classification of PDE's

The governing partial differential equations (PDE's), such as summarized in Sect. 8.2, can be mathematically classified into three categories: (1) elliptic, (2) parabolic and (3) hyperbolic. Most of the equations are 2nd-order PDE's. To classify the PDE's several procedures are available, where most common is the discriminant evaluation [486]: Let us consider the PDE of the form in a 2D domain  $x^T = (x \ y) \in \mathfrak{R}^2$

$$\mathcal{L}(\phi) = A \frac{\partial^2 \phi}{\partial x^2} + B \frac{\partial^2 \phi}{\partial x \partial y} + C \frac{\partial^2 \phi}{\partial y^2} + D \frac{\partial \phi}{\partial x} + E \frac{\partial \phi}{\partial y} + F \phi + G = 0 \quad (8.9)$$

where the coefficients  $A, B, C, D, E, F$  and  $G$  are constants or may be functions of both independent and/or dependent variables. Then, the three categories of PDE can be distinguished according to:



$$\begin{aligned}
 \text{elliptic PDE: } & B^2 - 4AC < 0, \\
 \text{parabolic PDE: } & B^2 - 4AC = 0, \\
 \text{hyperbolic PDE: } & B^2 - 4AC > 0
 \end{aligned}
 \tag{8.10}$$

It is apparent that the classification depends only on the highest-order derivatives in each independent variable. We note that the coefficients  $A$  to  $G$  of (8.9) can also vary as functions of  $x$ ,  $y$ ,  $\phi$ ,  $\partial\phi/\partial x$  or  $\partial\phi/\partial y$  and (8.10) can still be used if  $A$ ,  $B$  and  $C$  are given a local interpretation. This implies that an equation can belong to one classification in one part of the domain and another classification in another part of the domain. Typical examples of PDE classifications are given as follows:

(a) Elliptic equation

$$\begin{aligned}
 -K_x \frac{\partial^2 \phi}{\partial x^2} - K_y \frac{\partial^2 \phi}{\partial y^2} &= 0 \quad (K_x > 0, K_y > 0) \\
 A = -K_x, B = 0, C = -K_y \\
 B^2 - 4AC &= -4K_x K_y < 0
 \end{aligned}
 \tag{8.11}$$

(b) Parabolic equation

$$\begin{aligned}
 \frac{\partial \phi}{\partial t} + v \frac{\partial \phi}{\partial x} - K_x \frac{\partial^2 \phi}{\partial x^2} &= 0 \quad (v > 0, K_x > 0) \\
 A = -K_x, B = 0, C = 0 \\
 B^2 - 4AC &= 0
 \end{aligned}
 \tag{8.12}$$

(c) Hyperbolic equation

$$\begin{aligned}
 \frac{\partial \phi}{\partial t} + v \frac{\partial \phi}{\partial x} &= 0 \quad (v > 0) \\
 \text{differentiating with respect to } x \text{ and } t: \\
 \frac{\partial^2 \phi}{\partial t \partial x} + v \frac{\partial^2 \phi}{\partial x^2} = 0, \quad \frac{\partial^2 \phi}{\partial t^2} + v \frac{\partial^2 \phi}{\partial t \partial x} &= 0 \\
 \text{and combining:} \\
 \frac{\partial^2 \phi}{\partial t^2} - v^2 \frac{\partial^2 \phi}{\partial x^2} &= 0 \\
 A = 1, B = 0, C = -v^2 \\
 B^2 - 4AC &= 4v^2 > 0
 \end{aligned}
 \tag{8.13}$$

To generalize the PDE classification to more variables, a common way is to classify the PDE's via the 2nd-order differential operator defined as

$$\mathcal{D}(\phi) = \nabla \cdot (\mathbf{D} \cdot \nabla \phi)
 \tag{8.14}$$

where  $D$  is a symmetric, positive-definite tensor. Then, the three categories of PDE in the  $D$ -dimensional space  $\mathfrak{R}^D$  ( $D = 1, 2, 3$ ) are distinguished as follows:

$$\begin{aligned} \text{elliptic PDE:} \quad & -\mathcal{D}(\phi) - F\phi + G = 0, \\ \text{parabolic PDE:} \quad & a \frac{\partial \phi}{\partial t} + \mathbf{v} \cdot \nabla \phi - \mathcal{D}(\phi) - F\phi + G = 0, \\ \text{hyperbolic PDE:} \quad & a \frac{\partial \phi}{\partial t} + \mathbf{v} \cdot \nabla \phi + F\phi - G = 0 \end{aligned} \quad (8.15)$$

where  $a$ ,  $F$  and  $G$  are coefficients and  $\mathbf{v}$  represents a flux vector.

The classification of PDE's can be associated with the smoothness of the solution  $\phi$ . Elliptic PDE's produce solutions that are smooth (up to the smoothness of coefficients) even if BC's are not smooth. On the other hand, parabolic PDE's will cause the smoothness of solutions to increase with growing time and reducing influences by first-order derivatives, while hyperbolic PDE's preserve lack of smoothness.

## 8.4 Methods of Approximation

### 8.4.1 Approximate Solution

The sought approximation of the basic PDE's (8.3) and (8.5) with their BC's (8.4) and (8.6), respectively, starts with expressing a suitable approximate functional form for the solution  $\phi$ . The usual form is

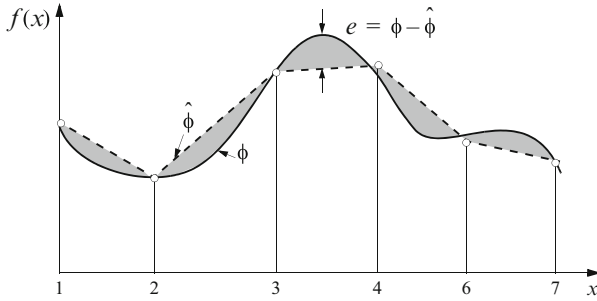
$$\phi(\mathbf{x}, t) \approx \hat{\phi}(\mathbf{x}, t) = \sum_j N_j(\mathbf{x}) \phi_j(t) \quad (8.16)$$

where  $\hat{\phi}$  is the approximate solution,  $N_j$  represent a set of given *basis functions* (or trial or interpolation functions) and  $\phi_j$  are a set of unknown coefficients (at the *nodes* of interpolation) to be determined. In the functional expression (8.16) the spatial and temporal variables are separated. This variable separation procedure is termed as *Kantorovich (semidiscrete) method* [149, 300, 377] and allows the discretization first in space followed by a time marching procedure for the temporal discretization, which is the usual practice in numerical analysis leading to efficient computational schemes, although alternative of (8.16) exists.<sup>1</sup>

---

<sup>1</sup>A continuous space-time approximation can be expressed in the form

$$\phi(\mathbf{x}, t) \approx \hat{\phi}(\mathbf{x}, t) = \sum_j N_j(\mathbf{x}, t) \phi_j$$



**Fig. 8.4** Approximating  $\hat{\phi}(x)$  (dashed line) of state function  $\phi$  (solid line) creating an error  $e = \phi - \hat{\phi}$  (shaded area)

It is to be noted that the sought approximate solution  $\hat{\phi}$  is a function distributed over the entire domain  $\Omega$  of  $\mathcal{L}(\phi)$  and its boundary  $\Gamma$ ; hence, it is a *global* function. Examples of the corresponding basis functions include polynomials (e.g., Lagrangian, Hermite or Chebyshev polynomials) or trigonometric functions (e.g., Fourier series). Approximating the solution to (8.3)–(8.6) with the series expression (8.16), an *error* will generally occur defined as the difference between the exact solution  $\phi$  and the approximate solution  $\hat{\phi}$ :

$$e = \phi - \hat{\phi} \tag{8.17}$$

The situation is sketched in Fig. 8.4, where the exact solution  $\phi$  shown as solid line is approximated by a piecewise continuous linear interpolation  $\hat{\phi}$  between selected locations at nodes  $j = 1, 2, \dots$  depicted as dashed line. The difference between  $\phi$  and  $\hat{\phi}$  represents the error  $e$  of the solution illustrated by the shaded area in Fig. 8.4. It indicates that the error is in general a function of space (and time). The error can also be measured only at the discrete locations of nodes  $j = 1, 2, \dots$ , providing a vector  $e$  of pointwise errors:

$$e = e_j = \phi_j - \hat{\phi}_j \quad (j = 1, 2, \dots) \tag{8.18}$$

The goal is now to make the error  $e$  as small as possible, and hence minimize the difference between  $\phi$  and  $\hat{\phi}$ . Since the exact solution  $\phi$  is generally unknown and  $e$  is variable in space and time, the minimization of the error  $e$  requires a general approach to be described in the following.

---

where the basis functions  $x, t$  have to be prescribed both in space  $x$  and time  $t$ . It requires a finite element in space-time and increases the computational dimension, e.g., a transient 3D problem needs a 4D trial space.

### 8.4.2 Definition of Errors and Related Functional Spaces

Defining in the  $D$ -dimensional Euclidean  $\mathfrak{R}^D$  space with

$$D^s e = \frac{\partial^{|s|} e(\mathbf{x})}{\partial x_1^{s_1} \partial x_2^{s_2} \dots \partial x_D^{s_D}} = \left( \frac{\partial^{s_1}}{\partial x_1^{s_1}} \right) \left( \frac{\partial^{s_2}}{\partial x_2^{s_2}} \right) \dots \left( \frac{\partial^{s_D}}{\partial x_D^{s_D}} \right) e(\mathbf{x}) \quad (8.19)$$

the generalized partial derivatives up to and including of the  $2m$ th order appearing in the governing PDE's ( $m = 0, 1, \dots$ ), where  $s$  is a multi-index  $s = (s_1, s_2, \dots)$  with  $|s| = \sum_{i=1}^D s_i$  and  $s_i = 0, 1, \dots$ , the following error norms are meaningful in the further analysis.

#### 8.4.2.1 Sobolev Space $W_p^m(\Omega)$ Norm Error

The Sobolev space norm error is defined as

$$\|e\|_{W_p^m(\Omega)} = \left\{ \int_{\Omega} \left[ |e|^p + \sum_{s=1}^m |D^s e|^p \right] d\Omega \right\}^{\frac{1}{p}} \quad (8.20)$$

where  $m$  denotes the highest order of the derivatives of the  $2m$ th governing PDE and  $p$  represents the power to which the derivatives are raised. Note that for a 2nd order PDE  $m = 1$ . The Sobolev space  $W_p^m(\Omega)$  is defined as the functional space which includes all  $p$  integrable functions ( $1 \leq p \leq +\infty$ ) with  $p$  integrable derivatives of  $m$ th order. Hence,  $W_p^m(\Omega)$  is a collection of functions on  $\Omega$  which are endowed with the associated norm (8.20), where any function  $\phi \in W_p^m(\Omega)$  is  $m$  times differentiable and  $p$ th-order integrable on  $\Omega$ .

#### 8.4.2.2 Hilbert Space $H^m(\Omega)$ Norm Error

The Hilbert space  $H^m(\Omega)$  corresponds to the Sobolev space  $W_p^m(\Omega)$  with  $p$  equal to 2, i.e.,  $H^m(\Omega) = W_2^m(\Omega)$ . Thus,

$$\|e\|_{H^m(\Omega)} = \|e\|_{W_2^m(\Omega)} = \left\{ \int_{\Omega} \left[ e^2 + \sum_{s=1}^m (D^s e)^2 \right] d\Omega \right\}^{\frac{1}{2}} \quad (8.21)$$

As seen the Hilbert space  $H^m(\Omega)$  is a functional space with square integrable functions and square integrable derivatives of  $m$ th order. Any function  $\phi \in H^m(\Omega)$  is  $m$  times differentiable and square integrable on  $\Omega$ .

### 8.4.2.3 Energy Norm Error

The energy norm error  $\|e\|_E$  is a special case of the Hilbert space norm  $H^m(\Omega)$  in the 2<sup>m</sup>th PDE. For a 2<sup>nd</sup>-order PDE ( $m = 1$ ) it reads

$$\|e\|_E = \|e\|_{H^1(\Omega)} = \|e\|_{W_2^1(\Omega)} = \left\{ \int_{\Omega} \left[ e^2 + \left( \frac{\partial e}{\partial x_1} \right)^2 + \left( \frac{\partial e}{\partial x_2} \right)^2 + \dots \right] d\Omega \right\}^{\frac{1}{2}} \tag{8.22}$$

The Hilbert space  $H^1(\Omega)$  is a functional space with square integrable functions and square integrable derivatives of 1<sup>st</sup> order. Any function  $\phi \in H^1(\Omega)$  is once differentiable and square integrable on  $\Omega$ . A (smaller) Hilbert subspace  $H_0^1(\Omega)$  can be defined for functions  $\phi$  which are zero on the boundary  $\Gamma$  of the domain  $\Omega$  at the same time, i.e.,  $\phi|_{\Gamma} = 0$ . Then, the Hilbert subspace  $H_0^1(\Omega)$  reads

$$H_0^1(\Omega) \equiv \{ \phi \in H^1(\Omega) : \phi = 0 \text{ on } \Gamma \} \tag{8.23}$$

so that any function  $\phi \in H_0^1(\Omega)$  is once differentiable, square integrable on  $\Omega$  and zero on  $\Gamma$ .

### 8.4.2.4 $L_p(\Omega)$ -Norm (Banach Space Norm) Error

The Banach space  $L_p(\Omega)$  is defined as the complete normed linear space such that

$$\|e\|_{L_p(\Omega)} = \left( \int_{\Omega} |e|^p d\Omega \right)^{\frac{1}{p}} \tag{8.24}$$

Using  $p = 2$  we obtain the  $L_2(\Omega)$  space, which is equivalent to the Hilbert space  $H^0(\Omega)$  with  $m = 0$ :

$$\|e\|_{L_2(\Omega)} = \|e\|_{H^0(\Omega)} = \|e\|_{W_2^0(\Omega)} = \left( \int_{\Omega} e^2 d\Omega \right)^{\frac{1}{2}} \tag{8.25}$$

The  $L_2(\Omega)$  space is a functional space with square integrable functions so that any function  $\phi \in L_2(\Omega)$  must be square integrable on  $\Omega$ . The  $L_2$  norm is one of the most widely used error norm. Another useful error norm is the maximum error norm  $L_{\infty}(\Omega)$  given for  $p = \infty$ :

$$\|e\|_{L_{\infty}(\Omega)} = \max_j |e_j| \tag{8.26}$$

where  $e_j$  is the discrete error at location  $j$ . More seldom used in practice is the  $L_1(\Omega)$  error norm:

$$\|e\|_{L_1(\Omega)} = \int_{\Omega} |e| d\Omega \tag{8.27}$$

### 8.4.2.5 Root Mean Square (RMS) and Other Pointwise Error Norms

The *RMS error norm* represents a pointwise  $L_2(\Omega)$  space norm, which can be expressed in different forms. Most useful is the normalized RMS error norm defined as

$$\|e\|_{\text{RMS}} = \left[ \frac{1}{N_P} \left( \frac{1}{\hat{\phi}_{\max}^2} \sum_{j=1}^{N_P} e_j^2 \right) \right]^{\frac{1}{2}} \quad (8.28)$$

where  $N_P$  is the number of components of the error vector  $e$  and  $\hat{\phi}_{\max}$  is the maximum value of the approximate solution  $\hat{\phi}$  to normalize the  $e_j$  components.

If focusing on the maximum error occurring in the approximate discrete solution, the *normalized maximum error norm* can be useful and is defined as

$$\|e\|_{L_\infty} = \frac{1}{\hat{\phi}_{\max}} \max_j |e_j| \quad (8.29)$$

It yields the strongest error measure and should be preferred if the local error is important in the numerical approximation (*'scheme listens to each sound'*).

As an alternative to the  $L_2$  RMS norm, the normalized  $L_1$  error norm can be applied:

$$\|e\|_{L_1} = \frac{1}{N_P \hat{\phi}_{\max}} \sum_{j=1}^{N_P} |e_j| \quad (8.30)$$

However, it should not be the first choice and the RMS norm is commonly more appropriate.

### 8.4.3 Method of Weighted Residuals (MWR)

There are two fundamental theories of constructing approximate solutions to the governing PDE's:

1. The classic *Rayleigh-Ritz method* [377,590], which is based on finding solutions via an equivalent variational problem. By extremization of the related variational functional (condition of stationarity) useful approximate solutions can be obtained. However, natural variational functionals only exist for *self-adjoint*<sup>2</sup>

---

<sup>2</sup>Let  $\mathcal{L}$  be a differential operator of a PDE defined in  $\Omega$  and let  $\phi$  and  $\psi$  be two functions in the field of definition of  $\mathcal{L}$ . The operator  $\mathcal{L}$  is said to be self-adjoint if identical to its own adjoint operator  $\mathcal{L}^*$ , i.e.,  $\mathcal{L} = \mathcal{L}^*$ , which must result from the integral statement

differential operator  $\mathcal{L}$  of the governing PDF. A self-adjoint PDE is given for a symmetric equation (containing no advective terms). However, ADE in the form of (8.3) or (8.5) possesses an unsymmetric non-self-adjoint differential operator for which a natural variational functionals cannot be found.<sup>3</sup>

2. The *method of weighted residuals* (MWR) [163] provides the most generality in applications and will be preferred usually. It can be applied to all type of PDE and systems of PDE's, even to those which cannot be cast in variational form. The following finite element approach will be exclusively based on MWR.

It is obvious that the approximate solution  $\hat{\phi}$  of (8.16) is not likely to satisfy exactly the governing PDE

$$\mathcal{L}(\phi) = 0 \tag{8.31}$$

in form of (8.3) or (8.5). Substituting  $\hat{\phi}$  in (8.31) yields a PDE for the error  $e = \phi - \hat{\phi}$ , (8.17), written as

$$\mathcal{L}(e) = \mathcal{L}(\phi) - \mathcal{L}(\hat{\phi}) = -\mathcal{L}(\hat{\phi}) \neq 0 \tag{8.32}$$

or

$$\mathcal{L}(\hat{\phi}) = R, \quad R = -\mathcal{L}(e) \neq 0 \tag{8.33}$$

where  $R = R(\mathbf{x}, t)$  is the *residual*, which is a measure of the *induced error* arising from the used approximation. It is commonly impossible and not reasonable to try to force  $R$  to be zero everywhere in  $\Omega$  and on  $\Gamma$  (it would meet the exact solution).

---


$$\int_{\Omega} \mathcal{L}(\phi)\psi d\Omega = \int_{\Omega} \phi \mathcal{L}^*(\psi) d\Omega + \text{boundary integral terms}$$

<sup>3</sup>For instance, the non-self-adjoint ADE in form of (8.5) can be transformed to a self-adjoint problem by introducing the new operator [129, 132, 218, 427]

$$\bar{\mathcal{L}} = \varphi \mathcal{L}$$

where the function  $\varphi = \varphi(\mathbf{x})$  is chosen by

$$\varphi = \exp(\beta), \quad \beta = -\frac{\mathbf{q} \cdot \mathbf{x}}{\|\mathbf{D}\|}$$

assuming a dispersion tensor  $\mathbf{D}$  with  $D_{ij} = 0$  for  $i \neq j$ . It yields the following variational functional

$$\mathcal{I} = \int_{\Omega} \left[ \frac{1}{2} \nabla \phi \cdot (\mathbf{D} \cdot \nabla \phi) + \left( \mathcal{R} \frac{\partial \phi}{\partial t} + \frac{\vartheta + Q}{2} \phi - H - Q_{\phi w} \right) \phi \right] \exp(\beta) d\Omega - \int_{\Gamma} (\mathbf{D} \cdot \nabla \phi) \cdot \mathbf{n} \phi \exp(\beta) d\Gamma$$

to be extremized. However, its application is clearly restricted because values of  $\exp(\beta)$  can become very large (small) for advection-dominated processes and the variational functional terms overflow (underflow) in practical computations [129, 485].

Instead, the pragmatic approach is to require the residual  $R$  to vanish in an overall integrated sense. The corresponding mathematical statement is that  $R$  must be *orthogonal*<sup>4</sup> to an arbitrary *weighting* (or test) *function*  $w(\mathbf{x}, t)$ , i.e.,

$$\int_{\Omega} w(\mathbf{x}, t) R d\Omega = 0, \quad \text{for all } w(\mathbf{x}, t) \quad (8.34)$$

The expression (8.34) is the core of MWR [163], which minimizes the residual  $R$  as a weighted average over the domain  $\Omega$ . This form is quite general and the arbitrariness in  $w(\mathbf{x}, t)$  provides theoretical generality for various numerical approaches. For specifying appropriate weighting functions  $w(\mathbf{x}, t)$  it is assumed that its interpolation, using any suitable polynomial basis, can be made sufficiently precise:

$$w(\mathbf{x}, t) \approx \hat{w}(\mathbf{x}, t) = \sum_i w_i(\mathbf{x}) W_i(t) \quad (8.35)$$

where  $w_i(\mathbf{x})$  is the set of interpolation polynomials and  $W_i(t)$  is the corresponding set of known coefficients at the nodes of interpolation. The coefficients  $W_i(t)$ ,

---

<sup>4</sup>We know from (2.26) when two vectors in space are at right angles, their dot product is zero and the vectors are orthogonal. While vectors have only a limited number of entries, any real-valued function  $f(\mathbf{x})$  is characterized by infinite number of points within its domain of definition  $\Omega$ . It is obvious to consider two functions  $f(\mathbf{x})$  and  $g(\mathbf{x})$  to be orthogonal, if the product  $f(\mathbf{x})g(\mathbf{x})$  ‘summed’ over all  $\mathbf{x}$  within the domain  $\Omega$  results zero. Since the amount of  $\mathbf{x}$  covers infinite real numbers, the product  $f(\mathbf{x})g(\mathbf{x})$  has to be integrated. Hence, the analogy for the dot product is the *inner product* given by

$$(f, g) = \int_{\Omega} f(\mathbf{x})g(\mathbf{x})d\Omega$$

Then, the two functions are orthogonal if  $(f, g) = 0$ . It is evident that functions if defined in the  $L_2(\Omega)$  space, cf. (8.25), e.g.,

$$\|f\| = \left( \int_{\Omega} f(\mathbf{x})^2 d\Omega \right)^{\frac{1}{2}} < \infty, \quad \|g\| = \left( \int_{\Omega} g(\mathbf{x})^2 d\Omega \right)^{\frac{1}{2}} < \infty$$

can be treated if they were vectors, where the *Schwarz’s inequality* holds

$$(f, g) \leq \|f\| \|g\|$$

or

$$|(f, g)|^2 \leq (f, f)(g, g)$$

The  $L_2(\Omega)$ –norm corresponds to a measure of the size of a function, which is in direct analogy with the vector norm (2.11). The Schwarz’s inequality ensures that the expression

$$\cos \theta = \frac{(f, g)}{\|f\| \|g\|}$$

yields well-defined angles  $\theta$  in space similar to the scalar product of vectors (2.24).



**Table 8.1** Suitable choices of weighting functions  $w_i(\mathbf{x})$ , ( $i = 1, 2, \dots, N_{\text{EQ}}$ ) and their resulting numerical methods

$w_i(\mathbf{x})$	WS (8.36) <sup>a</sup>	Method	Remark
$\delta(\mathbf{x} - \mathbf{x}_i)$	$\int_{\Omega} \delta(\mathbf{x} - \mathbf{x}_i) \mathcal{L}(\hat{\phi}) d\Omega = 0$	Point collocation	FDM
$\begin{cases} 1 & \text{for } \mathbf{x}_i \in \Omega^e \\ 0 & \text{for } \mathbf{x}_i \notin \Omega^e \end{cases}$	$\int_{\Omega} \mathcal{L}(\hat{\phi}) d\Omega = 0$	Subdomain collocation	FVM
$N_i$	$\int_{\Omega} N_i(\mathbf{x}) \mathcal{L}(\hat{\phi}) d\Omega = 0$	Galerkin (Bubnov-Galerkin)	GFEM (standard FEM)
$N_i + \tilde{F}_i(\mathbf{x})$	$\int_{\Omega} (N_i(\mathbf{x}) + \tilde{F}_i(\mathbf{x})) \mathcal{L}(\hat{\phi}) d\Omega = 0$	Petrov-Galerkin	PGFEM (upwind)
$\partial R / \partial \phi_i$	$\int_{\Omega} \mathcal{L}(N_i(\mathbf{x})) \mathcal{L}(\hat{\phi}) d\Omega = 0$	Least square Galerkin	LSGFEM (PGLS)

<sup>a</sup>  $\mathcal{L}(\hat{\phi}) = R = \mathcal{L}\left(\sum_{j=1}^{N_p} N_j(\mathbf{x})\phi_j(t)\right)$

however, quantify the specific weighting function  $w$ . To remove this dependence on a specific  $w$ , the weak statement (8.35) is extremized with respect to the parametric set  $W_i(t)$ . Thus, the following weak statement (WS) for minimizing the residual error  $R = \mathcal{L}(\hat{\phi})$  in any selected approximate solution  $\hat{\phi}$  (8.16) results

$$\begin{aligned} \text{WS} &= \frac{\partial}{\partial W_i} \int_{\Omega} \hat{w}(\mathbf{x}, t) \mathcal{L}(\hat{\phi}) d\Omega = 0 \\ &= \int_{\Omega} w_i(\mathbf{x}) \mathcal{L}\left(\sum_{j=1}^{N_p} N_j(\mathbf{x})\phi_j(t)\right) d\Omega = 0 \quad \text{for } (1 \leq i \leq N_{\text{EQ}}) \end{aligned} \tag{8.36}$$

where ( $i = 1, 2, \dots, N_{\text{EQ}}$ ) is chosen to produce exactly the correct number of equations required to determine the  $N_{\text{EQ}}$  unknown coefficients  $\phi_j(t)$  at any time  $t$ . We note that

$$N_{\text{EQ}} = N_p N_{\text{DOF}} \tag{8.37}$$

where  $N_p$  is the number of chosen nodes and  $N_{\text{DOF}}$  is the number of *degrees of freedom*. For example,  $N_{\text{DOF}} = 1$  for scalar equations of  $\phi$  in the form of (8.3) or (8.5) and  $N_{\text{DOF}} = N + 2$  for the vectorial variable  $\phi = (h \ C_k \ T)^T$  ( $k = 1, 2, \dots, N$ ) appearing in (8.2). Having the weak statement expressed in the form of (8.36) it remains to identify the two sets of known functions  $w_i(\mathbf{x})$  and  $N_j(\mathbf{x})$  spanning the domain  $\Omega \subset \mathfrak{R}^D$ . Usually, both the basis function set  $N_j(\mathbf{x})$  and the weighting function set  $w_i(\mathbf{x})$  are defined as interpolation polynomials, with a typical selection as Lagrange polynomials. Depending on the choice of the weighting functions  $w_i(\mathbf{x})$  various alternative (and familiar) methods can be generated. The most important methods are summarized in Table 8.1.

Viewing Table 8.1, we can recognize classic numerical techniques as special cases of MWR. In the *point collocation* approach a set of points  $\mathbf{x}_i$  is specified in the solution domain  $\Omega$  and Dirac delta functions are chosen as weighting functions.

It produces a discrete approximation referred to as a stencil, common in finite-difference schemes. The choices of polynomials  $N_j(\mathbf{x})$  determine finally the accuracy of the finite-difference approximation. In the *subdomain collocation* approach the solution domain is subdivided into a number of subdomains  $\Omega = \cup \Omega^e$  and weighting functions are unity for all  $i$  if  $\mathbf{x}_i \in \Omega^e$  and zero otherwise. It leads to finite volume approximations. As  $w_i$  are constant in each of the respective subdomains, any integration by part reduces to boundary integrals. First-order operations are obvious and give normal fluxes through the discretized subdomain boundaries. However, with derivatives higher than first-order, FVM approaches require specific treatment such as TPFA or MPFA schemes [412].

A suitable option for the set of weighting functions  $w_i(\mathbf{x})$  is to require it be identical to the set of basis functions  $N_i(\mathbf{x})$  by each term  $i$ :  $w_i = N_i$ . It means, the test functions are represented by a linear combination of the *same* basis functions as used to approximate the solution. This is known as the *Galerkin criterion* named after B.G. Galerkin [181] who originally introduced it for (non-discrete) structural formulations. This Galerkin method leads to the standard finite-element approximation, called as Galerkin-FEM or GFEM (sometimes termed as Bubnov-Galerkin method [590] to differ from the modified Petrov-Galerkin method). It is important to note that the Galerkin-based WS enforces the residual error  $R$  be orthogonal to every member of the basis functions, which provides an *optimal* approximation expressed by *Céa's lemma* [84, 193, 555] written in the form:

$$\|e\|_{E, G} \leq \|e\|_{E, O} \quad (8.38)$$

where  $\|e\|_{E, G}$  and  $\|e\|_{E, O}$  are the energy (Hilbert space) norm errors (8.22) produced by the Galerkin method and by any other approximation method, respectively. For elliptic boundary value problems the optimality (8.38) is explicitly shown in Appendix F. Extensions to GFEM are given in the so-called Petrov-Galerkin method, where the weighting functions differ from the basis functions. It allows the foundation of stabilized numerical techniques which are appropriate for solving advection-dominated transport problems.

In *least squares* (LS) the set of weighting functions is constructed via the PDE operation. The resulting schemes can provide better convergence properties. Furthermore, it can be exploited to derive stabilized methods for ADE with dominant advection. An advantageous and attractive feature of the LS method is that a non-self-adjoint (1st-order differential) operator of PDE is converted into a self-adjoint 2nd-order problem, which provides *symmetry* in the approximate equation system. The Galerkin choice  $w_i(\mathbf{x}) = N_i(\mathbf{x})$  is also optimal for LS approximations.

In the following Galerkin WS will be taken as the base finite-element weak statement. Extensions will be given for the Petrov-Galerkin and least square FEM to derive artificial diffusion stabilization mechanisms of upwind schemes applied to ADE.

## 8.5 Weak Forms

For the following finite element analysis any governing PDE  $\mathcal{L}(\phi) = 0$  (with its BC's and IC's) has to be recast into its weak form (or weak statement) according to (8.34)

$$\int_{\Omega} w(\mathbf{x}, t) \mathcal{L}(\phi) d\Omega = 0, \quad \forall w(\mathbf{x}, t) \quad (8.39)$$

where  $w(\mathbf{x}, t)$  is an arbitrary weighting function. It is important to note the difference between the original PDE formulation and the weak form from the mathematical point of view. While the classic statement of the initial boundary value problem is in general unique and unambiguous, there is usually no unique weak statement of the same problem because there are alternative choices for  $w$  and optional formulations for BC's. Each weak form, however, has usually a unique solution. Some weak statements are more useful than others and it is important to find the most appropriate weak form. In this sense, a weak form represents a formulation *equivalent* to the governing PDE. The weak form incorporates the BC's.

### 8.5.1 Divergence Form of ADE

The weak form (8.39) in application to the ADE (8.3) yields

$$\int_{\Omega} w \frac{\partial(\mathcal{R}\phi)}{\partial t} d\Omega + \int_{\Omega} w \nabla \cdot (\mathbf{q}\phi) d\Omega - \int_{\Omega} w \nabla \cdot (\mathbf{D} \cdot \nabla \phi) d\Omega + \int_{\Omega} w(\vartheta\phi - H - Q_{\phi w}) d\Omega = 0 \quad (8.40)$$

which is satisfied for any weighting function  $w = w(\mathbf{x}, t)$ . In the formulation of (8.40)  $w$  need not to be differentiable and it is sufficient to require that  $w$  is only square integrable:  $\forall w \in L_2(\Omega)$ .

However, let us restrict the class of weighting functions to those, which are at least once-differentiable, i.e.,  $\forall w \in H^1(\Omega)$ . The restriction on  $w$  permits to invoke the following identity via partial integration applied to the 1st-order advective term

$$\int_{\Omega} \nabla \cdot (w\mathbf{q}\phi) d\Omega = \int_{\Omega} w \nabla \cdot (\mathbf{q}\phi) d\Omega + \int_{\Omega} \phi \mathbf{q} \cdot \nabla w d\Omega \quad (8.41)$$

and to the 2nd-order dispersion term

$$\int_{\Omega} \nabla \cdot [w(\mathbf{D} \cdot \nabla \phi)] d\Omega = \int_{\Omega} w \nabla \cdot (\mathbf{D} \cdot \nabla \phi) d\Omega + \int_{\Omega} \nabla w \cdot (\mathbf{D} \cdot \nabla \phi) d\Omega \quad (8.42)$$

Now, let us apply the Gauss's integral theorem (2.77) to the LHS's of (8.41) and (8.42) to obtain

$$\begin{aligned} \int_{\Omega} \nabla \cdot (w\mathbf{q}\phi) d\Omega &= \int_{\Gamma} w\phi\mathbf{q} \cdot \mathbf{n} d\Gamma \\ \int_{\Omega} \nabla \cdot [w(\mathbf{D} \cdot \nabla\phi)] d\Omega &= \int_{\Gamma} w(\mathbf{D} \cdot \nabla\phi) \cdot \mathbf{n} d\Gamma \end{aligned} \quad (8.43)$$

and to find for (8.41)

$$\int_{\Omega} w\nabla \cdot (\mathbf{q}\phi) d\Omega = \int_{\Gamma} w\phi\mathbf{q} \cdot \mathbf{n} d\Gamma - \int_{\Omega} \phi\mathbf{q} \cdot \nabla w d\Omega \quad (8.44)$$

and for (8.42)

$$\int_{\Omega} w\nabla \cdot (\mathbf{D} \cdot \nabla\phi) d\Omega = \int_{\Gamma} w(\mathbf{D} \cdot \nabla\phi) \cdot \mathbf{n} d\Gamma - \int_{\Omega} \nabla w \cdot (\mathbf{D} \cdot \nabla\phi) d\Omega \quad (8.45)$$

Inserting (8.44) and (8.45) into (8.40), the weak form becomes

$$\begin{aligned} \int_{\Omega} w \frac{\partial(\mathcal{R}\phi)}{\partial t} d\Omega - \int_{\Omega} \phi\mathbf{q} \cdot \nabla w d\Omega + \int_{\Omega} \nabla w \cdot (\mathbf{D} \cdot \nabla\phi) d\Omega + \\ \int_{\Omega} w(\vartheta\phi - H - Q_{\phi w}) d\Omega + \int_{\Gamma} w(\phi\mathbf{q} - \mathbf{D} \cdot \nabla\phi) \cdot \mathbf{n} d\Gamma = 0, \quad \forall w \in H^1(\Omega) \end{aligned} \quad (8.46)$$

Recalling that the boundary is composed of three segments  $\Gamma = \Gamma_D \cup \Gamma_N \cup \Gamma_C$  imposed by the Dirichlet, Neumann and Cauchy-type BC's, we can separate the boundary integral of (8.46) into these three parts and invoke the BC's of (8.4) to obtain

$$\begin{aligned} \int_{\Omega} w \frac{\partial(\mathcal{R}\phi)}{\partial t} d\Omega - \int_{\Omega} \phi\mathbf{q} \cdot \nabla w d\Omega + \int_{\Omega} \nabla w \cdot (\mathbf{D} \cdot \nabla\phi) d\Omega + \\ \int_{\Omega} w(\vartheta\phi - H - Q_{\phi w}) d\Omega + \int_{\Gamma_D} w(\phi\mathbf{q} - \mathbf{D} \cdot \nabla\phi) \cdot \mathbf{n} d\Gamma + \\ \int_{\Gamma_N} wq_N^\dagger d\Gamma - \int_{\Gamma_C} w\Phi^\dagger(\phi_C - \phi) d\Gamma = 0, \quad \forall w \in H^1(\Omega) \end{aligned} \quad (8.47)$$

Now, we have to further restrict the class of test functions  $w$  to those that vanish on the Dirichlet boundary segment  $\Gamma_D$ , i.e., we require  $w = 0$  on  $\Gamma_D$ . This class of functions belongs to the  $H_0^1$  functional space (8.23). Using this restriction of  $\forall w \in H_0^1$ , the final weak form for the divergence form of ADE (8.3) with its BC's (8.4) results

$$\begin{aligned}
& \int_{\Omega} w \frac{\partial(\mathcal{R}\phi)}{\partial t} d\Omega - \int_{\Omega} \phi \mathbf{q} \cdot \nabla w d\Omega + \int_{\Omega} \nabla w \cdot (\mathbf{D} \cdot \nabla \phi) d\Omega + \\
& \int_{\Omega} w(\vartheta \phi - H) d\Omega + \sum_w w(\mathbf{x}_w) \phi_w Q_w(t) + \int_{\Gamma_N} w q_N^\dagger d\Gamma - \\
& \int_{\Gamma_C} w \Phi^\dagger (\phi_C - \phi) d\Gamma = 0, \quad \forall w \in H_0^1(\Omega) \quad (8.48)
\end{aligned}$$

which has to be solved for  $\phi \approx \hat{\phi}$ . We recognize from (8.48) that the sought solution must also only be once differentiable, i.e.,  $\phi \approx \hat{\phi} \in H^1(\Omega)$ . Note that in (8.48) the well-type SPC (8.4) has been inserted, where we made use of the integral over the SPC singularity, which simplifies

$$\int_{\Omega} w Q_{\phi_w} d\Omega = - \int_{\Omega} w \left( \sum_w \phi_w Q_w(t) \delta(\mathbf{x} - \mathbf{x}_w) \right) d\Omega = - \sum_w w(\mathbf{x}_w) \phi_w Q_w(t) \quad (8.49)$$

This SPC realization in the weak form implies that the entire amount of the sink/source  $Q_w$  of a well  $w$  fully pertains to the equation at the given point  $\mathbf{x}_w$ . In a finite element approximation it will be attained by enforcing that each well coincides with a node of the spatial discretization.

### 8.5.2 Convective Form of ADE

The weak form for the ADE (8.5) with its BC's (8.6) can be derived in a similar way as done in Sect. 8.5.1 for the divergence form. The weak statement (8.39) applied to (8.5) yields

$$\begin{aligned}
& \int_{\Omega} w \dot{\mathcal{R}} \frac{\partial \phi}{\partial t} d\Omega + \int_{\Omega} w \mathbf{q} \cdot \nabla \phi d\Omega - \int_{\Omega} w \nabla \cdot (\mathbf{D} \cdot \nabla \phi) d\Omega + \\
& \int_{\Omega} w [(\vartheta + Q)\phi - H - Q_{\phi_w}] d\Omega = 0, \quad \forall w \in L_2(\Omega) \quad (8.50)
\end{aligned}$$

In contrast to the weak form for the divergence form of ADE we restrict the partial integration only to the 2nd-order dispersion term in the convective form of ADE, i.e.,

$$\int_{\Omega} \nabla \cdot [w(\mathbf{D} \cdot \nabla \phi)] d\Omega = \int_{\Omega} w \nabla \cdot (\mathbf{D} \cdot \nabla \phi) d\Omega + \int_{\Omega} \nabla w \cdot (\mathbf{D} \cdot \nabla \phi) d\Omega \quad (8.51)$$

By employing the Gauss's integral theorem (2.77) on the LHS term of (8.51) we find

$$\int_{\Omega} w \nabla \cdot (\mathbf{D} \cdot \nabla \phi) d\Omega = \int_{\Gamma} w (\mathbf{D} \cdot \nabla \phi) \cdot \mathbf{n} d\Gamma - \int_{\Omega} \nabla w \cdot (\mathbf{D} \cdot \nabla \phi) d\Omega \quad (8.52)$$

Inserting (8.52) into (8.50) the weak form of the convective form of ADE results

$$\begin{aligned} & \int_{\Omega} w \dot{\mathcal{R}} \frac{\partial \phi}{\partial t} d\Omega + \int_{\Omega} w \mathbf{q} \cdot \nabla \phi d\Omega + \int_{\Omega} \nabla w \cdot (\mathbf{D} \cdot \nabla \phi) d\Omega + \\ & \int_{\Omega} w [(\vartheta + \mathcal{Q})\phi - H - \mathcal{Q}_{\phi w}] d\Omega - \int_{\Gamma} w (\mathbf{D} \cdot \nabla \phi) \cdot \mathbf{n} d\Gamma = 0, \quad \forall w \in H^1(\Omega) \end{aligned} \quad (8.53)$$

Separating the boundary integral of (8.53) into the three segments  $\Gamma = \Gamma_D \cup \Gamma_N \cup \Gamma_C$  imposed by the Dirichlet, Neumann and Cauchy-type BC's, respectively, we invoke the BC's of (8.6) to obtain

$$\begin{aligned} & \int_{\Omega} w \dot{\mathcal{R}} \frac{\partial \phi}{\partial t} d\Omega + \int_{\Omega} w \mathbf{q} \cdot \nabla \phi d\Omega + \int_{\Omega} \nabla w \cdot (\mathbf{D} \cdot \nabla \phi) d\Omega + \\ & \int_{\Omega} w [(\vartheta + \mathcal{Q})\phi - H - \mathcal{Q}_{\phi w}] d\Omega - \int_{\Gamma_D} w (\mathbf{D} \cdot \nabla \phi) \cdot \mathbf{n} d\Gamma + \\ & \int_{\Gamma_N} w q_N d\Gamma - \int_{\Gamma_C} w \Phi (\phi_C - \phi) d\Gamma = 0, \quad \forall w \in H^1(\Omega) \end{aligned} \quad (8.54)$$

Using this restriction  $\forall w \in H_0^1$ , the final weak form for the convective form of ADE (8.5) with its BC's (8.6) results

$$\begin{aligned} & \int_{\Omega} w \dot{\mathcal{R}} \frac{\partial \phi}{\partial t} d\Omega + \int_{\Omega} w \mathbf{q} \cdot \nabla \phi d\Omega + \int_{\Omega} \nabla w \cdot (\mathbf{D} \cdot \nabla \phi) d\Omega + \\ & \int_{\Omega} w [(\vartheta + \mathcal{Q})\phi - H] d\Omega + \sum_w w(\mathbf{x}_w) (\phi_w - \phi(\mathbf{x}_w)) Q_w(t) + \\ & \int_{\Gamma_N} w q_N d\Gamma - \int_{\Gamma_C} w \Phi (\phi_C - \phi) d\Gamma = 0, \quad \forall w \in H_0^1(\Omega) \end{aligned} \quad (8.55)$$

for solving  $\phi \approx \hat{\phi} \in H^1(\Omega)$ , where the well-type SPC has been incorporated according to (8.6) (see related discussion in Sect. 8.5.1).

### 8.5.3 Discussion of Both Weak Forms

We emphasize again that the BC's used in the weak form (8.55) for the convective form of ADE have different meaning in comparison with BC's embodied in the weak form (8.48) for the divergence form of ADE because in general  $q_N \neq q_N^\dagger$

and  $\Phi \neq \Phi^\dagger$ . Only in absence of the normal advective flux  $\mathbf{q} \cdot \mathbf{n} = 0$ , it becomes  $q_N = q_N^\dagger$  and accordingly  $\Phi = \Phi^\dagger$ . The consequences on boundary fluxes in both weak forms are obvious. For instance, a natural Neumann BC for the divergence form of ADE  $q_N^\dagger = 0$  implies that the boundary segment  $\Gamma_N$  is impervious for the total (both advective and dispersive) flux independent of the actual value of  $\mathbf{q} \cdot \mathbf{n}$ , which represents a stronger BC formulation in comparison with the convective form of ADE. On the other hand, a natural Neumann BC for the convective form of ADE  $q_N = 0$  ensures at first that the boundary segment  $\Gamma_N$  is only impervious for the dispersive flux, unless  $\mathbf{q} \cdot \mathbf{n} = 0$  can be additionally satisfied. In practical application, the differences between these two weak forms are often not relevant. In solving the convective form of ADE a preceding solution of a flow problem delivers a flow field which satisfies  $\mathbf{q} \cdot \mathbf{n}$  conditions on the boundary in a weak sense and implies appropriate formulations of BC's for both advective and dispersive fluxes in the convective form of ADE, which are equivalent to the divergence form of ADE. In cases, where a total load of a quantity  $\phi$  (consisting of advective plus dispersive fluxes) has to be imposed on a boundary section as formulated by (6.21), (6.28) or (8.7), the Cauchy BC term of (8.55) in the convective form of ADE can be easily utilized as

$$\int_{\Gamma_C} w\Phi(\phi_C - \phi)d\Gamma = - \int_{\Gamma_C} w\mathbf{q} \cdot \mathbf{n}(\phi_C - \phi)d\Gamma \quad (8.56)$$

where  $\mathbf{q} \cdot \mathbf{n}|_{\Gamma_C}$  is a known advective normal flux on  $\Gamma_C$  so that  $\mathbf{q} \cdot \mathbf{n}\phi_C|_{\Gamma_C}$  prescribes an advective load of quantity  $\phi$ , positive outward-directed on  $\Gamma_C$ .

As discussed in Sect. 6.5.7 outflow BC's (OBC's) can be imposed in two different ways. Commonly, for standard situations a zero-gradient condition, i.e., a natural Neumann BC with  $\nabla\phi \approx \mathbf{0}$  is applied. Denoting the boundary portion of the OBC by  $\Gamma_{No} \subset \Gamma_N \subset \Gamma$ , it is specified

$$\int_{\Gamma_{No}} wq_N d\Gamma = 0 \quad \text{on} \quad \Gamma_{No} \subset \Gamma_N \quad (8.57)$$

for the convective weak form (8.55) and

$$\int_{\Gamma_{No}} wq_N^\dagger d\Gamma = \int_{\Gamma_{No}} w\phi\mathbf{q} \cdot \mathbf{n}d\Gamma \quad \text{on} \quad \Gamma_{No} \subset \Gamma_N \quad (8.58)$$

for the divergence weak form (8.48). It is obvious, this type of OBC can be simply realized in the convective form, while for the divergence form a surface integral remains to be treated implicitly because  $\phi$  is unknown and the normal flux  $\mathbf{q} \cdot \mathbf{n}$  must be determined (or be known) on  $\Gamma_{No}$ . The second and alternative way is to impose the OBC fully implicitly, even for the gradient-driven dispersive boundary flux. Using this BC formulation the Neumann-type boundary integrals are replenished to specify the *implicit* OBC

$$\int_{\Gamma_{No}} w q_N d\Gamma = - \int_{\Gamma_{No}} w (\mathbf{D} \cdot \nabla \phi) \cdot \mathbf{n} d\Gamma \quad \text{on } \Gamma_{No} \subset \Gamma_N \quad (8.59)$$

for the convective weak form (8.55) and

$$\int_{\Gamma_{No}} w q_N^\dagger d\Gamma = \int_{\Gamma_{No}} w (\phi \mathbf{q} - \mathbf{D} \cdot \nabla \phi) \cdot \mathbf{n} d\Gamma \quad \text{on } \Gamma_{No} \subset \Gamma_N \quad (8.60)$$

for the divergence weak form (8.48), which must be treated with unknown  $\phi$ . We conclude that OBC requires in general an implicit treatment of the specific Neumann-type surface integrals for the divergence weak form, which is more complex. In contrast, however, the OBC in the convective weak form can be simply specified, unless the zero-gradient Neumann condition on the outflow boundary is not appropriate under specific situations (cf. discussion in Sect. 6.5.7).

## 8.6 Spatial Discretization by Finite Elements

The governing weak forms derived in Sect. 8.5 contains integral expressions which have to be solved. To accomplish an approximate solution via FEM the continuum domain with its boundary  $\bar{\Omega} = \Omega \cup \Gamma$  is subdivided into a set of nonoverlapping subdomains, called *finite elements* (see Fig. 8.5), such that

$$\bar{\Omega} \approx \hat{\hat{\Omega}} \equiv \bigcup_{e=1}^{N_E} \bar{\Omega}^e \quad \text{with } \bar{\Omega}^e = \Omega^e \cup \Gamma^e, \quad \bar{\Omega}^e \neq \emptyset \quad (8.61)$$

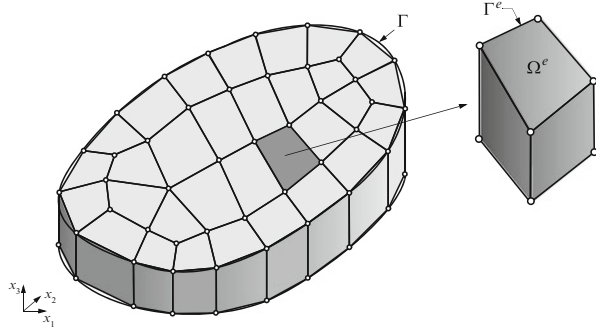
where  $N_E$  is the number of finite elements,  $\hat{\hat{\Omega}}$  is the approximate global domain,  $\Omega^e$  and  $\Gamma^e$  are the domain and the boundary of each finite element  $e$ , respectively. The *basic idea* of the finite element spatial discretization (8.61) is to split any integral that appears in the weak statements into a sum over the elements

$$\begin{aligned} \int_{\Omega} \{ \dots \} d\Omega &= \sum_{e=1}^{N_E} \int_{\Omega^e} \{ \dots \} d\Omega^e \\ \int_{\Gamma} \{ \dots \} d\Gamma &= \sum_{e=1}^{N_E} \int_{\Gamma^e} \{ \dots \} d\Gamma^e \end{aligned} \quad (8.62)$$

This nonoverlapping sum over all elements is called *assembly*. The actual domain  $\hat{\hat{\Omega}}$  assembled by all these elements  $\bigcup_e (\Omega^e \cup \Gamma^e)$  is termed *finite element mesh* (Fig. 8.5). The goal of the assembly (8.62) is to accomplish an easily tractable, sufficiently accurate and efficient integration on element level. This can be attained



**Fig. 8.5** Spatial discretization of a continuum domain with its boundary  $\bar{\Omega} = \Omega \cup \Gamma$  by finite elements  $\bar{\Omega}^e = \Omega^e \cup \Gamma^e$ , ( $e = 1, \dots, N_E$ ) forming a finite element mesh

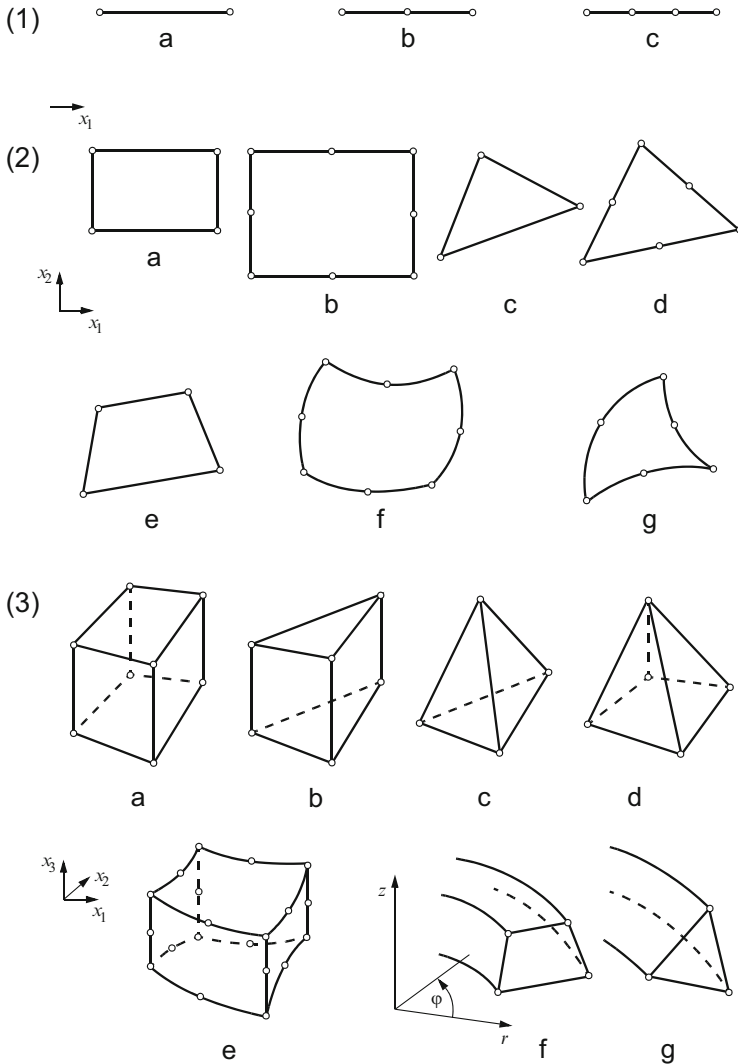


by choosing suitable shapes for the finite element  $\bar{\Omega}^e$  that have appropriate geometric entities (vertices, mid-sides) to match the interpolation for the approximate solution  $\hat{\phi}$  according to (8.16) with a desired accuracy. The finite element  $\bar{\Omega}^e$  can be a line, triangle or quadrilateral in 1D, 2D or 3D, respectively, and the degree of interpolation over it can be linear, quadratic or even higher. In practice, the phrase *finite element* refers to both the geometry of the element and degree of approximation used for the solution variable(s), e.g., a quadratic quadrilateral element is a 2D quadrilateral shape with a biquadratic (biparabolic) interpolation, a linear triangular prismatic element represents a 3D pentahedral shape with trilinear interpolation, and so forth. Commonly used finite elements in 1D, 2D and 3D are depicted in Fig. 8.6.

## 8.7 Elementwise Continuous Approximations

The assembly (8.62) of the finite elements is only valid if the basis (interpolation) functions (8.16) satisfy requirements on continuity. The basis functions have to be restricted to avoid any infinite terms in the integrals of the approximate weak statement. The situation is explained in Fig. 8.7. Let us consider the interfacing boundary of two adjacent finite elements, where we study the approximate function  $\hat{\phi}$  and its derivatives in a very small distance  $\delta \rightarrow 0$ . Within the elementwise interpolation procedure we can ensure that  $\hat{\phi}$  is continuous everywhere in  $\hat{\Omega}$  and also at the element interface(s). However, this must not be the case for the first derivative anymore, which can become discontinuous at element interfaces. While the first derivative is discontinuous, its value remains in a finite value and any integrand of the weak form containing up to a first-order derivative is finite and accordingly evaluable. In contrast, however, consider its second derivative, which tends to an infinite value at the element interface. Such a term is no more square integrable and the assembly (8.62) fails.

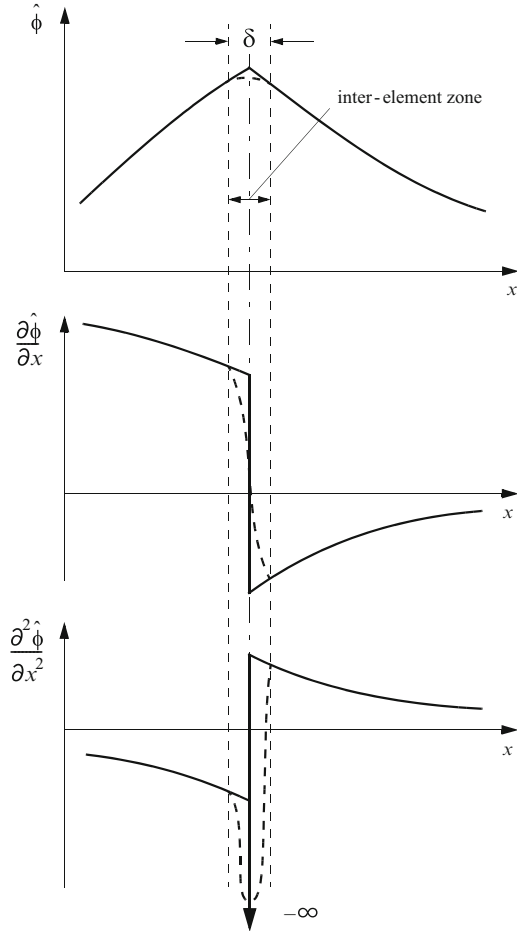
The continuity requirement can be generalized as follows. Suppose the integrand in the approximate weak statement contains up to  $(m + 1)$ th derivatives, then



**Fig. 8.6** Overview of commonly used finite elements. (1) 1D elements: (a) linear, (b) quadratic, (c) cubic; (2) 2D elements: (a) linear rectangular, (b) quadratic rectangular, (c) linear triangular, (d) quadratic triangular, (e) linear quadrilateral, (f) quadratic curved quadrilateral, (g) quadratic curved triangular; (3) 3D elements: (a) linear quadrilateral prism (hexahedron), (b) linear triangular prism (pentahedron), (c) linear tetrahedron, (d) linear pyramid, (e) quadratic curved hexahedron, (f) axisymmetric linear rectangular ring, (g) axisymmetric linear triangular ring (Modified from [76])

continuity in the  $m$ th derivative of the approximate function must be satisfied. This is called the  $C_m$ -continuity requirement. The validity of the assembly (8.62) requires the fulfillment of the  $C_m$ -continuity in any finite element basis function.

**Fig. 8.7** Inter-element behavior of  $C_0$  continuous approximate function  $\hat{\phi}$  and its derivatives: While  $\hat{\phi}$  is continuous at the element interface, its first derivative becomes discontinuous within the inter-element zone  $\delta \rightarrow 0$ , but is still finite. The second derivative, however, may become infinite (Modified from [590])



Now, having a look to the weak forms as derived in Sect. 8.5, we recognize that the highest derivatives are only of first order (thanks to the reduction of the 2nd-order derivatives in the dispersion term due to applying the Gauss’s integral theorem). Hence, it is sufficient to satisfy only  $C_0$ –continuity in the interpolation function(s) of the unknown variable(s), i.e., the element basis functions  $\hat{\phi}$  have to be chosen in such a way that the zero derivatives are continuous and their first derivatives, while discontinuous at the element interfaces (they actually suffer jumps at nodal points), need only to be square integrable.

The most important class of  $C_0$  basis functions refers to *Lagrangian polynomials*, which are standard in FEM.  $C_0$  functions are commonly sufficient for all problems of advection-dispersion type, which are encountered in the present flow and transport processes. On the other hand, a higher order continuity, e.g.,  $C_1$  functions satisfying continuity of both zero and first derivatives, can be provided by Hermitian polynomials [173, 280]. Although  $C_1$  Hermitian polynomials can achieve a higher

accuracy for the first derivatives, however, at the expense of additional degrees of freedom associated with computational extra costs, their practical applicability has shown limited (e.g., to undistorted elements) and rather cumbersome. Indeed, we need not a continuity higher than  $C_0$ . In the following we exclusively prefer  $C_0$  continuous basis functions for various element types in 1D, 2D and 3D.

## 8.8 Finite Element Basis Functions

### 8.8.1 Shape Function, Master Element and Isoparametric Element Type

In using assembly (8.62) it is advantageous to restrict the interpolation of the unknown variable(s) within each finite element  $\bar{\Omega}^e = \Omega^e \cup \Gamma^e$ , such that the approximation  $\hat{\phi}(\mathbf{x}, t)$  according to (8.16) can then be formed as the *union* of the finite element approximations  $\hat{\phi}^e(\mathbf{x}^e, t)$  on  $\bar{\Omega}^e$ , viz.,

$$\phi(\mathbf{x}, t) \approx \hat{\phi}(\mathbf{x}, t) = \bigcup_{e=1}^{N_E} \hat{\phi}^e(\mathbf{x}^e, t) \quad (8.63)$$

Note that it is not possible to simply sum  $\hat{\phi}^e$  over  $e$  since a double contribution would occur on every finite element boundary. Thus, a summation without overlap of element boundary will be indicated by the union symbol:

$$\bigcup_{e=1}^{N_E} (\dots) = \sum_{e=1}^{N_E} (\dots) \quad \text{without boundary overlap} \quad (8.64)$$

On any finite element domain  $\bar{\Omega}^e$ , the generic form for  $\hat{\phi}^e$  is

$$\hat{\phi}^e(\mathbf{x}^e, t) = \sum_{J=1}^{N_{BN}} N_J^e(\mathbf{x}^e) \phi_J^e(t) \quad (8.65)$$

where  $\phi_J^e$  are the set of unknown coefficients at the nodes  $J$  belonging to the element  $e$  and  $N_J^e(\mathbf{x}^e)$  are the set of given  $C_0$  continuous basis functions, called *shape functions*, associated with the element  $e$  and the *local* node number  $J$  (note that we shall differ between local and global node numbering as further discussed below). The element shape functions  $N_J^e(\mathbf{x}^e)$  represent polynomials of 1st, 2nd or even higher degree. In practice, however, we prefer polynomials of 1st degree and, optionally, 2nd degree. There are as many of these polynomials as there are nodal points  $N_{BN}$  in  $\bar{\Omega}^e$ . To achieve a continuous representation of  $\hat{\phi}$  (cf. Sect. 8.7) the element shape functions must satisfy  $C_0$ -continuity, for which the approximate solution is continuous and have piecewise continuous first-order

derivatives. Those element shape functions are referred to as  $C_0$ -class elements, which will be generally used in the following.

The element shape functions have the following property at the nodal points:

$$N_J^e(\mathbf{x}_I^e) = \begin{cases} \delta_{IJ} & \text{for } \mathbf{x}_I^e \in \bar{\Omega}^e \\ 0 & \text{otherwise} \end{cases} \quad (8.66)$$

where  $\delta_{IJ}$  is the Kronecker symbol (2.7) and  $\mathbf{x}_I^e$  are the Cartesian coordinates of local node  $I$  (cf. (2.30)). From (8.66) it directly follows that

$$\sum_{J=1}^{N_{\text{BN}}} N_J^e(\mathbf{x}^e) = 1, \quad \forall \mathbf{x}^e \in \bar{\Omega}^e \quad (8.67)$$

The ability of handling nonuniform and distorted geometries is an important feature of the FEM. A fundamental aspect of FEM is the use of a *master element*  $\bar{\Omega}_m^e = \Omega_m^e \cup \Gamma_m^e$ , where all element-related inner products and integrations are performed in local coordinates  $\boldsymbol{\eta}$  defined as

$$\boldsymbol{\eta}^T = \begin{cases} (\xi \ \eta \ \zeta) & \text{3D} \\ (\xi \ \eta) & \text{2D and axisymmetric} \\ (\xi) & \text{1D} \end{cases} \quad (8.68)$$

A one-to-one mapping (coordinate transformation, see Sect. 2.1.5) bridges the global Euclidean  $\mathbf{x}$ -space and the local (computational)  $\boldsymbol{\eta}$ -space of the master element  $\bar{\Omega}_m^e$ :

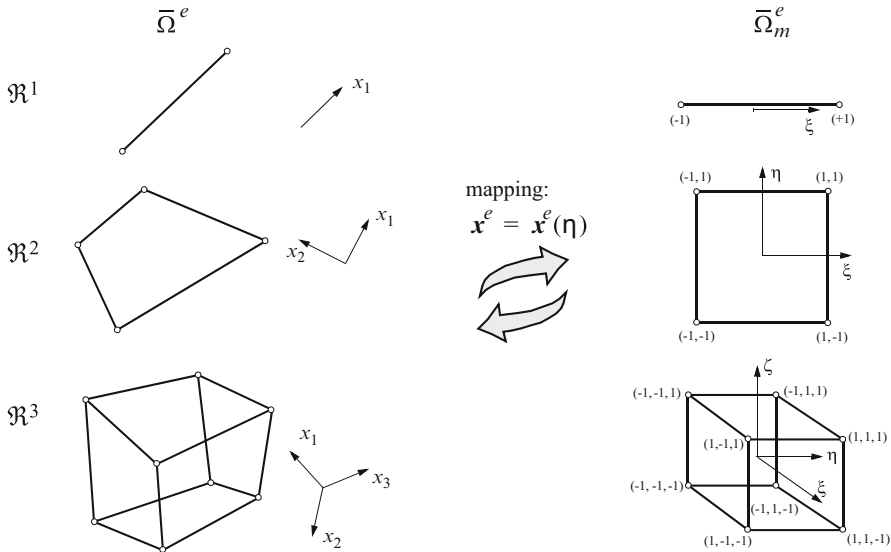
$$\mathbf{x}^e = \mathbf{x}^e(\boldsymbol{\eta}) \quad (8.69)$$

The element geometry of the master element  $\bar{\Omega}_m^e$  is always Cartesian (rectangular) so that the integration on such an element level can be efficiently computed. Based on this mapping the finite elements can be distorted easily to fit most applicable geometries (Fig. 8.8). For this purpose it is advantageous to define the element shape functions in their local coordinates  $N_J^e(\boldsymbol{\eta})$ , such that (8.65) becomes

$$\hat{\phi}^e(\mathbf{x}^e(\boldsymbol{\eta}), t) = \sum_{J=1}^{N_{\text{BN}}} N_J^e(\boldsymbol{\eta}) \phi_J^e(t) \quad (8.70)$$

and global coordinates  $\mathbf{x}$  are related to the local coordinates  $\boldsymbol{\eta}$  by using the interpolation

$$\mathbf{x}^e = \sum_{J=1}^{N_X} N_J^e(\boldsymbol{\eta}) \mathbf{x}_J^e \quad (8.71)$$



**Fig. 8.8** Finite elements with one-to-one mapping onto  $\mathfrak{R}^D$  ( $D = 1, 2, 3$ )

with

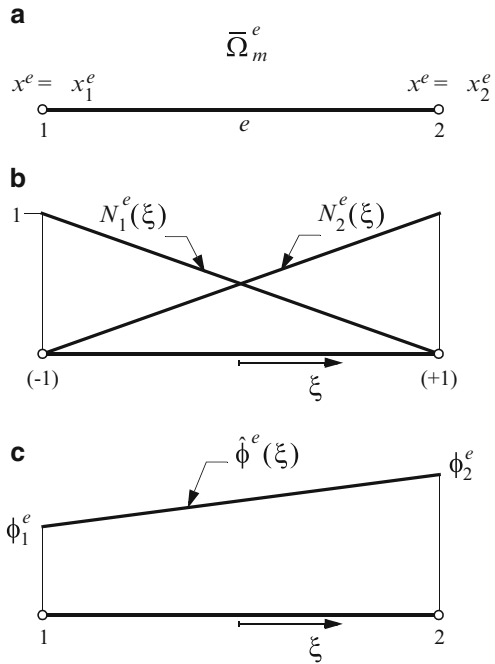
$$\mathbf{x} = \bigcup_{e=1}^{N_E} \mathbf{x}^e \tag{8.72}$$

where  $N_X$  is the number of element polynomials used for the geometry interpolation and  $\mathbf{x}_j^e$  are the global coordinates of node  $j$  on element  $e$ . According to the choice of  $N_X$  it is distinguished into (1) *isoparametric elements* with  $N_X = N_{BN}$ , i.e., polynomial approximation is used for both geometry and variables, (2) *superparametric elements* with  $N_X > N_{BN}$ , where a higher order approximation is used for the geometry, and (3) *subparametric elements* with  $N_X < N_{BN}$ , where a lower order approximation is used for the geometry compared to the variable approximation. Most efficient and ideal for our needs are isoparametric elements, which will be generally preferred in the present FEM. Appendix G summarizes the isoparametric finite elements used in FEFLOW for 1D, 2D (incl. axisymmetric) and 3D problems.

### 8.8.2 Local and Global Shape Functions

To illustrate the construction of finite element basis functions let us consider at first the simplest case: the use of linear isoparametric shape functions in a 1D geometry  $x \in \mathfrak{R}^1$  (see also Table G.1 in Appendix G). Figure 8.9 displays the master element  $\bar{\Omega}_m^e$  with the local node numbering  $J = 1, 2$ , the linear shape functions expressed

**Fig. 8.9** Piecewise-linear shape functions for 1D element: **(a)** master element  $\bar{\Omega}_m^e$  with local node numbering, **(b)** shape functions  $N_J^e(\xi)$  ( $J = 1, 2$ ) in local coordinate  $-1 \leq \xi \leq 1$ , **(c)** approximate variable  $\hat{\phi}^e(\xi)$  as linear function over element  $e$



in the local coordinate  $(-1 \leq \xi \leq 1)$

$$N_1^e(\xi) = \frac{1}{2}(1 - \xi) \quad N_2^e(\xi) = \frac{1}{2}(1 + \xi) \quad (8.73)$$

and the resulting approximate function  $\hat{\phi}^e$  over the element  $e$ . Using the mapping relation (8.71) for the linear 2-node element

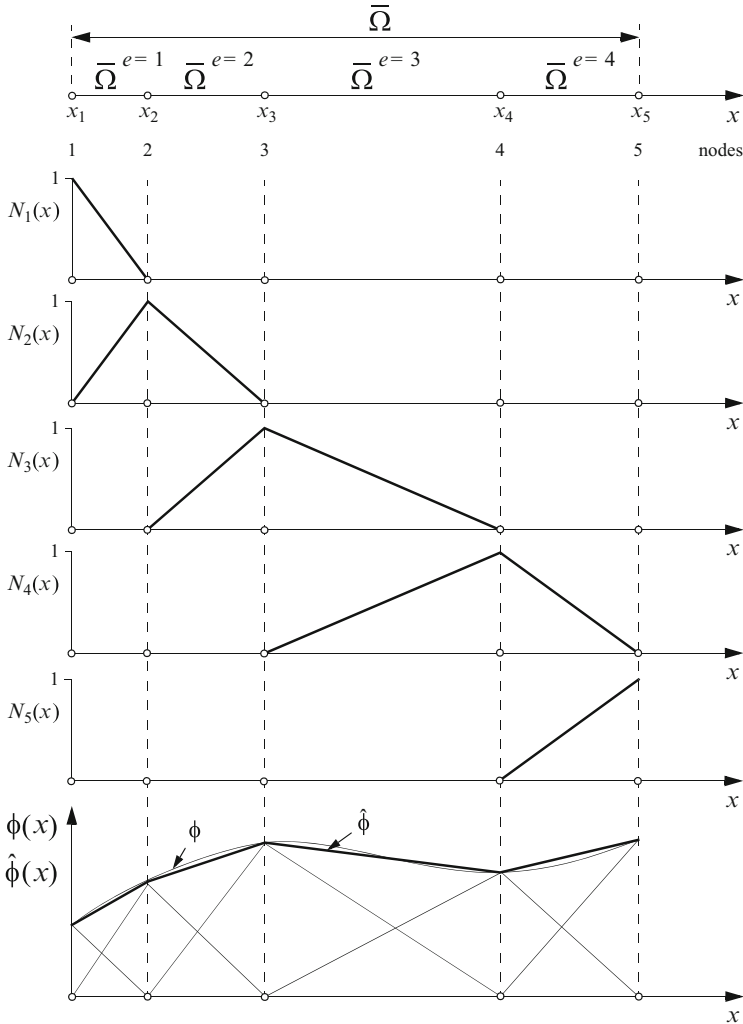
$$x^e = \sum_{J=1}^{N_{BN}=2} N_J^e(\xi) x_J^e \quad (8.74)$$

we find  $\xi = (2x^e - x_1^e - x_2^e)/(x_2^e - x_1^e)$  and the shape functions can also be written in the global coordinate  $x^e$ , viz.,

$$N_1^e(x^e) = \frac{x_2^e - x^e}{x_2^e - x_1^e} \quad N_2^e(x^e) = \frac{x^e - x_1^e}{x_2^e - x_1^e} \quad (8.75)$$

where  $x_1^e$  and  $x_2^e$  are the  $x$ -coordinates of local node number 1 and 2, respectively, of element  $e$ . Then, the approximate variable  $\hat{\phi}^e$  is linear over element  $e$  (Fig. 8.9c):

$$\begin{aligned} \hat{\phi}^e(\xi) &= \frac{1}{2}[(\phi_2^e - \phi_1^e)\xi + \phi_1^e + \phi_2^e] \quad \text{or} \\ \hat{\phi}^e(x^e) &= \frac{1}{x_2^e - x_1^e}[(\phi_2^e - \phi_1^e)x^e + x_2^e\phi_1^e - x_1^e\phi_2^e] \end{aligned} \quad (8.76)$$



**Fig. 8.10** Example of 1D finite element mesh consisting of four linear elements. Display of global basis function  $N_j(x)$ , ( $j = 1, \dots, 5$ ) and linear approximation of variable  $\phi$  by  $\hat{\phi}$

Now, let us consider a 1D mesh consisting of four linear elements and five global nodes as shown in Fig. 8.10. The approximate function  $\hat{\phi}$  is represented using global shape functions  $N_j$  that are equal to one at node  $j$  and zero at all other nodes. Accordingly, the global function  $\hat{\phi}$  in the mesh shown in Fig. 8.10 can be written as

$$\hat{\phi}(x, t) = \sum_{j=1}^{N_p=5} N_j(x)\phi_j(t) = \bigcup_{e=1}^{N_E=4} \hat{\phi}^e(x, t) \tag{8.77}$$



with

$$\hat{\phi}^e(x, t) = \sum_{J=1}^{N_{\text{BN}}=2} N_J^e(\xi) \phi_J^e \tag{8.78}$$

It can be observed that, as  $j$  and  $J$  vary, all elements contain shape functions, which are similar in global coordinates  $x$  and identical in the local coordinates  $\xi$ . This is a key issue for devising efficient operations on the master element level, which are generic and will be performed widely independent of the global (physical) coordinates as discussed further below.

Now, we have to emphasize the difference between the global node numbering used in  $N_j$  and the local node numbering used in  $N_J^e$  and to consider how we can relate properties between the global and the local systems: *Any uppercase nodal index  $J$  associated with an element-rank quantity represents a local node number, while a lowercase nodal index  $j$  of a quantity without element rank means a global node number.* For the mesh of Fig. 8.10 it is seen that

$$\begin{aligned} N_1^1(x^1) &= N_1(x), & N_2^1(x^1) &= N_2(x) & \text{over element} & e = 1 \\ N_1^2(x^2) &= N_2(x), & N_2^2(x^2) &= N_3(x) & \text{over element} & e = 2 \\ N_1^3(x^3) &= N_3(x), & N_2^3(x^3) &= N_4(x) & \text{over element} & e = 3 \\ N_1^4(x^4) &= N_4(x), & N_2^4(x^4) &= N_5(x) & \text{over element} & e = 4 \end{aligned} \tag{8.79}$$

which can be written in a matrix form as follows

$$\begin{pmatrix} N_1 \\ N_2 \end{pmatrix}^e = \Delta^e \cdot \begin{pmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \end{pmatrix}, \quad e = 1, 2, \dots, N_E \tag{8.80}$$

or more generally

$$N_J^e = \sum_{j=1}^{N_p} \Delta_{Jj}^e N_j, \quad J = 1, 2, \dots, N_{\text{BN}} \tag{8.81}$$

where  $\Delta^e = \Delta_{Jj}^e$  is the *Boolean matrix* of element  $e$  having the property:

$$\Delta_{Jj}^e = \begin{cases} 1 & \text{if the local node } J \text{ corresponds to the global node } j \\ 0 & \text{otherwise} \end{cases} \tag{8.82}$$

The Boolean matrix  $\Delta^e$  will prove to be convenient in derivations of finite element equations, where local quantities have to be related to properties of the global

coordinate system. The Boolean matrices for the present mesh of Fig. 8.10 result for example:

$$\Delta^1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}, \quad \Delta^2 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}, \quad \Delta^3 = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}, \quad \Delta^4 = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad (8.83)$$

Using the Boolean matrix (8.82), the global shape functions  $N_j(\mathbf{x})$  appearing in the global approximate solution

$$\hat{\phi}(\mathbf{x}, t) = \sum_{j=1}^{N_p} N_j(\mathbf{x}) \phi_j(t) \quad (8.84)$$

can be directly expressed by local shape functions  $N_j^e$  according to

$$N_j(\mathbf{x}) = \bigcup_{e=1}^{N_E} \left( \sum_{J=1}^{N_{BN}} N_J^e(\boldsymbol{\eta}) \Delta_{Jj}^e \right) \quad (8.85)$$

To increase the accuracy of interpolation a quadratic shape function rather than a linear shape function can be chosen. Quadratic interpolation functions are generated by adding an additional node at the midside of each element as shown in Fig. 8.11 for 1D geometry. The shape functions for this quadratic 3-node element are (cf. also Table G.1 in Appendix G)

$$N_1^e(\xi) = \frac{1}{2}\xi(\xi - 1) \quad N_2^e(\xi) = 1 - \xi^2 \quad N_3^e(\xi) = \frac{1}{2}\xi(\xi + 1) \quad (8.86)$$

Apart from the different polynomials and the number of polynomials (= number of nodes  $N_{BN}$ ) per master element appearing for the quadratic element type, the construction of the basis function is based on the same principles as stated above for the linear element. The same is also true for isoparametric elements in higher dimensions (see Tabs. G.2–G.4 in Appendix G for the family of 2D and 3D elements used in FEFLOW). An example of a 2D triangle mesh for a piecewise bilinear approximation of  $\hat{\phi}$  is shown in Fig. 8.12. The shape functions of each triangle for the three nodes written in the local coordinates ( $0 \leq \xi, \eta \leq 1$ ) are

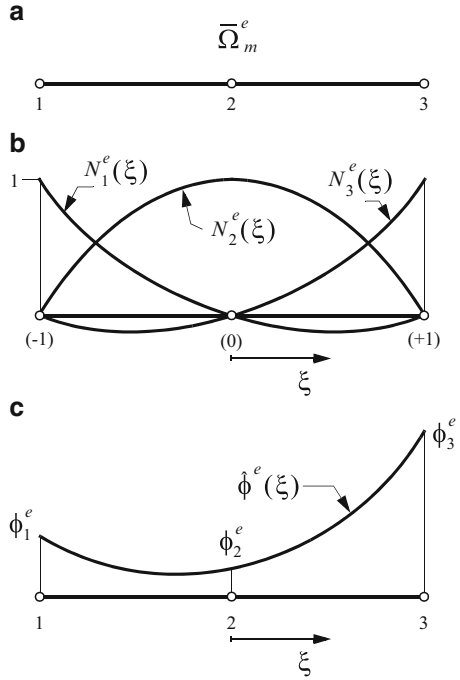
$$N_1^e(\xi, \eta) = 1 - \xi - \eta \quad N_2^e(\xi, \eta) = \xi \quad N_3^e(\xi, \eta) = \eta \quad (8.87)$$

Furthermore, using the mapping relation (8.71) as

$$x^e = \sum_{J=1}^{N_{BN}=3} N_J^e(\xi, \eta) x_J^e \quad y^e = \sum_{J=1}^{N_{BN}=3} N_J^e(\xi, \eta) y_J^e \quad (8.88)$$

written with (8.87) as

**Fig. 8.11** Piecewise-quadratic shape functions for 1D element: **(a)** master element  $\bar{\Omega}_m^e$  with local node numbering, **(b)** shape functions  $N_J^e(\xi)$  ( $J = 1, 2, 3$ ) in local coordinate  $-1 \leq \xi \leq 1$ , **(c)** approximate variable  $\hat{\phi}^e(\xi)$  as quadratic function over element  $e$



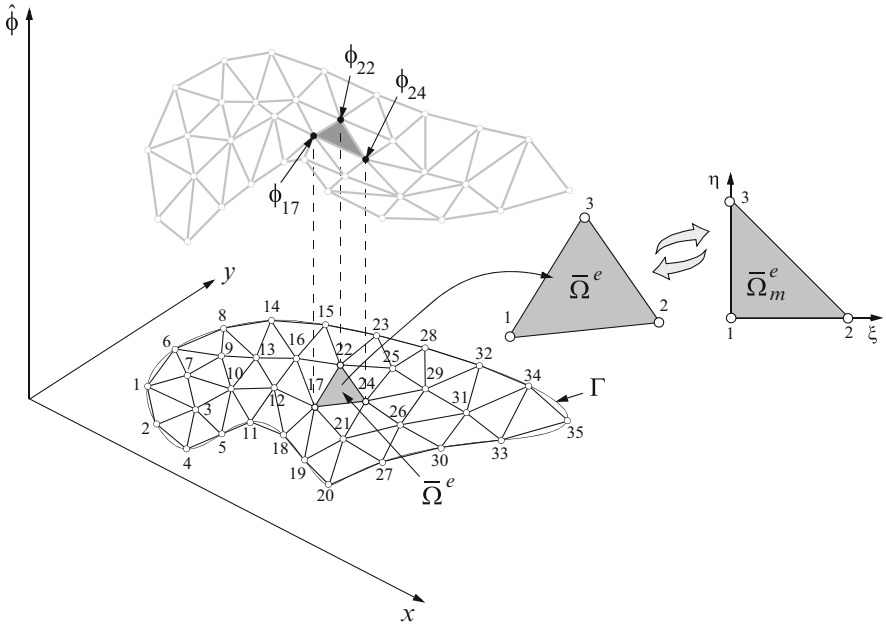
$$\begin{aligned} \begin{pmatrix} x \\ y \end{pmatrix}^e &= \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}^e + \begin{pmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{pmatrix}^e \cdot \begin{pmatrix} \xi \\ \eta \end{pmatrix} \\ &= \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}^e + \underbrace{\begin{pmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{pmatrix}^e}_{\mathbf{J}^e} \cdot \begin{pmatrix} \xi \\ \eta \end{pmatrix} \end{aligned} \tag{8.89}$$

we can express the local coordinates as

$$\begin{aligned} \begin{pmatrix} \xi \\ \eta \end{pmatrix} &= \frac{1}{|\mathbf{J}^e|} \begin{pmatrix} J_{22} & -J_{12} \\ -J_{21} & J_{11} \end{pmatrix}^e \cdot \begin{pmatrix} x - x_1 \\ y - y_1 \end{pmatrix}^e \\ &= \frac{1}{|\mathbf{J}^e|} \begin{pmatrix} y_3 - y_1 & x_1 - x_3 \\ y_1 - y_2 & x_2 - x_1 \end{pmatrix}^e \cdot \begin{pmatrix} x - x_1 \\ y - y_1 \end{pmatrix}^e \end{aligned} \tag{8.90}$$

and find finally the shape functions of the linear triangle in global coordinates according to

$$\begin{aligned} N_1^e(x^e, y^e) &= \frac{1}{|\mathbf{J}^e|} [x_2 y_3 - x_3 y_2 + (y_2 - y_3)x + (x_3 - x_2)y]^e \\ N_2^e(x^e, y^e) &= \frac{1}{|\mathbf{J}^e|} [x_3 y_1 - x_1 y_3 + (y_3 - y_1)x + (x_1 - x_3)y]^e \\ N_3^e(x^e, y^e) &= \frac{1}{|\mathbf{J}^e|} [x_1 y_2 - x_2 y_1 + (y_1 - y_2)x + (x_2 - x_1)y]^e \end{aligned} \tag{8.91}$$



**Fig. 8.12** 2D triangular element mesh with global node numbering and a piecewise linear approximation of  $\hat{\phi}$ . Selected triangular element  $\bar{\Omega}^e \subset \mathbb{R}^2$  with local node numbering and mapping onto master element  $\bar{\Omega}_m^e$  in local coordinates  $\xi$  and  $\eta$

where the determinant of the Jacobian  $\mathbf{J}^e$  (equal to twice the area of triangle) is given by

$$|\mathbf{J}^e| = [J_{11}J_{22} - J_{21}J_{12}]^e = [x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)]^e \quad (8.92)$$

Then, the approximate variable  $\hat{\phi}$  forms a piecewise-bilinear function over the solution domain as exemplified in Fig. 8.12

$$\hat{\phi}(x, y, t) = \sum_j N_j(x, y) \phi_j(t) \quad (8.93)$$

with  $N_j(x, y) = \bigcup_e (\sum_J N_j^e(\xi, \eta) \Delta_{J_j}^e)$ .

In the same way we are able to construct finite element basis functions for all element types we have in mind for 1D, 2D and 3D applications. The family of finite elements preferred in FEFLOW are summarized in Appendix G. While the finite element basis functions  $N_j^e(\boldsymbol{\eta})$  are expressed in analytical forms, the required coordinate transformation (mapping) between  $\bar{\Omega}^e$  in the global coordinate system and  $\bar{\Omega}_m^e$  in the local coordinate system represents a generic task. It can be performed by very efficient basis operations for each element, which will be thoroughly described next.

## 8.9 Galerkin Finite Element Weak Statement

Using the weak forms derived for the divergence and convective form of ADE according to (8.48) and (8.55), respectively, their weak statements (8.36) applied to the approximate variable  $\hat{\phi}$  result

$$\begin{aligned} \text{WS} = & \int_{\Omega} w_i \frac{\partial(\mathcal{R}\hat{\phi})}{\partial t} d\Omega - \int_{\Omega} \hat{\phi} \mathbf{q} \cdot \nabla w_i d\Omega + \int_{\Omega} \nabla w_i \cdot (\mathbf{D} \cdot \nabla \hat{\phi}) d\Omega + \\ & \int_{\Omega} w_i (\vartheta \hat{\phi} - H) d\Omega + \phi_w Q_w(t)|_i + \int_{\Gamma_N} w_i q_N^\dagger d\Gamma - \\ & \int_{\Gamma_C} w_i \Phi^\dagger (\phi_C - \hat{\phi}) d\Gamma = 0 \quad \forall w_i \in H_0^1(\Omega), \quad 1 \leq i \leq N_{\text{EQ}} \end{aligned} \quad (8.94)$$

for the divergence form of ADE and

$$\begin{aligned} \text{WS} = & \int_{\Omega} w_i \mathcal{R} \frac{\partial \hat{\phi}}{\partial t} d\Omega + \int_{\Omega} w_i \mathbf{q} \cdot \nabla \hat{\phi} d\Omega + \int_{\Omega} \nabla w_i \cdot (\mathbf{D} \cdot \nabla \hat{\phi}) d\Omega + \\ & \int_{\Omega} w_i [(\vartheta + Q)\hat{\phi} - H] d\Omega + (\phi_w - \phi(\mathbf{x}_w)) Q_w(t)|_i + \int_{\Gamma_N} w_i q_N d\Gamma - \\ & \int_{\Gamma_C} w_i \Phi (\phi_C - \hat{\phi}) d\Gamma = 0 \quad \forall w_i \in H_0^1(\Omega), \quad 1 \leq i \leq N_{\text{EQ}} \end{aligned} \quad (8.95)$$

for the convective form of ADE.

Now, we discretize the domain  $\Omega$  and its boundary  $\Gamma$  by finite elements via (8.62), introduce the semidiscrete finite element basis function for  $\hat{\phi} = \hat{\phi}(\mathbf{x}, t)$  over each element  $e$

$$\begin{aligned} \hat{\phi}(\mathbf{x}, t) = & \sum_{j=1}^{N_p} N_j(\mathbf{x}) \phi_j(t) = \bigcup_{e=1}^{N_E} \hat{\phi}^e(\mathbf{x}^e, t) \\ & \hat{\phi}^e(\mathbf{x}^e, t) = \sum_{j=1}^{N_p} \sum_{J=1}^{N_{\text{BN}}} N_J^e(\boldsymbol{\eta}) \Delta_{Jj}^e \phi_j(t) \end{aligned} \quad (8.96)$$

and choose the Galerkin method (Table 8.1), where the weighting function becomes identical to the basis function<sup>5</sup>

<sup>5</sup>The weak statements (8.94) and (8.95) imply that the weighting functions  $w_i$  belong to the  $H_0^1$  functional space (8.23). On the other hand, the basis functions  $N_i$  belong to the  $H^1$  functional space, i.e., they do not vanish on Dirichlet boundaries:  $N_i \neq 0$  on  $\Gamma_D$ . Nevertheless, we may use WS in form of (8.94) and (8.95) with  $w_i = N_i \in H^1(\Omega)$ ,  $1 \leq i \leq N_p$ , where we enforce at first a zero flux  $(\phi \mathbf{q} - \mathbf{D} \cdot \nabla \phi) \cdot \mathbf{n} \approx 0$  or  $-(\mathbf{D} \cdot \nabla \phi) \cdot \mathbf{n} \approx 0$  on  $\Gamma_D$  in the original weak statements (8.47) and (8.54), respectively, and incorporate the actual Dirichlet (essential) BC's afterwards via a direct manipulation of the resulting discrete matrix system as further discussed in Sect. 8.16.

$$w_i(\mathbf{x}) = N_i(\mathbf{x}) = \bigcup_{e=1}^{N_E} \left( \sum_{J=1}^{N_{BN}} N_J^e(\boldsymbol{\eta}) \Delta_{li}^e \right) \quad (8.97)$$

we find the following finite element forms of the Galerkin weak statement (GWS)

$$\begin{aligned} \text{GWS} = & \sum_e \int_{\Omega^e} N_i \frac{\partial}{\partial t} [\mathcal{R}(\sum_j N_j \phi_j)] d\Omega^e - \sum_e \int_{\Omega^e} (\sum_j N_j \phi_j) \mathbf{q} \cdot \nabla N_i d\Omega^e + \\ & \sum_e \int_{\Omega^e} \nabla N_i \cdot [\mathbf{D} \cdot \nabla (\sum_j N_j \phi_j)] d\Omega^e + \sum_e \int_{\Omega^e} N_i [\vartheta (\sum_j N_j \phi_j) - H] d\Omega^e + \\ & \phi_w \mathcal{Q}_w(t)|_i + \sum_e \int_{\Gamma_N^e} N_i q_N^\dagger d\Gamma^e - \sum_e \int_{\Gamma_C^e} N_i \Phi^\dagger [\phi_C - (\sum_j N_j \phi_j)] d\Gamma^e = 0 \\ & 1 \leq i, j \leq N_P \end{aligned} \quad (8.98)$$

for the divergence form of ADE and

$$\begin{aligned} \text{GWS} = & \sum_e \int_{\Omega^e} N_i \mathcal{R} \frac{\partial}{\partial t} (\sum_j N_j \phi_j) d\Omega^e + \sum_e \int_{\Omega^e} N_i \mathbf{q} \cdot \nabla (\sum_j N_j \phi_j) d\Omega^e + \\ & \sum_e \int_{\Omega^e} \nabla N_i \cdot [\mathbf{D} \cdot \nabla (\sum_j N_j \phi_j)] d\Omega^e + \sum_e \int_{\Omega^e} N_i [(\vartheta + \mathcal{Q})(\sum_j N_j \phi_j) - H] d\Omega^e + \\ & (\phi_w - \phi_i) \mathcal{Q}_w(t)|_i + \sum_e \int_{\Gamma_N^e} N_i q_N d\Gamma^e - \\ & \sum_e \int_{\Gamma_C^e} N_i \Phi [\phi_C - (\sum_j N_j \phi_j)] d\Gamma^e = 0 \quad 1 \leq i, j \leq N_P \end{aligned} \quad (8.99)$$

for the convective form of ADE. The indicated integrals in (8.98) and (8.99) are evaluated at the element level  $e$  and *assembled* (summed up) into a global matrix system of the form

$$\mathbf{O} \cdot \dot{\boldsymbol{\phi}} + \mathbf{K} \cdot \boldsymbol{\phi} - \mathbf{F} = \mathbf{0} \quad (8.100)$$

The assembly process in forming the global matrices and vectors from element contributions will be described more in detail in Sect. 8.10. In (8.100)  $\boldsymbol{\phi} = \boldsymbol{\phi}(t)$  is a column vector of the state-variable approximation coefficients

$$\boldsymbol{\phi} = \boldsymbol{\phi}_j = \begin{pmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_{N_P} \end{pmatrix} \quad (8.101)$$

to be solved as unknowns from the resulting equation system (8.100). The superposed dot in (8.100) means differentiation with respect to time  $t$

$$\dot{\boldsymbol{\phi}} = \frac{d}{dt}\boldsymbol{\phi}(t) = \frac{d}{dt}\phi_j(t) = \begin{pmatrix} \frac{d\phi_1}{dt} \\ \frac{d\phi_2}{dt} \\ \vdots \\ \frac{d\phi_{N_P}}{dt} \end{pmatrix} \quad (8.102)$$

The components of the global rank square matrices  $\mathbf{O}$  and  $\mathbf{K}$  as well as the global column vector  $\mathbf{F}$  are written in indicial notation as<sup>6</sup>

<sup>6</sup>We can alternatively write the matrices and vectors in using directly the global shape function (8.85) with the global node numbers  $i, j$ :

$$\begin{aligned} O_{ij} &= \begin{cases} \sum_e \int_{\Omega^e} \mathcal{R}^e N_i N_j d\Omega^e & \text{divergence form} \\ \sum_e \int_{\Omega^e} \dot{\mathcal{R}}^e N_i N_j d\Omega^e & \text{convective form} \end{cases} \\ A_{ij} &= \begin{cases} -\sum_e \int_{\Omega^e} \mathbf{q}^e \cdot \nabla N_i N_j d\Omega^e & \text{divergence form}^7 \\ \sum_e \int_{\Omega^e} N_i \mathbf{q}^e \cdot \nabla N_j d\Omega^e & \text{convective form} \end{cases} \\ C_{ij} &= \sum_e \int_{\Omega^e} \nabla N_i \cdot (\mathbf{D}^e \cdot \nabla N_j) d\Omega^e \\ R_{ij} &= \begin{cases} \sum_e \int_{\Omega^e} (\vartheta^e + \frac{\partial \mathcal{R}^e}{\partial t}) N_i N_j d\Omega^e & \text{divergence form} \\ \sum_e \int_{\Omega^e} (\vartheta^e + \mathcal{Q}^e) N_i N_j d\Omega^e - \delta_{ij} \mathcal{Q}_w(t)|_i & \text{convective form} \end{cases} \\ B_{ij} &= \begin{cases} \sum_e \left( \int_{\Gamma_C^e} \Phi^{\ddagger e} N_i N_j d\Gamma^e + \int_{\Gamma_{N_O}^e} N_i (\mathbf{q} N_j - \mathbf{D} \cdot \nabla N_j) \cdot \mathbf{n} d\Gamma^e \right) & \text{divergence form} \\ \sum_e \left( \int_{\Gamma_C^e} \Phi^e N_i N_j d\Gamma^e - \int_{\Gamma_{N_O}^e} N_i (\mathbf{D} \cdot \nabla N_j) \cdot \mathbf{n} d\Gamma^e \right) & \text{convective form} \end{cases} \\ H_i &= \begin{cases} \sum_e \left( \int_{\Gamma_C^e} N_i \Phi^{\ddagger e} \phi_C^e d\Gamma^e - \int_{\Gamma_N^e \setminus \Gamma_{N_O}^e} N_i q_N^{\ddagger e} d\Gamma^e \right) & \text{divergence form} \\ \sum_e \left( \int_{\Gamma_C^e} N_i \Phi^e \phi_C^e d\Gamma^e - \int_{\Gamma_N^e \setminus \Gamma_{N_O}^e} N_i q_N^e d\Gamma^e \right) & \text{convective form} \end{cases} \\ Q_i &= \sum_e \int_{\Omega^e} N_i H^e d\Omega^e - \phi_w \mathcal{Q}_w(t)|_i \end{aligned}$$

<sup>7</sup>Note that:

$$\begin{aligned} \mathbf{q}^e \cdot \nabla N_i N_j &= (N_i^e \mathbf{q}^e \cdot \nabla N_j^e)^T \\ \mathbf{q}^e \cdot \nabla N_i N_j &= (N_i \mathbf{q}^e \cdot \nabla N_j)^T \end{aligned}$$

$$\begin{aligned}
\mathbf{O} &= O_{ij} = \sum_e \left( \sum_I \sum_J O_{IJ}^e \Delta_{li}^e \Delta_{lj}^e \right) \\
\mathbf{K} &= \mathbf{A} + \mathbf{C} + \mathbf{R} + \mathbf{B} \\
\mathbf{A} &= A_{ij} = \sum_e \left( \sum_I \sum_J A_{IJ}^e \Delta_{li}^e \Delta_{lj}^e \right) \\
\mathbf{C} &= C_{ij} = \sum_e \left( \sum_I \sum_J C_{IJ}^e \Delta_{li}^e \Delta_{lj}^e \right) \\
\mathbf{R} &= R_{ij} = \sum_e \left( \sum_I \sum_J R_{IJ}^e \Delta_{li}^e \Delta_{lj}^e \right) \\
\mathbf{B} &= B_{ij} = \sum_e \left( \sum_I \sum_J B_{IJ}^e \Delta_{li}^e \Delta_{lj}^e \right) \\
\mathbf{F} &= \mathbf{H} + \mathbf{Q} \tag{8.103} \\
\mathbf{H} &= H_i = \sum_e \left( \sum_I H_I^e \Delta_{li}^e \right) \\
\mathbf{Q} &= Q_i = \sum_e \left( \sum_I Q_I^e \Delta_{li}^e \right)
\end{aligned}$$

with the element matrices

$$\begin{aligned}
O_{IJ}^e &= \begin{cases} \int_{\Omega^e} \mathcal{R}^e N_I^e N_J^e d\Omega^e & \text{divergence form} \\ \int_{\Omega^e} \acute{\mathcal{R}}^e N_I^e N_J^e d\Omega^e & \text{convective form} \end{cases} \\
A_{IJ}^e &= \begin{cases} - \int_{\Omega^e} \mathbf{q}^e \cdot \nabla N_I^e N_J^e d\Omega^e & \text{divergence form}^7 \\ \int_{\Omega^e} N_I^e \mathbf{q}^e \cdot \nabla N_J^e d\Omega^e & \text{convective form} \end{cases} \\
C_{IJ}^e &= \int_{\Omega^e} \nabla N_I^e \cdot (\mathbf{D}^e \cdot \nabla N_J^e) d\Omega^e \\
R_{IJ}^e &= \begin{cases} \int_{\Omega^e} (\vartheta^e + \frac{\partial \mathcal{R}^e}{\partial t}) N_I^e N_J^e d\Omega^e & \text{divergence form} \\ \int_{\Omega^e} (\vartheta^e + Q^e) N_I^e N_J^e d\Omega^e - \delta_{IJ} Q_w(t) |I & \text{convective form} \end{cases} \\
B_{IJ}^e &= \begin{cases} \int_{\Gamma_C^e} \Phi^{\dagger e} N_I^e N_J^e d\Gamma^e + \int_{\Gamma_{NO}^e} N_I^e (\mathbf{q} N_J^e - \mathbf{D} \cdot \nabla N_J^e) \cdot \mathbf{n} d\Gamma^e & \text{divergence form} \\ \int_{\Gamma_C^e} \Phi^e N_I^e N_J^e d\Gamma^e - \int_{\Gamma_{NO}^e} N_I^e (\mathbf{D} \cdot \nabla N_J^e) \cdot \mathbf{n} d\Gamma^e & \text{convective form} \end{cases} \tag{8.104}
\end{aligned}$$

and the element vectors

$$\begin{aligned}
H_I^e &= \begin{cases} \int_{\Gamma_C^e} N_I^e \Phi^{\dagger e} \phi_C^e d\Gamma^e - \int_{\Gamma_N^e \setminus \Gamma_{NO}^e} N_I^e q_N^{\dagger e} d\Gamma^e & \text{divergence form} \\ \int_{\Gamma_C^e} N_I^e \Phi^e \phi_C^e d\Gamma^e - \int_{\Gamma_N^e \setminus \Gamma_{NO}^e} N_I^e q_N^e d\Gamma^e & \text{convective form} \end{cases}
\end{aligned}$$



$$Q_I^e = \int_{\Omega^e} N_I^e H^e d\Omega^e - \phi_w Q_w(t)|_I \quad (8.105)$$

where  $(i, j = 1, \dots, N_P)$ ,  $(e = 1, \dots, N_E)$  and  $(I, J = 1, \dots, N_{BN})$ . Note that in the  $\mathbf{B}^e$  matrix we also include the implicit OBC of (8.59) and (8.60) on the specific Neumann boundary  $\Gamma_{N_O}^e \subset \Gamma_N^e$ . The local element shape functions  $N_I^e$  in the element-rank matrices and vectors of (8.104) and (8.105) are expressed in the local  $\eta$ -coordinate system. Hence, the integration will be done on element level in  $\eta$ -coordinates. We note in (8.103) and (8.104) that the advection matrix  $\mathbf{A}$  (and correspondingly  $\mathbf{A}^e$ ) as well as the boundary matrix  $\mathbf{B}$  (and correspondingly  $\mathbf{B}^e$ ) at the presence of implicit OBC are unsymmetric, while all other matrices are symmetric.

## 8.10 Assembly Process

### 8.10.1 General Procedure

The assembly process is a fundamental feature of finite element computations. Assembly represents the summation of matrix or vector contributions from element integrals to global matrices and vectors. It is mathematically expressed as follows

$$\mathbf{K} = K_{ij} = \sum_{e=1}^{N_E} \left( \sum_I \sum_J K_{IJ}^e \Delta_{ii}^e \Delta_{jj}^e \right), \quad \mathbf{F} = F_i = \sum_{e=1}^{N_E} \left( \sum_I F_I^e \Delta_{ii}^e \right) \quad (8.106)$$

exemplified for a square matrix  $\mathbf{A}$  and a column vector  $\mathbf{F}$ . In (8.106)  $i, j = 1, \dots, N_P$  are the global row and column indices,  $I, J = 1, \dots, N_{BN}$  represent local row and column indices, associated with the element matrix  $K_{IJ}^e$  and vector  $F_I^e$ , and  $\Delta_{ii}^e$  and  $\Delta_{jj}^e$  are Boolean matrices (8.82) consisting of  $N_{BN}$  rows and  $N_P$  columns, which relate the local indices  $I, J$  to the global indices  $i, j$ . However, in actual computational practice the Boolean matrices  $\Delta_{ii}^e, \Delta_{jj}^e$  will never be constructed. Instead, the relation between global and local node numbers is executed via a computer program based on a nodal correspondence table called *incidence matrix*,  $\mathbf{N} = N_{eJ}$  ( $e = 1, \dots, N_E, J = 1, \dots, N_{BN}$ ). To demonstrate this procedure, we consider an example as shown in Fig. 8.13. A 2D domain is discretized by six linear triangular elements forming a simple eight-noded finite element mesh. The element-node relations are tabulated in the incidence matrix  $\mathbf{N}$ . There is no need to be concerned with the element ordering, however, the assignment of the global node numbers for each element must be consistent and systematic. It is not crucial which first local node of a particular element is incident with one of the global nodes joining the element, however, the remaining global nodes must be counter-clockwise ordered in  $\mathbf{N}$ . This counter-clockwise ordering is consistent with

the order of numbering used for the local nodes associated with the master element. For instance, considering element 2 in Fig. 8.13: We have chosen that local node 1 is incident with the global node 2, then local nodes 2 and 3 must be incident with global nodes 4 and 3, respectively. Alternatively, we also could choose for example that the global node 3 is incident with the first local node, so that the global nodes 2 and 4 become incident with the local nodes 2 and 3, respectively. The following C-like pseudo-code explains how the global matrix  $\mathbf{K}$  and global vector  $\mathbf{F}$  are assembled from the element matrices  $\mathbf{K}^e$  and element vectors  $\mathbf{F}^e$ , respectively:

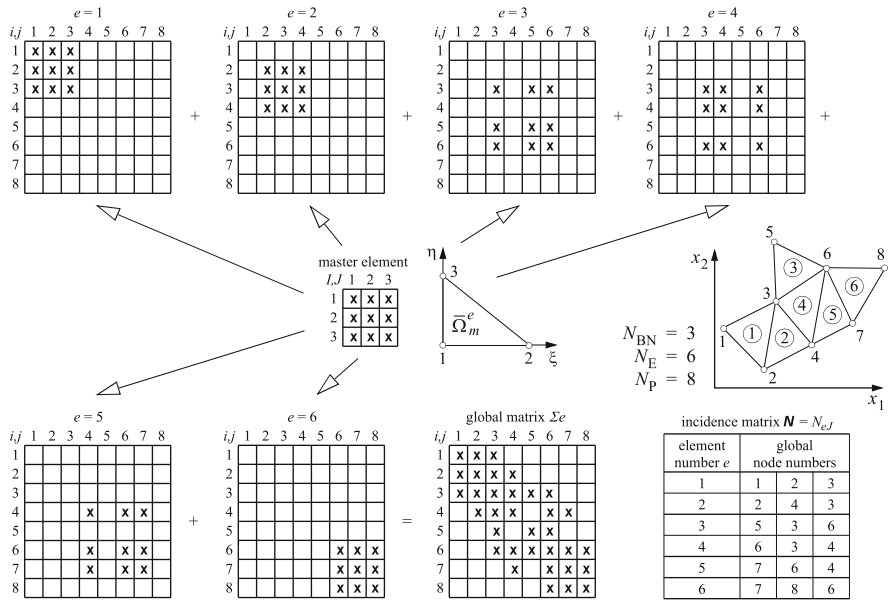
$\mathbf{K} = \mathbf{0}, \mathbf{F} = \mathbf{0}$	zeroing global matrix and vector
for ( $e = 0; e < N_E; e++$ ) {	global element loop
for ( $I = 0; I < N_{BN}; I++$ ) {	local element row loop
$i = N_{eI}$	global row index assignment
for ( $J = 0; J < N_{BN}; J++$ ) {	local element column loop
$j = N_{eJ}$	global column index assignment
$K_{ij} = K_{ij} + K_{IJ}^e$	addition of element to the global matrix
}	
$F_i = F_i + F_j^e$	addition of element to the global vector
}	
}	

(8.107)

For the example of Fig. 8.13 we obtain finally

$$\mathbf{K}_{ij} = \begin{pmatrix}
 K_{11}^1 & K_{12}^1 & K_{13}^1 & 0 \\
 K_{21}^1 & K_{22}^1 + K_{11}^2 & K_{23}^1 + K_{13}^2 & K_{12}^2 \\
 K_{31}^1 & K_{32}^1 + K_{31}^2 & K_{33}^1 + K_{33}^2 + K_{22}^3 + K_{22}^4 & \\
 0 & K_{21}^2 & K_{23}^2 + K_{32}^4 & K_{22}^2 + K_{33}^4 + K_{33}^5 \\
 0 & 0 & K_{12}^3 & 0 \\
 0 & 0 & K_{32}^3 + K_{12}^4 & K_{13}^4 + K_{23}^5 \\
 0 & 0 & 0 & K_{13}^5 \\
 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 \\
 K_{21}^3 & K_{23}^3 + K_{21}^4 & 0 & 0 \\
 0 & K_{31}^4 + K_{32}^5 & K_{31}^5 & 0 \\
 K_{11}^3 & K_{13}^3 & 0 & 0 \\
 K_{31}^3 & K_{33}^3 + K_{11}^4 + K_{22}^5 + K_{33}^6 & K_{21}^5 + K_{31}^6 & K_{32}^6 \\
 0 & K_{12}^5 + K_{13}^6 & K_{11}^5 + K_{11}^6 & K_{12}^6 \\
 0 & K_{23}^6 & K_{21}^6 & K_{22}^6
 \end{pmatrix} \quad (8.108)$$

and



**Fig. 8.13** Schematic diagram to illustrate the assembly of a global matrix from element matrices (Modified from [76])

$$F_i = \begin{pmatrix} F_1^1 \\ F_2^1 + F_1^2 \\ F_3^1 + F_3^2 + F_2^3 + F_4^4 \\ F_2^2 + F_3^4 + F_3^5 \\ F_1^3 \\ F_3^3 + F_1^4 + F_2^5 + F_3^6 \\ F_1^5 + F_1^6 \\ F_2^6 \end{pmatrix} \tag{8.109}$$

It can be seen that the resulting global matrix  $\mathbf{K}$  is sparse and banded. The sparsity structure of  $\mathbf{K}$  is an advantageous feature of the GFEM. The assembly of the global matrix requires the computation of  $N_E N_{BN}^2$  coefficients. However, it is not practicable to store the full matrix  $\mathbf{K}$ . In dependence on the preferred equations solvers (see Sect. 8.17.1) different storage management techniques have been developed. For standard iterative equation solvers only the nonzero entries of the matrix are compactly stored, which allows a very core-space saving strategy. Bookkeeping is required to localize the nonzero elements within the matrix. On the other hand, for Gaussian direct equations solvers the profile of the matrix (all elements of a row up to the most right-sided nonzero entry) must be stored, where also zero elements within the matrix profile have to be included, which is needed for the fill-in entries arising at later stages in the Gaussian forward elimination process. While for iterative solvers the order for the global numbering of the nodes is not crucial, the matrix profile used for direct solvers is strongly dependent on the ordering of nodes (and accordingly the demand on storage). As a general rule,

the numbering should be such as to minimize the nodal difference for each element (maximum node number minus minimum node number). For large and complex meshes automatic nodal renumbering schemes have to be utilized to minimize the matrix profile (cf. Sect. 8.17.1.4). We note that the order of numbering the elements is neither crucial for iterative nor for direct equation solvers. In cases where the element matrices are symmetric, the global matrix  $\mathbf{K}$  also becomes symmetric  $K_{ij} = K_{ji}$  and only the symmetric half of the matrix coefficients needs to be stored. The assembly of a symmetric global matrix requires the computation of  $\frac{1}{2}N_E N_{BN}(N_{BN} + 1)$  coefficients, which means a reduction of the computational effort by a factor of  $(N_{BN} - 1)/(2N_{BN})$  compared to an unsymmetric matrix, i.e., for instance  $\frac{1}{3}$  and  $\frac{5}{12}$  less for 2D triangular and 3D tetrahedron elements, respectively.

### 8.10.2 Parallelization via Element Agglomeration

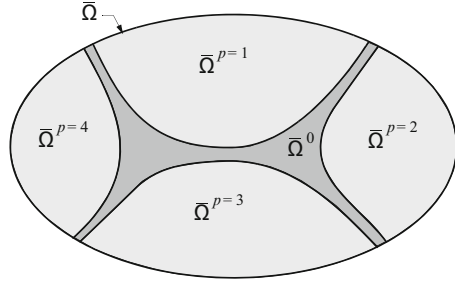
The element-by-element assembly procedure (8.106) and (8.107) in form of  $\mathbf{K} = \sum_e^{N_E} \mathbf{K}^e$  and so forth is ideally suitable for parallelization by agglomeration of elements for which the element matrices  $\mathbf{K}^e$  can be performed parallel on different CPU's or CPU cores to accelerate significantly the computations. The computational work of assembly is proportional to the number of elements  $N_E$ . Therefore, an efficient parallel assembly process can be achieved if the total number of elements of a mesh is suitably split into a certain number of subdomains called *partitions* of agglomerated elements so that the element summation is actually executed via

$$\mathbf{K} = \sum_p^{N_{PA}} \sum_e^{N_{E_p}} \mathbf{K}^e \quad \text{with} \quad N_E = \sum_p^{N_{PA}} N_{E_p} \quad (8.110)$$

where  $N_{PA}$  is the number of partitions and  $N_{E_p}$  is the number of elements agglomerated into partition  $p$ . Each partition is concurrently executed on different *threads* representing logical processors. A symmetric multiprocessing facility (SMP) of an operating system distributes all threads to the available physical processors and CPU cores during runtime. However, on computer systems with shared memory to which the threads simultaneously access the summation operations for an element  $e$  cannot be executed on different threads at the same time, otherwise the summation becomes erroneous due to occurring *race conditions*. To avoid multiple access during the assembly process the concerned element must be locked while it is summed up. But, *locking* is a rather inefficient process and slows down the computations in particular with increasing number of threads.

More useful and generally preferred in FEFLOW is a technique which is called *disjoint domain partitioning*. It does not need locking and provides an optimal speedup in the parallel assembly process based on (8.110). The computational domain  $\bar{\Omega} = \Omega \cup \Gamma$  is subdivided into a maximum number of partitions  $\bar{\Omega}^{PAD}$  of agglomerated elements  $\bar{\Omega}^e$ , which do not join each other, with a remaining (possibly small) border set of partition  $\bar{\Omega}^0$  (Fig. 8.14), which joins all the disjoint

**Fig. 8.14** Partitioning of computational domain  $\bar{\Omega}$  into disjoint subdomains  $\bar{\Omega}^{\text{PAD}} = \bigcup_p^{N_{\text{PAD}}} \bar{\Omega}^p$  (exemplified for  $N_{\text{PAD}} = 4$ ) and border set of partition  $\bar{\Omega}^0$  joining  $\bar{\Omega}^{\text{PAD}}$



partitions, i.e.,

$$\bar{\Omega} = \bar{\Omega}^{\text{PAD}} \cup \bar{\Omega}^0 = \bigcup_p^{N_{\text{PAD}}} \bar{\Omega}^p \cup \bar{\Omega}^0 \quad \text{with} \quad \bar{\Omega}^p = \bigcup_e^{N_{E_p}} \bar{\Omega}^e \quad (8.111)$$

where a partition  $p$  and any other partition  $q$  are disjoint if

$$\bar{\Omega}^p \cap \bar{\Omega}^q = \emptyset \quad \text{with} \quad \mathbf{x}_I^e \in \bar{\Omega}^p \wedge \mathbf{x}_I^e \notin \bar{\Omega}^q \quad (8.112)$$

$(p \neq q, \quad p, q = 1, \dots, N_{\text{PAD}})$

and

$$\bar{\Omega} \cap \bar{\Omega}^{\text{PAD}} = \bar{\Omega}^0 \neq \emptyset \quad (8.113)$$

for a node  $I$  at location  $\mathbf{x}_I^e$  in element  $e$ , where  $N_{\text{PAD}} = N_{\text{PA}} - 1$  is the number of disjoint partitions. While all elements belonging to the disjoint partitions can be concurrently assembled in the fast multithreading mode, the elements of the remaining partition  $\bar{\Omega}^0$  must be summed by single threaded execution. However, provided that  $\bar{\Omega}^0 \subset \bar{\Omega}$  is small compared to the disjoint partitions  $\bar{\Omega}^{\text{PAD}} = \bigcup_p^{N_{\text{PAD}}} \bar{\Omega}^p$ , the sequential part of the assembly is insignificant and the parallelized assembly in total provides superior speedups in practical computation. In FEFLOW an efficient and fast agglomeration algorithm is incorporated to find the suitable disjoint partitions of a mesh. To hold  $\bar{\Omega}^0$  small as possible, the algorithm runs recursively for cases where  $\bar{\Omega}^0$  can be further split into disjoint subpartitions. Practically, no more than three recursions are needed to find the minimum  $\bar{\Omega}^0$ , which disjoins all  $\bar{\Omega}^{\text{PAD}}$  by only one element distance (Fig. 8.15).

## 8.11 Finite Element Basis Operations

Recalling the basic calculus operations for coordinate transformation described in Sect. 2.1.5, the mapping (8.71) of isoparametric element geometry

$$\mathbf{x}^e = \sum_{J=1}^{N_{\text{BN}}} N_J^e(\boldsymbol{\eta}) \mathbf{x}_J^e \quad (8.114)$$



**Fig. 8.15** Example of a partitioned 2D triangle mesh: Elements drawn in *red* indicate the border set partition  $\Omega^0$

between global  $\mathbf{x}$  and local  $\boldsymbol{\eta}$  coordinates is associated with the nonsingular Jacobian  $\mathbf{J}^e$  defined in the  $\mathfrak{N}^3$  space

$$\mathbf{J}^e = \frac{\partial \mathbf{x}^e}{\partial \boldsymbol{\eta}} = \begin{pmatrix} \frac{\partial}{\partial \xi} \\ \frac{\partial}{\partial \eta} \\ \frac{\partial}{\partial \zeta} \end{pmatrix} (x_1^e \ x_2^e \ x_3^e) = \begin{pmatrix} J_{11} & J_{12} & J_{13} \\ J_{21} & J_{22} & J_{23} \\ J_{31} & J_{32} & J_{33} \end{pmatrix}^e = \begin{pmatrix} \frac{\partial x_1^e}{\partial \xi} & \frac{\partial x_2^e}{\partial \xi} & \frac{\partial x_3^e}{\partial \xi} \\ \frac{\partial x_1^e}{\partial \eta} & \frac{\partial x_2^e}{\partial \eta} & \frac{\partial x_3^e}{\partial \eta} \\ \frac{\partial x_1^e}{\partial \zeta} & \frac{\partial x_2^e}{\partial \zeta} & \frac{\partial x_3^e}{\partial \zeta} \end{pmatrix}$$

$$= \begin{pmatrix} \sum_{J=1}^{N_{\text{BN}}} \frac{\partial N_J^e(\xi, \eta, \zeta)}{\partial \xi} x_{1J}^e & \sum_{J=1}^{N_{\text{BN}}} \frac{\partial N_J^e(\xi, \eta, \zeta)}{\partial \xi} x_{2J}^e & \sum_{J=1}^{N_{\text{BN}}} \frac{\partial N_J^e(\xi, \eta, \zeta)}{\partial \xi} x_{3J}^e \\ \sum_{J=1}^{N_{\text{BN}}} \frac{\partial N_J^e(\xi, \eta, \zeta)}{\partial \eta} x_{1J}^e & \sum_{J=1}^{N_{\text{BN}}} \frac{\partial N_J^e(\xi, \eta, \zeta)}{\partial \eta} x_{2J}^e & \sum_{J=1}^{N_{\text{BN}}} \frac{\partial N_J^e(\xi, \eta, \zeta)}{\partial \eta} x_{3J}^e \\ \sum_{J=1}^{N_{\text{BN}}} \frac{\partial N_J^e(\xi, \eta, \zeta)}{\partial \zeta} x_{1J}^e & \sum_{J=1}^{N_{\text{BN}}} \frac{\partial N_J^e(\xi, \eta, \zeta)}{\partial \zeta} x_{2J}^e & \sum_{J=1}^{N_{\text{BN}}} \frac{\partial N_J^e(\xi, \eta, \zeta)}{\partial \zeta} x_{3J}^e \end{pmatrix}, \quad (8.115)$$

in the  $\mathfrak{N}^2$  space

$$\mathbf{J}^e = \begin{pmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{pmatrix}^e = \begin{pmatrix} \frac{\partial x_1^e}{\partial \xi} & \frac{\partial x_2^e}{\partial \xi} \\ \frac{\partial x_1^e}{\partial \eta} & \frac{\partial x_2^e}{\partial \eta} \end{pmatrix} = \begin{pmatrix} \sum_{J=1}^{N_{\text{BN}}} \frac{\partial N_J^e(\xi, \eta)}{\partial \xi} x_{1J}^e & \sum_{J=1}^{N_{\text{BN}}} \frac{\partial N_J^e(\xi, \eta)}{\partial \xi} x_{2J}^e \\ \sum_{J=1}^{N_{\text{BN}}} \frac{\partial N_J^e(\xi, \eta)}{\partial \eta} x_{1J}^e & \sum_{J=1}^{N_{\text{BN}}} \frac{\partial N_J^e(\xi, \eta)}{\partial \eta} x_{2J}^e \end{pmatrix} \quad (8.116)$$

and in the  $\mathfrak{N}^1$  space

$$\mathbf{J}^e = (J_{11})^e = \left( \frac{\partial x_1^e}{\partial \xi} \right) = \left( \sum_{J=1}^{N_{\text{BN}}} \frac{\partial N_J^e(\xi)}{\partial \xi} x_{1J}^e \right). \quad (8.117)$$

To evaluate the flux vector divergence terms in the element matrices of (8.104) the inverse Jacobian is required

$$\nabla N_J^e = (\mathbf{J}^e)^{-1} \cdot \begin{pmatrix} \frac{\partial N_J^e}{\partial \xi} \\ \frac{\partial N_J^e}{\partial \eta} \\ \frac{\partial N_J^e}{\partial \zeta} \end{pmatrix} \quad (8.118)$$

where

$$(\mathbf{J}^e)^{-1} = \frac{\partial \boldsymbol{\eta}}{\partial \mathbf{x}^e} = \begin{cases} \frac{1}{|\mathbf{J}^e|} \begin{pmatrix} (J_{22}^e J_{33}^e - J_{32}^e J_{23}^e) & (J_{13}^e J_{32}^e - J_{12}^e J_{33}^e) & (J_{12}^e J_{23}^e - J_{13}^e J_{22}^e) \\ (J_{31}^e J_{23}^e - J_{21}^e J_{33}^e) & (J_{11}^e J_{33}^e - J_{13}^e J_{31}^e) & (J_{21}^e J_{13}^e - J_{23}^e J_{11}^e) \\ (J_{21}^e J_{32}^e - J_{31}^e J_{22}^e) & (J_{12}^e J_{31}^e - J_{32}^e J_{11}^e) & (J_{11}^e J_{22}^e - J_{12}^e J_{21}^e) \end{pmatrix} & \text{in } \mathfrak{N}^3 \\ \frac{1}{|\mathbf{J}^e|} \begin{pmatrix} J_{22}^e & -J_{12}^e \\ -J_{21}^e & J_{11}^e \end{pmatrix} & \text{in } \mathfrak{N}^2 \\ \frac{1}{|\mathbf{J}^e|} & \text{in } \mathfrak{N}^1 \end{cases} \quad (8.119)$$

with the determinant of  $\mathbf{J}^e$

$$|\mathbf{J}^e| = \begin{cases} J_{11}^e (J_{22}^e J_{33}^e - J_{32}^e J_{23}^e) - J_{21}^e (J_{12}^e J_{33}^e - J_{13}^e J_{32}^e) + J_{31}^e (J_{12}^e J_{23}^e - J_{13}^e J_{22}^e) & \text{in } \mathfrak{N}^3 \\ J_{11}^e J_{22}^e - J_{21}^e J_{12}^e & \text{in } \mathfrak{N}^2 \\ J_{11}^e & \text{in } \mathfrak{N}^1 \end{cases} \quad (8.120)$$

Suppose the local shape functions are continuous and at least once-differentiable with respect to the local coordinates  $\boldsymbol{\eta}$ , a necessary and sufficient condition for  $(\mathbf{J}^e)^{-1}$  to exist is that the determinant of the Jacobian  $|\mathbf{J}^e|$  of element  $e$  be nonzero at every point  $\boldsymbol{\eta}$  in  $\bar{\Omega}^e$ :

$$|\mathbf{J}^e| \neq 0 \quad (8.121)$$

The master element matrices and vectors appearing in (8.104) and (8.105), respectively, are to be integrated over element volumes  $\Omega^e$  and surfaces  $\Gamma^e$ . The integration in local coordinates becomes for a differential ‘volume’ element

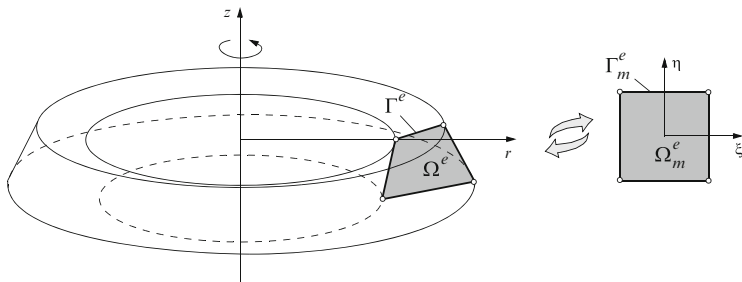
$$d\Omega^e = \begin{cases} \begin{cases} dx_1 dx_2 dx_3 = |\mathbf{J}^e| d\xi d\eta d\zeta \\ dx_1 dx_2 = |\mathbf{J}^e| d\xi d\eta \\ dx_1 = |\mathbf{J}^e| d\xi \end{cases} & \text{Cartesian } \mathfrak{R}^D \ (D = 1, 2, 3) \\ r dr d\phi dz = 2\pi |\mathbf{J}^e| r d\xi d\eta & \text{axisymmetric} \end{cases} \quad (8.122)$$

and for a differential ‘areal’ element in Cartesian coordinates of  $\mathfrak{R}^D$  ( $D = 1, 2, 3$ ) space:

$$d\Gamma^e = \begin{cases} \left\| \begin{pmatrix} \frac{\partial x_1}{\partial \xi} \\ \frac{\partial x_2}{\partial \xi} \\ \frac{\partial x_3}{\partial \xi} \end{pmatrix} \times \begin{pmatrix} \frac{\partial x_1}{\partial \eta} \\ \frac{\partial x_2}{\partial \eta} \\ \frac{\partial x_3}{\partial \eta} \end{pmatrix} \right\| d\xi d\eta = \left\| \det \begin{pmatrix} e_1 & e_2 & e_3 \\ J_{11}^e & J_{12}^e & J_{13}^e \\ J_{21}^e & J_{22}^e & J_{23}^e \end{pmatrix} \right\| d\xi d\eta \\ = \left\| \begin{pmatrix} J_{12}^e J_{23}^e - J_{13}^e J_{22}^e \\ J_{13}^e J_{21}^e - J_{11}^e J_{23}^e \\ J_{11}^e J_{22}^e - J_{12}^e J_{21}^e \end{pmatrix} \right\| d\xi d\eta & \text{at } \zeta = \pm 1 \text{ in } \mathfrak{R}^3 \\ \left\| \begin{pmatrix} \frac{\partial x_1}{\partial \eta} \\ \frac{\partial x_2}{\partial \eta} \\ \frac{\partial x_3}{\partial \eta} \end{pmatrix} \times \begin{pmatrix} \frac{\partial x_1}{\partial \zeta} \\ \frac{\partial x_2}{\partial \zeta} \\ \frac{\partial x_3}{\partial \zeta} \end{pmatrix} \right\| d\eta d\zeta = \left\| \det \begin{pmatrix} e_1 & e_2 & e_3 \\ J_{21}^e & J_{22}^e & J_{23}^e \\ J_{31}^e & J_{32}^e & J_{33}^e \end{pmatrix} \right\| d\eta d\zeta \\ = \left\| \begin{pmatrix} J_{22}^e J_{33}^e - J_{23}^e J_{32}^e \\ J_{23}^e J_{31}^e - J_{21}^e J_{33}^e \\ J_{21}^e J_{32}^e - J_{22}^e J_{31}^e \end{pmatrix} \right\| d\eta d\zeta & \text{at } \xi = \pm 1 \text{ in } \mathfrak{R}^3 \\ \left\| \begin{pmatrix} \frac{\partial x_1}{\partial \xi} \\ \frac{\partial x_2}{\partial \xi} \\ \frac{\partial x_3}{\partial \xi} \end{pmatrix} \times \begin{pmatrix} \frac{\partial x_1}{\partial \zeta} \\ \frac{\partial x_2}{\partial \zeta} \\ \frac{\partial x_3}{\partial \zeta} \end{pmatrix} \right\| d\xi d\zeta = \left\| \det \begin{pmatrix} e_1 & e_2 & e_3 \\ J_{11}^e & J_{12}^e & J_{13}^e \\ J_{31}^e & J_{32}^e & J_{33}^e \end{pmatrix} \right\| d\xi d\zeta \\ = \left\| \begin{pmatrix} J_{12}^e J_{33}^e - J_{13}^e J_{32}^e \\ J_{13}^e J_{31}^e - J_{11}^e J_{33}^e \\ J_{11}^e J_{32}^e - J_{12}^e J_{31}^e \end{pmatrix} \right\| d\xi d\zeta & \text{at } \eta = \pm 1 \text{ in } \mathfrak{R}^3 \\ \left\| \begin{pmatrix} \frac{\partial x_1}{\partial \xi} \\ \frac{\partial x_2}{\partial \xi} \end{pmatrix} \right\| d\xi = \left\| \begin{pmatrix} J_{11}^e \\ J_{12}^e \end{pmatrix} \right\| d\xi = \sqrt{(J_{11}^e)^2 + (J_{12}^e)^2} d\xi & \text{at } \eta = \pm 1 \text{ in } \mathfrak{R}^2 \\ \left\| \begin{pmatrix} \frac{\partial x_1}{\partial \eta} \\ \frac{\partial x_2}{\partial \eta} \end{pmatrix} \right\| d\eta = \left\| \begin{pmatrix} J_{21}^e \\ J_{22}^e \end{pmatrix} \right\| d\eta = \sqrt{(J_{21}^e)^2 + (J_{22}^e)^2} d\eta & \text{at } \xi = \pm 1 \text{ in } \mathfrak{R}^2 \\ \dots \Big|_{\xi=-1}^{\xi=+1} & \text{in } \mathfrak{R}^1 \end{cases} \quad (8.123)$$

and in cylindrical coordinates of  $\mathfrak{R}^2$  (meridional) space (Fig. 8.16):





**Fig. 8.16** Axisymmetric finite element in global (cylindrical) and local coordinates

$$d\Gamma^e = \begin{cases} \left\| \begin{pmatrix} \frac{\partial r}{\partial \xi} \\ \frac{\partial z}{\partial \xi} \end{pmatrix} \right\| rd\xi d\phi = 2\pi \left\| \begin{pmatrix} J_{11}^e \\ J_{12}^e \end{pmatrix} \right\| rd\xi = 2\pi \sqrt{(J_{11}^e)^2 + (J_{12}^e)^2} rd\xi & \text{at } \eta = \pm 1 \\ \left\| \begin{pmatrix} \frac{\partial r}{\partial \eta} \\ \frac{\partial z}{\partial \eta} \end{pmatrix} \right\| rd\eta d\phi = 2\pi \left\| \begin{pmatrix} J_{21}^e \\ J_{22}^e \end{pmatrix} \right\| rd\eta = 2\pi \sqrt{(J_{21}^e)^2 + (J_{22}^e)^2} rd\eta & \text{at } \xi = \pm 1 \end{cases} \quad (8.124)$$

where  $r = \sum_J^{N_{BN}} N_J^e(\xi, \eta) r_J^e$ .

In using (8.114)–(8.124) all integrals in (8.104) and (8.105) can be expressed in the  $\eta$ -coordinate system. For example, the element advection matrix  $A^e$  of the convective form (8.104) becomes in 3D

$$\begin{aligned} A_{IJ}^e &= \int_{\Omega^e} N_I^e \mathbf{q} \cdot \nabla N_J^e d\Omega^e \\ &= \iiint_{-1}^{+1} N_I^e(\boldsymbol{\eta}) \mathbf{q} \cdot \left( (\mathbf{J}^e)^{-1} \cdot \begin{pmatrix} \frac{\partial N_J^e(\boldsymbol{\eta})}{\partial \xi} \\ \frac{\partial N_J^e(\boldsymbol{\eta})}{\partial \eta} \\ \frac{\partial N_J^e(\boldsymbol{\eta})}{\partial \zeta} \end{pmatrix} \right) |J^e| d\xi d\eta d\zeta \end{aligned} \quad (8.125)$$

and similarly for all other element integrals appearing in (8.104) and (8.105). Typically, the integrals always take the form<sup>8</sup>:

<sup>8</sup>The local coordinates  $\boldsymbol{\eta}$  commonly range  $-1 \leq \boldsymbol{\eta} \leq +1$ , except in triangular or tetrahedral (and partly pentahedral and pyramidal) geometries, where the lower limit is zero:  $0 \leq \boldsymbol{\eta} \leq +1$ , see furthermore (8.128).

$$\int_{\Omega^e} f(\mathbf{x}) d\Omega^e = \begin{cases} \iiint_{-1}^{+1} f(\xi, \eta, \zeta) d\xi d\eta d\zeta & \text{in } \mathfrak{R}^3 \\ \iint_{-1}^{+1} f(\xi, \eta) d\xi d\eta & \text{in } \mathfrak{R}^2 \\ \int_{-1}^{+1} f(\xi) d\xi & \text{in } \mathfrak{R}^1 \end{cases} \quad (8.126)$$

$$\int_{\Gamma^e} g(\mathbf{x}) d\Gamma^e = \begin{cases} \iiint_{-1}^{+1} g(\xi, \eta, \zeta) (d\xi d\eta, d\eta d\zeta, d\xi d\zeta) & \text{in } \mathfrak{R}^3 \\ \iint_{-1}^{+1} g(\xi, \eta) (d\xi, d\eta) & \text{in } \mathfrak{R}^2 \\ g(\xi)|_{\xi=-1}^{\xi=+1} & \text{in } \mathfrak{R}^1 \end{cases}$$

where  $f(\cdot)$  and  $g(\cdot)$  are volume and surface integrand functions, respectively. We observe that the dependency of the element  $e$  on the physical (global) geometry only occurs in the Jacobian  $\mathbf{J}^e$ . Since the coordinate transformation is relatively simple, the Jacobian matrix should be easily evaluated.

## 8.12 Numerical and Analytical Integration

The transformation of the geometry and the variable coefficients in the element integrals (8.104) and (8.105) from the global coordinates  $\mathbf{x}$  to the local coordinates  $\boldsymbol{\eta}$  results in algebraically complex expressions, which cannot be analytically evaluated for distorted element geometries in general. However, in fact this is not an intrinsic disadvantage because very efficient and *exact* numerical integration techniques are available which makes the master element integration very cost-effective and highly flexible for a wide class of finite elements under general geometric (i.e., distorted) conditions.

For our needs the most efficient and, therefore, preferable numerical integration is the *Gauss-Legendre quadrature*, e.g., [590], providing an optimal degree of precision. Thus, the evaluation of the 3D, 2D and 1D elemental integrals of (8.126) reduces to expressions with a triple, double and single summation, respectively, in the following form:

$$\begin{aligned} \iiint_{-1}^{+1} f(\xi, \eta, \zeta) d\xi d\eta d\zeta &= \sum_{p=1}^n \sum_{q=1}^n \sum_{r=1}^n H_p H_q H_r f(\xi_p, \eta_q, \zeta_r) \\ \iint_{-1}^{+1} f(\xi, \eta) d\xi d\eta &= \sum_{p=1}^n \sum_{q=1}^n H_p H_q f(\xi_p, \eta_q) \\ \int_{-1}^{+1} f(\xi) d\xi &= \sum_{p=1}^n H_p f(\xi_p) \end{aligned} \quad (8.127)$$

**Table 8.2** Gauss-Legendre quadrature sampling points and weights for  $\int_{-1}^{+1} f(\xi)d\xi = \sum_{p=1}^n H_p f(\xi_p)$

Order	$n$	$p$	$\xi_p$	$H_p$	
Linear	1	1	0	2	
Quadratic	2	1	$+1/\sqrt{3}$	1	
		2	$-1/\sqrt{3}$	1	
Cubic	3	1	$+\sqrt{0.6}$	5/9	
		2	0	8/9	
		3	$-\sqrt{0.6}$	5/9	
Quintic	4	1	$+\sqrt{(3+a)/7}$	$0.5 - 1/(3a)$	$a = \sqrt{4.8}$
		2	$+\sqrt{(3-a)/7}$	$0.5 + 1/(3a)$	
		3	$-\sqrt{(3-a)/7}$	$0.5 + 1/(3a)$	
		4	$-\sqrt{(3+a)/7}$	$0.5 - 1/(3a)$	

where  $n \geq 1$  is the number of quadrature points in each direction,  $\xi_p, \eta_q, \zeta_r$  denote the Gauss point local coordinates in  $\bar{\Omega}_m^e$  and  $H_p, H_q, H_r$  are the associated quadrature rule weights. Note that for 1D elements the integrals are commonly evaluated analytically. Choosing  $n$  Gauss points a polynomial expression  $f(\cdot)$  of degree  $2n - 1$  can be exactly integrated. The positions and weights for the Gauss-Legendre quadrature rule up to order  $n = 4$  are listed in Table 8.2. When the integrand  $f(\cdot)$  is of different degree in  $\xi, \eta, \zeta$ , the number of Gauss points should be selected on the basis of the largest-degree polynomial. The minimum allowable quadrature is one that yields the volume or area of the element exactly [590]. For undistorted elements (such as rectangular or brick-shaped elements) the 2- and 3-point Gauss-Legendre rules (i.e., Gauss points in each direction) are sufficient to evaluate exactly all interesting integrals of linear and quadratic element types, respectively, because the Jacobian of the mapping  $\mathbf{J}^e$  is constant for these geometries. However, for distorted elements the Jacobian is no more constant and integrals involving more than one derivative cannot be exactly integrated since the integrand is a quotient of two polynomials [341]. In general,  $f(\cdot)$  may not be really polynomial due to the complex dependency of the integrand on  $\mathbf{J}^e$  and the possible presence of other variable coefficients such that the required number of Gauss points can only be estimated. In practice, an *optimal order of Gauss-Legendre integration* is used, which is defined as one that guarantees the highest possible accuracy and rate of convergence while minimizing the computational cost. For linear quadrilateral elements  $2 \times 2 \times 2$  and  $2 \times 2$ , and for quadratic quadrilateral elements  $3 \times 3 \times 3$  and  $3 \times 3$  are of optimal order in 3D and 2D, respectively. For the areal triangular element (triangle), the 3D linear tetrahedron, the 3D linear triangular prism (petahedron) and the 3D linear pyramidal element (pyramid) specific integration points and weights are applied [84, 280, 586, 590] as follows:

**Table 8.3** Quadrature points and weights of formulae (8.128) of quadratic order for linear triangle, linear tetrahedron, linear pentahedron and linear square pyramid

Type	$m$	$p$	$\xi_p$	$\eta_p$	$\zeta_p$	$H_p$	
Triangle	3	1	1/2	0		1/6	
		2	1/2	1/2		1/6	
		3	0	1/2		1/6	
Tetrahedron	4	1	$a$	$a$	$a$	1/24	$a = 0.13819660$
		2	$b$	$a$	$a$	1/24	$b = 0.58541020$
		3	$a$	$b$	$a$	1/24	
		4	$a$	$a$	$b$	1/24	
Pentahedron	6	1	1/2	0	$+1/\sqrt{3}$	1/6	
		2	1/2	1/2	$+1/\sqrt{3}$	1/6	
		3	0	1/2	$+1/\sqrt{3}$	1/6	
		4	1/2	0	$-1/\sqrt{3}$	1/6	
		5	1/2	1/2	$-1/\sqrt{3}$	1/6	
		6	0	1/2	$-1/\sqrt{3}$	1/6	
Pyramid	8	1	$+c/\sqrt{3}$	$+c/\sqrt{3}$	$1-c$	0.1007858820798250	$c = 0.455848155988775$
		2	$-c/\sqrt{3}$	$+c/\sqrt{3}$	$1-c$	0.1007858820798250	
		3	$+c/\sqrt{3}$	$-c/\sqrt{3}$	$1-c$	0.1007858820798250	
		4	$-c/\sqrt{3}$	$-c/\sqrt{3}$	$1-c$	0.1007858820798250	
		5	$+d/\sqrt{3}$	$+d/\sqrt{3}$	$1-d$	0.2325474512535080	$d = 0.877485177344559$
		6	$-d/\sqrt{3}$	$+d/\sqrt{3}$	$1-d$	0.2325474512535080	
		7	$+d/\sqrt{3}$	$-d/\sqrt{3}$	$1-d$	0.2325474512535080	
		8	$-d/\sqrt{3}$	$-d/\sqrt{3}$	$1-d$	0.2325474512535080	

triangle

$$\int_{\Gamma^e} f(\mathbf{x}) d\Gamma^e = \int_0^1 \int_0^{1-\xi} f(\xi, \eta) d\eta d\xi = \sum_{p=1}^m H_p f(\xi_p, \eta_p)$$

tetrahedron

$$\int_{\Omega^e} f(\mathbf{x}) d\Omega^e = \int_0^1 \int_0^{1-\xi} \int_0^{1-\xi-\eta} f(\xi, \eta, \zeta) d\zeta d\eta d\xi = \sum_{p=1}^m H_p f(\xi_p, \eta_p, \zeta_p)$$

pentahedron

$$\int_{\Omega^e} f(\mathbf{x}) d\Omega^e = \int_0^1 \int_0^{1-\xi} \int_{-1}^1 f(\xi, \eta, \zeta) d\zeta d\eta d\xi = \sum_{p=1}^m H_p f(\xi_p, \eta_p, \zeta_p)$$

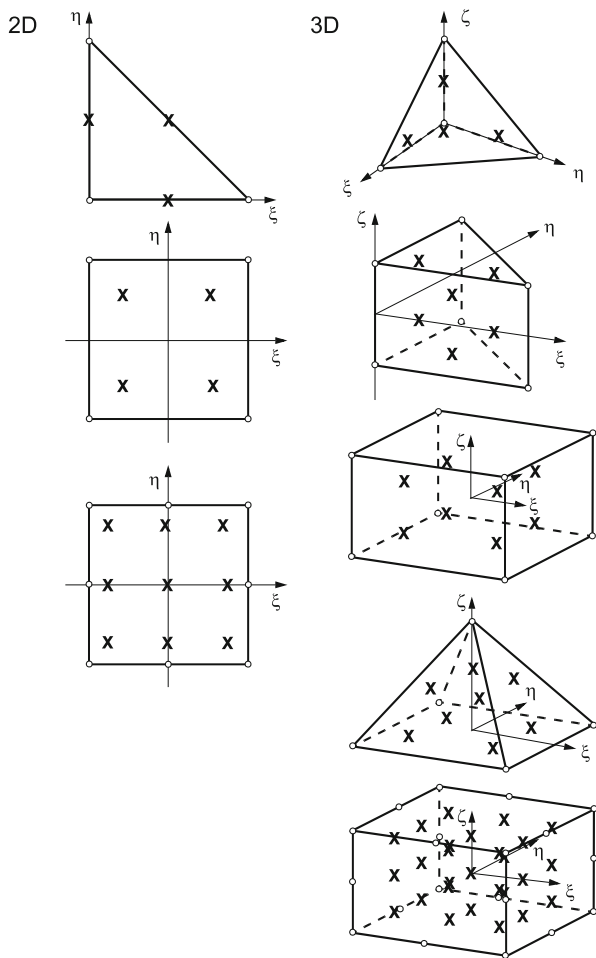
pyramid

$$\int_{\Omega^e} f(\mathbf{x}) d\Omega^e = \int_0^1 \int_{\zeta=-1}^{1-\zeta} \int_{\zeta=-1}^{1-\zeta} f(\xi, \eta, \zeta) d\zeta d\eta d\xi = \sum_{p=1}^m H_p f(\xi_p, \eta_p, \zeta_p)$$

(8.128)

where  $m$  is the total number of integration points. The sampling points and weights for the linear triangle, linear tetrahedron, linear pentahedron and linear square pyramid of quadratic order with  $m = 3$ ,  $m = 4$ ,  $m = 6$  and  $m = 8$ , respectively, are listed in Table 8.3. Figure 8.17 illustrates the locations of the integration points








**Fig. 8.17** Location of Gauss points symbolized by  $\times$  for linear and quadratic isoparametric elements used in FEFLOW in 2D (3-point rule for linear triangle, 4-point rule for linear quadrilateral and 9-point rule for quadratic quadrilateral) and 3D (4-point rule for linear tetrahedron, 6-point rule linear pentahedron, 8-point rule for linear hexahedron, 8-point rule for linear pyramid, 27-point rule for quadratic hexahedron)



in the master element  $\bar{\Omega}_m^e$  for the 2D and 3D finite elements used in FEFLOW. For axisymmetric elements the same quadrature rules are applied than in 2D.

Table 8.4 gives an estimation of arithmetic operations required for the numerical integration of one element. The working steps 1–7 in Table 8.4 indicate the effort from the coordinate transformation, which has to be performed basically, however, only once for all integrands in consideration. We recognize that the computation of the Jacobian and the terms related to the global derivatives are the most expensive steps. Once these terms are available the effort in computing additional terms and extensions in the integrand functions (e.g., introducing variable coefficients and anisotropic relations) remains low, which makes the numerical integration very flexible and efficient. We also see that the required number of total operations increases significantly with more complex (higher-order) finite elements, in particular for 3D finite elements, however, in favor of a higher accuracy in the

**Table 8.4** Estimation of computational effort for numerical integration of 2D and 3D finite elements: Number of required arithmetic operations (sum of multiplications, divisions, additions and subtractions) to build typical matrices of (8.104) for one element  $e$  via Gauss-Legendre quadrature

Element type	2D			3D			
	 3-node triangle	 4-node quadrilateral	 8-node quadrilateral	 4-node tetrahedron	 6-node pentahedron	 8-node hexahedron	 20-node hexahedron
$N_{BN}$	3	4	8	4	6	8	20
Gauss points $m$	3	4	9	4	6	8	27
1 $N_f^e$	2	17	48	3	22	68	255
2 $\frac{\partial N_f^e}{\partial \xi}$	0	4	22	0	4	16	112
3 $\frac{\partial N_f^e}{\partial \eta}$	0	4	22	0	4	16	112
4 $\frac{\partial N_f^e}{\partial \zeta}$	0	0	0	0	5	16	128
5 $\mathbf{J}^e$ ( $\propto 2 D^2 N_{BN}$ )	24	32	64	72	108	144	360
6 $ \mathbf{J}^e  d\Omega$	4	4	4	15	15	15	15
7 $(\mathbf{J}^e)^{-1}$	5	5	5	28	28	28	28
8 $\nabla N_f^e$ ( $\propto 2 D^2 N_{BN}$ )	24	32	64	72	108	144	360
9 $\int N_f^e$ ( $\propto N_{BN}$ )	3	4	8	4	6	8	20
10 $\int N_f^e N_f^e$ ( $\propto \frac{1}{2} N_{BN}(N_{BN} + 1)$ )	6	10	36	10	21	36	210
11 $\int N_f^e \mathbf{q} \cdot \nabla N_f^e$ ( $\propto 2 D N_{BN}^2$ )	36	64	256	96	216	384	2,400
12 $\int \nabla N_f^e \cdot (\mathbf{D} \cdot \nabla N_f^e)$ ( $\propto D^2 N_{BN}(N_{BN} + 1)$ )	48	80	288	180	378	648	3,780
Operations per Gauss point (summation of steps 1–12)	152	256	817	480	915	1,523	7,780
Operations per element (multiplied by $m$ )	456	1,024	7,353	1,920	5,490	12,184	210,060

approximation. Nevertheless, to maintain the generality in the geometric shapes of the used finite elements the numerical integration is indispensable. All quadrilateral, pentahedral and hexahedral elements lead to very complicated integral expression which can only be tackled numerically. Analytical evaluation is available only for specific element shapes. In Appendix H we evaluate the element matrices of (8.104) on an analytical basis for the linear 1D element, the linear 2D triangle and the linear 3D tetrahedron assuming constant coefficients. Only for these types of elements the Jacobians are always constant for every element shape, which make these elements favorable to analytical integration. In Appendix H we also discuss exceptions for the quadrilateral, hexahedral, pentahedral and pyramidal element, where a constant Jacobian is only attainable for an undisturbed element shape, such as the rectangle or parallelogram for the quadrilateral element, the brick or parallelepiped for the hexahedral element, the triangular prism with parallel top and bottom surfaces for

the pentahedral element and the pyramid with a parallelogram or rectangular base and oblique shape for the pyramidal element.

## 8.13 Temporal Discretization

### 8.13.1 General

The semidiscrete Galerkin approximation (8.96) of the governing ADE has led to system of ordinary differential equations (ODE's) written in the form (8.100)

$$\mathbf{O} \cdot \dot{\phi} + \mathbf{K} \cdot \phi = \mathbf{F} \quad (8.129)$$

for solving  $\phi = \phi(t)$  associated with IC's at  $t = t_0$

$$\phi(t_0) = \phi_0 \quad (8.130)$$

where  $\mathbf{O}$  is called a *consistent mass* (CM) matrix because it is defined consistent with the weak formulation in (8.98) and (8.99) assuming the separability of space  $\mathbf{x}$  and time  $t$ . It remains to solve the resulting semidiscrete equations (8.129) for  $\phi(t)$  via appropriate and cost-effective time-integration methods, which integrate (8.129) in time  $t$  to trace the temporal evolution of  $\phi(t)$  from the initial solution  $\phi_0$ .

Let us rewrite (8.129) in a normalized form, viz.,

$$\dot{\phi} + \mu \cdot \phi = \mathbf{f} \quad \text{with} \quad \mu = \mathbf{O}^{-1} \cdot \mathbf{K} \quad \text{and} \quad \mathbf{f} = \mathbf{O}^{-1} \cdot \mathbf{F} \quad (8.131)$$

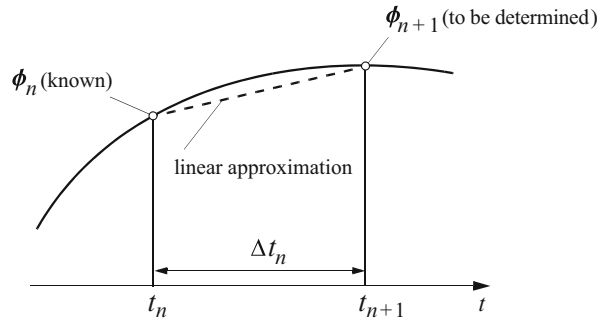
provided that  $\mathbf{O}$  is invertible with  $|\mathbf{O}| \neq 0$ , we find the solution of this first-order system of ODE's as [10]

$$\phi(t) = \underbrace{e^{-\mu(t-t_0)} \cdot \phi_0}_{\text{decay}} + \int_{t_0}^t \underbrace{e^{-\mu(t-\tau)} \cdot \mathbf{f}(\tau)}_{\text{forcing}} d\tau \quad (8.132)$$

consisting of two components: (1) the exponential decay of the homogeneous part and (2) the particular solution of the forcing contribution. However, we recognize that the exponential matrix  $e^{-\mu t}$  is complex and not an algebraic statement, hence not directly solvable. We have to conclude that, in general, it is not possible to integrate (8.132) (and accordingly (8.129)) on an analytical basis and further approximation methods are required to obtain a set of algebraic equations in terms of the nodal state-variable  $\phi$ .

There is an abundance of numerical methods for solving ODE's, which are categorized as *methods of lines* in the classic literature [331, 441, 462]. Among the wide variety of available methods, however, from the practical point of view, in particular the computational cost, we have interests only in efficient two-stage *single-step* time marching *recurrence* schemes, where for stability reasons implicit

**Fig. 8.18** Time marching recurrence of  $\phi$  in the finite time interval  $(t_n, t_{n+1})$ , where  $t_{n+1} = t_n + \Delta t_n$



or semi-implicit (so-called  $\mathcal{A}$ -stable [110]) algorithms will be preferred, which are stable independent of the used time step. Considering  $\phi(t)$  within the finite interval  $(t_n, t_{n+1})$  with

$$t_{n+1} = t_n + \Delta t_n \quad (8.133)$$

where the subscript  $n$  denotes the time plane and  $\Delta t_n = t_{n+1} - t_n$  is a variable time step length, the state-variable  $\phi(t)$  is defined as

$$\phi_n = \phi(t_n) \quad (8.134)$$

at the previous (old) time plane  $n$  and as

$$\phi_{n+1} = \phi(t_{n+1}) \quad (8.135)$$

at the new time plane  $n + 1$ . In each interval,  $\phi_{n+1}$  is recursively solved from the preceding values  $\phi_n$  at beginning of the time step  $\Delta t_n$  as shown in Fig. 8.18, and (8.132) can be recast into an incremental form, viz.,

$$\phi_{n+1} = e^{-\mu \Delta t_n} \cdot \phi_n + e^{-\mu t_{n+1}} \cdot \int_{t_n}^{t_{n+1}} e^{\mu \tau} \cdot \mathbf{f}(\tau) d\tau \quad (8.136)$$

While (8.136) is still an exact solution in time  $t$  without any approximation, it is necessary to expand the exponential decay matrix  $e^{-\mu \Delta t_n}$  within the time step  $\Delta t_n$  into a power series. The most typical linear approximations for  $e^{-\mu \Delta t_n}$  are listed in Table 8.5 and will be discussed in the following. However, before we proceed and introduce the appropriate time stepping schemes in detail, we have to consider first the approximation of the mass matrix  $\mathbf{O}$  called *mass lumping*.



**Table 8.5** Linear  $\theta$ -approximations<sup>a</sup> of the exponential decay matrix  $e^{-\mu\Delta t_n}$

Algorithm	$\theta$	Approximation
Forward	0	$\delta - \Delta t_n \mu$
Backward	1	$[\delta + \Delta t_n \mu]^{-1}$
Trapezoid	$\frac{1}{2}$	$[\delta + (\Delta t_n/2)\mu]^{-1} \cdot [\delta - (\Delta t_n/2)\mu]$
Galerkin	$\frac{2}{3}$	$[\delta + (2\Delta t_n/3)\mu]^{-1} \cdot [\delta - (\Delta t_n/3)\mu]$

<sup>a</sup> Weighting coefficient ( $0 \leq \theta \leq 1$ ) classifies approximation methods

### 8.13.2 Mass Lumping

The Galerkin formulation naturally leads to consistent mass matrices  $\mathbf{O} = \sum_e \mathbf{O}^e$ , which typically distribute the mass of an element over all associated nodes. This can be seen in the discrete element mass matrices  $\mathbf{O}^e$  of (H.8), (H.23) and (H.41) in Appendix H for the linear 1D, 2D triangular and 3D tetrahedral element, respectively. However, there can be different numerical reasons to concentrate (lump) the mass of an element on the mesh nodes. Mass lumping is a typical feature of the FDM and can also be useful in the finite element context for certain time stepping schemes. The principal motivation behind this technique is the generation of a mass matrix  $\mathbf{O}$ , which is *diagonal* and readily invertible to evaluate  $\mathbf{O}^{-1}$  of (8.131) in a trivial way. In contrast, the CM matrix, which is sparse and banded, has an inverse that is dense, such that the formulation (8.131) becomes inferior to (8.129) in practical computations.

To permit an equivalent formulation of the mass matrix  $\mathbf{O}^e$  of an element  $e$  we replace




$$\mathbf{O}^e = \mathbf{O}_{IJ}^e = \underbrace{\int_{\Omega^e} \hat{\mathcal{R}}^e N_I^e N_J^e d\Omega^e}_{\text{consistent}} \rightarrow \delta_{IJ} \underbrace{\int_{\Omega^e} \hat{\mathcal{R}}^e N_I^e d\Omega^e}_{\text{lumping}} \quad (8.137)$$

where  $\delta_{IJ}$  is the Kronecker symbol (2.7). Since  $\sum_{J=1}^{N_{\text{BN}}} N_J^e = 1$ , (8.67), the lumping procedure is equivalent to *summing* the rows of the CM matrix:  $O_{II}^e = \sum_J O_{IJ}^e$ . However, this row-summing technique of mass lumping is usually only applicable to linear elements, where the diagonals are always positive. We note that for higher-order elements specific rules of mass lumping are required [590], for instance [129]:

$$O_{IJ}^e = \begin{cases} \int_{\Omega^e} \hat{\mathcal{R}}^e d\Omega^e \int_{\Omega^e} N_I^e N_J^e d\Omega^e / \sum_{l=1}^{N_{\text{BN}}} \int_{\Omega^e} N_l^e N_l^e d\Omega^e & \text{for } I = J, \\ 0 & \text{for } I \neq J \end{cases} \quad (8.138)$$

In Table 8.6 analytical formulations of the consistent mass (CM) and the lumped mass (LM) matrix are compared for the linear 1D, 2D triangular and 3D tetrahedral element (cf. Appendix H). Using numerical integration, the quadrature rules are directly applied to the  $\delta_{IJ} \int_{\Omega^e} N_I^e d\Omega^e$  term to yield a diagonal matrix for  $\mathbf{O}^e$ .

**Table 8.6** Consistent versus lumped mass matrix for the linear 1D, 2D triangular and 3D tetrahedral element (cf. Appendix H)

Element type	Consistent $\int_{\Omega^e} N_j^e N_j^e d\Omega^e$	Lumping $\delta_{IJ} \int_{\Omega^e} N_j^e d\Omega^e$
	$\frac{\Delta x^e}{6} \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$	$\frac{\Delta x^e}{2} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
	$\frac{A^e}{12} \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix}$	$\frac{A^e}{3} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
	$\frac{V^e}{20} \begin{pmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 2 \end{pmatrix}$	$\frac{V^e}{4} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

Mass lumping is generally desired for an explicit time integration scheme to perform  $O^{-1}$  very easily in use of (8.131) because the application of a CM matrix for the explicit scheme becomes too expensive. On the other hand, for implicit time integration methods mass lumping is commonly neither necessary nor preferred. In general, CM matrix formulations provide higher accuracy, see [209, 210, 590]. Exception is given for unsaturated flow problems, where fully implicit time stepping schemes in combination with mass lumping have shown superior to CM.

### 8.13.3 Galerkin Approximation in Time

Similar to the spatial approximation (8.16) we can use the standard finite element expansion for the time-dependent state variable as

$$\phi(t) \approx \hat{\phi}(t) = \sum_j N_j(t) \phi_j \quad (j = n, n + 1, \dots) \tag{8.139}$$

within the time interval  $(t_n, t_n + \Delta t_n)$ . In the linear case with  $j = n, n + 1$  the interpolation functions are

$$N_n = 1 - \tau, \quad N_{n+1} = \tau \tag{8.140}$$

where

$$\tau = \frac{t - t_n}{\Delta t_n} \tag{8.141}$$

Inserting (8.139) into (8.129) a temporal weighted residual approximation, similar to a spatial weak statement (8.36), can be written

$$\int_{t_n}^{t_n + \Delta t_n} w_i(t) \left[ \mathbf{O} \cdot (\dot{N}_n \phi_n + \dot{N}_{n+1} \phi_{n+1}) + \mathbf{K} \cdot (N_n \phi_n + N_{n+1} \phi_{n+1}) - \mathbf{F} \right] dt = 0 \tag{8.142}$$

with

$$\dot{N}_n = -\frac{1}{\Delta t_n}, \quad \dot{N}_{n+1} = \frac{1}{\Delta t_n} \quad (8.143)$$

where  $w_i(t)$  represents the weighting function set to be specified below. If we substitute (8.141) with  $dt = \Delta t_n d\tau$  into (8.142) and divide by  $\Delta t_n$ , the following time marching recurrence formula results

$$\begin{aligned} & \left( \mathcal{O} \int_0^1 w_i \frac{1}{\Delta t_n} d\tau + \mathbf{K} \int_0^1 w_i \tau d\tau \right) \cdot \phi_{n+1} - \\ & \left( \mathcal{O} \int_0^1 w_i \frac{1}{\Delta t_n} d\tau - \mathbf{K} \int_0^1 w_i (1 - \tau) d\tau \right) \cdot \phi_n - \int_0^1 w_i \mathbf{F} d\tau = 0 \end{aligned} \quad (8.144)$$

Introducing a weighting coefficient  $\theta$  defined as

$$\theta = \frac{\int_0^1 w_i \tau d\tau}{\int_0^1 w_i d\tau} \quad (8.145)$$

and assuming a linear variation of  $\mathbf{F}(t) \approx N_n(t)\mathbf{F}_n + N_{n+1}(t)\mathbf{F}_{n+1}$  within the time interval where

$$\frac{\int_0^1 w_i \mathbf{F} d\tau}{\int_0^1 w_i d\tau} = \mathbf{F}_{n+1}\theta + \mathbf{F}_n(1 - \theta) \quad (8.146)$$

the final form of the recurrence scheme (8.144) is given by

$$\left( \frac{\mathcal{O}}{\Delta t_n} + \mathbf{K}\theta \right) \cdot \phi_{n+1} = \left( \frac{\mathcal{O}}{\Delta t_n} - \mathbf{K}(1 - \theta) \right) \cdot \phi_n + (\mathbf{F}_{n+1}\theta + \mathbf{F}_n(1 - \theta)) \quad (8.147)$$

to solve  $\phi_{n+1}$  at the new time plane  $n + 1$  from the preceding solution  $\phi_n$  at the previous time plane  $n$ . Using a Galerkin weighting with  $w_i = N_{n+1}$  the weighting coefficient (8.145) yields  $\theta = 2/3$ .

### 8.13.4 The $\theta$ -Family of Time Integration Methods

Introducing a more general weighting coefficient ( $0 \leq \theta \leq 1$ ), we can write

$$\begin{aligned} \phi(t_n + \theta\Delta t_n) &= \theta\phi(t_n + \Delta t_n) + (1 - \theta)\phi(t_n) \\ \mathbf{F}(t_n + \theta\Delta t_n) &= \theta\mathbf{F}(t_n + \Delta t_n) + (1 - \theta)\mathbf{F}(t_n) \\ \dot{\phi}(t_n + \theta\Delta t_n) &= \theta\dot{\phi}(t_n + \Delta t_n) + (1 - \theta)\dot{\phi}(t_n) \end{aligned} \quad (8.148)$$

Using the Taylor series expansion for  $\phi_{n+1} = \phi(t_n + \Delta t_n)$  about  $t_n$  and  $\phi_n = \phi(t_{n+1} - \Delta t_n)$  about  $t_{n+1}$ , respectively,

$$\begin{aligned}\phi_{n+1} &= \phi_n + \Delta t_n \dot{\phi}_n + \frac{\Delta t_n^2}{2} \ddot{\phi}_n + \frac{\Delta t_n^3}{6} \dddot{\phi}_n + \dots \\ \phi_n &= \phi_{n+1} - \Delta t_n \dot{\phi}_{n+1} + \frac{\Delta t_n^2}{2} \ddot{\phi}_{n+1} - \frac{\Delta t_n^3}{6} \dddot{\phi}_{n+1} + \dots\end{aligned}\quad (8.149)$$

we obtain a forward difference approximation, called *forward Euler* (FE)

$$\dot{\phi}_n = \frac{\phi_{n+1} - \phi_n}{\Delta t_n} - \mathcal{O}(\Delta t_n) - \mathcal{O}(\Delta t_n^2) \quad (8.150)$$

and a backward difference approximation, called *backward Euler* (BE)

$$\dot{\phi}_{n+1} = \frac{\phi_{n+1} - \phi_n}{\Delta t_n} + \mathcal{O}(\Delta t_n) - \mathcal{O}(\Delta t_n^2) \quad (8.151)$$

which are accurate to a first-order truncation error of  $\mathcal{O}(\Delta t_n)$ . Inserting (8.151) and (8.150) into (8.148) it results<sup>9</sup>

$$\dot{\phi}(t_n + \theta \Delta t_n) = \frac{\phi_{n+1} - \phi_n}{\Delta t_n} + \mathcal{O}((\theta - \frac{1}{2})\Delta t_n, \Delta t_n^2) \quad (8.152)$$

We recognize from (8.152) that the difference approximation is of second-order accuracy of  $\mathcal{O}(\Delta t_n^2)$  if (and only if)  $\theta = \frac{1}{2}$ , for all other values of  $\theta$  within ( $0 \leq \theta \leq 1$ ) the difference approximation is accurate to a first-order truncation error of  $\mathcal{O}(\Delta t_n)$ .

Common time stepping schemes result if choosing  $\theta$  in an appropriate manner, viz.,

$$\begin{aligned}\theta = 0 & \quad \text{explicit scheme, } \mathcal{O}(\Delta t_n) \\ \theta = \frac{1}{2} & \quad \text{trapezoid rule (Crank-Nicolson scheme), } \mathcal{O}(\Delta t_n^2) \\ \theta = \frac{2}{3} & \quad \text{Galerkin scheme, } \mathcal{O}(\Delta t_n) \\ \theta = 1 & \quad \text{implicit scheme, } \mathcal{O}(\Delta t_n)\end{aligned}\quad (8.153)$$

---

<sup>9</sup>The 2nd-order derivatives are obtained by repeated application of 1st-order approximations:

$$\ddot{\phi}_{n+1} = \frac{\dot{\phi}_{n+1} - \dot{\phi}_n}{\Delta t_n} + \mathcal{O}(\Delta t_n), \quad \ddot{\phi}_n = \frac{\dot{\phi}_{n+1} - \dot{\phi}_n}{\Delta t_n} - \mathcal{O}(\Delta t_n)$$

so that

$$\ddot{\phi}_{n+1} = \ddot{\phi}_n + \mathcal{O}(\Delta t_n)$$

Inserting (8.148) with (8.152) into (8.129) we obtain the following algebraic system of equations

$$\left(\frac{\mathbf{O}}{\Delta t_n} + \mathbf{K}\theta\right) \cdot \phi_{n+1} = \left(\frac{\mathbf{O}}{\Delta t_n} - \mathbf{K}(1-\theta)\right) \cdot \phi_n + (\mathbf{F}_{n+1}\theta + \mathbf{F}_n(1-\theta)) \quad (8.154)$$

and accordingly for the normalized form (8.131)

$$\left(\frac{\delta}{\Delta t_n} + \mu\theta\right) \cdot \phi_{n+1} = \left(\frac{\delta}{\Delta t_n} - \mu(1-\theta)\right) \cdot \phi_n + (\mathbf{f}_{n+1}\theta + \mathbf{f}_n(1-\theta)) \quad (8.155)$$

to recursively solve  $\phi_{n+1}$  at the new time plane  $n + 1$  from the preceding solution  $\phi_n$  at the previous time plane  $n$ , starting from the IC (8.130) at  $n = 0$ .

### 8.13.5 Predictor-Corrector Methods

A powerful alternative to the two-stage  $\theta$ -implicit/explicit recurrence solution (8.154) is the predictor-corrector method which was originally developed by Gresho et al. [209, 211, 212], hereafter referred to as GLS. This time integration method monitors the solution process via a local time truncation error estimation in which the time step size is cheaply and automatically varied in accordance with temporal accuracy requirements. It has been proven to be a cost-effective and robust procedure in that the time step size is increased whenever possible and decreased only if necessary. The predictor-corrector methods provide a rational mathematical basis for adaptively selecting the time step via error control. Such an adaptive time stepping is clearly superior to procedures based exclusively on empirical relations, e.g., a target-based or heuristic time stepping control as discussed in [124, 141, 582]. In the present analysis both 1st- and 2nd-order accurate variable step predictor-corrector schemes are of interest. The 1st-order accurate scheme refers to an explicit forward Euler (FE) formula as the predictor and the implicit backward Euler (BE) method as the corrector. It will hereafter be termed as the FE/BE predictor-corrector scheme. For the 2nd-order accurate method the explicit method is based on the Adams-Bashforth (AB) predictor, while the trapezoid rule (TR) is used as corrector with 2nd-order accuracy. It will be hereafter called as the AB/TR predictor-corrector scheme. The schemes are applied to the system of equations (8.129) or (8.131) written in a simplified form:

$$\dot{\phi} = \mathbf{r}(\phi) \quad \text{with} \quad \mathbf{r}(\phi) = \mathbf{f} - \mu \cdot \phi \quad (8.156)$$

### 8.13.5.1 GLS 1st-Order Forward Euler (FE)/Backward Euler (BE) Scheme

Predictor Solution

The FE scheme applied to  $\dot{\phi} = \mathbf{r}(\phi)$  gives, cf. (8.149),

$$\phi_{n+1}^p = \phi_n + \Delta t_n \mathbf{r}(\phi_n) = \phi_n + \Delta t_n \dot{\phi}_n \quad (8.157)$$

where the superscript  $p$  indicates the predictor values at the new time plane  $n + 1$ . The predictor provides a tentative solution at  $n + 1$ .

Corrector Solution

The BE corrector scheme applied to (8.156) is

$$\phi_{n+1} = \phi_n + \Delta t_n \mathbf{r}(\phi_{n+1}) = \phi_n + \Delta t_n \dot{\phi}_{n+1} \quad (8.158)$$

whose inversion yields the ‘acceleration’ vector

$$\dot{\phi}_{n+1} = \frac{\phi_{n+1} - \phi_n}{\Delta t_n} \quad (8.159)$$

to be used for preparing the next predictor step on the RHS of (8.157). The corrector (8.158) provides the actual solution at  $n + 1$ , which is commonly depart from the predictor solution (8.157).

Local Truncation Error (LTE) Estimation

The LTE  $\mathbf{d}_{n+1}$  is defined as the residual

$$\mathbf{d}_{n+1} = \phi_{n+1} - \phi(t_{n+1}) \quad (8.160)$$

between the approximate solution  $\phi_{n+1}$  and the exact solution  $\phi(t_{n+1})$  at the new time plane  $n + 1$ . Practically, we determine the exact solution via Taylor series analysis and assume that the exact solution is available at the beginning of the time step. We obtain for the FE formula

$$\begin{aligned} \mathbf{d}_{n+1}^p &= \phi_{n+1}^p - \phi(t_{n+1}) \\ &= \phi_n + \Delta t_n \dot{\phi}_n - \left( \phi_n + \Delta t_n \dot{\phi}_n + \frac{\Delta t_n^2}{2} \ddot{\phi}_n + \frac{\Delta t_n^3}{6} \ddot{\phi}_n + \dots \right) \\ &= -\frac{\Delta t_n^2}{2} \ddot{\phi}_n + \mathcal{O}(\Delta t_n^3) \end{aligned} \quad (8.161)$$

and for the BE scheme

$$\begin{aligned}
 \mathbf{d}_{n+1} &= \phi_{n+1} - \phi(t_{n+1}) \\
 &= \phi_n + \Delta t_n \dot{\phi}_{n+1} - \left( \phi_n + \Delta t_n \dot{\phi}_{n+1} - \frac{\Delta t_n^2}{2} \ddot{\phi}_{n+1} + \frac{\Delta t_n^3}{6} \dddot{\phi}_{n+1} - \dots \right) \\
 &= \frac{\Delta t_n^2}{2} \ddot{\phi}_{n+1} + \mathcal{O}(\Delta t_n^3) \\
 &= \frac{\Delta t_n^2}{2} \ddot{\phi}_n + \mathcal{O}(\Delta t_n^3)
 \end{aligned} \tag{8.162}$$

taking into account that the exact solution is available at  $t_n$  (and not at  $t_{n+1}$ ) by definition. From (8.161) and (8.162) it directly follows

$$\mathbf{d}_{n+1} = \frac{1}{2}(\phi_{n+1} - \phi_{n+1}^p) + \mathcal{O}(\Delta t_n^3) \tag{8.163}$$

by using  $\mathbf{d}_{n+1} = -\mathbf{d}_{n+1}^p + \mathcal{O}(\Delta t_n^3)$ . It provides an estimate of the LTE in a single BE step, where  $\phi_{n+1}$  and  $\phi_{n+1}^p$  are available from the corrector and predictor solution, respectively, at time plane  $n + 1$ .

### Time Step Selection

On this basis we can determine a useful formula for the acceptable size of the next time step as follows. From (8.162) we find

$$\frac{\|\mathbf{d}_{n+2}\|}{\|\mathbf{d}_{n+1}\|} = \left( \frac{\Delta t_{n+1}}{\Delta t_n} \right)^2 \frac{\|\ddot{\phi}_{n+1}\|}{\|\ddot{\phi}_n\|} \tag{8.164}$$

where  $\mathbf{d}_{n+1}$  is available from (8.163). The idea is now to keep the expected LTE at the next time plane  $n + 2$  equal to a pre-set (tolerable, target) error measure  $\epsilon$ , i.e.,  $\|\mathbf{d}_{n+2}\| = \epsilon$ . Since  $\ddot{\phi}_{n+1} = \ddot{\phi}_n + \mathcal{O}(\Delta t_n)$  (see<sup>9</sup>), (8.164) permits an estimate for the (potential) next time step size. Neglecting higher-order terms, we finally obtain from (8.164)

$$\Delta t_{n+1} = \Delta t_n \left( \frac{\epsilon}{\|\mathbf{d}_{n+1}\|} \right)^{1/2} \tag{8.165}$$

In this manner, the potential size of the next time step can be determined by the error norm  $\|\mathbf{d}_{n+1}\|$  estimated from the difference between the predicted and corrected solutions in (8.163). It can be used as a RMS error norm  $\|\mathbf{d}_{n+1}\|_{\text{RMS}}$ , cf. (8.28), or as a maximum error norm  $\|\mathbf{d}_{n+1}\|_{L_\infty}$ , cf. (8.29).

### 8.13.5.2 GLS 2nd-Order Adams-Bashforth (AB)/Trapezoid Rule (TR) Scheme

#### Predictor Solution

The 2nd-order AB formula applied to  $\dot{\phi} = r(\phi)$  is (e.g., see [211])

$$\phi_{n+1}^p = \phi_n + \frac{\Delta t_n}{2} \left[ \left( 2 + \frac{\Delta t_n}{\Delta t_{n-1}} \right) \dot{\phi}_n - \frac{\Delta t_n}{\Delta t_{n-1}} \dot{\phi}_{n-1} \right] \quad (8.166)$$

where  $\Delta t_n = t_{n+1} - t_n$  and  $\Delta t_{n-1} = t_n - t_{n-1}$ . It represents an explicit two-step method and requires two history vectors of acceleration at the current time plane  $\dot{\phi}_n$  and at the previous time plane  $\dot{\phi}_{n-1}$ . Since  $\dot{\phi}_{n-1}$  is additionally needed, the AB formula cannot be applied before the second step ( $n = 1$ ). Accordingly, the prediction has to be started with the FE predictor (8.157) and error estimation therefore begins at the completion of the second step.

#### Corrector Solution

The corrector step applied to (8.156) is based on the 2nd-order accurate TR, which reads

$$\phi_{n+1} = \phi_n + \frac{\Delta t_n}{2} [r(\phi_n) + r(\phi_{n+1})] = \phi_n + \frac{\Delta t_n}{2} (\dot{\phi}_n + \dot{\phi}_{n+1}) \quad (8.167)$$

whose inversion yields the history vector of acceleration

$$\dot{\phi}_{n+1} = \frac{2}{\Delta t_n} (\phi_{n+1} - \phi_n) - \dot{\phi}_n \quad (8.168)$$

to be used for preparing the next predictor step on the RHS of (8.166), where  $\dot{\phi}_n$  could be available from the previous application of the same equation. However, Bixler [48] has shown that the previous acceleration vector  $\dot{\phi}_n$  used in (8.168) can produce an oscillatory instability in the AB predictor in cases as a steady state is approached. Under such conditions  $\phi_{n+1} - \phi_n$  in (8.168) will go to zero, however,  $\dot{\phi}_n$  may not because of the recursive dependence on previous estimates of the acceleration vector. Bixler [48] proposed an alternative to  $\dot{\phi}_n$  in (8.168) by the following finite difference relation<sup>10</sup>:

<sup>10</sup>Truncated Taylor series expansions for  $\phi_{n-1}$  and  $\phi_{n+1}$  about  $t_n$  give:

$$\phi_{n-1} = \phi_n - \Delta t_{n-1} \dot{\phi}_n + \frac{\Delta t_{n-1}^2}{2} \ddot{\phi}_n, \quad \phi_{n+1} = \phi_n + \Delta t_n \dot{\phi}_n + \frac{\Delta t_n^2}{2} \ddot{\phi}_n$$



$$\dot{\phi}_n = \frac{\Delta t_{n-1}}{\Delta t_n + \Delta t_{n-1}} \left( \frac{\phi_{n+1} - \phi_n}{\Delta t_n} \right) + \frac{\Delta t_n}{\Delta t_n + \Delta t_{n-1}} \left( \frac{\phi_n - \phi_{n-1}}{\Delta t_{n-1}} \right) \quad (8.169)$$

which is also  $\mathcal{O}(\Delta t_n^2)$ . Inserting (8.169) into (8.168) the following formula is used to compute the acceleration vector for the next AB step (8.166), viz.,

$$\dot{\phi}_{n+1} = \left( 2 - \frac{\Delta t_{n-1}}{\Delta t_n + \Delta t_{n-1}} \right) \left( \frac{\phi_{n+1} - \phi_n}{\Delta t_n} \right) - \left( \frac{\Delta t_n}{\Delta t_n + \Delta t_{n-1}} \right) \left( \frac{\phi_n - \phi_{n-1}}{\Delta t_{n-1}} \right) \quad (8.170)$$

### Local Truncation Error (LTE) Estimation

In analogy to the FE/BE scheme in Sect. 8.13.5.1 the LTE is obtained for the AB predictor

$$\begin{aligned} \mathbf{d}_{n+1}^p &= \phi_{n+1}^p - \phi(t_{n+1}) \\ &= \phi_n + \frac{\Delta t_n}{2} \left[ \left( 2 + \frac{\Delta t_n}{\Delta t_{n-1}} \right) \dot{\phi}_n - \frac{\Delta t_n}{\Delta t_{n-1}} \dot{\phi}_{n-1} \right] - \phi(t_{n+1}) \\ &= \phi_n + \frac{\Delta t_n}{2} \left[ \left( 2 + \frac{\Delta t_n}{\Delta t_{n-1}} \right) \dot{\phi}_n - \frac{\Delta t_n}{\Delta t_{n-1}} (\dot{\phi}_n - \Delta t_{n-1} \ddot{\phi}_n + \frac{\Delta t_{n-1}^2}{2} \ddot{\phi}_n - \dots) \right] \\ &\quad - \left( \phi_n + \Delta t_n \dot{\phi}_n + \frac{\Delta t_n^2}{2} \ddot{\phi}_n + \frac{\Delta t_n^3}{6} \ddot{\phi}_n + \mathcal{O}(\Delta t_n^4) \right) \\ &= -\frac{1}{12} \left( 2 + 3 \frac{\Delta t_{n-1}}{\Delta t_n} \right) \Delta t_n^3 \ddot{\phi}_n + \mathcal{O}(\Delta t_n^4) \end{aligned} \quad (8.171)$$

where the exact solution is used at  $t_{n-1} = t_n - \Delta t_{n-1}$  to invoke Taylor series. Similarly, the LTE for the TR corrector results in

$$\begin{aligned} \mathbf{d}_{n+1} &= \phi_{n+1} - \phi(t_{n+1}) \\ &= \phi_n + \frac{\Delta t_n}{2} (\dot{\phi}_n + \dot{\phi}_{n+1}) - \phi(t_{n+1}) \\ &= \phi_n + \frac{\Delta t_n}{2} \left[ (\dot{\phi}_{n+1} - \Delta t_n \ddot{\phi}_{n+1} + \frac{\Delta t_n^2}{2} \ddot{\phi}_{n+1} - \dots) + \dot{\phi}_{n+1} \right] \\ &\quad - \left( \phi_n + \Delta t_n \dot{\phi}_{n+1} - \frac{\Delta t_n^2}{2} \ddot{\phi}_{n+1} + \frac{\Delta t_n^3}{6} \ddot{\phi}_{n+1} - \mathcal{O}(\Delta t_n^4) \right) \\ &= \frac{\Delta t_n^3}{12} \ddot{\phi}_{n+1} + \mathcal{O}(\Delta t_n^4) \\ &= \frac{\Delta t_n^3}{12} \ddot{\phi}_n + \mathcal{O}(\Delta t_n^4) \end{aligned} \quad (8.172)$$

---

Using the first expression to write  $\dot{\phi}_n = \frac{\phi_n - \phi_{n-1}}{\Delta t_{n-1}} + \frac{\Delta t_{n-1}}{2} \ddot{\phi}_n$  and inserting into the second formula with  $\frac{\Delta t_n}{2} \ddot{\phi}_n = \frac{\phi_{n+1} - \phi_n}{\Delta t_n} - \dot{\phi}_n$ , we obtain

$$\dot{\phi}_n = \frac{\phi_n - \phi_{n-1}}{\Delta t_{n-1}} + \frac{\Delta t_{n-1}}{\Delta t_n} \left( \frac{\phi_{n+1} - \phi_n}{\Delta t_n} - \dot{\phi}_n \right)$$

After some manipulations we finally find

$$\dot{\phi}_n = \frac{\Delta t_{n-1}}{\Delta t_n + \Delta t_{n-1}} \left( \frac{\phi_{n+1} - \phi_n}{\Delta t_n} \right) + \frac{\Delta t_n}{\Delta t_n + \Delta t_{n-1}} \left( \frac{\phi_n - \phi_{n-1}}{\Delta t_{n-1}} \right)$$

From (8.171) and (8.172) we can directly express the LTE of a single TR step as

$$\mathbf{d}_{n+1} = \frac{\phi_{n+1} - \phi_{n+1}^p}{3\left(1 + \frac{\Delta t_{n-1}}{\Delta t_n}\right)} + \mathcal{O}(\Delta t_n^4) \quad (8.173)$$

providing a function of the available predictor solution  $\phi_{n+1}^p$  and corrector solution  $\phi_{n+1}$  at the time plane  $n + 1$ .

### Time Step Selection

In analogy to Sect. 8.13.5.1 we can estimate the next time step size for the AB/TR scheme based on the requirement that an error norm for the next step should equal a pre-set tolerance measure  $\epsilon = \|\mathbf{d}_{n+2}\|$ . From (8.172) we find

$$\frac{\|\mathbf{d}_{n+2}\|}{\|\mathbf{d}_{n+1}\|} = \left(\frac{\Delta t_{n+1}}{\Delta t_n}\right)^3 \frac{\|\ddot{\phi}_{n+1}\|}{\|\ddot{\phi}_n\|} \quad (8.174)$$

where  $\mathbf{d}_{n+1}$  is known from (8.173). Neglecting higher-order terms and since  $\ddot{\phi}_{n+1} = \ddot{\phi}_n + \mathcal{O}(\Delta t_n)$ , we finally obtain from (8.174) the following relation

$$\Delta t_{n+1} = \Delta t_n \left(\frac{\epsilon}{\|\mathbf{d}_{n+1}\|}\right)^{1/3} \quad (8.175)$$

which is used to compute the potential next time step size.

### 8.13.5.3 Major Solution Steps and Tactic of Time Step Control

In Table 8.7 we summarize the major solution steps of the 1st-order accurate FE/BE and 2nd-order accurate AB/TR predictor-corrector schemes. In step 0 the time marching procedures are initialized by computing the acceleration vector  $\dot{\phi}_0$  based on the IC (8.130):  $\phi(t_0) = \phi_0$ . Furthermore, an initial time step size  $\Delta t_0$  is chosen, which should be kept sufficiently small. The error tolerance  $\epsilon$  is the only user-specified parameter to control the entire adaptive time marching process. It has significant effect on cost and accuracy. A too large value of  $\epsilon$  possesses a poor error estimate and the AB/TR becomes prone to oscillate when large time steps are used. Too small an  $\epsilon$ , however, will make the (albeit accurate) computations unacceptably expensive. In practice, it has been shown in many applications that a relative error per time step of  $\epsilon = 10^{-3}$ – $10^{-4}$  is quite optimal with respect to accuracy and performance. Note that a decrease of  $\epsilon$  by one power will approximately double the total number of time steps.

The computations per each time step consist of five major solution steps as listed in Table 8.7. At first, the predictor solution  $\phi_{n+1}^p$  at the new time plane  $n + 1$  has to be computed by using the explicit 1st-order accurate FE and 2nd-order accurate AB schemes. The AB scheme must start at  $n + 1$  as the FE scheme because the required acceleration vector  $\dot{\phi}_{n-1}$  is only available from the second step onward. Only at the second step  $n = 2$  the usual AB predictor procedure is started. All the predictors are cheaply computable and their extra effort is small. While these predictors are subsequently needed to estimate the truncation error for the time step control, they are also useful to linearize the governing PDE in the presence of nonlinearities. Step 2, the corrector  $\phi_{n+1}$ , is the actual solution of the governing PDE,  $\mathbf{O} \cdot \dot{\phi} + \mathbf{K} \cdot \phi = \mathbf{F}$ , via the implicit BE and TR schemes. If we additionally admit nonlinear dependencies in the form

$$\mathbf{O}(\phi) \cdot \dot{\phi} + \mathbf{K}(\phi) \cdot \phi = \mathbf{F}(\phi) \quad (8.176)$$

we can solve the linear system by using the predictor solution  $\phi_{n+1}^p$  at the new time plane  $n + 1$

$$\begin{aligned} \left( \frac{\mathbf{O}(\phi_{n+1}^p)}{\theta \Delta t_n} + \mathbf{K}(\phi_{n+1}^p) \right) \cdot \phi_{n+1} = \mathbf{O}(\phi_{n+1}^p) \cdot \left[ \frac{1}{\theta \Delta t_n} \phi_n + \left( \frac{1}{\theta} - 1 \right) \dot{\phi}_n \right] \\ + \mathbf{F}_{n+1}(\phi_{n+1}^p) \end{aligned} \quad (8.177)$$

where  $\theta = \frac{1}{2}$  for the TR scheme and  $\theta = 1$  for the BE scheme.

Once the corrector solution  $\phi_{n+1}$  is available at the new time plane  $n + 1$ , in step 3 the acceleration vectors  $\dot{\phi}_{n+1}$  can be computed, which will be needed in the following  $n + 2$  time step in the predictor and AB-corrector. With the known predictor and corrector solutions,  $\phi_{n+1}^p$  and  $\phi_{n+1}$ , respectively, in step 4 the LTE  $\mathbf{d}_{n+1}$  is determined at the current time plane  $n + 1$ . Appropriate error norms are applied to the vector  $\mathbf{d}_{n+1}$ . Commonly, the RMS  $L_2$  error norm (8.28)

$$\|\mathbf{d}_{n+1}\|_{\text{RMS}} = \left[ \frac{1}{N_P} \left( \frac{1}{\phi_{\max}^2} \sum_{j=1}^{N_P} d_{j,n+1}^2 \right) \right]^{\frac{1}{2}} \quad (8.178)$$

or the maximum  $L_\infty$  error norm (8.29)

$$\|\mathbf{d}_{n+1}\|_{L_\infty} = \frac{1}{\phi_{\max}} \max_j |d_{j,n+1}| \quad (8.179)$$

are chosen, where  $\phi_{\max}$  is the maximum value of the state variable  $\phi_{n+1}$  detected at the time plane  $n + 1$  to normalize the error vectors.

In step 5, the potential next time step size  $\Delta t_{n+1}$  is determined by using the just estimated error norm  $\|\mathbf{d}_{n+1}\| \in (\|\mathbf{d}_{n+1}\|_{\text{RMS}}, \|\mathbf{d}_{n+1}\|_{L_\infty})$  with the user-

**Table 8.7** Summarized solution steps of the 1st-order accurate FE/BE and 2nd-order accurate AB/TR predictor-corrector schemes applied to  $\mathbf{O} \cdot \dot{\phi} + \mathbf{K} \cdot \phi = \mathbf{F}$ , (8.129)

Step	$n$	FE/BE scheme $\mathcal{O}(\Delta t_n)$	AB/TR scheme $\mathcal{O}(\Delta t_n^2)$
0	Initialization	$n = 0$ $\mathbf{O} \cdot \dot{\phi}_0 = \mathbf{F} - \mathbf{K} \cdot \phi_0$ with given $\phi_0, \Delta t_0, \epsilon$	$\mathbf{O} \cdot \dot{\phi}_0 = \mathbf{F} - \mathbf{K} \cdot \phi_0$ with given $\phi_0, \Delta t_0, \epsilon$
1	Predictor	$n + 1$ $\phi_{n+1}^p = \phi_n + \Delta t_n \dot{\phi}_n$	$\phi_{n+1}^p = \begin{cases} \phi_n + \frac{\Delta t_n}{2} \left[ \left( 2 + \frac{\Delta t_n}{\Delta t_{n-1}} \right) \dot{\phi}_n - \frac{\Delta t_n}{\Delta t_{n-1}} \dot{\phi}_{n-1} \right] & \text{for } n > 1 \\ \phi_n + \Delta t_n \dot{\phi}_n & \text{for } n = 1 \end{cases}$
2	Corrector	$\left( \frac{\mathbf{O}}{\Delta t_n} + \mathbf{K} \right) \cdot \phi_{n+1} = \frac{\mathbf{O}}{\Delta t_n} \cdot \phi_n + \mathbf{F}_{n+1}$ $\dot{\phi}_{n+1} = \frac{\phi_{n+1} - \phi_n}{\Delta t_n}$	$\left( \frac{2\mathbf{O}}{\Delta t_n} + \mathbf{K} \right) \cdot \phi_{n+1} = \mathbf{O} \cdot \left( \frac{2}{\Delta t_n} \phi_n + \dot{\phi}_n \right) + \mathbf{F}_{n+1}$ $\dot{\phi}_{n+1} = \left( 2 - \frac{\Delta t_n}{\Delta t_{n-1}} \right) \left( \frac{\phi_{n+1} - \phi_n}{\Delta t_n} \right) - \left( \frac{\Delta t_n}{\Delta t_n + \Delta t_{n-1}} \right) \left( \frac{\phi_n - \phi_{n-1}}{\Delta t_{n-1}} \right)$
3	Updated accelerations		
4	Error estimation	$\mathbf{d}_{n+1} = \frac{1}{2}(\phi_{n+1} - \phi_{n+1}^p)$	$\mathbf{d}_{n+1} = \frac{\phi_{n+1} - \phi_{n+1}^p}{3 \left( 1 + \frac{\Delta t_{n-1}}{\Delta t_n} \right)}$
5	Time step control	$\Delta t_{n+1} = \Delta t_n \left( \frac{\epsilon}{\ \mathbf{d}_{n+1}\ } \right)^{1/2}$ if $\Delta t_{n+1} \geq \Delta t_n$ else if $\gamma \Delta t_n \leq \Delta t_{n+1} < \Delta t_n$ ( $\gamma = 0.85$ ) else if $\Delta t_{n+1} < \gamma \Delta t_n$	$\Delta t_{n+1} = \Delta t_n \left( \frac{\epsilon}{\ \mathbf{d}_{n+1}\ } \right)^{1/3}$ solution $\phi_{n+1}$ accepted, proceed with increased time step at $n + 2$ , solution $\phi_{n+1}$ accepted, but proceed with unchanged time step $\Delta t_{n+1} = \Delta t_n$ at $n + 2$ , solution $\phi_{n+1}$ cannot be accepted and must be rejected, repeat at $n + 1$ with reduced time step $\Delta t_n = \Delta t_{n+1}^{\text{red}}$ , where $\Delta t_{n+1}^{\text{red}} = \frac{\Delta t_n^2}{\Delta t_{n+1}} \left( \frac{\epsilon}{\ \mathbf{d}_{n+1}\ } \right)^\zeta$ ( $\zeta = 1$ for FE/BE and $\zeta = 2/3$ for AB/TR)

supplied error tolerance  $\epsilon$ . The following criteria are used to monitor the progress of solution:

1. If

$$\Delta t_{n+1} \geq \Delta t_n \quad (8.180)$$

the current solution  $\phi_{n+1}$  is accurate within the error bound defined by  $\epsilon$  and the increase of the time step is always accepted. In practice it has shown to be useful that the increase of the time step should be optionally constrained by further conditions. Firstly, the time step should not exceed a prescribed maximum size, i.e.,  $\Delta t_{n+1} \leq \Delta t_{\max}$ . Secondly, the rate for changing the time step size  $\mathcal{E} = \Delta t_{n+1}/\Delta t_n$  has also to be limited, where  $\mathcal{E} > 1$  can be 2, 3 or even more. Those constraints are beneficial to prevent inefficient oscillations in the time step size prediction. Then, the actually increased new time step is determined from

$$\Delta t_{n+1}^{\text{actual}} = \min(\Delta t_{n+1}, \Delta t_{\max}, \mathcal{E} \Delta t_n) \quad (8.181)$$

provided that  $\Delta t_{n+1}^{\text{actual}} \geq \Delta t_n$ .

2. Else if

$$\gamma \Delta t_n \leq \Delta t_{n+1} < \Delta t_n \quad (8.182)$$

where  $\gamma$  is typically 0.85, the solution  $\phi_{n+1}$  is accepted but the time step size is not changed, i.e.,  $\Delta t_{n+1} = \Delta t_n$ .

3. Else if

$$\Delta t_{n+1} < \gamma \Delta t_n \quad (8.183)$$

the solution  $\phi_{n+1}$  cannot be accepted within the required error tolerance  $\epsilon$  and has to be rejected. The current time step must be repeated with a reduced time step size. The reduced time step is computed from (8.165) and (8.175), respectively, by replacing  $\|\mathbf{d}\|_{n+1}$  and  $\Delta t_n$  with the just estimated  $\|\mathbf{d}\|_{n+2}$  and  $\Delta t_{n+1}$  to obtain

$$\Delta t_{n+1}^{\text{red}} = \frac{\Delta t_n^2}{\Delta t_{n+1}} \left( \frac{\epsilon}{\|\mathbf{d}^{n+1}\|} \right)^\zeta \quad (8.184)$$

where  $\zeta = 1$  for FE/BE and  $\zeta = 2/3$  for AB/TR scheme. The new solution restarted with this smaller time step is again tested against the error conditions and further step reduction can follow. However, up to 12 such reduction cycles are only allowed, then the algorithm signals to restart the overall time stepping procedure under stronger error bounds and initial time step (e.g., decrease  $\epsilon$  and/or  $\Delta t_0$ ).

After finishing solution step 5 for an acceptable solution  $\phi_{n+1}$ , the time stepping procedure proceeds to the next time plane  $n + 2$ , where it begins again with step 1 of Table 8.7. With the proposed predictor-corrector technique we can vary the size of the time step based solely on temporal accuracy requirements. Such an error-controlled adaptive time step selection strategy can follow the ‘physics’ of the underlying processes more intelligently and efficiently in comparison to heuristic rules. For example, the physics may require a small time step to follow a steep concentration or temperature profile over certain times or to adapt a sudden change in transient BC’s, while at later times it may be sufficient to follow a slow development of a flow or transport regime in time with reasonably large time steps. In either case, the predictor-corrector algorithm will usually automatically select the appropriate time step in a reliable manner, where the time step is increased whenever possible and decreased only when necessary.

### 8.13.6 Stability Properties

Any of the time marching recurrence schemes derived above can be written for the homogeneous solution (i.e., the source/sink of error is unimportant in the context so that we can assume  $F = \mathbf{0}$ ) in the form

$$\phi_{n+1} = \mathbf{A} \cdot \phi_n \quad (8.185)$$

where  $\mathbf{A}$  is the *amplification matrix*, which is given for the exact solution by the exponential decay relation (8.136)

$$\mathbf{A} = e^{-\mu \Delta t_n}, \quad \mu = \mathbf{O}^{-1} \cdot \mathbf{K} \quad (8.186)$$

and for the introduced time stepping schemes by the approximation

$$\begin{aligned} \mathbf{A} &= [\mathbf{O} + \theta \mathbf{K} \Delta t_n]^{-1} \cdot [\mathbf{O} - (1 - \theta) \mathbf{K} \Delta t_n] \\ &= [\boldsymbol{\delta} + \theta \boldsymbol{\mu} \Delta t_n]^{-1} \cdot [\boldsymbol{\delta} - (1 - \theta) \boldsymbol{\mu} \Delta t_n] \end{aligned} \quad (8.187)$$

in which  $\theta$  identifies the different recurrence algorithms. Table 8.5 lists the preferred linear single-step operators for particular  $\theta$  values. It is obvious as  $\mathbf{A}$  is recursively applied to each new vector  $\phi_n$ , the stability of the time integration method requires that any occurring approximation error must ultimately decay. Thus,  $\mathbf{A}$  must be a *bounded* operator and the time integration scheme is considered *stable* for  $|\mathbf{A}| < 1$ .

The stability of the time integration approximations can be further analyzed via modal decomposition. The solution  $\phi$  is expressed in terms of its linearly independent eigenvectors and eigenvalues by

$$\phi = \sum_{i=1}^{N_p} \varphi_i e^{-\delta \lambda_i t} \quad (8.188)$$

where  $\varphi_i$  are the eigenvectors and  $\lambda_i$  are the eigenvalues. Applying (8.188) to  $\dot{\phi} + \mu \cdot \phi = \mathbf{0}$ , (8.131), with  $\mathbf{f} = \mathbf{0}$ , it leads to the eigenproblem in the form

$$(\mu - \delta \lambda_i) \cdot \varphi_i = \mathbf{0}, \quad \forall i. \quad (8.189)$$

Since the eigenvectors have the properties of *modal orthogonality* in the form  $\varphi_j^T \cdot (\delta \cdot \varphi_i) = \delta_{ij}$ , we find after multiplying (8.189) by  $\varphi_j^T$

$$\varphi_j^T \cdot (\mu \cdot \varphi_i) = \lambda_i \delta_{ij} \quad (8.190)$$

showing that the eigenvectors are also orthogonal with respect to  $\mu$ . Now, we assume that the semidiscrete solution can be approximated in terms of the eigenvectors as

$$\phi = \sum_{i=1}^{N_p} \varphi_i y_i(t) \quad (8.191)$$

where  $y_i(t)$  represent the mode participation factors to be determined. Substituting (8.191) into  $\dot{\phi} + \mu \cdot \phi = \mathbf{0}$ , premultiplying with  $\varphi_j^T$  and applying the modal orthogonality conditions, leads to the result

$$\begin{aligned} \dot{y}_i [\varphi_j^T \cdot (\delta \cdot \varphi_i)] + y_i [\varphi_j^T \cdot (\mu \cdot \varphi_i)] &= 0 \quad \text{or} \\ \dot{y}_i \delta_{ij} + y_i \lambda_i \delta_{ij} &= 0 \quad \text{or} \\ \dot{y}_i + y_i \lambda_i &= 0 \quad \forall i. \end{aligned} \quad (8.192)$$

This modal formulation is very advantageous because it decouples the original equation into a sequence of scalar evolution equations for each mode  $i = 1, \dots, N_p$ . By applying the above time integration techniques, same as used for the original problem, now to the modal equations (8.192) we obtain similar to (8.185)

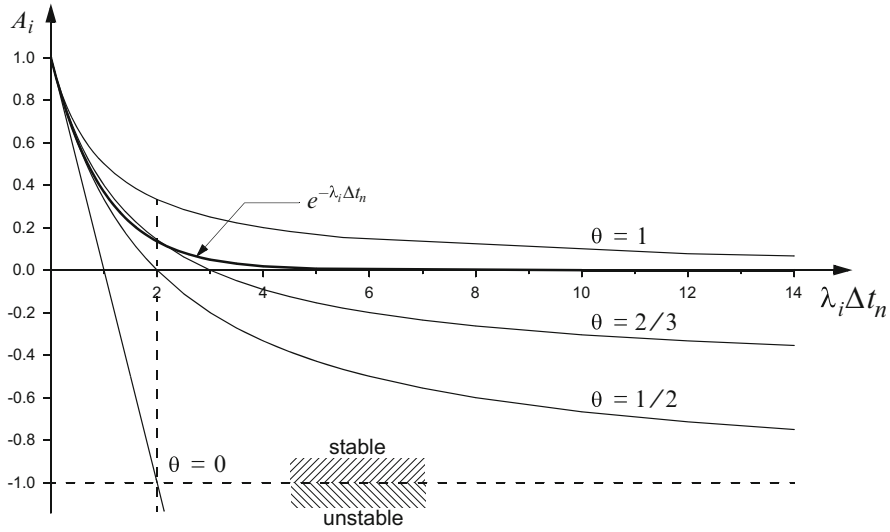
$$(y_i)_{n+1} = A_i (y_i)_n \quad (8.193)$$

where  $A_i$  is the scalar *amplification factor* of the  $i$ th mode, which is given for the exact solution by

$$A_i = e^{-\lambda_i \Delta t_n} \quad (8.194)$$

and for the introduced time stepping schemes by

$$A_i = \frac{1 - (1 - \theta) \lambda_i \Delta t_n}{1 + \theta \lambda_i \Delta t_n} \quad (8.195)$$



**Fig. 8.19** Amplification factor  $A_i$  (decay function) for various linear  $\theta$ -approximants (8.195) in comparison to the exact solution  $e^{-\lambda_i \Delta t_n}$  of a mode  $i$  with eigenvalue  $\lambda_i$

providing a scalar analog to (8.186) and (8.187), respectively. Taking into account that the eigenvalues  $\lambda_i$  cover the full eigenspectrum ranging between a maximum eigenvalue  $\lambda_{\max}$  and a minimum eigenvalue  $\lambda_{\min}$

$$\lambda_i = (\lambda_{\max}, \dots, \lambda_{\min}) \tag{8.196}$$

which can comprise several orders of magnitude, the requirement for stability is that the amplification factor  $A_i$  must be  $|A_i| < 1$ , i.e.,

$$-1 < \frac{1 - (1 - \theta)\lambda_i \Delta t_n}{1 + \theta\lambda_i \Delta t_n} < 1 \tag{8.197}$$

holding for all eigenvalues  $\lambda_i$  of the system.

Figure 8.19 illustrates how the amplification factor  $A_i$  of a mode  $i$  varies with  $\lambda_i \Delta t_n$  for various  $\theta$  of the four difference operators in comparison to the exact exponential decay  $e^{-\lambda_i \Delta t_n}$ , where  $\theta \in (0, \frac{1}{2}, \frac{2}{3}, 1)$  represents the explicit FE, the implicit TR (Crank-Nicolson), the implicit Galerkin and the fully implicit BE scheme, respectively. We easily recognize that the right-hand inequality of (8.197) imposes no restrictions on values of  $\lambda_i \Delta t_n$  or  $\theta$ . However, the left-hand inequality requires for stability that

$$(1 - 2\theta)\lambda_i \Delta t_n < 2 \tag{8.198}$$



when  $\theta < \frac{1}{2}$ . On the other hand, we see that there are no restrictions for  $\theta \geq \frac{1}{2}$ . Such algorithms satisfying (8.198) independent of the chosen time step size  $\Delta t_n$  are called *conditionally stable* ( $\mathcal{A}$ -stable). The TR (Crank-Nicolson) ( $\theta = \frac{1}{2}$ ), the Galerkin scheme ( $\theta = \frac{2}{3}$ ) and the implicit BE method ( $\theta = 1$ ) belong to this category. In contrast, the explicit FE scheme ( $\theta = 0$ ) is stable only if  $\lambda_i \Delta t_n < 2$ , otherwise  $A_i$  predicts unbounded (unstable) solutions with  $A_i \rightarrow -\infty$  as evidenced in Fig. 8.19. Therefore, an explicit scheme is not an  $\mathcal{A}$ -stable method. As also shown in Fig. 8.19 the 2nd-order accurate TR scheme ( $\theta = \frac{1}{2}$ ) fits very well with the exact solution and furnishes highest accuracy for  $\lambda_i \Delta t_n < 1$  in comparison to all other linear single-step schemes. The implicit BE scheme ( $\theta = 1$ ) approaches to the exact solution  $A_i \rightarrow 0$  for very large time steps  $\lambda_i \Delta t_n \rightarrow \infty$ , while the Galerkin and the TR schemes satisfy  $A_i \rightarrow -1$ . Apparently, the Galerkin method ( $\theta = \frac{2}{3}$ ) exhibits an optimal approximation behavior over the entire  $\lambda_i \Delta t_n$ -range.

The  $\mathcal{A}$ -stable time stepping algorithms satisfying (8.198) ensure boundedness and thus unconditional stability independently of the time step  $\Delta t_n$ . However,  $\mathcal{A}$ -stability is not sufficient to ensure smooth and wiggle-free (nonoscillatory) solutions. In fact, all algorithms which admit a negative amplification  $A_i$  (see Fig. 8.19) are prone to oscillatory behaviors if  $\Delta t_n$  becomes too large. The condition for nonoscillation (called  $\mathcal{L}$ -stability) requires  $0 < A_i < 1$ . The bound  $A_i > 0$  gives with (8.195) the criterion

$$(1 - \theta)\lambda_i \Delta t_n < 1 \quad (8.199)$$

to ensure nonoscillatory solutions. It is obvious that only the BE scheme ( $\theta = 1$ ) can satisfy this condition for arbitrary step sizes  $\Delta t_n$  assuring  $A_i \rightarrow 0$  for  $\lambda_i \Delta t_n \rightarrow \infty$ . Unlikely, in the TR scheme ( $\theta = \frac{1}{2}$ ) the time step has to be restricted by a critical time step  $\Delta t_n^{\text{crit}}$  such as

$$\Delta t_n < \Delta t_n^{\text{crit}} = \frac{2}{\lambda_{\max}} \quad (8.200)$$

to avoid oscillations in the solution (known as Crank-Nicolson noise [568]), which must be controlled by the maximum eigenvalue  $\lambda_{\max}$ .

Different methods exist for analyzing stability. One is the matrix method in which the eigenvalues of the matrix are estimated. To get a first (but simple) assessment of characteristic eigenvalues  $\lambda_i$  which are important to determine time step limitations, such as (8.200), we can use  $\delta\lambda_i = \mu$  from (8.189) and estimate  $\mu = \mathbf{O}^{-1} \cdot \mathbf{K}$  on an element level basis, e.g., [590]. Let us consider for the sake of simplicity the 1D linear element (cf. Table G.1a of Appendix G), for which the element matrices have been derived in Appendix H. We find for the diagonal contributions of the (lumped) mass matrix  $O_{ii}$  and diffusion matrix  $C_{ii}$  from (H.7) at a (global) mode  $i$ , assuming a uniform meshing with element length  $\Delta x$  and constant parameters (storage –  $\mathcal{R}$ , diffusion –  $D$ ):

$$\lambda_i \approx \frac{C_{ii}}{O_{ii}} = \frac{2D}{\Delta x} \bigg/ \frac{\hat{\mathcal{R}}\Delta x}{2} \Big|_{\text{LM}} = \frac{4D}{\hat{\mathcal{R}}\Delta x^2} \quad (8.201)$$

written without advection, where we also drop the source/sink terms appearing in (H.7) because stability is independent of the forcing functions. Then, the assessment of condition (8.200) for the TR scheme yields<sup>11</sup>

$$\Delta t_n < \Delta t_n^{\text{crit}} \approx \frac{\hat{\mathcal{R}}\Delta x^2}{2D} \quad (8.202)$$

which indicates that the critical time step is proportional to  $\Delta x^2$  for a diffusion-dominant problem  $D > 0$ . We see that the smallest element size  $\Delta x$  dictates the criterion. In practice, however, the TR (or Crank-Nicolson) criterion (8.202) is commonly insignificant. Possible oscillations produced by the TR scheme are strictly bounded and small if linear finite elements are used. This is exemplified in Fig. 8.20 illustrating slight, but quickly damped oscillations in the temporal development of the solution for the TR (Crank-Nicolson) scheme using a constant time step larger than the critical step size (8.202). It is shown in [568] that Crank-Nicolson noise is more significant for finite elements of quadratic or higher-order type.

For a further analysis let us consider the spatio-temporal discretization of the system (8.154) for a simplified 1D problem. Again, we use linear elements thoroughly described in Sect. H.1 of Appendix H for a 1D domain as shown in Fig. 8.21. As indicated in Fig. 8.21 the assembly of the elements leads to a tridiagonal global matrix, where the final discrete equations can be expressed for the row of global interior node  $i$  by using the specific matrix entries of (H.8) and (H.10). For simplicity we drop source/sink and boundary terms and find for (8.154) the following discrete equations of the 3-node ( $i + 1, i, i - 1$ )–stencil:

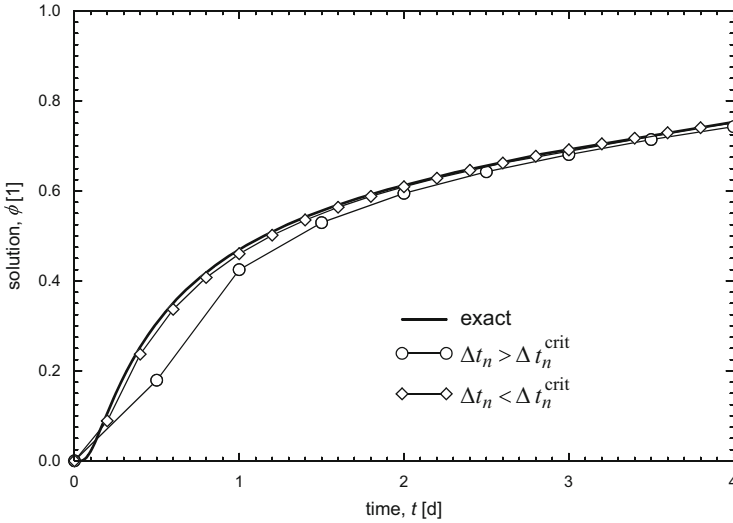
$$\begin{aligned} & \left[ \frac{\hat{\mathcal{R}}}{6} - \Delta t_n \theta \left( \frac{D}{\Delta x^2} - \frac{q}{2\Delta x} - \frac{\vartheta}{6} \right) \right] \phi_{i+1,n+1} + \left[ \frac{2\hat{\mathcal{R}}}{3} + \Delta t_n \theta \left( \frac{2D}{\Delta x^2} + \frac{2\vartheta}{3} \right) \right] \phi_{i,n+1} + \\ & \left[ \frac{\hat{\mathcal{R}}}{6} - \Delta t_n \theta \left( \frac{D}{\Delta x^2} + \frac{q}{2\Delta x} - \frac{\vartheta}{6} \right) \right] \phi_{i-1,n+1} = \left[ \frac{\hat{\mathcal{R}}}{6} + \Delta t_n (1 - \theta) \left( \frac{D}{\Delta x^2} - \frac{q}{2\Delta x} - \frac{\vartheta}{6} \right) \right] \phi_{i+1,n} + \\ & \left[ \frac{2\hat{\mathcal{R}}}{3} - \Delta t_n (1 - \theta) \left( \frac{2D}{\Delta x^2} + \frac{2\vartheta}{3} \right) \right] \phi_{i,n} + \left[ \frac{\hat{\mathcal{R}}}{6} + \Delta t_n (1 - \theta) \left( \frac{D}{\Delta x^2} + \frac{q}{2\Delta x} - \frac{\vartheta}{6} \right) \right] \phi_{i-1,n} \end{aligned} \quad (8.203)$$

written for the CM matrix and

---

<sup>11</sup>Since  $O_{ii}^e = \frac{\hat{\mathcal{R}}\Delta x}{3} \Big|_{\text{CM}}$  for a consistent mass (CM) matrix, the critical time step becomes even smaller:

$$\Delta t_n < \Delta t_n^{\text{crit}} \approx \frac{\hat{\mathcal{R}}\Delta x^2}{3D}$$



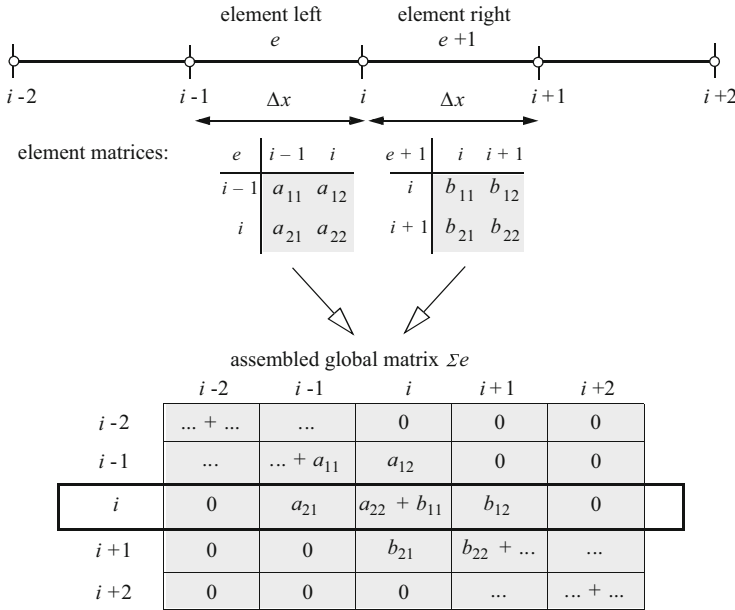
**Fig. 8.20** Example problem of a transient diffusion in a 1D domain of  $L = 1$  m discretized by five linear elements with  $\Delta x = 0.2$  m,  $\mathcal{R} = 1$  and  $D = 10^{-6} \text{ m}^2 \text{ s}^{-1}$  showing the history of Crank-Nicolson approximate solution  $\phi(t)$  at  $x = 0.3$  m in time for a constant time step  $\Delta t_n$  larger and smaller than the critical time step  $\Delta t_n^{\text{crit}} = 0.23$  d, (8.202), in comparison to the exact solution

$$\begin{aligned}
 & -\Delta t_n \theta \left( \frac{D}{\Delta x^2} - \frac{q}{2\Delta x} - \frac{\vartheta}{6} \right) \phi_{i+1,n+1} + \left[ \mathcal{R} + \Delta t_n \theta \left( \frac{2D}{\Delta x^2} + \frac{2\vartheta}{3} \right) \right] \phi_{i,n+1} - \\
 \Delta t_n \theta \left( \frac{D}{\Delta x^2} + \frac{q}{2\Delta x} - \frac{\vartheta}{6} \right) \phi_{i-1,n+1} &= \Delta t_n (1 - \theta) \left( \frac{D}{\Delta x^2} - \frac{q}{2\Delta x} - \frac{\vartheta}{6} \right) \phi_{i+1,n} + \\
 \left[ \mathcal{R} - \Delta t_n (1 - \theta) \left( \frac{2D}{\Delta x^2} + \frac{2\vartheta}{3} \right) \right] \phi_{i,n} &+ \Delta t_n (1 - \theta) \left( \frac{D}{\Delta x^2} + \frac{q}{2\Delta x} - \frac{\vartheta}{6} \right) \phi_{i-1,n}
 \end{aligned} \tag{8.204}$$

written for the LM matrix, where the diffusion  $D$ , the advective flux  $q$ , the storage  $\mathcal{R}$ , the decay rate  $\vartheta$  and the length of the linear 1D element  $\Delta x$  are assumed constant.

Based on discrete equations such as in form of (8.203) or (8.204) a very common and most useful method for analyzing stability is the classical Fourier analysis, called *von Neumann stability analysis*, e.g., [149, 209, 376]. Von Neumann stability results necessary conditions at least on a uniform mesh, regardless of BC's. On this basis it can be shown that nonoscillatory solutions for the TR (or Crank-Nicolson) method are bound to the pair of inequalities

$$Cr < Pg < 1/Cr, \quad \text{or} \quad Cr < Pg \quad \text{and} \quad Cr < 1/Pg \tag{8.205}$$



**Fig. 8.21** Node numbering and assembly to a tridiagonal global matrix for a uniform mesh of 1D linear elements

in which

$$Cr = \frac{q^* \Delta t_n}{\Delta x}, \quad \text{with } q^* = q/\hat{\mathcal{R}} \tag{8.206}$$

and

$$Pg = \frac{q^* \Delta x}{2D^*}, \quad \text{with } D^* = D/\hat{\mathcal{R}} \tag{8.207}$$

derived for the LM matrix, where  $Cr$  is the *Courant number* (named after the famous paper by Courant et al. [105]) and  $Pg$  defines the grid (mesh) *Péclet number*. The first limit  $Cr < Pg$  is the ‘diffusion limit’  $\Delta t_n < \hat{\mathcal{R}} \Delta x^2 / (2D)$  when  $Pg < 1$  as already stated in (8.202) and the second one  $PgCr < 1$  represents the ‘advection-diffusion limit’  $\Delta t_n < 2\hat{\mathcal{R}}D/q^2$  when  $Pg \geq 1$ . The second Crank-Nicolson criterion  $PgCr < 1$  was also found by Perrochet and Béroed [415] by using a matrix method.

While the discrete equations (8.203) and (8.204) are  $\mathcal{A}$ -stable for  $\theta \geq \frac{1}{2}$ , i.e., stability is guaranteed for any time step  $\Delta t_n$ , nonoscillatory results require additional limits which directly follow from (8.203) and (8.204). It can be easily seen from (8.203) that the term  $[\frac{\hat{\mathcal{R}}}{6} - \Delta t_n \theta (\frac{D}{\Delta x^2} + \frac{|q|}{2\Delta x} - \frac{\vartheta}{6})]$  must be negative to avoid oscillations in a CM formulation. It leads to a restriction for a minimum time step size, viz.,

$$\Delta t_n > \frac{\acute{R}\Delta x^2}{\theta(6D + 3|q|\Delta x - \vartheta\Delta x^2)} \quad \text{for } \theta \geq \frac{1}{2} \quad (8.208)$$

and additionally it should be required that  $(6D + 3|q|\Delta x - \vartheta\Delta x^2) > 0$  which arises a further constraint related to the decay rate

$$\vartheta < \frac{3}{\Delta x^2}(2D + |q|\Delta x) \quad (8.209)$$

or a limit for the element length determined by the decay rate  $\vartheta$

$$\Delta x < \frac{3|q| + \sqrt{9q^2 + 24D\vartheta}}{2\vartheta} \quad \text{if } \vartheta > 0 \quad (8.210)$$

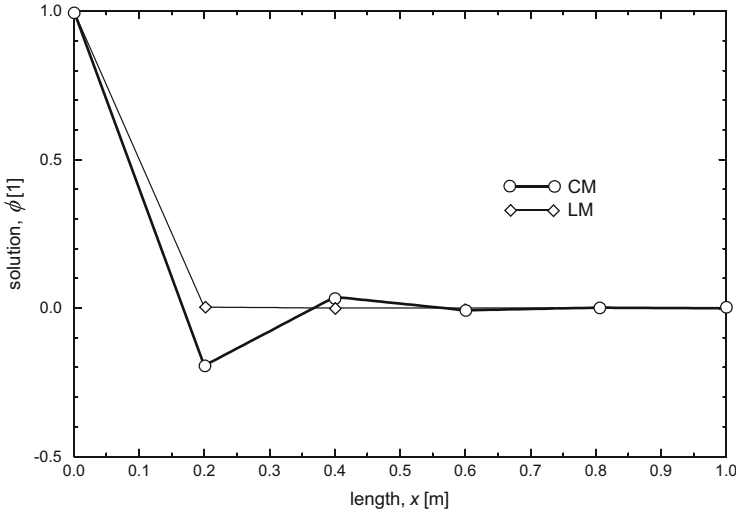
The physical interpretation of a minimum time step size (8.208) for a consistent mass formulation is that the mesh is too coarse to transmit and distribute a quantity to the nodes of an element in a very short time interval so that bounded oscillations become unavoidable. However, we observe from (8.204) that such a minimum time step constraint does not exist for mass lumping, since  $[-\Delta t_n\theta(\frac{D}{\Delta x^2} + \frac{|q|}{2\Delta x} - \frac{\vartheta}{6})]$  is always negative here, provided that (8.209) or (8.210) are satisfied. That means, the restriction for the decay rate (8.209) or (8.210) are present both for CM and LM matrix formulations. [Note that it could be possible to lump also the reaction matrix term similar to the LM matrix as discussed in Sect. 8.13.2, then the restrictions (8.209) or (8.210) would disappear.] We illustrate in Fig. 8.22 for a 1D example problem the oscillatory effect of CM if the time step is too small violating (8.208) and that mass lumping produces nonoscillatory results for the same time step. In practice, however, the restrictions (8.208), (8.209) or (8.210) are not really crucial because the mesh coarseness is commonly not achieved and even if oscillations of this type are caused by too small time step sizes in a coarse mesh they are quickly damped out in progressing the time steps. Nevertheless, due to the higher accuracy the CM formulation (cf. Sect. 8.13.2) is generally the first choice in the present finite element analysis.

The stability criterion (8.198) represents a serious restriction for the explicit FE scheme ( $\theta = 0$ ). A comprehensive stability analysis is given by Hindmarsh et al. [250]. The lumped explicit FE scheme becomes unstable, unless

$$Cr < \min(Pg, 1/Pg, 1) \quad (8.211)$$

or

$$\Delta t_n < \min\left(\frac{\acute{R}\Delta x^2}{2D}, \frac{2\acute{R}D}{q^2}, \frac{\acute{R}\Delta x}{q}\right) \quad (8.212)$$



**Fig. 8.22** Resulting distributions in a 1D domain of length  $L = 1$  m discretized by five linear elements with  $\Delta x = 0.2$  m,  $D = 10^{-6} \text{ m}^2 \text{ s}^{-1}$ ,  $q = 1 \text{ m d}^{-1}$ ,  $\hat{\mathcal{R}} = 1$  and  $\vartheta = 3 \cdot 10^{-4} \text{ s}^{-1}$ , satisfying the limit (8.209). Results are obtained for a full implicit scheme ( $\theta = 1$ ) at the first time step of  $\Delta t_n = 10^{-2} \text{ d}$  for CM and LM formulations, where only CM implies bounded oscillations

The first ‘diffusion limit’  $\Delta t_n < \hat{\mathcal{R}} \Delta x^2 / (2D)$  governs when  $Pg < 1$ , the second ‘advection-diffusion limit’ is restrictive when  $Pg \geq 1$ . The third restriction

$$Cr < 1 \tag{8.213}$$

is the *Courant-Friedrichs-Lewy (CFL) condition* [105], which is always a necessary condition for the stability of explicit schemes. If diffusion dominates the diffusion limit  $\Delta t_n < \Delta x^2 / (2D^*)$  possesses a terrible restriction for any explicit method. It means in practical terms: Assume  $L$  is the characteristic length of the computational domain, then the simulation time required for the full transient is  $t_{\text{end}} \simeq L^2 / D^*$ . For a typical (thermal) diffusivity  $D^*$  of  $10^{-6} \text{ m}^2 \text{ s}^{-1}$ , a length of  $L = 10$  m and the smallest element length of  $\Delta x = L / 1,000 = 10^{-2} \text{ m}$ , the diffusion limit (8.212) requires  $\Delta t_n < 50 \text{ s}$ . Since  $t_{\text{end}} = 10^8 \text{ s}$ , about  $2 \cdot 10^6$  time steps are required to perform the complete simulation with an explicit FE scheme. If we halve  $\Delta x$  the required time steps increase to  $8 \cdot 10^6 \text{ s}$ . This shows the serious drawback of explicit schemes, in particular for diffusion problems, where a very large number of tiny time steps becomes necessary, albeit each time step is computationally cheap because no equation systems must be solved. In contrast,  $\mathcal{A}$ -stable implicit schemes having no stability limitations can solve a diffusion problem with acceptable accuracy in, let’s say, less than 100 time steps, however, each time step is more expensive due to the solution of the equation system. Nevertheless, the implicit time stepping schemes have shown clearly superior to explicit methods, at least for diffusion-dominant problems, due to their clearly higher computational performance and robustness.

On the other hand, for dominant advection ( $Pg \gg 1$ ), the CFL condition (8.213) becomes important for explicit schemes

$$\Delta t_n < \frac{\Delta x}{q^*} \quad (8.214)$$

which implies only a linear dependence on  $\Delta x$ . For example, choosing  $L = 10$  m,  $\Delta x = 10^{-2}$  m and  $q^* = 10^{-4}$  m s<sup>-1</sup>, the time step limit (8.214) requires  $\Delta t_n < 100$  s and accordingly  $10^3$  time steps are needed to perform a full transient for the advection problem up to  $t_{\text{end}} \simeq L/q^* = 10^5$  s. It illustrates that the performance of explicit schemes considerably improves for advection-dominated (hyperbolic) problems (Sect. 8.3) and could be in fact more affordable compared to implicit techniques. But, taking into consideration more complex flow situations where locally (in space or time) the advection can be small compared to the diffusivity or even zero, the possible benefit of the computational performance of explicit methods can easily get lost again due to the strong limitation (8.212) in the time step control.

Finally, the advection-diffusion limit ( $Cr < 1/Pg$  or  $\Delta t_n < 2\mathcal{R}D/q^2$ ) can be too restrictive for advection-dominated simulations via explicit methods [250]. However, it is common practice to incorporate the temporal truncation error and upwind stabilization techniques (see following Sect. 8.14), where the physical diffusion  $D^*$  is artificially increased by  $q^{*2}\Delta t_n/2$  and by  $q^*\Delta x/2$ , respectively [91,250]. Then, the explicit method becomes tractable for hyperbolic problems with the changed advection-diffusion limits according to [250]

$$Cr < \frac{\sqrt{1 + 4Pg^2} - 1}{2Pg} \quad \text{and} \quad Cr < \frac{Pg}{1 + Pg} \quad (8.215)$$

In a resumé, due to the desired generality and robustness of the finite element strategy we prefer usually implicit time stepping schemes for the present class of problems. Explicit techniques (such as FE and AB) only occur in the context of predictor-corrector time marching schemes, for which no time step restrictions exist because the corrector solutions are generally implicit in form of the  $\mathcal{A}$ -stable BE or TR methods.

## 8.14 Upwinding

### 8.14.1 Pros and Cons of Upwind Methods

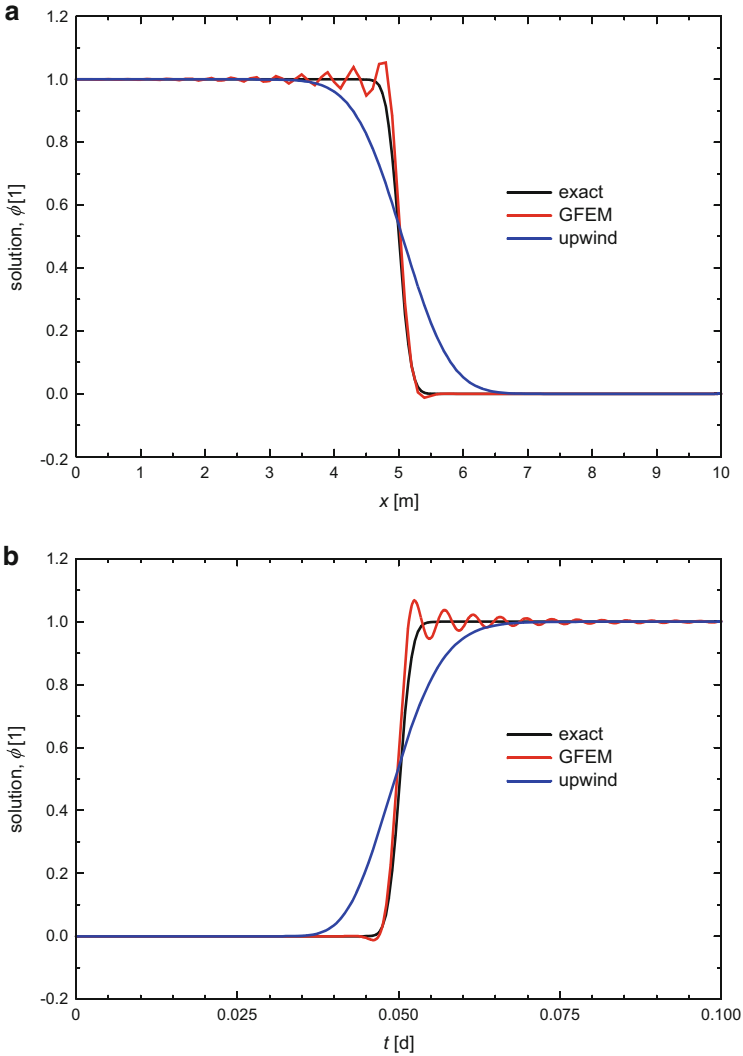
In computing transport-flow processes the FEM must be applied in situations where the advection dominates over diffusion/dispersion. For the numerical solution stability and boundedness (definitions given in Sect. 1.2.2) should be guaranteed. Numerical solutions should lie within proper bounds. Physically, nonnegative quantities (e.g., density, mass concentration, absolute temperature) should always

be positive. But, boundedness is difficult to guarantee under all circumstances. Unbounded solutions can occur on too coarse meshes in form of *wiggles*, i.e., oscillatory results generally occurring in a node-to-node manner which overshoot and undershoot the solution (Fig. 8.23). GFEM (and the equivalent central difference approximations) are prone to generate those spurious oscillations in space if the chosen mesh is inappropriate. In FDM it is popular to approximate the advective terms of the ADE by first-order onesided (flow direction-biased) differences, a process often referred to as *upwinding*. However, upwind methods precluding unwanted oscillations have disadvantages with regard to accuracy. It is to emphasize that stability does not imply accuracy – although it is true that instability implies inaccuracy. The resort to upwinding is usually a reduction of accuracy in favor of stability, where wiggles are artificially suppressed via damping mechanisms.

To treat advection-dominated transport problems by the FEM various upwind formulations have been developed in past. Pioneering work was given by Christie et al. [82], Heinrich et al. [236], Heinrich and Zienkiewicz [235] and Zienkiewicz et al. [595]. Asymmetric weighting functions were introduced such that the element upstream of a node is weighted more heavily than the element located downstream of a node equivalent to an upwind differencing. This type of upwind distortion of the weighting function represents a generalization of the standard (Galerkin-based) FEM and is called Petrov-Galerkin finite element method (PGFEM), cf. Table 8.1. Hughes [266] has shown that the upwind effect can also be achieved by asymmetry in the numerical quadrature rule for the advection terms. It was recognized that the PGFEM stabilization is equivalent to adding artificial (numerical) diffusion to the GFEM, termed as *balancing diffusion*, e.g., [307]. Unfortunately, many of the upwind methods reveal over-diffusive properties and there was a demand for alternative upwind techniques possessing reduced spurious numerical diffusion. While a scalar artificial diffusion often suffers from a considerable smearing effects [446, 541], the streamline-upwind (SU) method adds artificial diffusion only in the flow direction and not transversely [57]. The upwind finite element strategy have been further developed in a number of works, see e.g., [131, 149, 267–269, 272–276, 584, 585, 592]. The Petrov-Galerkin least square (PGLS) FEM [276, 385] appeared as a promising stabilization technique. This procedure results in an artificial diffusion concept of a built-in streamline-like upwinding similar to the SU method, however, leads to symmetric matrix systems. However, it has been found [274] that the streamline is not always the appropriate upwind direction. A generalization of the streamline concept in form of adding an additional discontinuity-capturing term was presented by Hughes and Mallet [269]. The shock capturing (SC) method applied to finite elements has been developed by Johnson et al. [292] and Codina [90, 92, 93].

It becomes clear that upwinding is a compromise between the requirements of accuracy and stability. There is (also) ‘no free lunch’ in numerics: stability must be paid by a reduction of accuracy. The question arises how much reduction in accuracy is acceptable or to which level wiggling can be tolerated. The most important pros and cons of upwind methods can be summarized as follows:





**Fig. 8.23** (a) Profiles and (b) breakthrough curves for 1D advection-dominant transport in a uniform flow field obtained by GFEM and upwind method simulated with AB/TR predictor-corrector time stepping on a coarse mesh consisting of 100 linear elements with  $\Delta x = 0.1$  m,  $D^* = 2.5 \cdot 10^{-6} \text{ m}^2 \text{ s}^{-1}$  and  $Pg = 23.15$  in comparison to the exact solution.<sup>12</sup> Oscillations are generated for GFEM, while smooth and overdiffusive solution results for upwinding, where physical diffusion  $D^*$  is artificially increased to  $D^* + q^* \Delta x / 2 = 6.04 \cdot 10^{-5} \text{ m}^2 \text{ s}^{-1}$ , which is more than 24 times higher

<sup>12</sup>The analytical (exact) solution of a 1D ADE is [71], p. 388, [540], cf. also Sect. 12.5.1

### Pros

- GFEM has serious deficiencies in solving problems with dominant advection, which are prone to generate spurious (nonphysical) node-to-node oscillations. Upwinding can *stabilizes* the solutions and is beneficial to obtain realistic (though not always accurate) solutions.
- Upwind methods allow *efficient* solutions without the ultimate need for fine (sometimes extremely dense) meshes.
- Upwinding makes difficult problems *computable* under given computational constraints. Extremely fine meshes and expensive computations could be caused for tough physical situations (e.g., shock-like front displacements of mass or energy, very thin boundary layers, high density contrasts in a large-scale problem) if upwind methods would not be admitted.
- There are certain situations where any wiggles in the solution become absolutely devastating and would totally preclude the possibility of obtaining a solution, e.g., strong advection in multispecies mass transport processes with nonlinear chemical reaction.

### Cons

- ‘Don’t suppress the wiggles – they’re telling you something!’ as stated in the famous paper by Gresho and Sani [208] who oppose, in principle, any artificial damping measures by upwinding: Wiggles are usually a signal that the spatial (and temporal) discretization is poor and some mesh refinements (at least locally) are required to obtain a physically adequate solution.
- A positive aspect of wiggles is that in signaling improper discretization they present *self-diagnosis property*. A method with such a self-diagnostic property is often superior to schemes which give smooth, and totally wiggle-free, but inaccurate and possibly overdamped solutions for any discretization.
- Upwinding is a method of damping and smoothing. *It solves the problem by changing the physics of the problem*. Robustness is obtained at the expense of accuracy. Diffusion is artificially increased in dependence on the chosen mesh, i.e., the solution becomes mesh-dependent. With other words: For a coarse mesh the solution is independent of the physical diffusion and can depart from the physics of the original problem. Upwinding could be only acceptable

---


$$\phi(x, t) = \phi_0 + \frac{1}{2}(\phi_D - \phi_0) \left[ \operatorname{erfc}\left(\frac{x - q^*t}{2\sqrt{D^*t}}\right) + \exp\left(\frac{xq^*}{D^*}\right) \operatorname{erfc}\left(\frac{x + q^*t}{2\sqrt{D^*t}}\right) \right]$$

valid for the IC:  $\phi(x, 0) = \phi_0$ , and BC’s:  $\phi(0, t) = \phi_D$  and  $\frac{\partial \phi}{\partial x}(\infty, t) = 0$ , where

$$\operatorname{erfc}(a) = \frac{2}{\sqrt{\pi}} \int_a^\infty \exp(-\xi^2) d\xi$$

is the complementary error function [71],  $\phi_0 = 0$  is the used initial value and  $\phi_D = 1$  is the used Dirichlet-type BC at  $x = 0$ . Note that for evaluating the analytical  $\exp(\cdot)\operatorname{erfc}(\cdot)$  expression the more suitable  $\operatorname{exf}(\cdot, \cdot)$  function is applied which will be further discussed in Sect. 12.5.1.

and reasonably accurate if the numerical diffusion is significantly less than the physical diffusion.

- Upwinding is potentially dangerous because it often leads to a false sense of security: ‘Any mesh works for any advection-diffusion relation’. Upwind schemes can damp more than just wiggles; this is particularly true for more complex nonlinear problems. J. Ferziger (noted in [209]) stated: *The greatest disaster one can encounter in computation is not instability or lack of convergence, but results that are good enough to be believable but bad enough to cause trouble.*

The decision between the pros and cons is often not easy. There is, unfortunately, no panacea, but the practitioner should be aware of the necessary compromises involved, and use a given method with due caution and ‘healthy skepticism’. Finally, our recommended strategy is to solve a problem without upwinding whenever possible, and to resort to an upwind method only if necessary and unavoidable. In the following, appropriate upwind methods available in FEFLOW will be described.

### 8.14.2 Petrov-Galerkin Finite Element Method (PGFEM)

The most common technique for introducing the upwind concept into the FEM is the Petrov-Galerkin finite element method (PGFEM), where the element weighting functions differ from the element basis functions  $w_I^e \neq N_I^e$  (cf. Table 8.1) and are appropriately designed to incorporate asymmetry with respect to the flow field. The weighting functions of an element  $e$  are constructed in general as

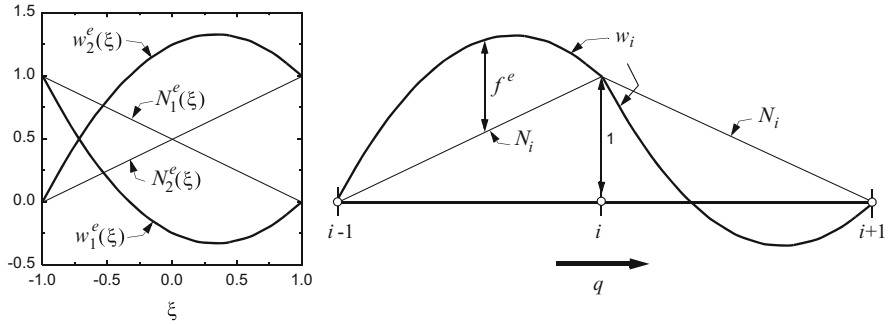
$$w_I^e(\boldsymbol{\eta}) = N_I^e(\boldsymbol{\eta}) + \alpha F_I^e(\boldsymbol{\eta}) \quad (8.216)$$

where  $F_I^e$  are modifying functions with the sign depending on the sign of the advective flux  $\mathbf{q}$  and  $\alpha$  is a free, so-called *upwind parameter* ( $0 \leq \alpha \leq 1$ ), which has to be determined. We note if  $\alpha = 0$ , (8.216) corresponds to the standard GFEM. The modifying functions  $F_I^e$  can be appropriately chosen either as continuous and discontinuous relations. Let us consider for convenience firstly the 1D case: In the continuous definition  $F_I^e$  are chosen as a polynomial one degree higher than  $N_I^e$ , e.g., [236]. For a linear 1D element we introduce  $F_I^e(\xi) = \mp f^e(\xi)$ , where  $f^e(\xi) = a\xi^2 + b\xi + c$ , written in the local coordinate ( $-1 \leq \xi \leq +1$ ), and determine its polynomial coefficients  $a$ ,  $b$  and  $c$  such that  $f^e(-1) = f^e(1) = 0$  and  $\int_{-1}^{+1} f^e(\xi) d\xi = 1$ . It leads to

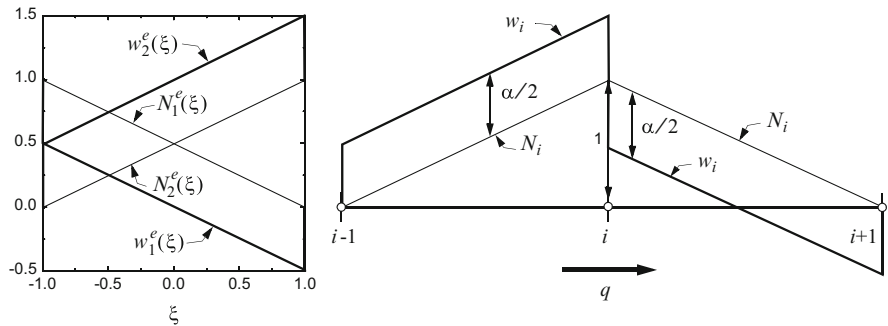
$$f^e(\xi) = \frac{3}{4}(1 - \xi)(1 + \xi) \quad (8.217)$$

Then, the following continuous weighting functions result as shown in Fig. 8.24 for a linear 1D element at the local nodes 1 and 2:

$$\begin{aligned} w_1^e(\xi) &= N_1^e(\xi) - \alpha f^e(\xi) \\ w_2^e(\xi) &= N_2^e(\xi) + \alpha f^e(\xi) \end{aligned} \quad (8.218)$$



**Fig. 8.24** Continuous Petrov-Galerkin weighting functions ( $\alpha = 1$ ) for the linear 1D element



**Fig. 8.25** Discontinuous Petrov-Galerkin weighting functions ( $\alpha = 1$ ) for the linear 1D element

where the basis functions are  $N_1^e(\xi) = \frac{1}{2}(1 - \xi)$ ,  $N_2^e(\xi) = \frac{1}{2}(1 + \xi)$ , cf. (H.1) of Appendix H.

Much more convenient are discontinuous weighting functions, e.g., [584]. In 1D one simply chooses:

$$\alpha F_I^e(\xi) = \alpha \frac{\Delta x^e}{2} \frac{q^e}{|q^e|} \frac{dN_I^e}{dx} \tag{8.219}$$

where  $\Delta x^e$  is the length of the finite element  $e$ . By using the derivations (H.6) of Appendix H we obtain the following asymmetric discontinuous weighting functions for a linear 1D element at the local nodes 1 and 2 with a positive advective flux  $q^e > 0$

$$\begin{aligned} w_1^e(\xi) &= N_1^e(\xi) - \frac{\alpha}{2} \\ w_2^e(\xi) &= N_2^e(\xi) + \frac{\alpha}{2} \end{aligned} \tag{8.220}$$

which are displayed in Fig. 8.25.

It is important to note that the discontinuous weighting functions  $w_I^e$  must result finite contributions for first derivatives in the integrand of the approximate weak statement in order satisfy the requirement on continuity as stated in Sect. 8.7. The discontinuity of the formulations (8.219) or (8.220) is considered within the element such that any first-order derivative of  $w_I^e$  is finite and valuable due to the  $C_0$ -continuity in  $N_I^e$ .

Usually, the asymmetric weighting functions  $w_I^e$  are only applied to the terms of the homogeneous solution of the governing PDE, i.e., in particular the advection and diffusion/dispersion terms. In doing so, for example for the 1D discrete finite element equations of Sect. H.1 of Appendix H, we find for (H.10) a modified formulation of the semidiscrete ADE convective form<sup>13</sup> (for sake of simplicity we drop BC and SPC terms):

$$\sum_e \left\{ \frac{\hat{\mathcal{R}}^e \Delta x^e}{6} \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} \frac{d\phi_1^e}{dt} \\ \frac{d\phi_2^e}{dt} \end{pmatrix} + \underbrace{\left[ \frac{q^e}{2} \begin{pmatrix} -1 + \alpha & 1 - \alpha \\ -1 - \alpha & 1 + \alpha \end{pmatrix} \right]}_{A^e} + \underbrace{\frac{D^e}{\Delta x^e} \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}}_{C^e} \right. \\ \left. + \frac{(\vartheta^e + Q^e) \Delta x^e}{6} \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \right] \cdot \begin{pmatrix} \phi_1^e \\ \phi_2^e \end{pmatrix} - \frac{H^e \Delta x^e}{2} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \Big\} = \mathbf{0} \quad (8.221)$$

We recognize from (8.221) that the upwind parameter  $\alpha$  is indeed only effective in the advection matrix  $A^e$ , while it is canceled out in the diffusion matrix  $C^e$ . Furthermore, it is easy to see that the sum of  $A^e + C^e$  can be alternatively written as<sup>14</sup>

$$A^e + C^e = \frac{q^e}{2} \begin{pmatrix} -1 & 1 \\ -1 & 1 \end{pmatrix} + (D^e + \alpha \frac{q^e \Delta x^e}{2}) \frac{1}{\Delta x^e} \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \quad (8.222)$$

where the physical diffusion  $D^e$  is increased by  $\alpha \frac{q^e \Delta x^e}{2}$ , which represents the artificial diffusion introduced by the PGFEM upwind method. Similar to (8.203) the assembly of the linear 1D elements (8.221) and applying the temporal  $\theta$ -integration scheme of (8.154) the following discrete equations of the 3-node  $(i + 1, i, i - 1)$ -stencil can be written assuming constant parameter properties (and for convenience also dropping source/sink terms):

<sup>13</sup>For the ADE convective form the continuous weighting functions (8.218) and the discontinuous weighting functions (8.220) lead to the same result. However, for the ADE divergence form only the continuous weighting functions (8.218) are applicable, where the element advection matrix  $A^e$  (8.104) becomes

$$A^e = \frac{q^e}{2} \begin{pmatrix} 1 + \alpha & 1 - \alpha \\ -1 - \alpha & -1 + \alpha \end{pmatrix}$$

<sup>14</sup>A similar expression can be obtained for the ADE divergence form.

$$\begin{aligned}
& \left[ \frac{c}{6} - \Delta t_n \theta (1 + (\alpha - 1)Pg) \right] \phi_{i+1,n+1} + \left[ \frac{2c}{3} + \Delta t_n \theta (2 + 2\alpha Pg) \right] \phi_{i,n+1} + \\
& \left[ \frac{c}{6} - \Delta t_n \theta (1 + (\alpha + 1)Pg) \right] \phi_{i-1,n+1} = \left[ \frac{c}{6} + \Delta t_n (1 - \theta) (1 + (\alpha - 1)Pg) \right] \phi_{i+1,n} + \\
& \left[ \frac{2c}{3} - \Delta t_n (1 - \theta) (2 + 2\alpha Pg) \right] \phi_{i,n} + \left[ \frac{c}{6} + \Delta t_n (1 - \theta) (1 + (\alpha + 1)Pg) \right] \phi_{i-1,n}
\end{aligned} \tag{8.223}$$

where  $c = \Delta x^2/D^*$ ,  $D^* = D/\mathcal{R}$  and  $Pg$  is the grid Réclet number as defined in (8.207).

To analyze the upwind parameter  $\alpha$  let us at first turn to the steady-state ADE formulation. Then, (8.223) reduces to

$$[1 + (\alpha - 1)Pg]\phi_{i+1} - 2(1 + \alpha Pg)\phi_i + [1 + (\alpha + 1)Pg]\phi_{i-1} = 0 \tag{8.224}$$

which represents a PGFEM formulation of the simplified 1D ADE:

$$q\nabla\phi - D\nabla^2\phi = 0 \quad \text{where in 1D} \quad \nabla = \partial/\partial x \tag{8.225}$$

We can solve (8.225) within the interval  $x_{i-1} \leq x \leq x_{i+1}$  for the local boundary value problem:  $\phi(x_{i-1}) = \phi_{i-1}$  and  $\phi(x_{i+1}) = \phi_{i+1}$ . The exact solution of this local 1D problem is

$$\phi(x) = \phi_{i-1} + (\phi_{i+1} - \phi_{i-1}) \frac{\exp\left[\frac{2Pg}{\Delta x}(x - x_{i-1})\right] - 1}{\exp(4Pg) - 1} \tag{8.226}$$

which can be taken to express the solution for  $\phi(x_i) = \phi_i$  leading to the ‘locally-exact’ formula of the 3-node  $(i + 1, i, i - 1)$ -stencil

$$\phi_{i+1} - (1 + a)\phi_i + a\phi_{i-1} = 0 \tag{8.227}$$

where

$$a = \exp(2Pg) \tag{8.228}$$

with

$$\begin{aligned}
a &> 0 & \text{for} & \quad q > 0 \\
\frac{1}{a} &> 0 & \text{for} & \quad q < 0
\end{aligned} \tag{8.229}$$

In comparison of the scheme (8.224) with the exact formula (8.227) it must be required due to (8.229) for  $q > 0$

$$a = \frac{1 + Pg(\alpha + 1)}{1 + Pg(\alpha - 1)} > 0 \quad (8.230)$$

which yields

$$\begin{aligned} \alpha &\geq \alpha^{\text{crit}} = 1 - \frac{1}{Pg} && \text{for } Pg \geq 1 \\ \alpha &= 0 && \text{for } Pg < 1 \end{aligned} \quad (8.231)$$

Apparently, for the standard GFEM with  $\alpha = 0$  once the grid Péclet number  $Pg > 1$  node-to-node oscillations will occur since the denominator in (8.230) becomes negative. To avoid oscillations the upwind parameter  $\alpha$  must be greater than the critical value  $\alpha^{\text{crit}}$  defined in (8.231). It shows that for  $\alpha = 1$  the scheme is unconditionally stable and corresponds to a *full upwind* scheme. Furthermore, a complete accuracy is obtained for a given Péclet number  $Pg$  if the parameter  $a$  of the exact solution (8.228) is equated to  $a$  of the approximate solution (8.230). It gives the so-called *optimal upwind parameter*

$$\alpha^{\text{opt}} = \coth(Pg) - \frac{1}{Pg} \quad (8.232)$$

It is obvious that the  $\alpha^{\text{opt}}$  satisfies the stability criterion (8.231) with

$$\alpha^{\text{opt}} \geq \alpha^{\text{crit}} \quad (8.233)$$

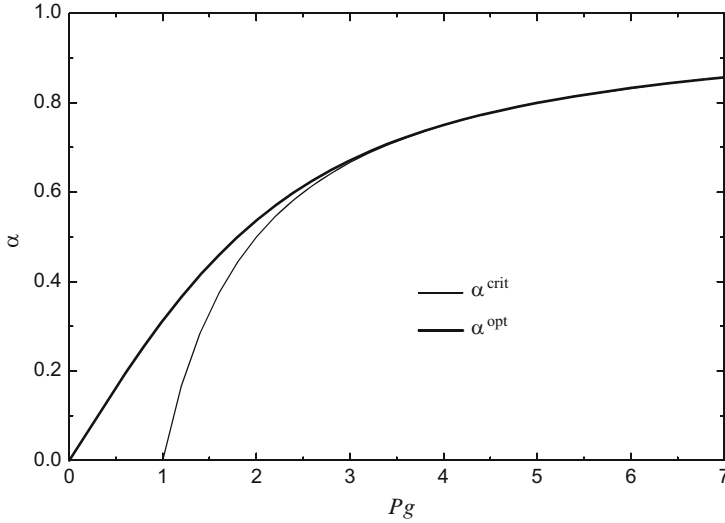
and so, indeed, it is optimal for this class of problems (Fig. 8.26). Upwind parameter relations for higher-order finite elements have been derived in [81, 131, 235].

The extension of the PGFEM to multidimensional and transient ADE problems can be done straightforward, e.g., [278, 279, 592]. However, in 2D and particularly in 3D the use of continuous weighting functions in form of (8.216) is cumbersome and ineffective, so that discontinuous weighting functions are often preferred. An appropriate discontinuous weighting function is [585]

$$w_i^e(\boldsymbol{\eta}) = N_i^e(\boldsymbol{\eta}) + \frac{h^e}{2\|\mathbf{q}^e\|} \left( \alpha + \frac{\beta \Delta t_n}{2} \frac{\partial}{\partial t} \right) \mathbf{q}^e \cdot \nabla N_i^e(\boldsymbol{\eta}) \quad (8.234)$$

where  $h^e$  is a characteristic element length which is defined further below,  $\alpha$  is a first upwind parameter as already introduced above for steady-state problems and  $\beta$  is a second upwind parameter related to the transient terms of the ADE. The intent and result of (8.234) is to add artificial diffusion into the discrete finite element equations. With the upwind parameter  $\alpha$  the diffusion is increased by  $\alpha\|\mathbf{q}^e\|h^e/2$  and with the upwind parameter  $\beta$  an added diffusion term is in the order of  $\beta\|\mathbf{q}^e\|h^e\Delta t_n/4$ . The upwind parameters can be determined by Yu and Heinrich [584]

$$\begin{aligned} \alpha &= \coth(Pg) - \frac{1}{Pg} \\ \beta &= \frac{Cr}{3} - \frac{\alpha}{PgCr} \end{aligned} \quad (8.235)$$



**Fig. 8.26** Critical upwind parameter  $\alpha^{\text{crit}}$  and optimal upwind parameter  $\alpha^{\text{opt}}$  in dependence on the grid Péclet number  $Pg$  based on linear finite elements

where the mesh Péclet number  $Pg$  (8.207) and the Courant number  $Cr$  (8.206) are defined in multidimensions

$$Pg = \frac{\|\mathbf{q}^{*e}\| h^e}{2D^{*e}}, \quad Cr = \frac{\|\mathbf{q}^{*e}\| \Delta t_n}{h^e} \quad \text{with} \quad \|\mathbf{q}^{*e}\| = \|\mathbf{q}^e\|/\hat{\mathcal{R}}, \quad D^{*e} = D^e/\hat{\mathcal{R}} \quad (8.236)$$

The resulting PGFEM upwind scheme [584, 585] shows the best accuracy with  $\alpha \neq 0$  and  $\beta \neq 0$  according to (8.235). However, unconditionally stable algorithms also result for  $\alpha \neq 0$  and  $\beta = 0$ , albeit more artificial diffusion is produced.

The PGFEM upwind scheme in multidimensions requires the determination of the characteristic element length  $h^e$ . Figure 8.27 shows typical isoparametric finite elements in 2D and 3D over which the parametric vectors  $\mathbf{h}_\xi$ ,  $\mathbf{h}_\eta$  and  $\mathbf{h}_\zeta$  are defined and computed in 2D as

$$\left. \begin{aligned} \mathbf{h}_\xi &= h_{\xi i} = \frac{1}{2}[(x_{i2} + x_{i3}) - (x_{i1} + x_{i4})] \\ \mathbf{h}_\eta &= h_{\eta i} = \frac{1}{2}[(x_{i3} + x_{i4}) - (x_{i1} + x_{i2})] \end{aligned} \right\} \quad i = 1, 2 \quad (8.237)$$

and in 3D as

$$\left. \begin{aligned} \mathbf{h}_\xi &= h_{\xi i} = \frac{1}{4}[(x_{i2} + x_{i3} + x_{i6} + x_{i7}) - (x_{i1} + x_{i4} + x_{i5} + x_{i8})] \\ \mathbf{h}_\eta &= h_{\eta i} = \frac{1}{4}[(x_{i3} + x_{i4} + x_{i7} + x_{i8}) - (x_{i1} + x_{i2} + x_{i5} + x_{i6})] \\ \mathbf{h}_\zeta &= h_{\zeta i} = \frac{1}{4}[(x_{i1} + x_{i2} + x_{i3} + x_{i4}) - (x_{i5} + x_{i6} + x_{i7} + x_{i8})] \end{aligned} \right\} \quad i = 1, 2, 3 \quad (8.238)$$



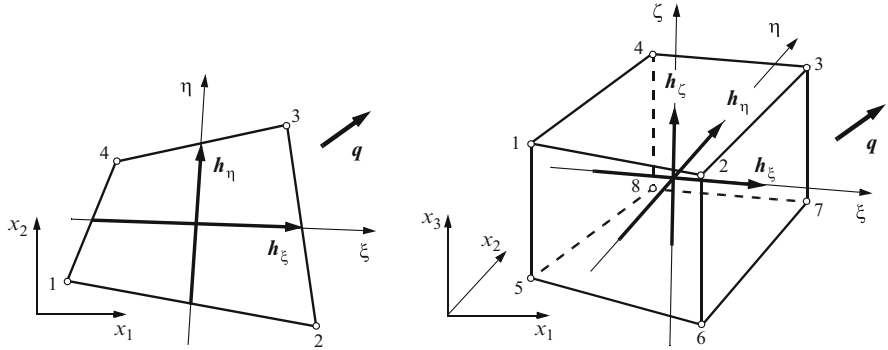


Fig. 8.27 2D quadrilateral and 3D hexahedral element used in definition of element length  $h^e$

Similar relations can be obtained for the other finite elements listed in Tables G.2–G.4 of Appendix G. The characteristic element length  $h^e$  then results

$$h^e = \begin{cases} |h_1| + |h_2| & \text{for 2D} \\ |h_1| + |h_2| + |h_3| & \text{for 3D} \end{cases} \quad (8.239)$$

where in 2D

$$\begin{aligned} h_1 &= \frac{1}{\|q^e\|} (q^e \cdot h_\xi) = \frac{1}{\|q^e\|} (q_1^e h_{\xi 1} + q_2^e h_{\xi 2}) \\ h_2 &= \frac{1}{\|q^e\|} (q^e \cdot h_\eta) = \frac{1}{\|q^e\|} (q_1^e h_{\eta 1} + q_2^e h_{\eta 2}) \end{aligned} \quad (8.240)$$

and in 3D

$$\begin{aligned} h_1 &= \frac{1}{\|q^e\|} (q^e \cdot h_\xi) = \frac{1}{\|q^e\|} (q_1^e h_{\xi 1} + q_2^e h_{\xi 2} + q_3^e h_{\xi 3}) \\ h_2 &= \frac{1}{\|q^e\|} (q^e \cdot h_\eta) = \frac{1}{\|q^e\|} (q_1^e h_{\eta 1} + q_2^e h_{\eta 2} + q_3^e h_{\eta 3}) \\ h_3 &= \frac{1}{\|q^e\|} (q^e \cdot h_\zeta) = \frac{1}{\|q^e\|} (q_1^e h_{\zeta 1} + q_2^e h_{\zeta 2} + q_3^e h_{\zeta 3}) \end{aligned} \quad (8.241)$$

are the projections of  $h_\xi$ ,  $h_\eta$  and  $h_\zeta$  in the direction of the local flow vector  $q^e$ . We note that for rectangular geometries the expression (8.239) reduces to  $h^e = (|q_1^e| \Delta x_1^e + |q_2^e| \Delta x_2^e + |q_3^e| \Delta x_3^e) / \|q^e\|$  in 3D, where  $\Delta x_i^e$ , ( $i = 1, 2, 3$ ) are the lengths of element edges in the coordinate directions. In 1D geometries it is simply  $h^e = \Delta x^e$ .

### 8.14.3 Streamline Upwind (SU) and Full Upwind (FU) Method

We have seen in the preceding Sect. 8.14.2 that PGFEM is designed to add an appropriate amount of artificial diffusion for stabilization purposes. The use of the

asymmetric weighting function takes effect only on the advective term and ends up with a diffusion increased for instance by  $\alpha \|q^e\| h^e/2$  for a linear finite element. It is obvious that such a type of stabilization should be correlated with the flow direction only and should not be effective in the transverse direction of advection to avoid an overly diffusion due to an excess of so-called *crosswind diffusion*.

The avoidance of crosswind diffusion leads to the concept of the streamline upwind (SU) method. The basic ideas were given by Kelly et al. [307] and Brooks and Hughes [57] who constructed the artificial diffusion operator in tensorial form acting only in the flow direction and not transversely, termed as *anisotropic balancing dissipation*. The idea of SU is to extend the tensor of physical dispersion/diffusion  $D$  defined for example in a porous medium of a single-species solute transport as (cf. (3.180), Tables 3.7 and 3.9)

$$\begin{aligned} D &= \varepsilon s D \delta + D_{\text{mech}} \\ D_{\text{mech}} &= \beta_T \|q\| \delta + (\beta_L - \beta_T) \frac{q \otimes q}{\|q\|} \end{aligned} \quad (8.242)$$

where  $D_{\text{mech}}$  is the (physical) tensor of mechanical dispersion and  $D$  is the molecular diffusion, by the tensor of numerical dispersion  $D_{\text{num}}$  in the form

$$D_{\text{num}} = \beta_{\text{num}} \frac{q \otimes q}{\|q\|} \quad (8.243)$$

so that

$$D = \varepsilon s D \delta + D_{\text{mech}} + D_{\text{num}} = \varepsilon s D \delta + \beta_T \|q\| \delta + (\beta_L + \beta_{\text{num}} - \beta_T) \frac{q \otimes q}{\|q\|} \quad (8.244)$$

where  $\beta_{\text{num}}$  represents the parameter of *numerical longitudinal dispersivity* which must be specified for each element. For example, in case of linear elements one takes for the element  $e$

$$\beta_{\text{num}}^e = \alpha \frac{h^e}{2} \quad (8.245)$$

where  $0 \leq \alpha \leq 1$  is the upwind parameter introduced above ( $\alpha = 0$  is the standard GFEM,  $\alpha = 1$  is the full upwind,  $\alpha = \alpha^{\text{opt}}$  is the optimal parameter defined in (8.232)) and  $h^e$  is the characteristic element length defined in (8.239). Note that for quadratic elements  $\beta_{\text{num}}^e = \alpha h^e/4$  as derived in [131].

Now, if looking to the resulting weak statement we have to modify for the advective and dispersive terms of the governing ADE written in its convective form according to (8.55)

$$\text{WS} = \int_{\Omega} wq \cdot \nabla \phi d\Omega + \int_{\Omega} \nabla w \cdot [(D + D_{\text{num}}) \cdot \nabla \phi] d\Omega \quad (8.246)$$

Since

$$\int_{\Omega} \nabla w \cdot (\mathbf{D}_{\text{num}} \cdot \nabla \phi) d\Omega = \int_{\Omega} \frac{\beta_{\text{num}}}{\|\mathbf{q}\|} (\mathbf{q} \cdot \nabla w) (\mathbf{q} \cdot \nabla \phi) d\Omega \quad (8.247)$$

Eq. (8.246) can be rewritten as

$$\text{WS} = \int_{\Omega} [w + \frac{\beta_{\text{num}}}{\|\mathbf{q}\|} (\mathbf{q} \cdot \nabla w)] (\mathbf{q} \cdot \nabla \phi) d\Omega + \int_{\Omega} \nabla w \cdot (\mathbf{D} \cdot \nabla \phi) d\Omega \quad (8.248)$$

As a result, a modified SU weighting function can be found in the form

$$\tilde{w} = w + \frac{\beta_{\text{num}}}{\|\mathbf{q}\|} (\mathbf{q} \cdot \nabla w) \quad (8.249)$$

which only affects the advective term and is similar to the discontinuous weighting function (8.234) (for  $\beta = 0$ ) used by the PGFEM. Finally, the SU method is recognized as the standard GFEM plus an extra term introducing the SU added numerical dispersion term:

$$\text{GWS} = \underbrace{\sum_e \int_{\Omega^e} [N_i \mathbf{q} \cdot \nabla \phi + \nabla N_i \cdot (\mathbf{D} \cdot \nabla \phi)] d\Omega^e}_{\text{standard GFEM}} + \underbrace{\sum_e \int_{\Omega^e} \frac{\beta_{\text{num}}^e}{\|\mathbf{q}\|} (\mathbf{q} \cdot \nabla N_i) (\mathbf{q} \cdot \nabla \phi) d\Omega^e}_{\text{added SU stabilization term}} \quad (8.250)$$

where  $\mathbf{D}$  represents the physical dispersion tensor (8.242),  $\beta_{\text{num}}^e = \alpha h^e / 2$  for linear elements and  $\beta_{\text{num}}^e = \alpha h^e / 4$  for quadratic elements. In practice, however, the second SU stabilization term in (8.250) is not directly executed. Instead, the modified dispersion tensor (8.244) is employed in the standard GFEM term, which is equivalent to (8.250).

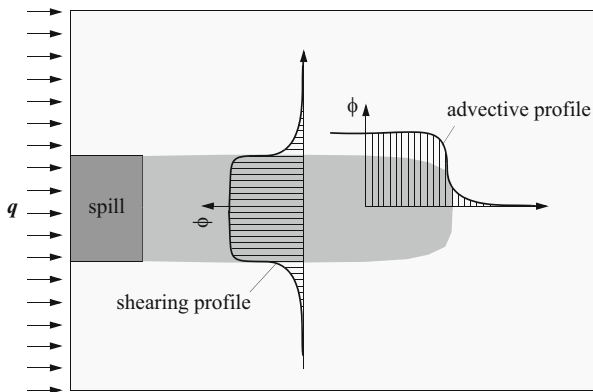
Commonly, the SU method is used with  $\alpha = 1$ . In case of need the SU stabilization can be turned back to a *full upwinding* (FU), where the stabilization is performed in all coordinate directions, i.e., independent of the flow field. In the full upwind case the dispersion tensor (8.244) is then used in the form:

$$\mathbf{D} = \varepsilon_S \mathbf{D} \boldsymbol{\delta} + (\beta_T + \beta_{\text{num}}) \|\mathbf{q}\| \boldsymbol{\delta} + (\beta_L - \beta_T) \frac{\mathbf{q} \otimes \mathbf{q}}{\|\mathbf{q}\|} \quad (8.251)$$

However, it should be aware that a full upwind scheme usually produces a large amount of crosswind diffusion.

### 8.14.4 Shock Capturing (SC) Method

SU stabilization is only effective in longitudinal direction of the advective flow and avoids any crosswind damping. This is motivated by the fact that often the gradient of a transported quantity  $\phi$  establishes in the direction of flow. However, under



**Fig. 8.28** Solution profiles  $\phi(x, t)$  in longitudinal and transverse direction to a flow field  $q$  forming an advection-displaced front and shearing transition layers, respectively

more complex flow conditions steep gradients can also occur in directions normal or skewed to the advective flow forming shearing (or flushing) transition layers such as illustrated in Fig. 8.28. Then, oscillations cannot be stabilized via the SU method. Indeed, it has been shown by Hughes et al. [274] that the streamline is not always the appropriate upwind direction. To increase the robustness of upwind methods it is obvious that the control of gradients must be required, i.e., the upwind direction should be aligned to the direction of gradients  $\nabla\phi$  of the transported quantity  $\phi$  rather than exclusively oriented to the trajectory of flow. This was basically proposed by Hughes and Mallet [269] who generalized the SU concept by adding an additional diffusion in the gradient direction which is called *discontinuity capturing* or *shock capturing* (SC).

The SC method has been further developed by Johnson et al. [292] and Codina [90]. The SC technique appears as a nonlinear method because the gradient  $\nabla\phi$  is part of the numerical solution. The main idea behind SC is to increase the amount of damping in the neighborhood of gradients. Then, the damping to be added must be proportional to the discrete residual of the governing ADE within each element and must vanish in regions where the solution is smooth and also where the advective term of the residual is small. Hence, SC stabilizes in dependence on the solution gradient and is accordingly operational both in longitudinal and transverse direction. It admits an optimal amount of crosswind damping necessary to stabilize also the shearing profiles (Fig. 8.28).

We have shown in the preceding Sect. 8.14.3 that the SU method is characterized by introducing an additional term  $\frac{\beta_{num}}{\|q\|} (q \cdot \nabla w)$  to the weighting function  $\tilde{w}$  in form of (8.249). Now, the basic idea of SC is to use  $\tilde{w}$  with a further additional term, the SC term, such that

$$\tilde{w} = \underbrace{w}_{\text{GFEM}} + \underbrace{\tau_1 (q \cdot \nabla w)}_{\text{SU}} + \underbrace{\tau_2 (q_{\parallel} \cdot \nabla w)}_{\text{SC}} \tag{8.252}$$

where the first term is the standard Galerkin weighting function, the second term is the linear SU modification and the third term is the new nonlinear SC extension. The vector  $\mathbf{q}_{\parallel}$  is the projection of the flux vector  $\mathbf{q}$  onto the direction of the local gradient  $\nabla\phi$  of the solution  $\phi$ , viz.,

$$\mathbf{q}_{\parallel} = \frac{\mathbf{q} \cdot \nabla\phi}{\|\nabla\phi\|^2} \nabla\phi \quad (8.253)$$

provided that  $\|\nabla\phi\| \neq 0$ . It is easy to see that  $\mathbf{q}_{\parallel} \cdot \nabla\phi = \mathbf{q} \cdot \nabla\phi$ . The upwind parameters  $\tau_1$  and  $\tau_2$  are defined on element level as

$$\tau_1 = \frac{\alpha h^e}{2\|\mathbf{q}\|}, \quad \tau_2 = \max\left(0, \frac{\alpha h^e}{2\|\mathbf{q}_{\parallel}\|} - \tau_1\right) \quad (8.254)$$

written for linear elements, where  $0 \leq \alpha \leq 1$  is the known upwind parameter defined above, (8.231) or (8.232), and  $h^e$  is the characteristic element length according to (8.239). In using (8.252), the SC method is recognized as the standard GFEM plus two extra terms introducing the SU added numerical dispersion term and the SC added numerical dispersion term applied to the Galerkin weak statement of the advective and dispersive terms of the governing ADE:

$$\begin{aligned} \text{GWS} = & \underbrace{\sum_e \int_{\Omega^e} [N_i \mathbf{q} \cdot \nabla\phi + \nabla N_i \cdot (\mathbf{D} \cdot \nabla\phi)] d\Omega^e}_{\text{standard GFEM}} + \underbrace{\sum_e \int_{\Omega^e} \tau_1 (\mathbf{q} \cdot \nabla N_i) (\mathbf{q} \cdot \nabla\phi) d\Omega^e}_{\text{added SU stabilization term}} \\ & + \underbrace{\sum_e \int_{\Omega^e} \tau_2 (\mathbf{q}_{\parallel} \cdot \nabla N_i) (\mathbf{q}_{\parallel} \cdot \nabla\phi) d\Omega^e}_{\text{added SC stabilization operator}} \end{aligned} \quad (8.255)$$

where the SC method is constructed to keep unaltered the added numerical dispersion in the streamline direction and to modify only the crosswind (transverse) dispersion. This crosswind dispersion must satisfy two conditions [90]. First, to avoid overdamped crosswind effects, it must be small in regions where the advective transport is not important, that is where  $\mathbf{q} \cdot \nabla\phi$  is small. Second, the measure of crosswind damping should be proportional to the element residual, e.g., for the ADE convective form

$$R(\phi) = \mathbf{q} \cdot \nabla\phi - \nabla \cdot (\mathbf{D} \cdot \nabla\phi) + (\vartheta + Q)\phi - H - Q_{\phi w} \quad (8.256)$$

to be evaluated on element basis. Using  $R(\phi)$  we can determine an isotropic SC dispersion coefficient as [90, 292]

$$D_{\text{sc}} = \frac{1}{2} \alpha_c h^e \frac{|R(\phi)|}{\|\nabla\phi\|} \quad (8.257)$$

if  $\|\nabla\phi\| \neq 0$  and zero otherwise. If we use the element residual only for the advective term of the ADE convective form by  $R(\phi) \approx \mathbf{q} \cdot \nabla\phi$ , a useful and simplified estimate of the isotropic SC dispersion coefficient results

$$D_{\text{sc}} \approx \frac{1}{2} \alpha_c h^e \|\mathbf{q}_{\parallel}\| \quad (8.258)$$

where with (8.253) it is  $\|\mathbf{q}_{\parallel}\| = |\mathbf{q} \cdot \nabla\phi| / \|\nabla\phi\|$ . The upwind parameter  $\alpha_c$  is given by

$$\alpha_c = \max\left(0, a - \frac{1}{Pg_{\parallel}}\right) \quad \text{with} \quad Pg_{\parallel} = \frac{\|\mathbf{q}_{\parallel}\| h^e}{2D^e} \quad (8.259)$$

where it is proposed, e.g., [90]

$$a = \begin{cases} \left. \begin{array}{l} 0.7 \quad \text{for linear element} \\ 0.35 \quad \text{for quadratic element} \end{array} \right\} & \text{in 2D} \\ \left. \begin{array}{l} 1.0 \quad \text{for linear element} \\ 0.5 \quad \text{for quadratic element} \end{array} \right\} & \text{in 3D} \end{cases} \quad (8.260)$$

The isotropic SC dispersion coefficient  $D_{\text{sc}}$  (8.257) or (8.258) is added to the hydrodynamic dispersion tensor  $\mathbf{D}$  (8.242). It yields

$$\mathbf{D} = (\varepsilon_S \mathbf{D} + D_{\text{sc}}) \boldsymbol{\delta} + \mathbf{D}_{\text{mech}} \quad (8.261)$$

The SC dispersion coefficient  $D_{\text{sc}} = D_{\text{sc}}(\phi)$  is nonlinear due to the solution dependency and an appropriate numerical treatment is required. In the practical implementation the SC method is not used in combination with the SU stabilization, i.e., SC stabilizes completely the solution via the isotropic SC dispersion coefficient  $D_{\text{sc}}$ .

### 8.14.5 Petrov-Galerkin Least Square (PGLS) Finite Element Method

The Petrov-Galerkin least square (PGLS) FEM represents an alternative stabilization technique to solve transient ADE in the convective form [276]. Its special feature is in introducing a *symmetric* stabilization term. In contrast to the PGFEM, SU and SC upwind methods as described in Sects. 8.14.2–8.14.4, PGLS leads to symmetric matrix systems and possesses built-in streamline-like upwind characteristics. The PGLS symmetrization is superior to symmetric-matrix time integration schemes, where the advective term is treated only explicitly so as done by Leismann and Frind [338]. The effect of PGLS has similarities to the SU upwinding, where

an anisotropic (streamline-oriented) balancing dissipation (dispersion) is added to the physical longitudinal dispersion parameter. However, in the PGLS method the artificial dispersion (diffusion) is directly derived from the least-square (LS) finite element concept and requires no ‘free’ upwind parameter such as  $\alpha$  of the preceding upwind methods.

As indicated in Table 8.1 the LS minimization by PGLS has to be done with respect to the nodal values of the state variable(s). Due to the square operations in the inner products of the governing PDE, higher order derivatives remain, which usually require higher order basis functions, i.e., a  $C_0$  continuity (cf. Sect. 8.7) in the interpolation functions is no more sufficient, unless the LS operation is only restricted to first-order terms while the higher order terms are treated in the standard Galerkin-based manner via an operator splitting approach. Basic work was given by Nguyen and Reynen [385] and further developments can be found in [319, 559], among others. König [319] used an operator splitting method in a two-pass strategy, where the separate equations for the diffusive and the advective parts are solved successively. On the other hand, Wendland [559] improved the operator splitting technique by introducing a suited one-pass approach termed as *symmetric streamline stabilization*, where the diffusive and advective parts are reassembled in one symmetric matrix system.

### 8.14.5.1 Operator Splitting

The basic ADE in the convective form (8.5)

$$\hat{\mathcal{R}}\dot{\phi} + \mathbf{q} \cdot \nabla\phi - \nabla \cdot (\mathbf{D} \cdot \nabla\phi) + (\vartheta + Q)\phi - H - Q_{\phi_w} = 0 \quad (8.262)$$

can be written in an operator-split formulation

$$\hat{\mathcal{R}}\dot{\phi} + (\mathcal{L}^d + \mathcal{L}^a)\phi = H + Q_{\phi_w} \quad (8.263)$$

with

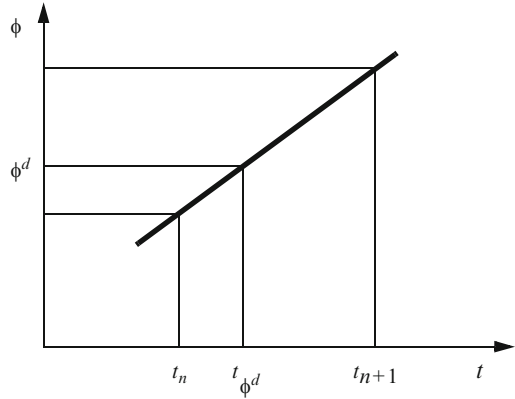
$$\begin{aligned} \mathcal{L}^d &= -\nabla \cdot (\mathbf{D} \cdot \nabla) + (\vartheta + Q) \\ \mathcal{L}^a &= \mathbf{q} \cdot \nabla \end{aligned} \quad (8.264)$$

where  $\mathcal{L}^d$  is a diffusion differential operator and  $\mathcal{L}^a$  is an advection differential operator. We can also split the solution  $\phi$  into the diffusive and the advective part such that

$$\phi = \phi^d + \phi^a \quad (8.265)$$

Then, we transform (8.263) into two separate equations: first, the diffusive PDE

**Fig. 8.29** Temporally discrete interpolation of the intermediate diffusive solution  $\phi^d$



$$\hat{\mathcal{R}}\dot{\phi}^d + \mathcal{L}^d \phi = H + Q_{\phi w} \tag{8.266}$$

and, second, the purely advective (hyperbolic) PDE

$$\hat{\mathcal{R}}(\dot{\phi} - \dot{\phi}^d) + \mathcal{L}^a \phi = 0 \tag{8.267}$$

Summing (8.266) and (8.267) we realize the original ADE (8.263).

The idea of the operator splitting technique is in approximating the diffusive PDE (8.266) and advective PDE (8.267) in a separate manner. After completion the total discrete ADE is obtained by assembling the diffusive and advective parts. In doing so, we consider the variables  $\phi^d(t)$  and  $\phi^a(t)$  in the time interval  $(t_n, t_{n+1})$  and assume at the beginning of the interval the following IC's for the diffusive variable  $\phi_n^d = \phi^d(t_n)$  and for the advective variable  $\phi_n^a = \phi^a(t_n)$ :

$$\phi_n^d = \phi_n, \quad \phi_n^a = 0, \quad \dot{\phi}_n^a = 0 \tag{8.268}$$

It is to be noted that the diffusive solution  $\phi^d$  can be considered as an intermediate solution which represents a temporally discrete interpolation between the previous and the new time plane as evidenced in Fig. 8.29.

### 8.14.5.2 Approximation of the Diffusive Part

In the context of FEM, the two variables  $\phi$  and  $\phi^d$  are replaced by a continuous approximation (8.16) that assumes the separability of space and time, thus

$$\begin{aligned} \phi(\mathbf{x}, t) &\approx \sum_i N_i(\mathbf{x}) \phi_i(t) \\ \phi^d(\mathbf{x}, t) &\approx \sum_i N_i(\mathbf{x}) \phi_i^d(t) \end{aligned} \tag{8.269}$$



where the subscript  $i = 1, \dots, N_p$  denotes the global nodal indices. The Galerkin weak statement of (8.266)

$$\text{GWS} = \int_{\Omega} N_i \left( \mathcal{R} \dot{\phi}^d + \mathcal{L}^d \phi - H - Q_{\phi_w} \right) d\Omega = 0 \quad (8.270)$$

leads after inserting the semidiscrete basis functions (8.269) to the following global matrix system (cf. Sect. 8.9)

$$\mathbf{O} \cdot \dot{\phi}^d + \mathbf{K}^d \cdot \phi - \mathbf{F} = \mathbf{0} \quad (8.271)$$

with

$$\mathbf{K}^d = \mathbf{C} + \mathbf{R} + \mathbf{B} \quad (8.272)$$

where the matrices  $\mathbf{O}$ ,  $\mathbf{C}$ ,  $\mathbf{R}$ ,  $\mathbf{B}$  and the vector  $\mathbf{F}$  are given in (8.103)–(8.105) referred to the convective form. By using the methods of time integration introduced above in Sect. 8.13 and invoking (8.265) and (8.268) with

$$\phi(t_n + \theta \Delta t_n) = \theta(\phi_{n+1}^d + \phi_{n+1}^a) + (1 - \theta)\phi_n \approx \theta\phi_{n+1}^d + (1 - \theta)\phi_n \quad (8.273)$$

the following matrix systems of the intermediate (diffusive) part result

$$\left( \frac{\mathbf{O}}{\Delta t_n} + \mathbf{K}^d \theta \right) \cdot \phi_{n+1}^d = \left[ \frac{\mathbf{O}}{\Delta t_n} - \mathbf{K}^d (1 - \theta) \right] \cdot \phi_n + (\mathbf{F}_{n+1} \theta + \mathbf{F}_n (1 - \theta)) \quad (8.274)$$

for the  $\theta$ -family of time stepping methods (cf. Sect. 8.13.4) and

$$\left( \frac{\mathbf{O}}{\theta \Delta t_n} + \mathbf{K}^d \right) \cdot \phi_{n+1}^d = \mathbf{O} \cdot \left[ \frac{1}{\theta \Delta t_n} \phi_n + \left( \frac{1}{\theta} - 1 \right) \dot{\phi}_n \right] + \mathbf{F}_{n+1} \quad (8.275)$$

for the predictor-corrector methods (cf. Sect. 8.13.5), where the weighting coefficient  $\frac{1}{2} \leq \theta \leq 1$  identifies the different time integration methods.

### 8.14.5.3 Approximation of the Advective Part

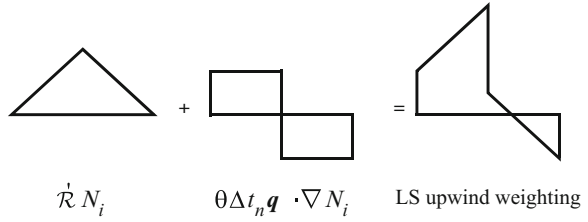
The residual of the advective part (8.267) in form of

$$R = \mathcal{R}(\dot{\phi} - \dot{\phi}^d) + \mathbf{q} \cdot \nabla \phi \quad (8.276)$$

will be treated by the LS method

$$\frac{\partial}{\partial \phi_i} \int_{\Omega} \frac{1}{2} R^2 d\Omega = 0 \quad (8.277)$$

**Fig. 8.30** LS weighting function of the operator splitting



which is equivalent to

$$\int_{\Omega} w_i R d\Omega = 0 \quad (8.278)$$

with the nodal test function  $w_i$  given by

$$w_i = \frac{\partial R}{\partial \phi_i} \quad (i = 1, \dots, N_P) \quad (8.279)$$

or with (8.269) in (8.276)

$$w_i = \frac{\partial N_i}{\partial t} + \mathbf{q} \cdot \nabla N_i \quad (8.280)$$

Since  $N_i = N_i(\mathbf{x})$  is not a function of time, the residual (8.276) is to be expressed in its temporally discrete form, such that

$$R = \dot{\mathcal{R}} \left[ \sum_i N_i (\phi_{i,n+1} - \phi_{i,n+1}^d) - \sum_i N_i (\phi_{i,n} - \phi_{i,n}^d) \right] + \Delta t_n \mathbf{q} \cdot \nabla \left[ \theta \sum_i N_i \phi_{i,n+1} + (1 - \theta) \sum_i N_i \phi_{i,n} \right] \quad (8.281)$$

which yields

$$w_i = \frac{\partial R}{\partial \phi_{i,n+1}} = \dot{\mathcal{R}} N_i + \theta \Delta t_n \mathbf{q} \cdot \nabla N_i \quad (8.282)$$

Then, the LS weak statement (8.278) results

$$\text{LSWS} = \int_{\Omega} (\dot{\mathcal{R}} N_i + \theta \Delta t_n \mathbf{q} \cdot \nabla N_i) [\dot{\mathcal{R}} (\dot{\phi} - \dot{\phi}^d) + \mathbf{q} \cdot \nabla \phi] d\Omega = 0 \quad (8.283)$$

where the residual is weighted by the LS test function (8.282) consisting of two parts as displayed in Fig. 8.30.

The LS weak statement (8.283) leads to the following semidiscrete matrix system

$$(\mathbf{O} + \theta \Delta t_n \mathbf{V}) \cdot \dot{\phi} + (\mathbf{A} + \theta \mathbf{T}) \cdot \phi = (\mathbf{O} + \theta \Delta t_n \mathbf{V}) \cdot \dot{\phi}^d \quad (8.284)$$

with (cf. (8.103))

$$\begin{aligned}
 \mathbf{O} &= O_{ij} = \sum_e \left( \sum_I \sum_J O_{IJ}^e \Delta_{Ii}^e \Delta_{Jj}^e \right) \\
 \mathbf{A} &= A_{ij} = \sum_e \left( \sum_I \sum_J A_{IJ}^e \Delta_{Ii}^e \Delta_{Jj}^e \right) \\
 \mathbf{V} &= V_{ij} = \sum_e \left( \sum_I \sum_J V_{IJ}^e \Delta_{Ii}^e \Delta_{Jj}^e \right) \\
 \mathbf{T} &= T_{ij} = \sum_e \left( \sum_I \sum_J T_{IJ}^e \Delta_{Ii}^e \Delta_{Jj}^e \right)
 \end{aligned} \tag{8.285}$$

and the element matrices

$$\begin{aligned}
 O_{IJ}^e &= \int_{\Omega^e} \hat{\mathcal{R}}^e N_I^e N_J^e d\Omega^e \\
 A_{IJ}^e &= \int_{\Omega^e} N_I^e (\mathbf{q}^e \cdot \nabla N_J^e) d\Omega^e \\
 V_{IJ}^e &= \int_{\Omega^e} (\mathbf{q}^e \cdot \nabla N_I^e) N_J^e d\Omega^e \\
 T_{IJ}^e &= \int_{\Omega^e} \Delta t_n \frac{1}{\mathcal{R}} (\mathbf{q}^e \cdot \nabla N_I^e) (\mathbf{q}^e \cdot \nabla N_J^e) d\Omega^e
 \end{aligned} \tag{8.286}$$

where  $\Delta^e$  is the Boolean matrix defined in (8.82). The time discretization of (8.284) for the  $\theta$ -family of time stepping methods (cf. Sect. 8.13.4) results

$$\begin{aligned}
 (\mathbf{O} + \theta \Delta t_n \mathbf{V}) \cdot \left( \frac{\phi_{n+1} - \phi_n}{\Delta t_n} \right) + \theta (\mathbf{A} + \theta \mathbf{T}) \cdot \phi_{n+1} + (1 - \theta) (\mathbf{A} + \theta \mathbf{T}) \cdot \phi_n \\
 = (\mathbf{O} + \theta \Delta t_n \mathbf{V}) \cdot \left( \frac{\phi_{n+1}^d - \phi_n}{\Delta t_n} \right)
 \end{aligned} \tag{8.287}$$

and finally

$$\left[ \frac{\mathbf{O}}{\Delta t_n} + \theta (\mathbf{V} + \mathbf{A} + \theta \mathbf{T}) \right] \cdot \phi_{n+1} = \left( \frac{\mathbf{O}}{\Delta t_n} + \theta \mathbf{V} \right) \cdot \phi_{n+1}^d - (1 - \theta) (\mathbf{A} + \theta \mathbf{T}) \cdot \phi_n \tag{8.288}$$

Regarding the predictor-corrector strategy based on the BE and TR schemes, if taking

$$\begin{aligned}
 \dot{\phi}_{n+1} &= \frac{1}{\theta \Delta t_n} (\phi_{n+1} - \phi_n) - \left( \frac{1}{\theta} - 1 \right) \dot{\phi}_n \\
 \dot{\phi}_{n+1}^d &= \frac{1}{\theta \Delta t_n} (\phi_{n+1}^d - \phi_n) - \left( \frac{1}{\theta} - 1 \right) \dot{\phi}_n
 \end{aligned} \tag{8.289}$$

and using (8.284), the following matrix system for the predictor-corrector schemes is obtained

$$\left( \frac{\mathbf{O}}{\theta \Delta t_n} + \mathbf{V} + \mathbf{A} + \theta \mathbf{T} \right) \cdot \phi_{n+1} = \left( \frac{\mathbf{O}}{\theta \Delta t_n} + \mathbf{V} \right) \cdot \phi_{n+1}^d \tag{8.290}$$

#### 8.14.5.4 Assembly of the Diffusive and Advective Parts

To obtain the matrix system for the complete ADE (8.262) the diffusive and advective parts have to be added. For the  $\theta$ -family of time stepping methods the summation of (8.274) and (8.288) yields

$$\begin{aligned} \left[ \frac{\mathcal{O}}{\Delta t_n} + \theta(\mathbf{V} + \mathbf{A} + \theta\mathbf{T}) \right] \cdot \phi_{n+1} + \left( \frac{\mathcal{O}}{\Delta t_n} + \theta\mathbf{K}^d \right) \cdot \phi_{n+1}^d &= \left( \frac{\mathcal{O}}{\Delta t_n} + \theta\mathbf{V} \right) \cdot \phi_{n+1}^d \\ - (1 - \theta)(\mathbf{A} + \theta\mathbf{T}) \cdot \phi_n + \left[ \frac{\mathcal{O}}{\Delta t_n} - (1 - \theta)\mathbf{K}^d \right] \cdot \phi_n + (\mathbf{F}_{n+1}\theta + \mathbf{F}_n(1 - \theta)) & \end{aligned} \quad (8.291)$$

The term  $\frac{\mathcal{O}}{\Delta t_n} \cdot \phi_{n+1}^d$  can be eliminated from (8.291). The remaining terms correlating with the intermediate solution  $\phi_{n+1}^d$  will be transformed in the following way [559]: All terms related to  $\phi_{n+1}^d$  on the LHS of (8.291) are replaced by  $\phi_{n+1}$ , while such terms on the RHS of (8.291) are substituted by  $\phi_n$ . In doing so, the following matrix system results

$$\begin{aligned} \left[ \frac{\mathcal{O}}{\Delta t_n} + \theta(\mathbf{K}^d + \mathbf{V} + \mathbf{A} + \theta\mathbf{T}) \right] \cdot \phi_{n+1} &= \\ \left[ \frac{\mathcal{O}}{\Delta t_n} - (1 - \theta)(\mathbf{K}^d + \mathbf{A} + \theta\mathbf{T}) + \theta\mathbf{V} \right] \cdot \phi_n + (\mathbf{F}_{n+1}\theta + \mathbf{F}_n(1 - \theta)) & \end{aligned} \quad (8.292)$$

Analogously, for the BE and TR predictor-corrector schemes we add (8.275) and (8.290)

$$\begin{aligned} \left( \frac{\mathcal{O}}{\theta\Delta t_n} + \mathbf{V} + \mathbf{A} + \theta\mathbf{T} \right) \cdot \phi_{n+1} + \left( \frac{\mathcal{O}}{\theta\Delta t_n} + \mathbf{K}^d \right) \cdot \phi_{n+1}^d &= \\ \left( \frac{\mathcal{O}}{\theta\Delta t_n} + \mathbf{V} \right) \cdot \phi_{n+1}^d + \mathcal{O} \cdot \left[ \frac{1}{\theta\Delta t_n} \phi_n + \left( \frac{1}{\theta} - 1 \right) \dot{\phi}_n \right] + \mathbf{F}_{n+1} & \end{aligned} \quad (8.293)$$

which gives

$$\left( \frac{\mathcal{O}}{\theta\Delta t_n} + \mathbf{K}^d + \mathbf{V} + \mathbf{A} + \theta\mathbf{T} \right) \cdot \phi_{n+1} = \mathbf{V} \cdot \phi_n + \mathcal{O} \cdot \left[ \frac{1}{\theta\Delta t_n} \phi_n + \left( \frac{1}{\theta} - 1 \right) \dot{\phi}_n \right] + \mathbf{F}_{n+1} \quad (8.294)$$

The final matrix systems (8.292) and (8.294) for the  $\theta$ -family of time stepping methods and the BE and TR predictor-corrector schemes, respectively, are symmetric and positive definite. It results from the fact that the advective matrices  $\mathbf{V}$  and  $\mathbf{A}$  form a symmetric contribution as the sum  $(\mathbf{V} + \mathbf{A})$  because  $\mathbf{A} = \mathbf{V}^T$  is the transpose as easily seen from (8.286). This is only attainable for the ADE convective form (8.5) or (8.262). Unlikely, the ADE divergence form (8.3) is rather inappropriate for the PGLS method.<sup>15</sup>

<sup>15</sup>The ADE divergence form (8.3) contains a divergence expression of the advection term in form of  $\nabla \cdot (\mathbf{q}\phi)$ . For the split advective part (8.267) the advective operator  $\mathcal{L}^a$  would be

$$\mathcal{L}^a = \mathbf{q} \cdot \nabla + (\nabla \cdot \mathbf{q})$$

The symmetric term  $\mathbf{T}$  appearing in (8.292) and (8.294) can be interpreted as an additional term of artificial diffusion. This naturally results from the LS weighting procedure (8.277). In comparison to the SU method (see Sect. 8.14.3) where an anisotropic balancing dissipation tensor  $\mathbf{D}_{\text{num}} = \beta_{\text{num}} \frac{\mathbf{q} \otimes \mathbf{q}}{\|\mathbf{q}\|}$  (8.243) with  $\beta_{\text{num}}$  (8.245) as a function of the element length  $h^e$  and the upwind parameter  $\alpha$  is added, it is obvious that the LS damping matrix  $\mathbf{T}$  (8.285), (8.286) is identical to  $\mathbf{D}_{\text{num}}$  except for the parameter  $\beta_{\text{num}}$ , which becomes for the PGLS method

$$\beta_{\text{num}} = \frac{\Delta t_n \|\mathbf{q}^e\|}{\tilde{\mathcal{R}}} = Cr h^e \quad (8.295)$$

where  $Cr$  is the Courant number defined in (8.236). It reveals that the PGLS upwinding is quite similar to a SU method, where the damping is performed in the longitudinal direction of flow via the added damping parameter  $\beta_{\text{num}}$ . While in the SU method  $\beta_{\text{num}}$  is a function on the element length  $h^e$  and the free upwind parameter  $0 \leq \alpha \leq 1$ , in the PGLS the damping parameter  $\beta_{\text{num}}$  is dependent on the time step size  $\Delta t_n$  and the quotient  $\|\mathbf{q}^e\|/\tilde{\mathcal{R}}$ . Hence, PGLS is recognized as a built-in streamline-like upwind strategy, however, without any free upwind parameter. Comparing  $\beta_{\text{num}}$  for the PGLS of (8.295) with the SU method of (8.245) it is apparent that the Courant number should be  $Cr \leq 0.5$  for the PGLS (at linear elements) to avoid an overdamping larger than in the SU method.

### 8.14.6 An Illustrative Example

To demonstrate the impact of the different finite element schemes introduced above on stability and accuracy we consider a representative example of an advection-

---

Then, the LS weak statement of the advective part is

$$\text{LSWS} = \int_{\Omega} [\tilde{\mathcal{R}} N_i + \theta \Delta t_n \nabla \cdot (\mathbf{q} N_i)] [\tilde{\mathcal{R}} (\dot{\phi} - \dot{\phi}^d) + \nabla \cdot (\mathbf{q} \phi)] d\Omega = 0$$

which leads to a matrix system equivalent to (8.284), but having different element matrices

$$\begin{aligned} \mathbf{A}^e &= \int_{\Omega^e} N_i^e [(q^e \cdot \nabla N_j^e) + (\nabla \cdot \mathbf{q}^e) N_j^e] d\Omega^e \\ \mathbf{V}^e &= \int_{\Omega^e} [(q^e \cdot \nabla N_i^e) + (\nabla \cdot \mathbf{q}^e) N_i^e] N_j^e d\Omega^e \\ \mathbf{T}^e &= \int_{\Omega^e} \Delta t_n \frac{1}{\tilde{\mathcal{R}}} [\nabla \cdot (\mathbf{q}^e N_i^e)] [\nabla \cdot (\mathbf{q}^e N_j^e)] d\Omega^e \end{aligned}$$

While the symmetry of the matrix system is still maintained since  $\mathbf{A}^e = \mathbf{V}^{eT}$ , the divergence expressions  $(\nabla \cdot \mathbf{q}^e)$  appearing in  $\mathbf{A}^e$ ,  $\mathbf{V}^e$  and  $\mathbf{T}^e$  can cause difficulties if the flow is not selenoidal (i.e., not divergence-free:  $\nabla \cdot \mathbf{q} \neq 0$ ) at the presence of storage and sources/sinks. This makes the LS technique rather inappropriate for the ADE divergence form.

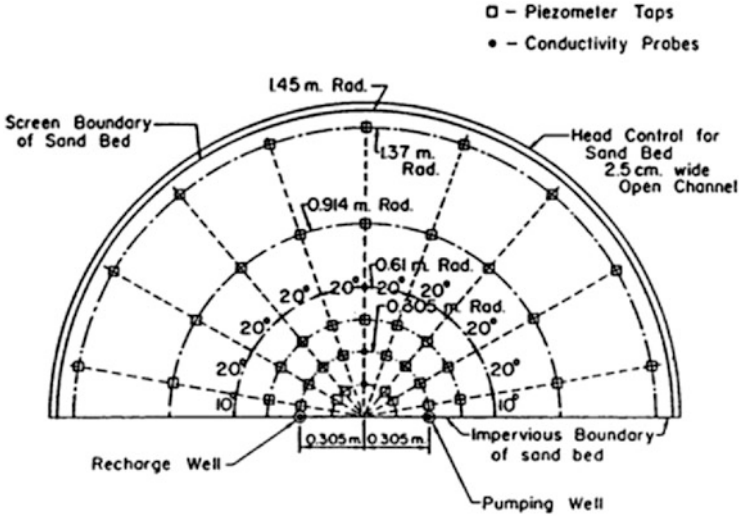


Fig. 8.31 Plane view of Hoopes and Harleermann’s sand-filled semi-cylinder [257]

dominated solute transport on a nonuniform flow field to which analytical and experimental results are available. It is known as the Hoopes and Harleermann’s two-well problem [257, 470]. Hoopes and Harleermann [257] performed a lab-scale experiment in a semi-cylinder filled with sand as shown in Fig. 8.31. They measured the distribution of a solute between a recharge and a pumping well. For an analytical solution they set up a conceptual model of a 2D horizontal confined aquifer which is homogeneous and isotropic. The nonuniform flow between the well doublet at a distance,  $2d = 0.61$  m, is isothermal and in a steady state. The solute transport is only affected by advection and dispersion. Comparisons of the analytical result with experiments and various numerical solution schemes have already been performed elsewhere [136, 257, 282, 470]. The Hoopes and Harleermann’s problem is now used to compare the different numerical schemes with the analytical (exact) results.

One obtains the 2D analytical solution in terms of the velocity potential  $\Phi$  and the streamline function  $\Psi$  (cf. Sect. 2.1.11). They are related to the original  $x_1, x_2$ -coordinates via the conformal transformation

$$\Phi + i\Psi = \text{Ln}(z + d)/(z - d), \quad \text{with } z = x_1 + ix_2 \tag{8.296}$$

where  $\text{Ln}$  is the complex natural logarithm,  $i$  corresponds to the imaginary unit and  $d$  is the half well spacing. This transformation maps the area of the half circle with radius  $r \leq d$  onto a strip of infinite length and width  $\pi/2$ . The governing transport equation can now be transformed to a 1D equation written in the form

$$\frac{\partial \phi}{\partial t} + v^2 \left( \frac{\partial \phi}{\partial \Phi} + D \frac{\partial^2 \phi}{\partial \Phi^2} \right) = 0 \tag{8.297}$$

where  $v$  is the intrinsic velocity and  $D = D_o + \beta_L v$  is the dispersion coefficient ( $D_o =$  molecular diffusion,  $\beta_L =$  longitudinal dispersivity). The intrinsic velocity  $v$  at a flux rate  $Q_w$  of the recharge well is given by

$$v = \frac{Q_w}{2\pi B \mathcal{R}d} (\cosh(\Phi_D) + \cos(\Psi_D)) \quad (8.298)$$

with the dimensionless quantities

$$\begin{aligned} \Phi_D &= \frac{1}{2} \ln \left( \frac{(x_1+d)^2 + x_2^2}{(x_1-d)^2 + x_2^2} \right) \\ \Psi_D &= \arctan \left( \frac{-2x_2 d}{x_1^2 + x_2^2 - d^2} \right) \end{aligned} \quad (8.299)$$

where  $B$  is the aquifer thickness. The velocity potential  $\Phi(x_1, x_2)$  and the stream-function  $\Psi(x_1, x_2)$  are obtained after multiplication with  $Q_w/(2\pi B)$ . The IC and BC are

$$\phi(\Phi, \Psi, t_0) = 0, \quad \text{and} \quad \phi(\Phi(-d, 0), \Psi(-d, 0), t) = \phi_w \quad (8.300)$$

The dimensionless solutal quantity at arbitrary time is given by

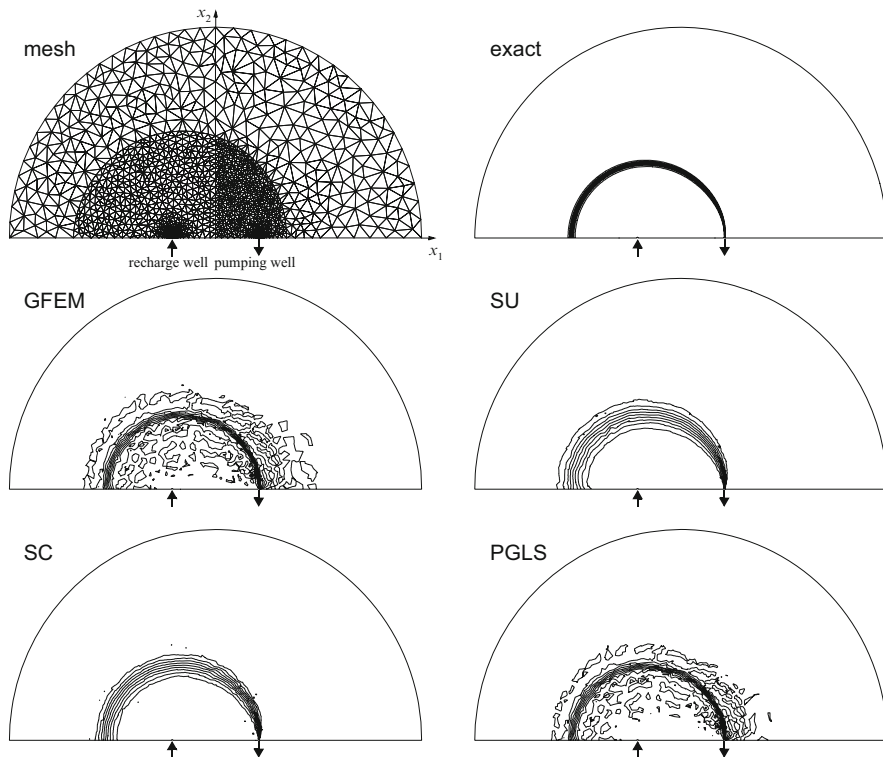
$$\phi_D = \frac{\phi}{\phi_w} = \frac{1}{2} \operatorname{erfc} \left( \frac{I_D - t_D}{2\sqrt{J_D}} \right) \quad (8.301)$$

where  $\operatorname{erfc}()$  is the complementary error function and  $t_D = Q_w t / (2\pi B \mathcal{R}d^2)$  is the dimensionless time. Owing to the properties of the conformal transformation (8.296), the solution (8.301) can be calculated for given spatial points, where the integrals are

$$I_D = \int_{-\infty}^{\Phi_D} \frac{d\Phi_D}{v_D^2}, \quad \text{and} \quad J_D = \int_{-\infty}^{\Phi_D} \frac{D_D}{v_D^4} d\Phi_D \quad (8.302)$$

in which  $v_D = 2\pi B \mathcal{R}d v / Q_w$  and  $D_D = 2\pi B \mathcal{R}D / Q_w$  are likewise dimensionless. The complete analytical solution is given in [257, 470], but, its evaluation is cumbersome.

For the present comparative study the finite element computations are performed on a triangle mesh of the symmetric half of the circular problem as shown in Fig. 8.32. The mesh is consciously chosen relatively coarse and only slightly refined in the vicinity of the recharge and pumping wells, where high velocities are expected. The used model parameters are listed in Table 8.8. The AB/TR predictor-corrector method with automatic time stepping is firstly employed. Hoopes and Harlemann [257] assumed no dispersion across streamlines in their formulation of (8.297). The longitudinal dispersivity  $\beta_L = 0.0015$  m is very small and gives rise to a steep front of solute in space and time. Hence, the transport is dominated



**Fig. 8.32** Used 2D mesh consisting of 4,890 triangles with 2,544 nodes and resulting isocontours of solution  $\phi$  at  $t = 0.2$  d computed by GFEM, SU, SC and PGLS using adaptive AB/TR time stepping in comparison to the exact distribution

by advection as evidenced in Fig. 8.33 for breakthrough curves of two observation points located at  $x_2 = 0.145$  m and  $x_2 = 0.305$  m along the symmetry line with  $x_1 = 0$  m between the wells. The breakthrough histories obtained for the different finite element schemes are compared to the exact curve. As seen the standard GFEM method provides oscillating numerical solutions, while the SU and SC schemes can completely dampen out the oscillations but introduce in turn spurious numerical dispersion. Figure 8.33 reveals that the PGLS scheme is not able to produce wigggle-free solutions. Since the stabilization in the PGLS is dependent on the actual time step size, cf. the relationship of (8.295), it is obvious that the time steps generated by the adaptive predictor-corrector procedure are unsuitably small to achieve sufficient damping via the LS mechanism.

The simulated solute distributions at  $t = 0.2$  d for the GFEM, SU, SC and PGLS schemes by using AB/TR time stepping are depicted in Fig. 8.32 for isocontours of ten solute levels spanning between the maximum and minimum values of the attained numerical results. The exact solution also shown in Fig. 8.32 reveals a sharp circular-shaped front, which can be hardly modeled by the numerical methods on the used coarse mesh. Thus, the GFEM and the PGLS schemes exhibit bounded

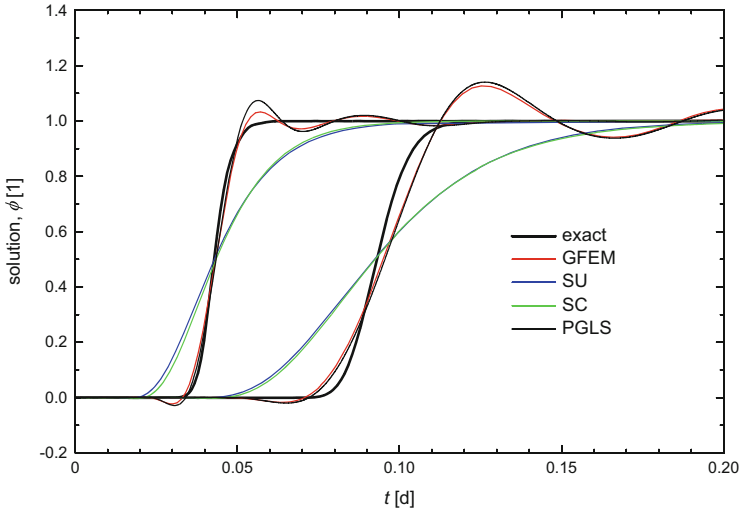


**Table 8.8** Simulation parameters of the Hoopes and Harlemann's two-well problem

Quantity	Symbol	Value	Unit
<i>Flow</i>			
Well discharge	$Q_w$	$2.339 \cdot 10^{-6}$	$\text{m}^3 \text{s}^{-1}$
Flux at recharge well	$\bar{q}_{n_h} = \frac{Q_w}{2\pi R}$	6.4327	$\text{m}^2 \text{d}^{-1}$
Head at pumping well	$h_D$	0	m
Isotropic aquifer transmissivity	$T$	$10^{-4}$	$\text{m}^2 \text{s}^{-1}$
<i>Solute transport</i>			
Initial condition	$\phi(\mathbf{x}, t_0)$	0	$\text{mg l}^{-1}$
Solute at recharge well	$\phi_w$	1	$\text{mg l}^{-1}$
Aquifer thickness	$B$	0.089	m
Porosity	$\varepsilon$	0.374	
Molecular diffusion	$D_o$	0	$\text{m}^2 \text{s}^{-1}$
Longitudinal dispersivity	$\beta_L$	0.0015	m
Transverse dispersivity	$\beta_T$	0	m
<i>FEM</i>			
Half well spacing	$d$	0.305	m
Wellbore radius	$R$	0.005	m
Outer boundary radius	$R_{\Omega}$	1.45	m
Number of triangular elements	$N_E$	4,890	
Number of mesh nodes	$N_P$	2,544	
Initial time step size	$\Delta t_0$	$10^{-5}$	d
RMS error tolerance (AB/TR method)	$\epsilon$	$10^{-4}$	1
Simulation time period	$t_{\text{end}}$	0.2	d

oscillations, which also spread in a distance from the front. On the other hand, the SU scheme results a quite wiggle-free distribution, however, the front is significantly widened due to the numerical dispersion effect. However, it is interesting that the SC scheme, while also providing essentially non-oscillatory solutions, can remarkably reduce the amount of spurious numerical dispersion and gives reasonably better results than the SU method.

To complete the comparison let us also investigate the 1st-order accurate FE/BE time stepping method, which promises a higher stability. Indeed, in this case even the GFEM leads to well-stabilized solutions as displayed in Fig. 8.34, however, we note a remarkable influence of numerical dispersion if the time steps are chosen large (we enforce large time steps by relaxing the RMS error criterion in the FE/BE predictor corrector method according to  $\epsilon = 10^{-2}$ ). To better understand the reason and measure of this effect, in the next section the quantities of numerical dispersion will be estimated for the different methods. Further numerical comparisons in 2D and 3D applications can be found in [136].

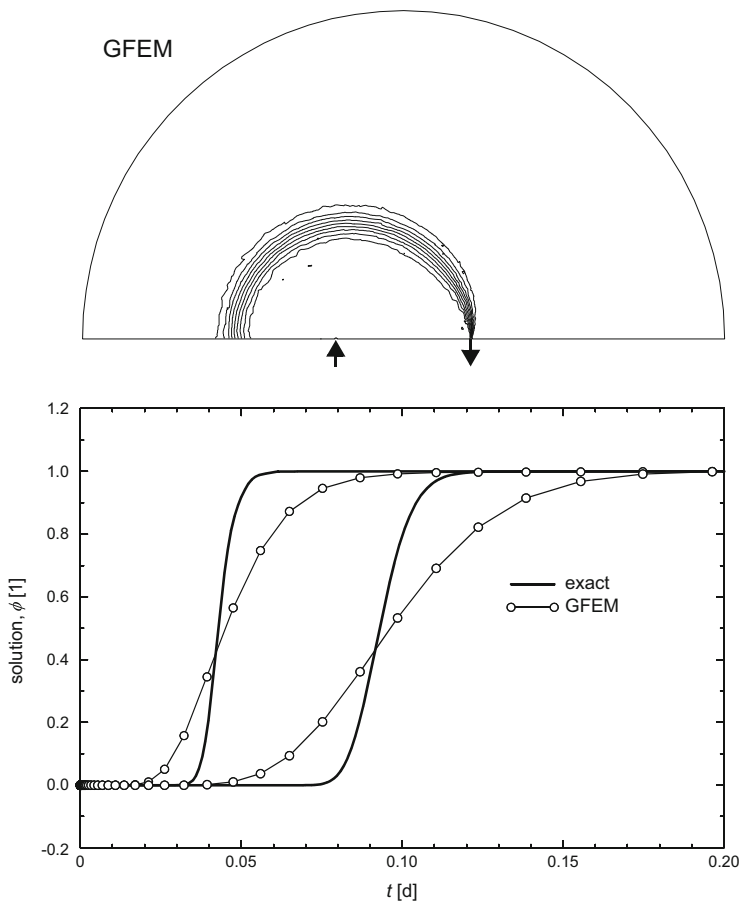


**Fig. 8.33** Breakthrough of approximate solutions obtained with the 2nd-order accurate AB/TR time integration method in comparison to the exact history at two points located at  $x_2 = 0.145$  m (*left*) and 0.305 m (*right*) on the symmetric line  $x_1 = 0$  between the wells

## 8.15 Summarized Quantitative Discussion of Error and Stability for the Favorite Schemes

A quantitative error estimate of the spatio-temporally discrete equations can be obtained by evaluating the LTE associated with the temporal and spatial derivatives. For this purpose let us consider the discrete equations (8.223) resulting from a PGFEM approximation of the 1D ADE convective form without sources/sinks and boundary terms for a uniform mesh with linear elements of length  $h$  ( $= \Delta x$ ), CM matrix and constant parameters written as

$$\begin{aligned}
 & \frac{\hat{\mathcal{R}}}{6} \left( \frac{\phi_{i+1,n+1} - \phi_{i+1,n}}{\Delta t_n} \right) + \frac{2\hat{\mathcal{R}}}{3} \left( \frac{\phi_{i,n+1} - \phi_{i,n}}{\Delta t_n} \right) + \frac{\hat{\mathcal{R}}}{6} \left( \frac{\phi_{i-1,n+1} - \phi_{i-1,n}}{\Delta t_n} \right) + \\
 & \theta \left[ -\frac{D}{h^2} + (1-\alpha) \frac{q}{2h} \right] \phi_{i+1,n+1} + (1-\theta) \left[ -\frac{D}{h^2} + (1-\alpha) \frac{q}{2h} \right] \phi_{i+1,n} + \\
 & \quad \theta \left( \frac{2D}{h^2} + \alpha \frac{q}{h} \right) \phi_{i,n+1} + (1-\theta) \left( \frac{2D}{h^2} + \alpha \frac{q}{h} \right) \phi_{i,n} + \\
 & \theta \left[ -\frac{D}{h^2} - (1+\alpha) \frac{q}{2h} \right] \phi_{i-1,n+1} + (1-\theta) \left[ -\frac{D}{h^2} - (1+\alpha) \frac{q}{2h} \right] \phi_{i-1,n} = 0
 \end{aligned} \tag{8.303}$$



**Fig. 8.34** Isocontours of solution at  $t = 0.2$  d and breakthrough curves simulated for GFEM by using the 1st-order accurate FE/BE predictor-corrector method with a relaxed RMS error criterion of  $\epsilon = 10^{-2}$

A Taylor series expansion in time for  $\phi_{i+1,n}$  about the time plane  $n + 1$  and for  $\phi_{i+1,n+1}$  about the time plane  $n$  gives, cf. (8.149)

$$\begin{aligned} \left( \frac{\phi_{i+1,n+1} - \phi_{i+1,n}}{\Delta t_n} \right) &= \theta \left[ \dot{\phi}_{i+1} - \frac{\Delta t_n}{2} \ddot{\phi}_{i+1} + \frac{\Delta t_n^2}{6} \ddot{\phi}_{i+1} - \mathcal{O}(\Delta t_n^3) \right]_{n+1} + \\ &\quad (1 - \theta) \left[ \dot{\phi}_{i+1} + \frac{\Delta t_n}{2} \ddot{\phi}_{i+1} + \frac{\Delta t_n^2}{6} \ddot{\phi}_{i+1} + \mathcal{O}(\Delta t_n^3) \right]_n \end{aligned} \tag{8.304}$$

Similar expressions result for  $(\phi_{i,n+1} - \phi_{i,n})/\Delta t_n$  and  $(\phi_{i-1,n+1} - \phi_{i-1,n})/\Delta t_n$ .

Now, we can also apply a Taylor series expansion in space for  $\phi_{i+1,n+1}$  and  $\phi_{i-1,n+1}$ , respectively, about  $i$  at a given time plane  $n + 1$  to obtain

$$\begin{aligned}\phi_{i+1,n+1} &= \left[ \phi_i + h\phi'_i + \frac{h^2}{2}\phi''_i + \frac{h^3}{6}\phi'''_i + \mathcal{O}(h^4) \right]_{n+1} \\ \phi_{i-1,n+1} &= \left[ \phi_i - h\phi'_i + \frac{h^2}{2}\phi''_i - \frac{h^3}{6}\phi'''_i + \mathcal{O}(h^4) \right]_{n+1}\end{aligned}\quad (8.305)$$

where  $'$  denotes differentiation with respect to the 1D space coordinate  $\partial/\partial x$ . Similar expressions result for  $\phi_{i+1}$  and  $\phi_{i-1}$  at the other time planes  $n$  and  $n - 1$ , i.e.,  $\phi_{i+1,n}$ ,  $\phi_{i-1,n}$ ,  $\phi_{i+1,n-1}$  and  $\phi_{i-1,n-1}$ , respectively.

To obtain expressions for higher order time derivatives  $\ddot{\phi}$  and  $\dddot{\phi}$  in terms of spatial derivatives, the governing 1D ADE  $\dot{\phi} + q\phi' - D\phi'' = 0$  may be rewritten in the form

$$\dot{\phi} = -q\phi' + D\phi'' \quad (8.306)$$

Differentiating (8.306) with respect to time, rearranging the differentials and successively substituting again (8.306) in the resulting expression, gives

$$\ddot{\phi} = \frac{\partial}{\partial t}(-q\phi' + D\phi'') = -q\frac{\partial}{\partial x}\dot{\phi} + D\frac{\partial}{\partial x^2}\dot{\phi} = q^2\phi'' - 2qD\phi''' + D^2\phi^{(4)} \quad (8.307)$$

and similarly

$$\dddot{\phi} = -q^3\phi''' + 3q^2D\phi^{(4)} - 3qD^2\phi^{(5)} + D^3\phi^{(6)} \quad (8.308)$$

Now, inserting (8.304) and the related expressions into (8.303), replacing all higher order time derivatives by spatial derivatives via (8.307) and (8.308) as well as substituting with (8.305) all  $(i + 1)$ th and  $(i - 1)$ th terms, we find after some manipulations the following approximate representation of the governing ADE at node  $i$  and time plane  $n + \theta$  (note that  $t_{n+\theta} = \theta t_{n+1} + (1 - \theta)t_n$ ):

$$\begin{aligned}(\dot{\mathcal{R}}\dot{\phi} + q\phi' - D\phi'')_{i,n+\theta} &= \left(\alpha\frac{qh}{2} + \frac{\Delta t_n}{2}q^2\right)\theta\phi''_{i,n+1} + \left(\alpha\frac{qh}{2} - \frac{\Delta t_n}{2}q^2\right)(1-\theta)\phi''_{i,n} \\ &+ \left(\frac{\Delta t_n^2}{6}q^3 - \Delta t_n qD\right)\theta\phi'''_{i,n+1} + \left(\frac{\Delta t_n^2}{6}q^3 + \Delta t_n qD\right)(1-\theta)\phi'''_{i,n} \\ &+ \left(\frac{h^2}{12}D - \frac{\Delta t_n}{2}D^2 + \frac{\Delta t_n^2}{2}q^2D + \frac{\Delta t_n^3}{24}q^4 - \alpha\frac{q}{24}h^3\right)\theta\phi^{(4)}_{i,n+1} \\ &+ \left(\frac{h^2}{12}D + \frac{\Delta t_n}{2}D^2 + \frac{\Delta t_n^2}{2}q^2D + \frac{\Delta t_n^3}{24}q^4 - \alpha\frac{q}{24}h^3\right)(1-\theta)\phi^{(4)}_{i,n} \\ &+ \text{HOT}\end{aligned}\quad (8.309)$$

or simply

$$\dot{\mathcal{R}}\dot{\phi} + q\phi' = (D + D_{\text{num}})\phi'' + \mathcal{O}(h^2, (2\theta - 1)\Delta t_n, \Delta t_n^2) \quad (8.310)$$

where the *numerical dispersion* coefficient  $D_{\text{num}}$  associated with the second spatial derivatives on the RHS of (8.309) appears

$$D_{\text{num}} = \alpha \frac{qh}{2} + \Delta t_n q^2 (\theta - \frac{1}{2}) \quad (8.311)$$

The RHS of (8.309) encompasses the total truncation error of the spatio-temporal discretization, which has no physical basis. The coefficient  $D_{\text{num}}$  of the *spurious* (unphysical) numerical dispersion covers the leading terms of the truncation errors, which are of 1st order in space and time.

In generalization of (8.310) we can find a semidiscrete representation of the governing ADE convective form in multidimensions as

$$\hat{\mathcal{R}} \frac{\partial \phi}{\partial t} + \mathbf{q} \cdot \nabla \phi - \nabla \cdot [(\mathbf{D} + \mathbf{D}_{\text{num}}) \cdot \nabla \phi] + (\vartheta + Q)\phi - H - Q_{\phi w} = \mathcal{O}(h^{e^2}, (2\theta - 1)\Delta t_n, \Delta t_n^2) \quad (8.312)$$

with the tensor of numerical dispersion

$$\mathbf{D}_{\text{num}} = D_{\text{num}} \boldsymbol{\delta} + \beta_{\text{num}} \frac{\mathbf{q} \otimes \mathbf{q}}{\|\mathbf{q}\|} \quad (8.313)$$

where  $\beta_{\text{num}}$  is the streamline-oriented coefficient of numerical dispersion while the scalar numerical dispersion  $D_{\text{num}}$  consists of the spatial part  $D_{\text{num}}^{\text{space}}$  and the temporal part  $D_{\text{num}}^{\text{time}}$ , viz.,

$$D_{\text{num}} = D_{\text{num}}^{\text{space}} + D_{\text{num}}^{\text{time}} \quad (8.314)$$

with

$$\begin{aligned} D_{\text{num}}^{\text{space}} &\sim \text{different for GFEM, SU, FU, SC, PGLS} \\ D_{\text{num}}^{\text{time}} &= \Delta t_n q^{e^2} (\theta - \frac{1}{2}) \end{aligned} \quad (8.315)$$

The coefficients of numerical dispersion, the orders of accuracy and the stability restrictions for the different favorite schemes discussed above in Sects. 8.13 and 8.14 are summarize in Table 8.9, where the parameters are evaluated at element level. The standard Galerkin FEM with the TR (Crank-Nicolson) time stepping is recognized as the most accurate method which is 2nd-order accurate in space and time  $\mathcal{O}(h^{e^2}, \Delta t_n^2)$  without numerical dispersion, however, it is only conditionally stable. Most restrictive and crucial for the GFEM is the  $Pg < 1$  condition, unless oscillatory solutions can be produced. The diffusion limit  $Cr < Pg$  is commonly not important for the TR (Crank-Nicolson) scheme, while the advection-diffusion limit  $PgCr < 1$  can be more serious for dominant advection if using the TR time stepping. In Table 8.9 the accuracy of the schemes decreases from top to down in favor of increasing stability. However, the higher stability is paid by an increased amount of spurious numerical dispersion which can significantly exceed the physical dispersion/diffusion  $D^e$  if the mesh is coarse and/or the time steps are large. Here, the FU scheme with fully implicit time stepping of 1st-order accuracy  $\mathcal{O}(h^e, \Delta t_n)$ , while unconditionally stable, tends to produce very overdiffusive results if  $h^e$  and/or  $\Delta t_n$  are large. Compromises between stability and accuracy are possible

**Table 8.9** Estimated accuracy and stability restrictions of the favorite schemes using linear finite elements and semi-implicit or implicit ( $\mathcal{A}$ -stable) time integration in solving the ADE (8.5) or (8.3) at presence of advection  $\mathbf{q} \neq \mathbf{0}$ . Note that for diffusion/conduction problems ( $\mathbf{q} \equiv \mathbf{0}$ ) only the standard GFEM is applied possessing no numerical dispersion and no stability restrictions, except the diffusion limit  $Cr < Pg$ , ( $\Delta t_n < \mathcal{R}h^{e2}/(2D^e)$ ), for the 2nd-order accurate TR time integration method

Scheme	Time integration <sup>c</sup>	Accuracy			Stability <sup>a</sup>		
		Numerical dispersion <sup>b</sup>		Order	Temporal limits <sup>d</sup>	Spatial limits	
		$D_{num}^e$	$\beta_{num}^e$				
GFEM <sup>f</sup>	TR	–	–	$\mathcal{O}(h^{e2}, \Delta t_n^2)$	↑ high	$Cr < Pg < \frac{1}{Cr}$	$Pg < 1$
	BE	$\frac{\Delta t_n}{2} q^{e2}$	–	$\mathcal{O}(h^{e2}, \Delta t_n)$		–	–
PGLS <sup>g</sup>	TR	–	$Cr h^e$	$\mathcal{O}(h^{e2}, \Delta t_n)$	↑	$Cr < Pg < \frac{1}{Cr}$	–
	BE	$\frac{\Delta t_n}{2} q^{e2}$	$Cr h^e$	$\mathcal{O}(h^{e2}, \Delta t_n)$		–	–
SU <sup>h</sup>	TR	–	$\frac{h^e}{2}$	$\mathcal{O}(h^e, \Delta t_n^2)$	↑	$Cr < Pg < \frac{1}{Cr}$	–
	BE	$\frac{\Delta t_n}{2} q^{e2}$	$\frac{h^e}{2}$	$\mathcal{O}(h^e, \Delta t_n)$		–	–
SC <sup>i</sup>	TR	$\frac{1}{2} \alpha_c h^e \ \mathbf{q}_{  }\ $	–	$\mathcal{O}(h^e, \Delta t_n^2)$	↑	$Cr < Pg < \frac{1}{Cr}$	–
	BE	$\frac{1}{2} \alpha_c h^e \ \mathbf{q}_{  }\  + \frac{\Delta t_n}{2} q^{e2}$	–	$\mathcal{O}(h^e, \Delta t_n)$		–	–
FU <sup>j</sup>	TR	$\frac{h^e}{2} \ \mathbf{q}^e\ $	–	$\mathcal{O}(h^e, \Delta t_n^2)$	↑	$Cr < Pg < \frac{1}{Cr}$	–
	BE	$\frac{1}{2}(h^e + \Delta t_n \ \mathbf{q}^e\ ) \ \mathbf{q}^e\ $	–	$\mathcal{O}(h^e, \Delta t_n)$		low	–

<sup>a</sup> Necessary but not always sufficient to ensure boundedness and prevent oscillations  
<sup>b</sup> Expressed in the element tensor  $D_{num}^e = D_{num}^e \delta + \beta_{num}^e \frac{\mathbf{q}^e \otimes \mathbf{q}^e}{\|\mathbf{q}^e\|}$  of numerical dispersion (8.313)  
<sup>c</sup>  $\theta$ -family and corrector methods:  $\theta = \frac{1}{2}$ , TR (Crank-Nicolson);  $\theta = 1$ , BE (fully implicit)  
<sup>d</sup> Conditionally stable,  $Cr = \frac{\|\mathbf{q}^e\| \Delta t_n}{\mathcal{R}h^e}$ ,  $Pg = \frac{\|\mathbf{q}^e\| h^e}{2\|D^e\|}$  ( $D^e$  = physical dispersion,  $h^e$  by (8.239))  
<sup>f</sup> Standard Galerkin without any upwinding,  $\alpha = 0$   
<sup>g</sup> Least square strategy suitable for ADE convective form  
<sup>h</sup> Streamline upwinding used with  $\alpha = 1$ , (8.245)  
<sup>i</sup> Shock capturing with projected flux  $\|\mathbf{q}_{||}^e\| = \frac{|\mathbf{q}^e \cdot \nabla \phi^e|}{\|\nabla \phi^e\|}$  and upwind parameter  $\alpha_c$ , (8.259)  
<sup>j</sup> Full upwinding equivalent to (8.251) with  $\beta_{num}^e = \frac{h^e}{2}$

by resorting to the PGLS, SU or SC schemes, where always the 1st-order accurate BE time stepping provides a higher stability in contrast to the 2nd-order accurate TR time stepping scheme.

In predictor-corrector time stepping only the corrector solutions are important for the stability analysis, while the explicit predictors provide prolonged solutions, which are primarily used to estimate the accuracy in comparison to the corrector solutions needed in the adaptive time stepping control. However, the accuracy of the predictor and corrector must be consistent, so that FE and BE are both 1st-order accurate in time as well as the AB and TR are both 2nd-order accurate in time, cf. Sects. 8.13.5.1 and 8.13.5.2, respectively. No upwinding is used for the explicit FE and AB predictor schemes, i.e.,  $\alpha = 0$ . However, we note from (8.311) with  $\theta = 0$  a negative numerical dispersion coefficient  $D_{num}^e = -\Delta t_n q^{e2}/2$  arises for the FE predictor.

It is important to note that the stability bounds and errors of numerical dispersion listed in Table 8.9 only occur in the presence of advection  $\mathbf{q} \neq \mathbf{0}$ . Without advection  $\mathbf{q} \equiv \mathbf{0}$  there is no need to use the PGLS, SU, SC and FU schemes.

For diffusion/conduction problems the standard GFEM is most optimal and unconditionally stable, except for the diffusion limit  $Cr < Pg$ , ( $\Delta t_n < \hat{\mathcal{R}}h^{e^2}/(2D^e)$ ), arising for the 2nd-order accurate TR time integration method, which is, however, commonly noncrucial.

## 8.16 Implementation of Dirichlet-Type BC's in the Resulting Matrix System

In formulating the weak statements in Sect. 8.9 the specification of Dirichlet boundaries  $\Gamma_D$  remains of particular concern. So far, the resulting weak statements do not incorporate any BC's of Dirichlet type, now we have to make up for it. In principle, there are two ways for implementing Dirichlet-type (essential) BC's. First, they can be mimicked via a Cauchy-type BC if the transfer coefficient  $\Phi^e$  (or  $\Phi^{\dagger e}$ ) appearing in (8.104) and (8.105), associated with a global node  $i$  and the adjacent elements, is set to an arbitrary large value (theoretically,  $\Phi^e \rightarrow \infty$ ). It enforces that the condition  $\phi_i \approx \phi_C$  is satisfied in a reasonable approximation (the larger  $\Phi^e$ , the better the approximation of  $\phi_i \approx \phi_C$ ), such that the Dirichlet BC appears as a special case of the Cauchy BC. While this method is very easy and efficient, it has a clear disadvantage; namely, the large value required for the transfer coefficient increases significantly the parameter contrast in the resulting matrix system, which can deteriorate the properties of the system matrix causing negative impact on the solution of the sparse equation system.

The second method which is preferred here avoids those circumstances and satisfies the Dirichlet-type BC's in an exact manner without deteriorating the matrix system. The basic idea is that the solution at a Dirichlet boundary  $\Gamma_D$  is known and accordingly all nodes sharing  $\Gamma_D$  can be eliminated from the computations by a simple bookkeeping procedure. Let us consider a matrix system resulting from a spatio-temporal discretization such as given in (8.154) or (8.177), which leads always to a linear (or linearized) sparse system of equations written in a compact matrix form (note that the time plane indexing is dropped for convenience)

$$\mathbf{A} \cdot \boldsymbol{\phi} = \mathbf{r} \quad (8.316)$$

or

$$\begin{pmatrix} A_{11} & A_{12} & \dots & A_{1i} & \dots & A_{1N_p} \\ A_{21} & A_{22} & \dots & A_{2i} & \dots & A_{2N_p} \\ \vdots & \vdots & & \vdots & & \vdots \\ A_{i1} & A_{i2} & \dots & A_{ii} & \dots & A_{iN_p} \\ \vdots & \vdots & & \vdots & & \vdots \\ A_{N_p1} & A_{N_p2} & \dots & A_{N_pi} & \dots & A_{N_pN_p} \end{pmatrix} \cdot \begin{pmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_i \\ \vdots \\ \phi_{N_p} \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_i \\ \vdots \\ r_{N_p} \end{pmatrix} \quad (8.317)$$

which has to be solved at each time stage for the unknown solution vector  $\phi$  consisting of  $N_p$  components, where  $\mathbf{A}$  is the sparse system matrix of dimension  $N_p \times N_p$  comprising all terms of the LHS and  $\mathbf{r}$  is the  $N_p$ -dimensional RHS vector comprising all RHS terms of the basic discrete system (8.154) or (8.177). Now, assuming that the solution at the  $i$ th node is known, i.e.,  $\phi_i = \phi_D$ , where  $\phi_D$  is a prescribed Dirichlet value, then (8.317) can be rewritten

$$\begin{pmatrix} A_{11} & A_{12} & \dots & A_{1i} & \dots & A_{1N_p} \\ A_{21} & A_{22} & \dots & A_{2i} & \dots & A_{2N_p} \\ \vdots & \vdots & & \vdots & & \vdots \\ 0 & 0 & \dots & 1 & \dots & 0 \\ \vdots & \vdots & & \vdots & & \vdots \\ A_{N_p 1} & A_{N_p 2} & \dots & A_{N_p i} & \dots & A_{N_p N_p} \end{pmatrix} \cdot \begin{pmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_i \\ \vdots \\ \phi_{N_p} \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ \phi_D \\ \vdots \\ r_{N_p} \end{pmatrix} \quad (8.318)$$

To retrieve a possible symmetry of  $\mathbf{A}$ , it is useful to shift the  $i$ th column to the RHS. It yields the equivalent formulation

$$\begin{pmatrix} A_{11} & A_{12} & \dots & 0 & \dots & A_{1N_p} \\ A_{21} & A_{22} & \dots & 0 & \dots & A_{2N_p} \\ \vdots & \vdots & & \vdots & & \vdots \\ 0 & 0 & \dots & 1 & \dots & 0 \\ \vdots & \vdots & & \vdots & & \vdots \\ A_{N_p 1} & A_{N_p 2} & \dots & 0 & \dots & A_{N_p N_p} \end{pmatrix} \cdot \begin{pmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_i \\ \vdots \\ \phi_{N_p} \end{pmatrix} = \begin{pmatrix} r_1 - A_{1i}\phi_D \\ r_2 - A_{2i}\phi_D \\ \vdots \\ \phi_D \\ \vdots \\ r_{N_p} - A_{N_pi}\phi_D \end{pmatrix} \quad (8.319)$$

This bookkeeping procedure can be done for all Dirichlet nodes. Assuming there are  $N_D$  Dirichlet BC's in total, which are implemented in the matrix the system, the actual number of equations  $N_{EQ}$  which has to be solved is

$$N_{EQ} = N_p - N_D \quad (8.320)$$

and the final matrix system is compressed to the actual set of equations in the form

$$\begin{pmatrix} A_{11} & A_{12} & \dots & A_{1N_{EQ}} \\ A_{21} & A_{22} & \dots & A_{2N_{EQ}} \\ \vdots & \vdots & \ddots & \vdots \\ A_{N_{EQ}1} & A_{N_{EQ}2} & \dots & A_{N_{EQ}N_{EQ}} \end{pmatrix} \cdot \begin{pmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_{N_{EQ}} \end{pmatrix} = \begin{pmatrix} r_1 - A_{1i}\phi_D \\ r_2 - A_{2i}\phi_D \\ \vdots \\ r_{N_{EQ}} - A_{N_{EQ}i}\phi_D \end{pmatrix} \quad (8.321)$$

where all rows and columns of  $\mathbf{A}$  are removed to which Dirichlet-type BC's are associated. In the practical solution, a *profiling* of the matrix system can be easily performed, where all Dirichlet equations are determined and eliminated from the matrix system which is actually solved. The procedure is accurate and efficient because the properties of  $\mathbf{A}$  remain unchanged and the equation system is reduced by the  $N_D$  entries.



## 8.17 Solution of Linear Systems of Algebraic Equations

The spatio-temporal finite element approximation such as given by (8.154) or (8.177) leads to a matrix system in form of (8.316). After elimination the  $N_D$  Dirichlet-type BC's from the  $N_P$  equations as described in the preceding Sect. 8.16, we end up with a system of simultaneous linear (or linearized) algebraic equations written in matrix form (for sake of simplicity we drop the time plane indexing in case of transient ADE) as

$$\mathbf{A} \cdot \boldsymbol{\phi} = \mathbf{b} \quad (8.322)$$

or in index notation

$$A_{ij}\phi_j = b_i \quad (1 \leq i, j \leq N_{EQ}) \quad (8.323)$$

or

$$\begin{pmatrix} A_{11} & A_{12} & \dots & A_{1N_{EQ}} \\ A_{21} & A_{22} & \dots & A_{2N_{EQ}} \\ \vdots & \vdots & \ddots & \vdots \\ A_{N_{EQ}1} & A_{N_{EQ}2} & \dots & A_{N_{EQ}N_{EQ}} \end{pmatrix} \cdot \begin{pmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_{N_{EQ}} \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_{N_{EQ}} \end{pmatrix} \quad (8.324)$$

which has to be solved for the  $N_{EQ}$ -dimensional solution vector  $\boldsymbol{\phi}$ , where the system matrix  $\mathbf{A}$  has  $N_{EQ}$  rows and columns. The RHS-vector  $\mathbf{b}$ , containing additionally the Dirichlet BC terms according to (8.321), has  $N_{EQ}$  components. It is to be noted that for transient problems we always prefer implicit or semi-implicit time integration schemes due to stability and performance reasons, which essentially require the solution of equation systems (in contrast to temporally explicit schemes, where in combination with mass lumping, cf. Sect. 8.13.2, there is no need to solve a system of simultaneous equations, however, at the expense of a commonly huge number of time steps as discussed in Sect. 8.13.6). Furthermore, steady-state problems, in which the time step is deemed infinitely large  $\Delta t_n \rightarrow \infty$ , an equation system in form of (8.322) has inevitably to be solved.

The unique solution of (8.322) at given  $\mathbf{A}$  and  $\mathbf{b}$  in a form

$$\boldsymbol{\phi} = \mathbf{A}^{-1} \cdot \mathbf{b} \quad (8.325)$$

only exists when  $\mathbf{A}$  is non-singular, i.e.,  $\mathbf{A}$  must have a non-vanishing determinant  $|\mathbf{A}| \neq 0$ . The system matrix  $\mathbf{A}$  is usually unsymmetric, i.e.,  $\mathbf{A} \neq \mathbf{A}^T$ , when advection terms occur in the discrete formulation (except for the PGLS method introduced in Sect. 8.14.5). On the other hand,  $\mathbf{A}$  can also be symmetric, i.e.,  $\mathbf{A} = \mathbf{A}^T$ , for instance when terms of advection are absent. As a consequence of the used finite element discretization the system matrix  $\mathbf{A}$  is *sparse*, i.e., many of its components are zero, and possesses a definite structure which is determined by its

**Table 8.10** Advantages versus disadvantages of direct and iterative solution techniques

Method	Advantage	Disadvantage
Direct	Solution of $\mathbf{A} \cdot \phi = \mathbf{b}$ is exact. Sequence of operations only performed once. No initial estimates and iterations are required	May be inefficient for large problems, in particular in 3D. Can produce round-off errors
Iterative	Efficient with respect to storage demand and CPU time	Solution of $\mathbf{A} \cdot \phi = \mathbf{b}$ is approximative. Initial estimates and iteration parameters are required. System matrix $\mathbf{A}$ should be well-conditioned

non-zero components. A method of inversion of  $\mathbf{A}$ , in particular when the order  $N_{\text{EQ}}$  of the matrix becomes large, depends very much on the structure of  $\mathbf{A}$ . Accordingly, efficient solution methods will utilize the sparsity structure of  $\mathbf{A}$  under exploitation of its symmetry if occurring. In general, we can differ into two major solution strategies: (1) direct and (2) iterative techniques, e.g., [15, 376, 430, 453, 590]. For large problems, particularly in 3D applications, iterative solution methods are more efficient than direct solution techniques, however, they may suffer sometimes from a poor convergence behavior. The relative merits of direct and iterative solution techniques are listed in Table 8.10. In recent years, due to the increases in computer memory and the suitability for shared-memory multiprocessing there is a revival of direct solution methods, e.g., [461].

### 8.17.1 Direct Solution Methods

#### 8.17.1.1 Gaussian Elimination

The classic direct solution method is the *Gaussian elimination*. Its objective is to subtract appropriately scaled rows in the system (8.324) to arrive at an upper triangular matrix equation in the form

$$\mathbf{A} \cdot \phi = \mathbf{b} \longrightarrow \mathbf{U} \cdot \phi = \mathbf{b}' \quad (8.326)$$

where  $\mathbf{U}$  is an upper triangular matrix. We obtain  $\mathbf{U}$  by following procedure termed as forward elimination: We choose the first row as the *pivot* equation and eliminate  $\phi_1$  from each equation below it. This is achieved by multiplying the first equation by  $A_{21}/A_{11}$  provided the *pivot element*  $A_{11} \neq 0$ , which is then subtracted from the second equation. It is continued similarly until  $\phi_1$  is eliminated from all equations. Now, we eliminate  $\phi_2, \phi_3, \dots$  in the same manner until the upper triangular form is attained

$$\begin{pmatrix} U_{11} & U_{12} & \dots & U_{1N_{\text{EQ}}} \\ 0 & U_{22} & \dots & U_{2N_{\text{EQ}}} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & U_{N_{\text{EQ}}N_{\text{EQ}}} \end{pmatrix} \cdot \begin{pmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_{N_{\text{EQ}}} \end{pmatrix} = \begin{pmatrix} b'_1 \\ b'_2 \\ \vdots \\ b'_{N_{\text{EQ}}} \end{pmatrix} \quad (8.327)$$

where the components of the first row are  $U_{1j} = A_{1j}$ , ( $j=1, \dots, N_{\text{EQ}}$ ) and  $b'_1 = b_1$ . The solution  $\phi$  is then easily be performed by a recursive bottom-up *backsubstitution* as follows

$$\begin{aligned} \phi_{N_{\text{EQ}}} &= b'_{N_{\text{EQ}}} / U_{N_{\text{EQ}}N_{\text{EQ}}} \\ \phi_i &= (b'_i - U_{ij}\phi_j) / U_{ii}, \quad j > i \end{aligned} \quad (8.328)$$

We recognize that the Gaussian elimination changes the RHS vector  $\mathbf{b}$  to  $\mathbf{b}'$ , which makes this technique rather inappropriate for systems with multiple RHS's. This can be circumvented by the following decomposition solution strategy.

### 8.17.1.2 LU Matrix Decomposition and Crout Method

The preferred variant of Gaussian elimination is the *Crout method*, in which the RHS vector  $\mathbf{b}$  is not affected by the matrix decomposition. It is very advantageous for matrix systems where  $\mathbf{A}$  does not change in time (when using constant time steps) so that  $\mathbf{A}$  needs to be decomposed only once. In such cases  $\phi$  can be easily computed via simple backsubstitution for every time-varying RHS vector  $\mathbf{b}$ , a considerably fast computational process termed as *resolution*. In the Crout method the matrix  $\mathbf{A}$  is decomposed into a lower triangular matrix  $\mathbf{L}$  and an upper triangular matrix  $\mathbf{U}$ , viz.,

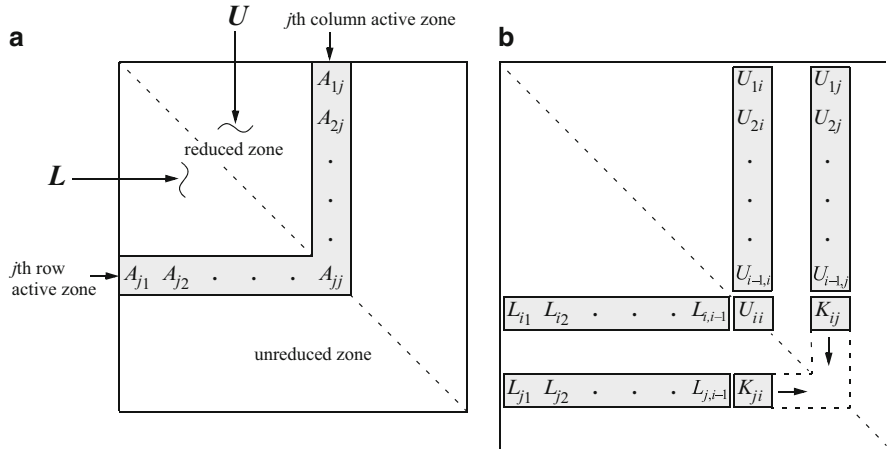
$$\mathbf{A} = \mathbf{L} \cdot \mathbf{U} \quad (8.329)$$

where

$$\mathbf{L} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ L_{21} & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ L_{N_{\text{EQ}}1} & L_{N_{\text{EQ}}2} & \dots & 1 \end{pmatrix} \quad (8.330)$$

and

$$\mathbf{U} = \begin{pmatrix} U_{11} & U_{12} & \dots & U_{1N_{\text{EQ}}} \\ 0 & U_{22} & \dots & U_{2N_{\text{EQ}}} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & U_{N_{\text{EQ}}N_{\text{EQ}}} \end{pmatrix} \quad (8.331)$$



**Fig. 8.35** *LU* decomposition of matrix *A* in the Crout elimination method: (a) reduced, active and unreduced zones, (b) terms used to construct  $U_{ij}$  and  $L_{ji}$

Then, the linear equation system (8.322) expressed with (8.329)

$$A \cdot \phi = (L \cdot U) \cdot \phi = L \cdot \underbrace{(U \cdot \phi)}_y = b \tag{8.332}$$

can now be solved via the pair of equations

$$L \cdot y = b \tag{8.333}$$

and

$$U \cdot \phi = y \tag{8.334}$$

The *LU* decomposition (also called factorization) of  $A = L \cdot U$  represents the crucial and most costly solution step. The Crout method computes *L* and *U* by a continuous accumulation of products and does not need to record the intermediate reduced matrices. In this *LU* decomposition process the matrix *A* divides into three zones as outlined in Fig. 8.35. There is a region that is fully reduced, in the second (called active) zone the matrix is currently being reduced and there is a third zone, which contains the original unreduced matrix components. Taking

$$L_{ii} \equiv 1 \quad (i = 1, 2, \dots, N_{EQ}) \tag{8.335}$$

for each active zone *j* the entries of *U* and *L* are given by

$$\begin{aligned}
 U_{ij} &= A_{ij} - \sum_{m=1}^{i-1} L_{im}U_{mj} & (j = i, i+1, \dots, N_{\text{EQ}}) \\
 L_{ji} &= \left( A_{ji} - \sum_{m=1}^{i-1} L_{jm}U_{mi} \right) / U_{ii} & (j = i+1, i+2, \dots, N_{\text{EQ}})
 \end{aligned}
 \tag{8.336}$$

for  $(i = 1, 2, \dots, N_{\text{EQ}})$ . We note that the summation in (8.336) is ignored when the lower limit of the index  $m$  exceeds the upper limit.

It is obvious from (8.336) that the diagonal entry in the matrix  $U$  must be non-zero. This can be assumed for a matrix  $A$  which is diagonally dominant, i.e.,

$$|A_{ii}| \geq \sum_{j \neq i} |A_{ij}| \quad (i = 1, 2, \dots, N_{\text{EQ}}) \tag{8.337}$$

In other cases partial *pivoting* is required in which rows of  $A$  are appropriately interchanged to meet non-zero diagonals. The above decomposition can be used for both unsymmetric and symmetric matrices  $A$ . However, if  $A$  is symmetric,  $A = A^T$ , the relation exists

$$U_{ij} = L_{ji}U_{ii} \tag{8.338}$$

and it is no more necessary to store the complete matrix. Only the diagonals and the components above the diagonals need to be stored, while (8.338) is utilized to construct the missing part. It reduces the decomposition costs by nearly 50% [509].

Having completed the decomposition of  $A$ , it is now trivial to solve  $\phi$  by utilizing (8.333) and (8.334) in a forward elimination and backward substitution procedure, viz.,

$$y_i = b_i - \sum_{j=1}^{i-1} L_{ij}y_j \quad (i = 1, 2, \dots, N_{\text{EQ}}) \tag{8.339}$$

and

$$\phi_i = \left( y_i - \sum_{j=i+1}^{N_{\text{EQ}}} U_{ij}\phi_j \right) / U_{ii} \quad (i = N_{\text{EQ}}, N_{\text{EQ}} - 1, \dots, 1) \tag{8.340}$$

respectively, which are computationally cheap. We see that in (8.339) the RHS vector  $b$  is not destroyed during the forward elimination, which makes the Crout method very useful for multiple RHS's in a resolution process avoiding a repeated  $LU$  decomposition.

### 8.17.1.3 Other Methods

A symmetric matrix  $\mathbf{A}$  which is *positive definite*, i.e.,

$$\phi \cdot (\mathbf{A} \cdot \phi) > 0 \quad \text{for all } \phi \neq \mathbf{0} \quad (8.341)$$

can be decomposed in the form

$$\mathbf{A} = \mathbf{L} \cdot \mathbf{U} = \mathbf{L} \cdot (\mathbf{D} \cdot \mathbf{L}^T) = \underbrace{(\mathbf{L} \cdot \mathbf{D}^{1/2})}_{\tilde{\mathbf{L}}} \cdot \underbrace{(\mathbf{D}^{1/2} \cdot \mathbf{L}^T)}_{\tilde{\mathbf{L}}^T} \quad (8.342)$$

where  $\mathbf{D}$  is the diagonal matrix defined as

$$\mathbf{D} = [U_{11}, U_{22}, \dots, U_{N_{\text{EQ}}N_{\text{EQ}}}] \quad (8.343)$$

The decomposition (8.342) in the form

$$\mathbf{A} = \tilde{\mathbf{L}} \cdot \tilde{\mathbf{L}}^T \quad (8.344)$$

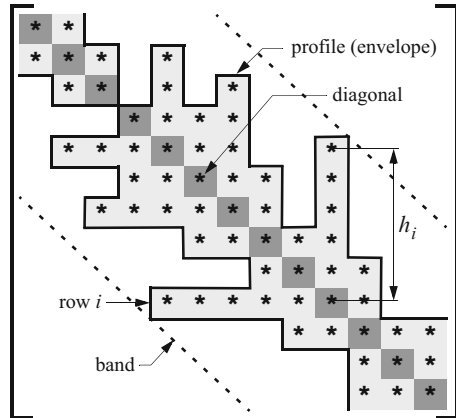
is used in the *Cholesky method*, e.g., [456,468], in which the lower triangular matrix  $\tilde{\mathbf{L}}$  is related to  $\mathbf{L}$  appearing in the Gaussian method by

$$\tilde{\mathbf{L}} = \mathbf{L} \cdot \mathbf{D}^{1/2} \quad (8.345)$$

The Cholesky method introduces a little extra-effort in computing the square root  $\mathbf{D}^{1/2}$  compared to the Crout method for symmetric matrices. However, just this square root operation accounts for small round-off errors, which is a striking feature of the Cholesky method.

Further variants of the Gaussian elimination method differ in strategies of sparse matrix storage and bookkeeping, elimination sequences, pivoting techniques and round-off error minimizations. Active column profile solvers [509,590] based on the Crout method reduce the required storage and computational effort for unsymmetric and symmetric sparse matrices, where their columns and rows are stored only within the non-zero *profile* (also termed as envelope or skyline) of  $\mathbf{A}$ , see Fig. 8.36. It has a definite advantage over the method of a fixed banded storage. The profile can be very variable so that long and small columns can be compactly stored. The column heights are, however, dependent on the node (equation) numbering used in forming  $\mathbf{A}$ . An interesting alternative Gaussian elimination is the *frontal method* [256, 287, 291], which operates in a wave-front advancing through a finite element mesh. In contrast to a profile solution strategy, the operation sequences of the frontal method are determined by the element numbering, rather than by the node numbering. The advantage is that at no time the complete sparse matrix  $\mathbf{A}$  must exist. Only parts of the matrix are assembled as they enter the front. However, it implies a considerable amount of bookkeeping compared to an active column profile

**Fig. 8.36** The profile (envelope) and band of a matrix  $\mathbf{A}$ . The profile height  $h_i$  at a matrix row  $i$  is the number of columns (respectively rows) included between the first non-zero column entry and the diagonal. The bandwidth  $\text{Bwd}(\mathbf{A})$ , (8.347), is formed by the maximum profile height occurring in  $\mathbf{A}$



solver where the processing overhead remains relatively small. The frontal method is attractive in treating large matrices out-of-core, where the storage of a complete matrix would exceed the capacity of computer memory. Today, however, this is often no more a serious constraint.

**8.17.1.4 Fill-in Reduction and Nodal Reordering**

A profile of a sparse matrix structure as exemplified in Fig. 8.36 is formed by the non-zero entries being in a largest distance from the diagonal. This is described by the profile heights  $h_i$  ( $i = 1, \dots, N_{EQ}$ ) for each row  $i$  defined as

$$h_i = i - \min(j | A_{ij} \neq 0, j \leq i) + 1, \quad (i = 1, 2, \dots, N_{EQ}) \tag{8.346}$$

Note that the *bandwidth*  $\text{Bwd}(\mathbf{A})$  of  $\mathbf{A}$  is the maximum of all profile heights occurring in the matrix:

$$\text{Bwd}(\mathbf{A}) = \max_{1 \leq i \leq N_{EQ}} h_i \tag{8.347}$$

However, it is the nature of the finite element discretization that not all matrix entries between the first non-zero column entry and the diagonal are non-zero. Quite contrarily, a large number of entries in between can be zeros. Now, the consequence of the elimination process according to formulae (8.336) is that those zero entries within the profile of  $\mathbf{A}$  become replaced by non-zero entries. Such entries are called *fill-in*. Since fill-in entries cause further fill-in, the complete matrix profile must be stored to perform the matrix elimination. The required storage amounts to the *envelope* (or total profile) given by the sum of the profile heights, viz.,

$$\text{Env}(\mathbf{A}) = \begin{cases} \sum_{i=1}^{N_{\text{EQ}}} h_i & \text{for symmetric } \mathbf{A} \\ \sum_{i=1}^{N_{\text{EQ}}} (2h_i - 1) & \text{for unsymmetric } \mathbf{A} \end{cases} \quad (8.348)$$

On the other hand, the solution effort  $\text{Ecp}(\mathbf{A})$  of matrix elimination is proportional to the square of each profile heights

$$\text{Ecp}(\mathbf{A}) \sim \sum_{i=1}^{N_{\text{EQ}}} h_i^2 \quad (8.349)$$

In order to minimize both the storage size  $\text{Env}(\mathbf{A})$  and the computational effort  $\text{Ecp}(\mathbf{A})$  for a matrix  $\mathbf{A}$  it is obvious that the profile heights  $h_i$  should be hold small as possible. Indeed, the profile heights are determined by the global nodal numbering used in a finite element mesh. In practical terms: the larger the difference between the highest and lowest node number occurring in a finite element, the larger the profile heights at the corresponding matrix index. This is evidenced in the example mesh shown in Fig. 8.37. While an inappropriate nodal numbering used in the mesh of Fig. 8.37a leads to a wide-spread pattern of non-zero entries in the matrix with a consequent large storage demand and a significant amount of fill-in, an intelligent nodal reordering as outlined for the mesh of Fig. 8.37b accomplishes a significant reduction of the storage demand, fill-in and computational effort.

There are different techniques [189, 418] which are useful to automatically renumber the mesh nodes with the aim to bring all matrix entries closer to the diagonal. Most important are:

- The Reverse Cuthill-McKee (RCM) method [108], which reorders the nodes according to the lowest connectivity with surrounding nodes at each level of the corresponding graph of spatial discretization.
- The Multilevel Nested Dissection (MLNDS) method [301, 302], in which the reduction of nodal interconnectivities is employed via a recursive partitioning of domains.

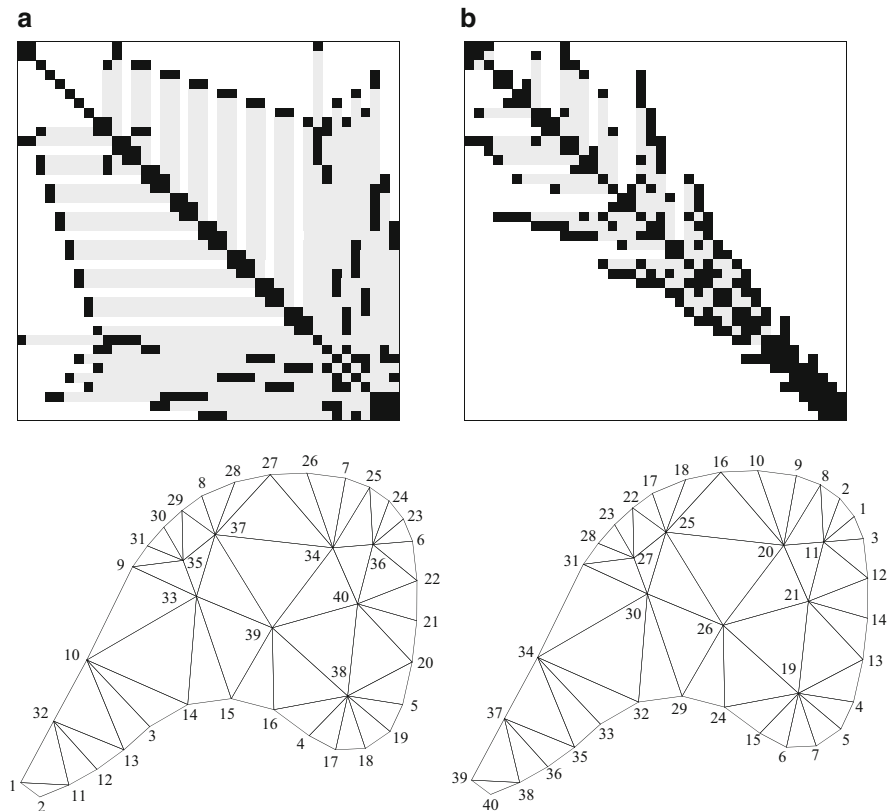
The RCM usually gives excellent reductions. The MLNDS is to be preferred for bigger meshes. It accomplishes a reasonable profile reduction (albeit often not so much as via RCM), however, at lower computational costs. Furthermore, MLNDS is better suitable for parallel processing. In practice, nodal reordering schemes are obligatory when direct equation solvers become in use. The nodal reordering is performed before Dirichlet-type BC's are implemented according to (8.321). It ends up with a compressed matrix system which is optimal for direct profile solvers.

### 8.17.2 Iterative Solution Methods

The solution of the matrix equation (8.322)

$$\mathbf{A} \cdot \phi = \mathbf{b} \quad (8.350)$$





**Fig. 8.37**  $A$ -matrix occupations for a simple 2D triangle mesh (a) before and (b) after RCM nodal reordering. Matrix entries drawn in *black* relate to intrinsic non-zero coefficients, entries drawn in *gray* identify fill-in. The nodal reordering reduces the total profile  $\text{Env}(\mathbf{A})$  for the present mesh to about one third

by using direct methods can be rather inefficient for large systems. Their computational effort  $\text{Ecp}(\mathbf{A})$  is proportional to the square sum of all matrix profile heights (8.349), for a band structure of  $\mathbf{A}$  it is proportional to  $N_{\text{EQ}}(\text{Bwd}(\mathbf{A}))^2$  and for a full matrix it is even proportional to  $N_{\text{EQ}}^3$ . However, there are reliable alternatives in form of iterative solution methods, which solve (8.350) on an efficient approximate basis possessing a computational effort having only a more or less linear proportion to the equation number  $N_{\text{EQ}}$  and, however, a dependence on an iterative cycle. The faster the convergence of the iterative procedure, the smaller the required number of iterations and the better and efficient will be the iterative solution.

The principle of all iterative solution procedures is to make a first guess  $\phi^0$ , then apply a recurrence scheme to generate a sequence of new approximations  $\phi^1, \phi^2, \dots$ , that converge to  $\phi$ . A simple recurrence scheme, known as *Richardson iteration*, could have the form  $\phi^{\tau+1} = \phi^{\tau} - \chi(\mathbf{A} \cdot \phi^{\tau} - \mathbf{b})$ , ( $\tau = 0, 1, \dots$ ), where

$\tau$  is an iteration counter and  $\chi \neq 0$  is an acceleration parameter. The advantages of such an approach are obvious: (1) the system matrix  $\mathbf{A}$  must not be inverted directly anymore and (2) the sparsity of  $\mathbf{A}$  can be fully exploited, where only the non-zero entries are stored in a dense manner (need not to consider fill-in). The disadvantage of iterative methods is that (1) the rate of convergence may be slow or even divergence may occur and (2) an error criteria has to be chosen at which the iteration is terminated to consider the approximate solution as sufficiently accurate. It becomes clear that the crucial point of each iteration method is to find an acceleration strategy for a fast rate of convergence. Today, there is a wide variety of iterative methods for solving both symmetric and unsymmetric systems, see e.g., [15, 453]. Most important for the present class of problems are the following:

- The Conjugate Gradient (CG) method.
- The Orthogonal Minimum Residual (ORTHOMIN) method.
- The Generalized Minimal Residual (GMRES) method.
- The Lanczos Conjugate Gradient Square (CGS) method.
- The Lanczos Bi-conjugate Gradient Stabilized (BiCGSTAB) method.
- The Multigrid (MG), in particular Algebraic Multigrid (AMG) method.

To improve the convergence behavior of these iterative methods, they are usually applied in combination with so-called preconditioning techniques which transform the basic matrix system into a form that is more suitable for the iterative procedure.

### 8.17.2.1 Preconditioning

An important property of the matrix  $\mathbf{A}$  is given by the *condition number*  $\kappa(\mathbf{A})$  defined as [453]

$$\kappa(\mathbf{A}) = \|\mathbf{A}\| \|\mathbf{A}^{-1}\| \quad (8.351)$$

which characterizes the ratio between the maximum and minimum eigenvalues  $\kappa(\mathbf{A}) = \lambda_{\max}(\mathbf{A})/\lambda_{\min}(\mathbf{A})$ . Problems for which  $\kappa$  is large are called *ill-conditioned problems*, otherwise if  $\kappa$  is not too large they are called *well-conditioned problems*. Typically, a high parameter contrast in the coefficients of  $\mathbf{A}$  causes a high condition number  $\kappa$ . Unfortunately, the eigenvalue distribution significantly influences the convergence behavior of an iterative method. For instance in case of the CG method, if  $S(P)$  is the number of iterative steps required to decrease the error  $\|\phi^\tau - \phi\|$  by a factor of  $P$ , then  $S(P) \leq \frac{1}{2}\sqrt{\kappa} \ln(2/P) + 1$ . That means, the number of iterations needed to reach convergence is  $\mathcal{O}(\sqrt{\kappa})$ . It suggests that the rate of convergence can be significantly improved if we could decrease  $\kappa \rightarrow 1$ . Indeed, this is possible by a suited transformation of the basic matrix system in such a way that an iterative method will converge much faster than without this modification. Such a type of transformation is termed *preconditioning*.

To construct appropriate preconditioners for matrix  $\mathbf{A}$ , we differ between explicit and implicit preconditioning methods. To solve  $\mathbf{A} \cdot \boldsymbol{\phi} = \mathbf{b}$ , an explicit method transforms the system into

$$(\mathbf{C}^{-1} \cdot \mathbf{A}) \cdot \boldsymbol{\phi} = \mathbf{C}^{-1} \cdot \mathbf{b} \quad (8.352)$$

where  $\mathbf{C}$  is the preconditioning matrix to be chosen. Then, (8.352) is solved on an iterative basis, e.g.,

$$\boldsymbol{\phi}^{\tau+1} = (\delta - \tilde{\mathbf{A}}) \cdot \boldsymbol{\phi}^{\tau} + \tilde{\mathbf{b}}, \quad \text{where } \tilde{\mathbf{A}} = \mathbf{C}^{-1} \cdot \mathbf{A} \quad \text{and} \quad \tilde{\mathbf{b}} = \mathbf{C}^{-1} \cdot \mathbf{b} \quad (8.353)$$

In implicit preconditioning the original problem  $\mathbf{A} \cdot \boldsymbol{\phi} = \mathbf{b}$  is replaced by a sequence of solutions of the form

$$\mathbf{C} \cdot (\boldsymbol{\phi}^{\tau+1} - \boldsymbol{\phi}^{\tau}) = -\mathbf{r}^{\tau} \quad (8.354)$$

with the residual vector

$$\mathbf{r}^{\tau} = \mathbf{A} \cdot \boldsymbol{\phi}^{\tau} - \mathbf{b} \quad (8.355)$$

Both forms (8.352) and (8.354) are equivalent. Their use is only dependent on how the inverse  $\mathbf{C}^{-1}$  is explicitly known. In any cases, the preconditioning matrix  $\mathbf{C}$  should require only little computational extra-effort. On the other hand, the chosen preconditioning matrix must significantly improve the eigenspectrum of  $\mathbf{C}^{-1} \cdot \mathbf{A}$  in comparison to  $\mathbf{A}$ , i.e.,  $\kappa(\mathbf{C}^{-1} \cdot \mathbf{A}) < \kappa(\mathbf{A})$ . Both requirements are somewhat contradictory: the better the preconditioning, the higher often the costs. Note that for the case  $\mathbf{C} = \mathbf{A}$  in (8.354) the scheme corresponds to a direct solution and the sequence of solutions stops after one iteration. In approximating  $\mathbf{A}$  with  $\mathbf{C}$  it tends to drop low eigenvalues (i.e., long-wavelength eigenmodes), what can be a deficiency. Hence, a good and optimal choice of  $\mathbf{C}$  (or explicitly  $\mathbf{C}^{-1}$ ) is desired. Today, a large family of preconditioners is available, cf. [15, 45, 453]. Our preferred conditioning methods are:

- The incomplete  $LU$  (ILU) decomposition method:  $\mathbf{C} = \mathbf{L} \cdot \mathbf{U}$  for unsymmetric and  $\mathbf{C} = \mathbf{L} \cdot (\mathbf{D} \cdot \mathbf{L}^T)$  for symmetric systems.
- The modified ILU (MILU) method, where diagonal entries of  $\mathbf{U}$  are additionally modified to tackle ill-conditioned problems.
- The polynomial preconditioning:  $\mathbf{C}^{-1} = p(\mathbf{A})$ , where  $p(\mathbf{A})$  is a polynomial of lower degree, commonly Chebyshev polynomials.

Most useful is the incomplete lower-upper (ILU) preconditioning in the form

$$\mathbf{C} = \begin{cases} \mathbf{L} \cdot \mathbf{U} & \text{unsymmetric} \\ \mathbf{L} \cdot (\mathbf{D} \cdot \mathbf{L}^T) & \text{symmetric} \end{cases} \quad (8.356)$$

which is achieved by a matrix decomposition with a Crout method (8.336), however, the fill-in that occurs for all the off-diagonals within the matrix profile is completely or partly neglected. The simplest and usually preferred method is the ILU decomposition (factorization) with *no fill-in*, termed by ILU(0). It works very fast, is robust and needs only a comparatively small extra-storage. For certain applications an extended ILU preconditioning can be suitable in which some fill-in is allowed in the incomplete *LU* decomposition, e.g., ILU(1) which accomplishes 1st-order fill-ins, see [453] for more. Commonly, nodal reordering (see Sect. 8.17.1.4) is not needed for ILU(0), but in using ILU(1) it can improve the accuracy of the preconditioner due to the fill-in-minimized rearranged structure of the matrix.

Improvements of ILU preconditioning can be attained by so-called modified ILU (MILU) preconditioners [40]. Basically, these techniques are zero fill-in ILU(0) strategies, however, the fill-in entries occurring during the matrix decomposition process are kept (e.g., adding up positive off-diagonal entries) in order to put the lumped sum to the diagonal entries. In this way one attempts to compensate the discarded entries, which can be important for the lower eigenvalues if the matrix is ill-conditioned. A favorite is the Gustaffson MILU preconditioning [217], which is designed and specialized for symmetric matrices including a high coefficient contrast. However, the MILU strategy has shown often insufficiently robust and should not be applied in general.

### 8.17.2.2 The Preconditioned Conjugate Gradient (PCG) Method

The conjugate gradient (CG) method goes back to Hestenes and Stiefel [247], who presented a new iterative method with a significantly increased rate of convergence in solving sparse *symmetric* positive-definite equation systems. Bizarrely, it took many years until this powerful method has found acceptance in the numerical analysis community that were exclusively fixed on direct solution methods over long time. But, beginning in the 1970s and in particular once computers became powerful enough to tackle real 3D problems, the CG method has started its triumph and gained considerable attraction in numerical modeling. Today, the CG method has become the standard iterative method for sparse symmetric equation systems. Its major advantages are: (1) the number of operations per iterative step  $\tau$  is only proportional to the number of equations  $N_{EQ}$  and (2) it converges in at most  $N_{EQ}$  iterations in the absence of round-off errors. In practice, however, the method already converges after a relatively small number of iterations much faster than the pessimistic estimate of  $N_{EQ}$ . The rate of convergence depends on the distribution of eigenvalues. Accordingly, the use of appropriate preconditioning further increases the rate of convergence of the CG method. The *preconditioned CG (PCG) method* is the preferred iterative solution method for symmetric matrix systems.

The iterative algorithm for solving  $\mathbf{A} \cdot \boldsymbol{\phi} = \mathbf{b}$  by the PCG method is given as follows (see e.g., [15, 453]):

Let  $\mathbf{A}$  and  $\mathbf{C}$  be symmetric and positive definite. Guess initially  $\phi^0$  and set:  $\mathbf{r}^0 = \mathbf{A} \cdot \phi^0 - \mathbf{b}$ ,  $\mathbf{h}^0 = \mathbf{C}^{-1} \cdot \mathbf{r}^0$  and  $\mathbf{d}^0 = -\mathbf{h}^0$ , with known values:  $\epsilon$  and  $ITMAX$

For iterations  $\tau = 0, 1, 2, \dots$  compute until convergence:

$$\phi^{\tau+1} = \phi^\tau + \alpha^\tau \mathbf{d}^\tau \quad \text{where} \quad \alpha^\tau = \frac{\mathbf{h}^{\tau T} \cdot \mathbf{r}^\tau}{\mathbf{d}^{\tau T} \cdot (\mathbf{A} \cdot \mathbf{d}^\tau)} \quad (8.357)$$

$$\mathbf{r}^{\tau+1} = \mathbf{r}^\tau + \alpha^\tau (\mathbf{A} \cdot \mathbf{d}^\tau)$$

$$\mathbf{h}^{\tau+1} = \mathbf{C}^{-1} \cdot \mathbf{r}^{\tau+1}$$

$$\mathbf{d}^{\tau+1} = -\mathbf{h}^{\tau+1} + \beta^\tau \mathbf{d}^\tau \quad \text{where} \quad \beta^\tau = \frac{\mathbf{h}^{\tau+1 T} \cdot \mathbf{r}^{\tau+1}}{\mathbf{h}^{\tau T} \cdot \mathbf{r}^\tau}$$

Stop if  $\frac{\mathbf{r}^{\tau+1 T} \cdot \mathbf{r}^{\tau+1}}{\mathbf{b}^T \cdot \mathbf{b}} < \epsilon^2$  or  $\tau > ITMAX$

where  $\mathbf{r}$  is the residual vector,  $\mathbf{h}$  is the pseudoresidual vector,  $\mathbf{d}$  is the search direction vector,  $\mathbf{C}$  is the preconditioning matrix, commonly  $\mathbf{C} = \mathbf{L} \cdot (\mathbf{D} \cdot \mathbf{L}^T)$  by using an ILU(0) preconditioner,  $\epsilon$  is the termination criterion (default  $10^{-8}$ ) and  $ITMAX$  is the allowed maximum number of iterations (e.g., 200) to be chosen in dependence on  $N_{EQ}$ .

### 8.17.2.3 The Preconditioned Restarted ORTHOMIN Method

The orthogonal minimum residual (ORTHOMIN) method belongs to a family of generalized conjugate gradient methods. It was firstly presented by Vinsome [547] and widely used in petroleum reservoir simulation. A biorthogonal vector algorithm, originally attributed to Lanczos [332], forms the basis for solving sparse unsymmetric matrix systems. Most of the following variants of iterative methods applied to unsymmetric matrices are based on that biorthogonalization procedure, termed as *Lanczos algorithm*.

In iteratively solving  $\mathbf{A} \cdot \phi = \mathbf{b}$  by the ORTHOMIN method, the orthogonality of  $\mathbf{A} \cdot \mathbf{q}^\tau$  is required, that is  $(\mathbf{A} \cdot \mathbf{q}^\tau) \cdot (\mathbf{A} \cdot \mathbf{q}^k) = 0$  for  $\tau \neq k$ , where  $\mathbf{q}^\tau$  and  $\mathbf{q}^k$  are the search directions at iterative steps  $\tau$  and  $k$ , respectively. The ORTHOMIN method converges to the exact solution  $\phi$  within  $N_{EQ}$  iterations, however, it is necessary to store up  $N_{EQ}$  search directions  $\mathbf{q}^\tau$  and  $N_{EQ}$  products  $\mathbf{A} \cdot \mathbf{q}^\tau$ . This implies large summation which makes the procedure sensitive for accumulating round-off errors in the computation of the search directions. To overcome this deficiency the orthogonalization is restarted every  $K$  iterations, i.e., the ORTHOMIN procedure runs for  $K$  steps to get an approximation  $\phi^K$ , then setting  $\phi^0 = \phi^K$  and restart the iterations until convergence is reached. This is referred to as *restarted* ORTHOMIN or ORTHOMIN( $K$ ), e.g., [342]. Since the computational cost increases as  $\mathcal{O}(K^2 N_{EQ})$  and the memory cost increases as  $\mathcal{O}(K N_{EQ})$ , it is required to hold  $K$  small relative to  $N_{EQ}$ . Usually,  $K$  is chosen in the range between 4 and 10, depending inversely on the condition number of the matrix. In practice, it has been shown that the restarted ORTHOMIN converges in a similar number of iterations as the non-restarted version [42].

The iterative algorithm for solving  $\mathbf{A} \cdot \phi = \mathbf{b}$  by the preconditioned restarted ORTHOMIN( $K$ ) method is given as follows (e.g., [346, 547]):

Let  $\mathbf{A}$  and  $\mathbf{C}$  be unsymmetric. Guess initially  $\phi^0$  and set:  $\mathbf{r}^0 = \mathbf{b} - \mathbf{A} \cdot \phi^0$ ,  $i = 0$ , with the known values:  $\epsilon$ ,  $ITMAX$  and  $K$

(1) For iterations  $\tau = 1, 2, \dots, K$  do:

$$\begin{aligned} i &= i + 1 \\ \mathbf{u}^\tau &= \mathbf{C}^{-1} \cdot \mathbf{r}^{\tau-1} \\ \mathbf{v}^\tau &= \mathbf{A} \cdot \mathbf{u}^\tau \\ \mathbf{p}^\tau &= \mathbf{u}^\tau \\ \mathbf{q}^\tau &= \mathbf{v}^\tau \end{aligned}$$

For  $1 \leq k \leq \tau$ , do  $\begin{cases} \alpha^{k\tau} = (\mathbf{q}^{k\tau} \cdot \mathbf{v}^\tau) / (\mathbf{q}^{k\tau} \cdot \mathbf{q}^k) \\ \mathbf{p}^\tau = \mathbf{p}^\tau - \alpha^{k\tau} \mathbf{p}^k \\ \mathbf{q}^\tau = \mathbf{q}^\tau - \alpha^{k\tau} \mathbf{q}^k \end{cases}$

$$\begin{aligned} \beta^\tau &= (\mathbf{q}^{\tau T} \cdot \mathbf{r}^{\tau-1}) / (\mathbf{q}^{\tau T} \cdot \mathbf{q}^\tau) \\ \phi^\tau &= \phi^{\tau-1} + \beta^\tau \mathbf{p}^\tau \\ \mathbf{r}^\tau &= \mathbf{r}^{\tau-1} - \beta^\tau \mathbf{q}^\tau \end{aligned}$$

Compute  $e_1^\tau = \frac{\|\mathbf{r}^\tau\|_{L_\infty}}{\|\mathbf{b}\|_{L_\infty}}$ ,  $e_2^\tau = \frac{\beta^\tau \|\mathbf{p}^\tau\|_{L_\infty}}{\|\phi^\tau\|_{L_\infty}}$

Stop if  $e_1^\tau < \epsilon$  or  $e_2^\tau < \epsilon$  or  $i \geq ITMAX$

(2) End do

$$\begin{aligned} \mathbf{r}^0 &= \mathbf{r}^K \\ \phi^0 &= \phi^K \end{aligned}$$

Go to (1) for restarting

(8.358)

where  $\mathbf{r}$  is the residual vector,  $\mathbf{q}$  is the search direction vector,  $\mathbf{u}$ ,  $\mathbf{v}$ ,  $\mathbf{p}$  are auxiliary vectors,  $\mathbf{C}$  is the preconditioning matrix, preferentially  $\mathbf{C} = \mathbf{L} \cdot \mathbf{U}$  by using an ILU(0) Crout preconditioner,  $\epsilon$  is the termination criterion (default  $10^{-6}$ ),  $\|\cdot\|_{L_\infty}$  is the maximum norm defined by (8.26),  $K$  is the number of iterations (default 5) after which the algorithm is periodically restarted and  $ITMAX$  is the tolerated maximum of total iterations (e.g., 200) to be chosen in dependence on  $N_{EQ}$ .

The ORTHOMIN( $K$ ) method is only guaranteed to converge for *positive real* matrices  $\mathbf{A}$ , that is if  $\phi \cdot (\mathbf{A} \cdot \phi) > 0$  for all  $\phi \neq \mathbf{0}$ . In other cases, there is no guarantee anymore for convergence, unless an appropriate preconditioning matrix  $\mathbf{C}$  can be found which creates a positive real matrix  $\mathbf{C}^{-1} \cdot \mathbf{A}$ .

### 8.17.2.4 The Preconditioned Restarted GMRES Method

The generalized minimal residual (GMRES) method was introduced by Saad and Schultz [454] for solving unsymmetric matrix systems. It is mathematically equivalent to a generalized conjugate gradient method, however, the orthogonal

vector algorithm is based on the Arnoldi method (see e.g., [15, 453]), which saves computational effort and improves robustness in particular for large equation systems. Li et al. [346] have shown that GMRES can be one-third faster than ORTHOMIN in various large-scale petroleum reservoir applications. GMRES is guaranteed to converge in at most  $N_{\text{EQ}}$  steps, provided that  $\mathbf{A}$  or  $\mathbf{C}^{-1} \cdot \mathbf{A}$  is positive real. However, similar to the ORTHOMIN method the storage demand of GMRES increases linearly with the iteration  $\tau$  and the number of operations increases as  $\mathcal{O}(\tau^2 N_{\text{EQ}})$ , which is rather computationally impractical for large matrices. The alternative is that GMRES is to restart after a fixed number of iterations  $K$ , similar to a restarted ORTHOMIN( $K$ ) procedure. In GMRES( $K$ ), the GMRES method is periodically restarted after every  $K$  iterations until reaching convergence.

The iterative algorithm for solving  $\mathbf{A} \cdot \phi = \mathbf{b}$  by the preconditioned restarted GMRES( $K$ ) method is given as follows (e.g., [342, 346, 453, 454]):

*Let  $\mathbf{A}$  and  $\mathbf{C}$  be unsymmetric. Choose a first guess  $\phi^0$ . Set up the  $(K + 1) \times K$  Hessenberg matrix:  $\mathbf{H}^K = (\mathbf{H}^{k\tau})_{1 \leq k \leq K+1, 1 \leq \tau \leq K} = \mathbf{0}$ ,  $i = 0$ , with the known values:  $\epsilon_1$ ,  $\epsilon_2$ ,  $ITMAX$  and  $K$*

*(1) Arnoldi process:*

*Compute  $\mathbf{r}^0 = \mathbf{C}^{-1} \cdot (\mathbf{b} - \mathbf{A} \cdot \phi^0)$ ,  $\beta = \|\mathbf{r}^0\|_{L_2}$  and  $\mathbf{v}^1 = \mathbf{r}^0/\beta$*

*For iterations  $\tau = 1, 2, \dots, K$  do:*

$$i = i + 1$$

$$\mathbf{w}^\tau = \mathbf{C}^{-1} \cdot (\mathbf{A} \cdot \mathbf{v}^\tau)$$

*For  $1 \leq k \leq \tau$ , do if  $(\mathbf{w}^{\tau T} \cdot \mathbf{v}^k) > \epsilon_1 \|\mathbf{w}^\tau\|_{L_2}$ :*

$$\begin{cases} \mathbf{H}^{k\tau} = \mathbf{w}^{\tau T} \cdot \mathbf{v}^k \\ \mathbf{w}^\tau = \mathbf{w}^\tau - \mathbf{H}^{k\tau} \mathbf{v}^k \end{cases}$$

$$\mathbf{H}^{\tau+1,\tau} = \|\mathbf{w}^\tau\|_{L_2}$$

$$\mathbf{v}^{\tau+1} = \mathbf{w}^\tau / \|\mathbf{w}^\tau\|_{L_2}$$

*End do*

*Define  $\mathbf{V}^K = (\mathbf{v}^1, \dots, \mathbf{v}^K)^T$  and update  $\phi^K = \phi^0 + \mathbf{V}^K \mathbf{y}^K$ ,*

*where  $\mathbf{y}^K$  is the minimizer of  $\|\beta \mathbf{e}_1 - \mathbf{H}^K \mathbf{y}\|_{L_2}$  with  $\mathbf{e}_1 = (1, 0, 0, \dots, 0)^T$*

*Stop if  $\frac{\|\mathbf{r}^K\|_{L_2}}{\|\mathbf{b}\|_{L_2}} \leq \epsilon_2$  or  $i \geq ITMAX$ , where  $\mathbf{r}^K = \mathbf{C}^{-1} \cdot (\mathbf{b} - \mathbf{A} \cdot \phi^K)$ ,*

*otherwise set  $\phi^0 = \phi^K$  and go to (1) for restarting*

(8.359)

where  $\mathbf{r}$  is the residual vector,  $\mathbf{H}$  is the Hessenberg matrix [453],  $\mathbf{v}$  and  $\mathbf{w}$  are auxiliary vectors,  $\mathbf{C}$  is the preconditioning matrix, preferentially  $\mathbf{C} = \mathbf{L} \cdot \mathbf{U}$  by using an ILU(0) Crout preconditioner,  $\epsilon_1$  is the criterion for checking orthogonality (default  $10^{-10}$ ),  $\epsilon_2$  is the convergence criterion to terminate the iteration (default  $10^{-6}$ ),  $\|\cdot\|_{L_2}$  is the  $L_2$  error norm defined by (8.25),  $K$  is the number of iterations (default 5) after which the algorithm is periodically restarted and  $ITMAX$  is the tolerated maximum of total iterations (e.g., 200) to be chosen in dependence on  $N_{\text{EQ}}$ .

### 8.17.2.5 The Preconditioned Lanczos Conjugate Gradient Square (CGS) Method

The CGS method is a variant of the Lanczos-type biorthogonalization (biconjugate gradient) method for solving unsymmetric matrix systems in which biorthogonal sets of vectors are generated. It has been proposed by Sonneveld [487]. CGS is a highly efficient iterative method for unsymmetric matrices which converges about twice as fast than standard biconjugate gradient methods. It is based on squaring the residual polynomials. While CGS works well in many applications, it is prone to rounding errors due to the squared polynomials and even breakdowns cannot be fully precluded in cases causing a divide by zero. On the other hand, the residual error is not strictly decreasing in the progress of iterations. Nevertheless, its properties regarding the smallest storage demand and accelerated convergence in the most applications make the CGS method in combination with an appropriate preconditioning a powerful recurrence scheme for solving unsymmetric equation systems.

The iterative algorithm for solving  $\mathbf{A} \cdot \boldsymbol{\phi} = \mathbf{b}$  by the preconditioned CGS method can be written in the following form (e.g., [453, 487]):

*Let  $\mathbf{A}$  and  $\mathbf{C}$  be unsymmetric. Guess initially  $\boldsymbol{\phi}^0$  and set:  $\mathbf{r}^0 = \mathbf{q}^0 = \mathbf{C}^{-1} \cdot (\mathbf{b} - \mathbf{A} \cdot \boldsymbol{\phi}^0)$ ,  $\mathbf{g}^0 = \mathbf{h}^0 = \mathbf{0}$ ,  $\beta^0 = \mathbf{q}^{0T} \cdot \mathbf{r}^0$  and  $\gamma^0 = 0$  with known values:  $\epsilon$  and  $ITMAX$*

*For iterations  $\tau = 0, 1, 2, \dots$  compute until convergence:*

$$\begin{aligned} \mathbf{y}^{\tau+1} &= \mathbf{r}^{\tau} + \gamma^{\tau} \mathbf{h}^{\tau} \\ \mathbf{g}^{\tau+1} &= \mathbf{y}^{\tau+1} + \gamma^{\tau} (\mathbf{h}^{\tau} + \gamma^{\tau} \mathbf{g}^{\tau}) \\ \mathbf{h}^{\tau+1} &= \mathbf{y}^{\tau+1} - \alpha^{\tau+1} \mathbf{C}^{-1} \cdot (\mathbf{A} \cdot \mathbf{g}^{\tau+1}) \quad \text{where } \alpha^{\tau+1} = \frac{\beta^{\tau}}{\mathbf{q}^{0T} \cdot [\mathbf{C}^{-1} \cdot (\mathbf{A} \cdot \mathbf{g}^{\tau+1})]} \\ \boldsymbol{\phi}^{\tau+1} &= \boldsymbol{\phi}^{\tau} + \alpha^{\tau+1} (\mathbf{y}^{\tau+1} + \mathbf{h}^{\tau+1}) \\ \mathbf{r}^{\tau+1} &= \mathbf{r}^{\tau} - \alpha^{\tau+1} \mathbf{C}^{-1} \cdot [\mathbf{A} \cdot (\mathbf{y}^{\tau+1} + \mathbf{h}^{\tau+1})] \\ \beta^{\tau+1} &= \mathbf{q}^{0T} \cdot \mathbf{r}^{\tau+1} \\ \gamma^{\tau+1} &= \beta^{\tau+1} / \beta^{\tau} \\ \text{Stop if } \frac{\|\mathbf{r}^{\tau+1}\|_{L_2}}{\|\mathbf{r}^0\|_{L_2}} &< \epsilon \quad \text{or } \tau > ITMAX \end{aligned}$$

(8.360)

where  $\mathbf{r}$  is the residual vector,  $\mathbf{q}$  is the shadow residual vector,  $\mathbf{y}$ ,  $\mathbf{h}$ ,  $\mathbf{g}$  are auxiliary vectors,  $\mathbf{C}$  is the preconditioning matrix, preferentially  $\mathbf{C} = \mathbf{L} \cdot \mathbf{U}$  by using an ILU(0) Crout preconditioner,  $\epsilon$  is the termination criterion (default  $10^{-8}$ ) and  $ITMAX$  is the allowed maximum number of iterations (e.g., 200) to be chosen in dependence on  $N_{EQ}$ . The preconditioned CGS method converges in at most  $N_{EQ}$  iterations for positive real matrices.



### 8.17.2.6 The Preconditioned Lanczos Bi-conjugate Gradient Stabilized (BiCGSTAB) Method

Van der Vorst [535] proposed the BiCGSTAB algorithm as an improved variant of CGS. The BiCGSTAB method stabilizes and smoothes the convergence behavior. It leads to a more robust and usually faster converging iterative technique for solving unsymmetric equations systems with small storage demand and low computational cost. The BiCGSTAB iteration steps are only slightly more expensive than the CGS steps. But, similar to CGS the BiCGSTAB method cannot fully exclude the risk of a computational breakdown if the system matrix  $\mathbf{A}$  is not positive real. In most applications, however, BiCGSTAB has shown a superior behavior by what it has become a preferred iterative method for solving unsymmetric matrix systems.

The iterative algorithm for solving  $\mathbf{A} \cdot \phi = \mathbf{b}$  by the preconditioned BiCGSTAB method is given as follows (e.g., [453, 535]):

*Let  $\mathbf{A}$  and  $\mathbf{C}$  be unsymmetric. Guess initially  $\phi^0$   
and set:  $\mathbf{r}^0 = \mathbf{b} - \mathbf{A} \cdot \phi^0$  and  $\mathbf{p}^0 = \mathbf{q}^0 = \mathbf{C}^{-1} \cdot \mathbf{r}^0$   
with known values:  $\epsilon$  and  $ITMAX$*

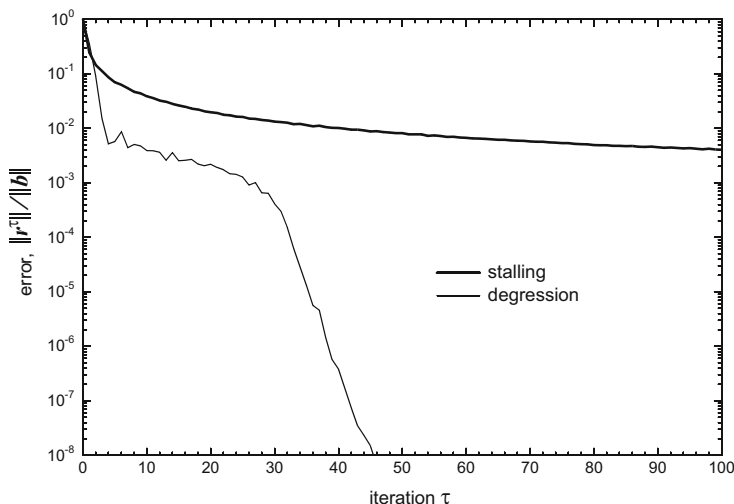
*For iterations  $\tau = 0, 1, 2, \dots$  compute until convergence:*

$$\begin{aligned} \mathbf{s}^\tau &= \mathbf{r}^\tau - \alpha^\tau \mathbf{A} \cdot \mathbf{p}^\tau & \text{where } \alpha^\tau &= \frac{\mathbf{q}^{0T} \cdot (\mathbf{C}^{-1} \cdot \mathbf{r}^\tau)}{\mathbf{q}^{0T} \cdot [\mathbf{C}^{-1} \cdot (\mathbf{A} \cdot \mathbf{p}^\tau)]} \\ \mathbf{t}^\tau &= \mathbf{C}^{-1} \cdot \mathbf{s}^\tau \\ \phi^{\tau+1} &= \phi^\tau + \alpha^\tau \mathbf{p}^\tau + \omega^\tau \mathbf{t}^\tau & \text{where } \omega^\tau &= \frac{(\mathbf{A} \cdot \mathbf{t}^\tau)^T \cdot \mathbf{s}^\tau}{(\mathbf{A} \cdot \mathbf{t}^\tau)^T \cdot (\mathbf{A} \cdot \mathbf{t}^\tau)} \\ \mathbf{r}^{\tau+1} &= \mathbf{s}^\tau - \omega^\tau \mathbf{A} \cdot \mathbf{t}^\tau \\ \beta^\tau &= \frac{\alpha^\tau}{\omega^\tau} \frac{\mathbf{q}^{0T} \cdot (\mathbf{C}^{-1} \cdot \mathbf{r}^{\tau+1})}{\mathbf{q}^{0T} \cdot (\mathbf{C}^{-1} \cdot \mathbf{r}^\tau)} \\ \mathbf{p}^{\tau+1} &= \mathbf{C}^{-1} \cdot \mathbf{r}^{\tau+1} + \beta^\tau [\mathbf{p}^\tau - \omega^\tau \mathbf{C}^{-1} \cdot (\mathbf{A} \cdot \mathbf{p}^\tau)] \\ \text{Stop if } & \frac{\|\mathbf{r}^{\tau+1}\|_{L_2}}{\|\mathbf{r}^0\|_{L_2}} < \epsilon \text{ or } \tau > ITMAX \end{aligned} \tag{8.361}$$

where  $\mathbf{r}$  is the residual vector,  $\mathbf{q}$  is the shadow residual vector,  $\mathbf{s}$ ,  $\mathbf{t}$ ,  $\mathbf{p}$  are auxiliary vectors,  $\mathbf{C}$  is the preconditioning matrix, preferentially  $\mathbf{C} = \mathbf{L} \cdot \mathbf{U}$  by using an ILU(0) Crout preconditioner,  $\epsilon$  is the termination criterion (default  $10^{-8}$ ) and  $ITMAX$  is the allowed maximum number of iterations (e.g., 200) to be chosen in dependence on  $N_{EQ}$ . The preconditioned BiCGSTAB method converges in at most  $N_{EQ}$  iterations for positive real matrices.

### 8.17.2.7 Multigrid (MG) Methods

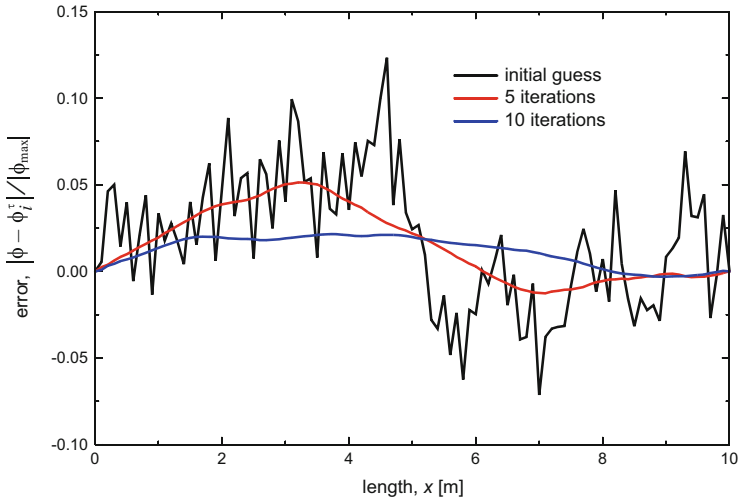
The iterative solution methods treated so far suffer from disabling limitations: (1) Their convergence rates are dependent on the number of equations  $N_{EQ}$  to be solved. This has severe implications in the numerical solution of very large problems involving millions and billions of mesh nodes. The required number of iterations inevitably increases and can reach an unacceptable size. (2) Their convergence rate



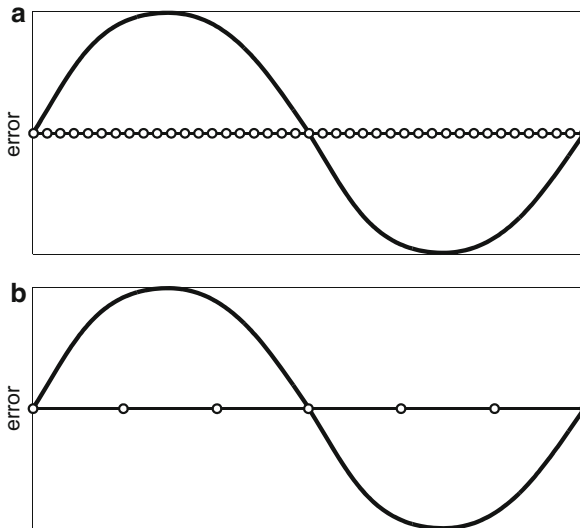
**Fig. 8.38** Convergence behavior of PCG method showing a typical well-behaved degression of residual error and the occurrence of stalling in a large problem solution

has a tendency to stall, in particular for large problems, that means the reduction rate of errors becomes slow or even practically stagnant. In fact, these iterative methods converge very rapidly for the few iterations and very slowly thereafter, see Fig. 8.38. The reason for that unfavorable behavior is obvious: The convergence rate is a function of the error field frequency, i.e., the measure of change of the error from node to node. All high error frequencies or small wavelength components which are comparable to the mesh size can be effectively reduced (smoothed out), see Fig. 8.39, however, low error frequencies or large wavelength components of error can only be badly annihilated such that the convergence rate automatically deteriorates. As the mesh is refined, the low error frequencies dominate the solution error and additional iterations become progressively less productive. Indeed, this represents a serious limitation of those iterative methods. But, the remedy is possible by using multigrid (MG) methods.

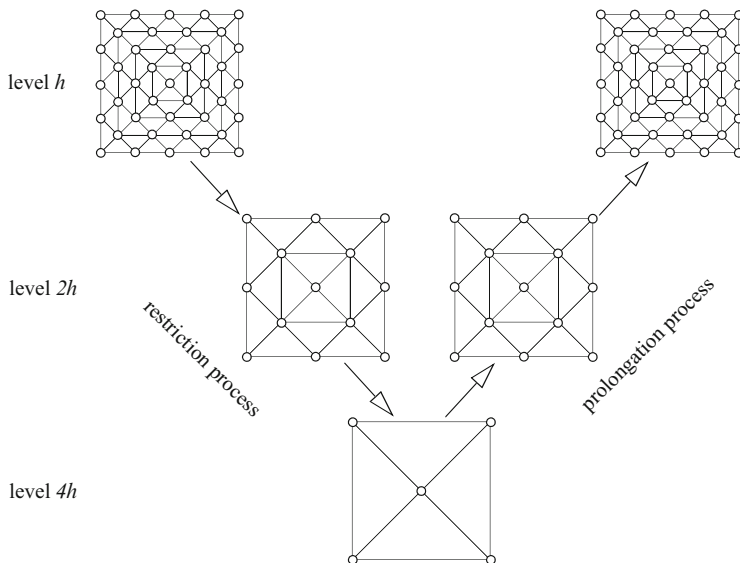
The basic idea of MG methods is likewise simple and intuitive: Since low frequency errors remain widely hidden for fine grids (meshes), it should be more efficient to reduce those errors on coarser grids (Fig. 8.40). In using both fine and coarse grids in an appropriate interplay it must per se lead to a highly powerful iterative strategy, where both high and low error frequencies are reduced at the same time with fast convergence. The natural way of transferring between fine and coarser grids firstly results in the traditional MG method, called geometric multigrid (GMG).



**Fig. 8.39** Error  $|\phi - \phi_i^\tau|/|\phi_{\max}|$  ( $i = 1, \dots, N_p$ ) reduction (smoothing) of PCG method in the course of iterations  $\tau$  applied to a 1D diffusion problem



**Fig. 8.40** (a) On a fine mesh, the error is visible as low frequency and large wavelength, (b) on a coarse mesh, the error is seen as high(er) frequency and small(er) wavelength. Smooth modes on a fine grid look less smooth on a coarse grid



**Fig. 8.41** Example of a three-level multigrid hierarchy consisting of uniform triangle meshes

### Geometric Multigrid (GMG) Method

The concept of the multigrid algorithm dates back to the 1960s when Fedorenko [161] and Bakhvalov [21] published their first studies. Other multigrid pioneers are Brandt [51] and Hackbusch [219], who have recognized in the 1970s the actual efficiency of the MG method and started fundamental developments. Today, for MG methods an extensive mathematical basis exists and a variety of efficient numerical strategies for many applications have been worked out. A good overview is given in the textbook by Trottenberg et al. [519]. The classic and standard MG approach refers to the geometric MG (GMG) method.

The usual practice in GMG consists of a successive *nested* structured grid (mesh) procedure in which the coarse grid has twice the grid spacing  $2h$  of the next finer grid with the grid size  $h$  so that all nodes in the coarse grid also appear in the fine grid. The use of grid spacings with a ratio of 2 allows very efficient intergrid transfer operations and lead to *hierarchical* meshes where typically fine mesh elements result from coarse mesh elements by a simple subdivision via element halving [219]. A hierarchical mesh can also be locally refined. An example for a three-level successive nested multigrid hierarchy consisting of uniform triangle meshes is exhibited in Fig. 8.41 which features a V-cycle.

To illustrate the GMG procedure for solving the finite element matrix system  $\mathbf{A} \cdot \boldsymbol{\phi} = \mathbf{b}$ , the sequence of only two mesh levels identified by subscripts  $h$  for the fine grid and  $2h$  for the next coarse grid are considered at first. It begins with solving  $\mathbf{A}_h \cdot \boldsymbol{\phi}_h = \mathbf{b}_h$  on the fine mesh for a small number of iterations  $< 10$  by using an appropriate iterative method (e.g., PCG or others) until the residual

$$\mathbf{r}_h^\tau = \mathbf{b}_h - \mathbf{A}_h \cdot \phi_h^\tau \quad (8.362)$$

is sufficiently smooth, i.e., high error frequencies are suitably reduced. This solution step is termed *presmoothing*. Then, to effectively reduce low error frequencies the so-called *coarse grid correction* starts. It transfers the residual  $\mathbf{r}_h^\tau$  to the coarse grid, a process called *restriction*, in the following form of a coarse grid defect matrix equation

$$\mathbf{A}_{2h} \cdot \Delta \phi_{2h}^\tau = \mathbf{b}_{2h}^\tau \quad \text{with} \quad \mathbf{b}_{2h}^\tau = \mathbf{I}_h^{2h} \cdot \mathbf{r}_h^\tau \quad \text{and} \quad \Delta \phi_{2h}^\tau = \phi_{2h}^{\tau+1} - \phi_{2h}^\tau \quad (8.363)$$

where  $\mathbf{I}_h^{2h}$  is a nonsquare matrix, known as the *restriction operator*. The solution increment  $\Delta \phi_{2h}^\tau$  results from (8.363) and can now be transferred back to the fine grid by interpolation, a process called *prolongation*, to obtain

$$\Delta \phi_h^\tau = \mathbf{I}_{2h}^h \cdot \Delta \phi_{2h}^\tau \quad (8.364)$$

where  $\mathbf{I}_{2h}^h$  is a nonsquare matrix, known as the *prolongation operator*. It gives the new approximation

$$\phi_h^{\tau+1} = \phi_h^\tau + \Delta \phi_h^\tau \quad (8.365)$$

on the fine mesh. A *postsmoothing* solution step can now follow in which the residual  $\mathbf{r}_h^{\tau+1} = \mathbf{b}_h - \mathbf{A}_h \cdot \phi_h^{\tau+1}$  is further reduced via a standard iterative solver to obtain an improved solution  $\phi_h^{\tau+1}$  on the fine mesh.

The construction of the restriction operator  $\mathbf{I}_h^{2h}$  and the prolongation operator  $\mathbf{I}_{2h}^h$  can be rather simple when using nested grids in which all coarse grid nodes appear in the fine grid nodes. In the FEM context the natural choice is the use of interpolations based on the basis functions (8.16) such that

$$\sum_j N_{2h,j} \phi_{2h,j} \approx \sum_l N_{h,l} \phi_{h,l} \quad (8.366)$$

where  $N_{2h,j}$  and  $N_{2h,l}$  denote the basis functions of meshes  $2h$  with nodes  $j$  and  $h$  with nodes  $l$ , respectively. To minimize the approximation error a Galerkin-weighting approach for (8.366) becomes useful

$$\int_\Omega N_{2h,i} N_{2h,j} d\Omega \phi_{2h,j} = \int_\Omega N_{2h,i} N_{h,l} d\Omega \phi_{h,l} \quad (8.367)$$

$$\mathbf{O}_{2h} \cdot \phi_{2h} = \mathbf{M}_h^{2h} \cdot \phi_h$$

where  $\mathbf{O}_{2h}$  is a consistent mass matrix for the coarse mesh, which can also be lumped (see Sect. 8.13.2), and  $\mathbf{M}_h^{2h}$  forms a new integral that consists of the inner product of basis functions from the different meshes. The restriction operator directly follows from (8.367) as

$$\mathbf{I}_h^{2h} = \mathbf{O}_{2h}^{-1} \cdot \mathbf{M}_h^{2h} \quad (8.368)$$

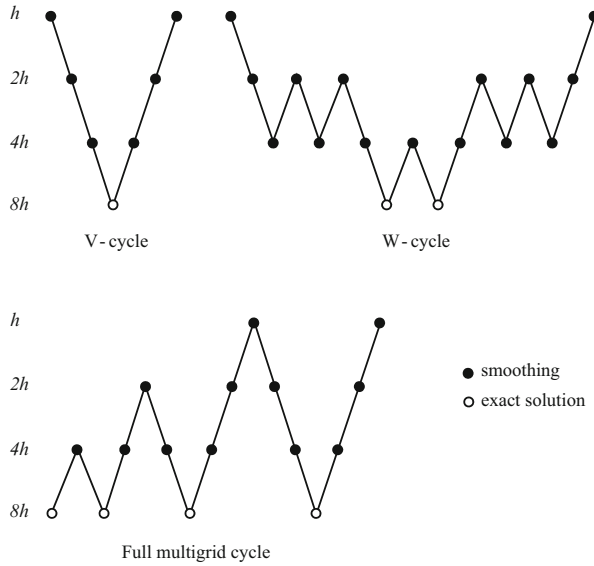
which is simply evaluable for the lumped matrix  $\mathbf{O}_{2h}^{-1} = \delta$ . A similar relation can be developed for the prolongation operator. We recognize that (8.368) implies operations in form of averages between adjacent grid points. More difficulties arise in using unnested and unstructured grids, see e.g., [354], where the operators have to be carefully derived to avoid additional unacceptable approximation errors appearing in the intergrid interpolations.

The principle of the two-grid procedure as stated above can be generalized to sequences of multiple grids. The reason for using more sequences of grids is obvious: The solution of (8.363) on the coarse grid may not be much different from the next fine grid. Hence, we can recursively repeat this two-grid procedure on successively coarser grids, creating coarser and coarser grids, down to some coarsest grid. Then, on the coarsest grid the remaining defect equation of the type (8.363) can be usually solved exactly via a direct solver. The solution is then prolonged successively to the finer grids. The multigrid algorithm takes the form:

$$\begin{aligned}
 &\text{smooth } \mathbf{r}_h^\tau = \mathbf{b}_h - \mathbf{A}_h \cdot \phi_h^\tau \quad n \text{ times} \\
 &\text{restrict } \mathbf{b}_{2h}^\tau = \mathbf{I}_h^{2h} \cdot \mathbf{r}_h^\tau \\
 &\quad \text{smooth } \mathbf{r}_{2h}^\tau = \mathbf{b}_{2h}^\tau - \mathbf{A}_{2h} \cdot \Delta \phi_{2h}^\tau \quad n \text{ times} \\
 &\quad \text{restrict } \mathbf{b}_{4h}^\tau = \mathbf{I}_{2h}^{4h} \cdot \mathbf{r}_{2h}^\tau \\
 &\quad \quad \text{smooth } \mathbf{r}_{4h}^\tau = \mathbf{b}_{4h}^\tau - \mathbf{A}_{4h} \cdot \Delta \phi_{4h}^\tau \quad n \text{ times} \\
 &\quad \quad \text{restrict } \mathbf{b}_{8h}^\tau = \mathbf{I}_{4h}^{8h} \cdot \mathbf{r}_{4h}^\tau \\
 &\quad \quad \quad \vdots \\
 &\quad \quad \quad \text{prolongate } \Delta \phi_{4h}^\tau = \mathbf{I}_{8h}^{4h} \cdot \Delta \phi_{8h}^\tau \\
 &\quad \quad \quad \text{smooth } \mathbf{r}_{4h}^\tau = \mathbf{b}_{4h}^\tau - \mathbf{A}_{4h} \cdot \Delta \phi_{4h}^\tau \quad m \text{ times} \\
 &\quad \quad \text{prolongate } \Delta \phi_{2h}^\tau = \mathbf{I}_{4h}^{2h} \cdot \Delta \phi_{4h}^\tau \\
 &\quad \quad \text{smooth } \mathbf{r}_{2h}^\tau = \mathbf{b}_{2h}^\tau - \mathbf{A}_{2h} \cdot \Delta \phi_{2h}^\tau \quad m \text{ times} \\
 &\quad \text{prolongate } \Delta \phi_h^\tau = \mathbf{I}_{2h}^h \cdot \Delta \phi_{2h}^\tau \\
 &\quad \text{compute } \phi_h^{\tau+1} = \phi_h^\tau + \Delta \phi_h^\tau \\
 &\quad \text{smooth } \mathbf{r}_h^{\tau+1} = \mathbf{b}_h - \mathbf{A}_h \cdot \phi_h^{\tau+1} \quad m \text{ times to finalize } \phi_h^{\tau+1}
 \end{aligned} \tag{8.369}$$

encompassing *one* iteration step (cycle)  $\tau$  of the multigrid procedure. Such a consecutive fine-to-coarse and coarse-to-fine multigrid cycle is called V-cycle, sketched in Fig. 8.42. However, there are much more options of how to cycle the multiple grids. Another possibility is the W-cycle, where more coarse grids are visited to drive the residuals down as much as possible before returning to the more expensive finer grids (Fig. 8.42). In cases where the initial solution on the fine mesh may be too poor, a full multigrid cycle (Fig. 8.42) is appropriate to obtain better starting solutions on the coarse grids.

It can be shown for the GMG method [519] that its *convergence is independent of the size of the finest grid*. Solving a problem in  $D$  dimensions, the reduction in the number of nodal points  $N_P$  between subsequent grids is of the order of  $N_P^{2h}/N_P^h \sim 1/2^D$ . Assuming that  $n$  smoothing steps are required on each grid and the computational work of each smoothing processes is proportional to the effective



**Fig. 8.42** Types of multigrid cycle exemplified for a four-grid method

number of equations  $N_{EQ}$ , the total computational work needed for a full V-cycle is only of the order  $\mathcal{O}(n N_{EQ}^h \log_D(N_{EQ}^h))$  and the associated storage requirement only of the order  $\mathcal{O}(N_{EQ}^h \log_D(N_{EQ}^h))$ . In fact, these estimates are extremely favorable and hardly to be beaten by any other iterative strategy. For a W-cycle the amount of work is only slightly larger. In particular for 3D problems, we observe that the number of grid points on the coarser grids drops dramatically.

### Algebraic Multigrid (AMG) Method

While the GMG method has shown very efficient in particular for large and very large problems, there are unfortunately a number of serious deficiencies which hamper its use in the finite element modeling practice. Detrimental is that GMG often deteriorates for problems with anisotropic and discontinuous coefficients. More important is that GMG depends fundamentally on the availability of an underlying grid. The treatment of complex meshes in 3D has shown often rather cumbersome. The FEM generally uses unstructured, nonhierarchical meshes. For that mesh complexity it is difficult if not impossible to construct reliable GMG methods. However, the basic principles of GMG can be exploited in a generalized strategy without suffering from GMG's fundamental restrictions. Such a strategy has become true with the *algebraic multigrid* (AMG) method [453, 519].

Brandt [52] and Stüben [497] can be seen as the major protagonists of the AMG method, who started AMG's development in the early 1980s. It was motivated by the observation that straightforward geometric grid transfer operations of restriction and prolongation can be alternatively formulated on the basis of the underlying matrices without any reference to grids (meshes), i.e., the construction of these operators can be done purely algebraically. On the other hand, AMG's algorithmic components of smoothing and coarse-grid correction remain completely analogous to the classical GMG method maintaining its computational power and efficiency for solving large matrix systems. In contrast to GMG where coarse-grid discretizations are used to reduce low-frequency error components, the AMG method reduces the low error frequencies on matrix equations of a reduced dimension defining a certain level. As a consequence, AMG does not require anymore fixed grid hierarchies. Accordingly, in AMG one should better use the term *multilevel* rather than multigrid (but for historical reasons the term multigrid is often further preferred in the AMG context).

AMG is best developed for scalar elliptic PDE, however, recent progress has also been attained for systems of PDE's and ADE's. It has been proven to be a very robust and efficient solution method applicable to both structured and unstructured meshes. Stüben [498] gives a comprehensive overview on AMG in different fields of application. The key feature of AMG is the exploitation of the Galerkin approach comparable to (8.367) of GMG, however, in the context of AMG the required coarse-grid operators of restriction and prolongation are based on interpolation that maps a coarse node into a fine one of a given matrix system. This coarsening process is fully automatic. Suppose the finite element matrix system  $\mathbf{A} \cdot \phi = \mathbf{b}$  is given at the highest level (finest mesh)

$$\mathbf{A}_h \cdot \phi_h = \mathbf{b}_h \quad (8.370)$$

similar to a geometric two-grid description, we can define the matrix system for the next coarse-level problem identified by subscript  $H$  as

$$\mathbf{A}_H \cdot \phi_H = \mathbf{b}_H \quad (8.371)$$

The coarse-level AMG system (8.371) is constructed by means of the Galerkin approach. In doing so, the coarse matrix  $\mathbf{A}_H$  results from

$$\mathbf{A}_H = (\mathbf{I}_h^H \cdot \mathbf{A}_h) \cdot \mathbf{I}_H^h \quad (8.372)$$

where  $\mathbf{I}_h^H$  and  $\mathbf{I}_H^h$  denote the restriction and prolongation operators, respectively. Their construction forms the major task of AMG's setup process, see [498] for more details. Having these operators a two-level process and by its recursive application any multilevel process can be easily performed in an analogy to GMG's multigrid cycle and smoothing algorithms described above.



## 8.18 Treatment of Nonlinearities

In the previous Sect. 8.17 we have described techniques for solving the resulting algebraic equations systems  $\mathbf{A} \cdot \phi = \mathbf{b}$ , provided that the system is *linear*, i.e., the solution  $\phi$  does not occur in any other combination than in a linear one. However, in a number of applications the parameters in the governing finite element equations can contain dependencies on the solution  $\phi$  itself. Typical examples are variable density and variable saturation problems, where the advective and diffusive (conductive) terms in  $\mathbf{A}$  become a function of  $\phi$ . Furthermore, nonlinear BC's and higher order reaction processes imply nonlinear dependencies in the RHS vector  $\mathbf{b}$ . In such cases a nonlinear matrix system results in a form

$$\mathbf{A}(\phi) \cdot \phi = \mathbf{b}(\phi) \quad (8.373)$$

where the main nonlinear functional dependence is identified by parentheses. Before we can solve (8.373) for  $\phi$  the system of equations must be *linearized* by using appropriate iterative methods. Most important are the Picard iteration method, which is a linearly convergent algorithm, and the Newton iteration method, which normally converges quadratically. A specific concern is suitable for transient problems.

### 8.18.1 Fixed Point Form and Picard Iteration Method

The system of nonlinear equations (8.373) written as

$$\mathbf{R}(\phi) = \mathbf{0} \quad \text{with} \quad \mathbf{R}(\phi) = \mathbf{A}(\phi) \cdot \phi - \mathbf{b}(\phi) \quad (8.374)$$

can be given in its *fixed point form* as

$$\phi = \mathbf{G}(\phi) \quad \text{with} \quad \mathbf{G}(\phi) = \mathbf{A}^{-1}(\phi) \cdot \mathbf{b}(\phi) \quad (8.375)$$

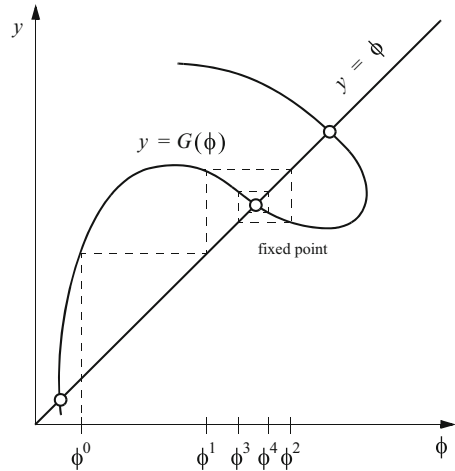
Solutions of (8.375) are called fixed points of the mapping function  $\mathbf{G}(\phi)$ , which represent solutions of (8.374). In graphical terms, fixed points are the intersections of the graph  $\mathbf{y} = \mathbf{G}(\phi)$  with the line  $\mathbf{y} = \phi$  as illustrated in Fig. 8.43 for a scalar functional dependence.

The fixed point form (8.375) immediately suggests the following iteration scheme

$$\phi^{\tau+1} = \mathbf{G}(\phi^\tau) \quad \tau = 0, 1, 2, \dots \quad (8.376)$$

where  $\tau$  is the iteration counter. The formulation (8.376) is the method of *successive substitution* known as the *Picard iteration method*. The iteration is started with a first

**Fig. 8.43** Fixed points of  $y = G(\phi)$  and fixed point (Picard) iteration starting from the initial value  $\phi^0$



guess  $\phi^0$  so that  $G(\phi^0)$  can be evaluated to obtain  $\phi^1$ . Repeating this procedure a sequence of successive solutions for  $\phi^{\tau+1}$  is obtained (see illustration in Fig. 8.43). In the practical application, however, the matrix  $G$  is not directly formed. Instead, the Picard iteration is executed in the basic matrix system in the form

$$A(\phi^\tau) \cdot \phi^{\tau+1} = b(\phi^\tau) \quad \tau = 0, 1, 2, \dots \tag{8.377}$$

to obtain  $\phi^{\tau+1}$ . The iteration method *linearizes* the matrix system so that the equation system (8.377) can be easily solved by using solution techniques of Sect. 8.17. An advantage of the Picard method is that the structural matrix properties remain unchanged, in particular, if  $A$  in (8.373) is *symmetric* the matrix system (8.377) remains symmetric. During the iterative loop the matrix system  $A$  and the RHS  $b$  must be updated (reassembled) with the previous solution and the equation system (8.377) has to be repeatedly solved until satisfactory convergence is achieved. A typical convergence criterion is

$$\frac{\|\phi^{\tau+1} - \phi^\tau\|}{\|\phi^{\tau+1}\|} \leq \epsilon \tag{8.378}$$

where  $\epsilon$  is an error tolerance to be prescribed and  $\|\cdot\|$  corresponds to a suitable error norm, e.g., RMS error norm (8.28) or maximum error norm (8.29).

The proof of the convergence for the Picard iteration is given by the *Banach fixed point theorem*, e.g., [199]. It is shown that the iteration error of the Picard method decreases *linearly* with the error of the previous iteration step, viz.,

$$\|\phi^{\tau+1} - \phi^\tau\| < \|\phi^\tau - \phi^{\tau-1}\| \quad \text{for } \tau > 0 \tag{8.379}$$

provided that the initial estimate for the solution  $\phi^0$  is within a contracting distance

$$\|\phi^0 - \phi\| \leq r_c \quad (8.380)$$

to converge to the unique solution  $\phi$ , where  $r_c$  is referred to as the *radius of convergence* of the Picard iteration scheme. In general, for the present class of problems it is not possible to determine  $r_c$ . For strong nonlinearities the convergence radius can be very small and a good first guess of the solution is usually needed to attain a converging solution, otherwise the method diverges and fails. Nevertheless, the Picard iteration method has shown relatively robust in many applications. Its robustness is however paid by an only linear (1st-order) convergence rate.

### 8.18.2 Newton Iteration Method

The *Newton iteration method*, also known as the *Newton-Raphson method*, possesses a more rapid convergence behavior in form of a quadratic convergence rate. Considering the nonlinear matrix equation (8.374) written as

$$\mathbf{R}(\phi) = \mathbf{A}(\phi) \cdot \phi - \mathbf{b}(\phi) = \mathbf{0} \quad (8.381)$$

and assuming that the residual  $\mathbf{R}(\phi)$  is continuous and differentiable, a Taylor series expansion for the residual at the new iteration  $\mathbf{R}(\phi^{\tau+1})$  about the previous iterative solution  $\phi^\tau$  yields

$$\mathbf{R}(\phi^{\tau+1}) = \mathbf{R}(\phi^\tau) + \frac{\partial \mathbf{R}(\phi^\tau)}{\partial \phi^\tau} \cdot \Delta \phi^\tau + \mathcal{O}(\Delta \phi^{\tau 2}) + \dots \quad (8.382)$$

with

$$\Delta \phi^\tau = \phi^{\tau+1} - \phi^\tau \quad (8.383)$$

Assuming  $\mathbf{R}(\phi^{\tau+1}) = \mathbf{0}$  and neglecting 2nd and higher order terms, we obtain from (8.382) the Newton iteration scheme in the form

$$\mathbf{J}(\phi^\tau) \cdot \Delta \phi^\tau = -\mathbf{R}(\phi^\tau) \quad \tau = 0, 1, 2, \dots \quad (8.384)$$

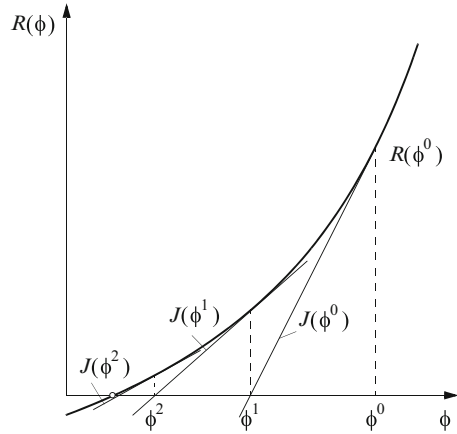
where

$$\mathbf{R}(\phi^\tau) = \mathbf{A}(\phi^\tau) \cdot \phi^\tau - \mathbf{b}(\phi^\tau) \quad (8.385)$$

and the tangential matrix

$$\mathbf{J}(\phi^\tau) = \frac{\partial \mathbf{R}(\phi^\tau)}{\partial \phi^\tau} = \frac{\partial}{\partial \phi^\tau} [\mathbf{A}(\phi^\tau) \cdot \phi^\tau - \mathbf{b}(\phi^\tau)] = \mathbf{A}(\phi^\tau) + \hat{\mathbf{J}}(\phi^\tau) \quad (8.386)$$

**Fig. 8.44** Illustration of Newton iteration method for solving the nonlinear function  $R(\phi) = 0$ . Case of convergence. Iteration starts from the initial value  $\phi^0$ . The Jacobians  $J(\phi^\tau) = \partial R(\phi^\tau)/\partial \phi^\tau$  represent the tangential slopes which are variable for each iteration  $\tau = 0, 1, 2, \dots$



with

$$\hat{J}(\phi^\tau) = \frac{\partial \mathbf{A}(\phi^\tau)}{\partial \phi^\tau} \cdot \phi^\tau - \frac{\partial \mathbf{b}(\phi^\tau)}{\partial \phi^\tau} \quad (8.387)$$

in which  $\mathbf{J}(\phi^\tau)$  and  $\hat{\mathbf{J}}(\phi^\tau)$  are the Jacobian matrix and the *partial* Jacobian matrix, respectively. It is important to note that the Jacobian has to be updated for each iteration (*full Newton method*) to realize the quadratic convergence rate as illustrated in Fig. 8.44 for a scalar functional dependence. On the other hand, the partial Jacobian  $\hat{\mathbf{J}}$  causes always an *unsymmetric* matrix even if the system matrix  $\mathbf{A}$  is symmetric. For these reasons the Newton iteration method is relatively expensive. The partial Jacobian  $\hat{\mathbf{J}}$  can be computed either analytically or numerically.<sup>16</sup> Usually, the analytical evaluation is preferred because it provides a more efficient implementation. From (8.386) we can recognize that the full Newton method reduces to the Picard method (8.377) when the partial Jacobian  $\hat{\mathbf{J}}$  (8.387) is dropped such that  $\mathbf{J}(\phi^\tau) \approx \mathbf{A}(\phi^\tau)$ .

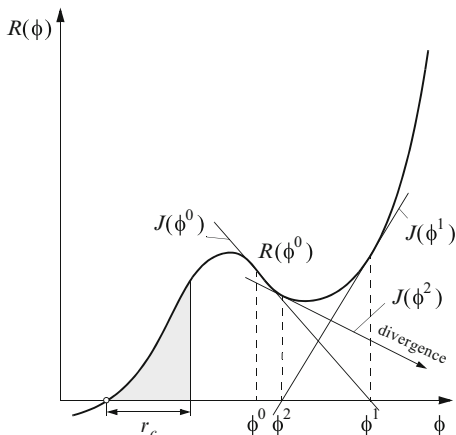
For terminating the Newton iteration scheme (8.384) the deviatory convergence criterion of (8.378) may be applied. However, the convergence of the Newton

<sup>16</sup>If the Jacobian  $\mathbf{J}(\phi^\tau) = \partial \mathbf{R}(\phi^\tau)/\partial \phi^\tau$  is not analytically available or too difficult for an analytical evaluation, it can be constructed numerically via a secant approximation by using a possibly very small increment  $\delta$  in a form such as

$$\mathbf{J}(\phi^\tau) \approx \frac{\mathbf{R}(\phi^\tau + \delta) - \mathbf{R}(\phi^\tau)}{\delta}$$

The increment  $\delta$  should not be chosen too small to avoid roundoff errors. On the other hand, a too large  $\delta$  leads to a poor approximation of the Jacobian. A reasonable choice is the square root of the unit roundoff being about  $\epsilon_R = 10^{-12}$  in double precision arithmetic, accordingly  $\delta = \sqrt{\epsilon_R} = 10^{-6}$ . The extra effort of the numerical evaluation consists of additional  $N_{EQ}$  evaluation of residual  $\mathbf{R}$ .

**Fig. 8.45** Diverging Newton iteration in solving the nonlinear function  $R(\phi) = 0$  if initial value  $\phi^0$  is outside the convergence radius  $r_c$



method can easily (and additionally) be controlled by another useful error criterion, viz., the test of the minimal residual, e.g., written in the form

$$\frac{\|R(\phi^{\tau+1})\|}{\|F(\phi^{\tau+1})\|} \leq \epsilon_2 \tag{8.388}$$

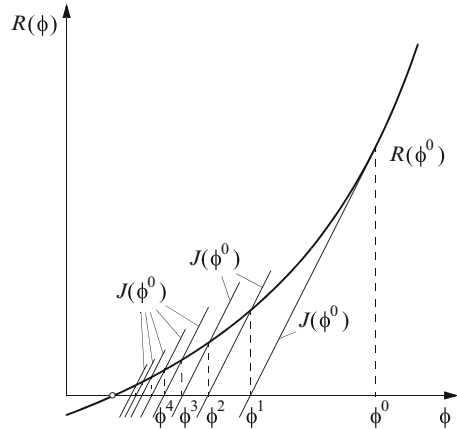
normalized for instance by the RHS vector  $F$  (appearing in (8.154) or (8.177)), where  $\epsilon_2$  represents a second convergence criterion. The advantage of this test is that the global balance error of the spatio-temporal matrix system is directly controlled. The acceptable measure of the minimal residual  $\epsilon_2$  can be chosen suitably small, possibly in the range of the roundoff error.

It is known that the Newton method requires a good first guess of the solution  $\phi^0$ , otherwise if the starting solution is too far from the correct solution the method can ‘blow up’ and quickly diverges (see Fig. 8.45). The convergence radius  $r_c$  as defined in (8.380) is generally smaller for the Newton method than for the Picard iteration method. It is complicated further in the Newton method that  $r_c$  decreases as the number of equations  $N_{EQ}$  increases so that  $\phi^0$  must be closer to the correct solution for bigger meshes. There are various cost-effective modifications in the Newton iteration method to reduce the increased computational effort in updating the Jacobian, where a concomitantly slower convergence rate (commonly linear) has to be accepted. Most important are the modified Newton method and the quasi-Newton method introduced next.

### 8.18.3 Modified Newton and Quasi-Newton Iteration Method

A major drawback of the full Newton method is that the Jacobian  $J(\phi^\tau)$  has to be updated in each iteration  $\tau$ . Although its quadratic convergence leads usually

**Fig. 8.46** Illustration of modified Newton iteration method for solving the nonlinear function  $R(\phi) = 0$ . Case of convergence. Iteration starts from the initial value  $\phi^0$ . The Jacobian  $J(\phi^0) = \partial R(\phi^0)/\partial \phi^0$  represents the initial tangential slope at  $\phi^0$  which is kept and used in all subsequent iterations



to a small number of iterations, each iteration is accordingly expensive. There are variants of the Newton method which can result in fewer costly iterations, however, at the expense of a slower convergence. Nevertheless, since each iterative step is ostensibly cheaper one can afford more iterations.

To obviate the need of Jacobian updating the *modified Newton method* can be used in which the Jacobian is only built once at the initial step, i.e.,  $J(\phi^0)$  is formed with the initial solution  $\phi^0$ . Then, all subsequent iterations leave this initial Jacobian  $J(\phi^0)$  unchanged (see Fig. 8.46), i.e.,

$$J(\phi^0) \cdot \Delta \phi^\tau = -R(\phi^\tau) \quad \text{with} \quad J(\phi^0) = \frac{\partial R(\phi^0)}{\partial \phi^0} \quad (8.389)$$

The convergence rate of the modified Newton iteration method is only linear. However, in comparison to the Picard method, which is also linearly convergent, the modified Newton algorithm is usually cheaper because it needs only one matrix update per iteration cycle.

Another possible cost-effective modification is the *quasi-Newton method*. In this case the Jacobian  $J(\phi^\tau)$  can be thought of as approximations to the system matrix  $A(\phi^\tau)$ . The quasi-Newton iteration can be written in the form

$$\begin{aligned} \phi^{\tau+1} &= \phi^\tau - s^\tau A^{-1}(\phi^\tau) \cdot R(\phi^\tau) \\ A^{-1}(\phi^{\tau+1}) &= A^{-1}(\phi^\tau) + \Delta A^{-1}(\phi^\tau) \end{aligned} \quad (8.390)$$

where  $A(\phi^\tau)$  has to satisfy the secant condition

$$A(\phi^\tau) \cdot (\phi^\tau - \phi^{\tau-1}) = R(\phi^\tau) - R(\phi^{\tau-1}) \quad (8.391)$$

in which  $s^\tau$  is an acceleration factor (usually,  $s^\tau = 1$ ) and the update of the system matrix is expressed directly via an incremental correction  $\Delta A^{-1}(\phi^\tau)$  to its inverse.

The efficiency of the quasi-Newton method is dependent on finding good choices of inverse update forms. For a symmetric matrix  $\mathbf{A}$  the *Broyden-Fletcher-Goldfarb-Shannon (BFGS)* update [121] has shown most successful. Broyden's update is also available for unsymmetric matrices, for more see, e.g., [156]. The quasi-Newton method possesses usually a better than linear convergence rate and is accordingly superior to the modified Newton method.

### 8.18.4 Transient Nonlinear Problem Solution

For transient problems the nonlinear matrix system has to be solved at the time plane  $n + 1$ :

$$\mathbf{A}(\phi_{n+1}) \cdot \phi_{n+1} = \mathbf{b}(\phi_{n+1}) \quad (8.392)$$

In accordance with the used time integration methods different strategies have found appropriate. In principle, at each time plane the nonlinear system (8.392) must be iteratively solved to achieve convergence. The iteration procedure reads for the Picard method

$$\mathbf{A}(\phi_{n+1}^\tau) \cdot \phi_{n+1}^{\tau+1} = \mathbf{b}(\phi_{n+1}^\tau) \quad \tau = 0, 1, 2, \dots \quad (8.393)$$

and for the full Newton method

$$\begin{aligned} \mathbf{J}(\phi_{n+1}^\tau) \cdot \Delta\phi_{n+1}^\tau &= -\mathbf{R}(\phi_{n+1}^\tau) \quad \tau = 0, 1, 2, \dots \\ \Delta\phi_{n+1}^\tau &= \phi_{n+1}^{\tau+1} - \phi_{n+1}^\tau \\ \mathbf{J}(\phi_{n+1}^\tau) &= \frac{\partial \mathbf{R}(\phi_{n+1}^\tau)}{\partial \phi_{n+1}^\tau} \\ \mathbf{R}(\phi_{n+1}^\tau) &= \mathbf{A}(\phi_{n+1}^\tau) \cdot \phi_{n+1}^\tau - \mathbf{b}(\phi_{n+1}^\tau) \end{aligned} \quad (8.394)$$

The iteration usually starts at time plane  $n + 1$  with the first guess taking from the previous time  $n$ , i.e.,  $\phi_{n+1}^0 = \phi_n$ . The process is repeated within each time plane until the following convergence criteria are satisfied:

$$\frac{\|\phi_{n+1}^{\tau+1} - \phi_{n+1}^\tau\|}{\|\phi_{n+1}^\tau\|} \leq \epsilon \quad (8.395)$$

and/or

$$\frac{\|\mathbf{R}(\phi_{n+1}^{\tau+1})\|}{\|\mathbf{F}(\phi_{n+1}^{\tau+1})\|} \leq \epsilon_2 \quad (8.396)$$

For transient problems a good first guess  $\phi_{n+1}^0$  is always available since the solution usually changes little between time steps, provided the time step length  $\Delta t_n$  is

sufficiently small. All the more, if we use the error-controlled predictor-corrector methods as described in Sect. 8.13.5 an even better first guess can be obtained by using the predictor solution  $\phi_{n+1}^p$  at the current time plane  $n + 1$ , viz.,

$$\phi_{n+1}^0 = \phi_{n+1}^p \quad (8.397)$$

where  $\phi_{n+1}^p$  is given by (8.157) and (8.166) for the FE and AB scheme, respectively. Now, it is argued [211] that (1) the required degree of convergence is reached in just one iteration per time step when the predictor furnishes a sufficiently accurate first guess, and (2) the pre-set error measure  $\epsilon$  used in the predictor-corrector schemes is recognized as the controlling parameter when keeping the time discretization error small. It leads to the so-called *one-step Newton method* (or alternatively, *one-step Picard method*), in which the predictor value  $\phi_{n+1}^p$  is generally utilized to linearize the complete nonlinear system without any need for a repeated iteration within each time step. The following procedures result

$$\begin{aligned} \mathbf{J}(\phi_{n+1}^p) \cdot \Delta\phi_{n+1} &= -\mathbf{R}(\phi_{n+1}^p) \\ \Delta\phi_{n+1} &= \phi_{n+1} - \phi_{n+1}^p \\ \mathbf{J}(\phi_{n+1}^p) &= \frac{\partial \mathbf{R}(\phi_{n+1}^p)}{\partial \phi_{n+1}^p} \\ \mathbf{R}(\phi_{n+1}^p) &= \mathbf{A}(\phi_{n+1}^p) \cdot \phi_{n+1}^p - \mathbf{b}(\phi_{n+1}^p) \end{aligned} \quad (8.398)$$

for the one-step Newton method and

$$\mathbf{A}(\phi_{n+1}^p) \cdot \phi_{n+1} = \mathbf{b}(\phi_{n+1}^p) \quad (8.399)$$

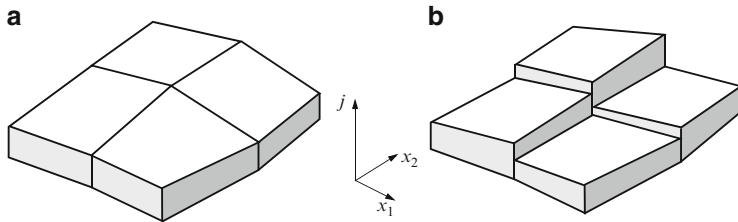
for the one-step Picard method. The one-step Newton (or Picard) method embedded in the predictor-corrector scheme with its automatic step-size (time approximation error) control has shown a cost-effective and favorable approach in many applications. In comparison to the Picard method the extra work for the one-step Newton method is small in forming the Jacobian and residual matrices which can be done simultaneously with assembling the matrix system, however, in favor of achieving a quadratic convergence behavior. This is particularly true for an unsymmetric system matrix  $\mathbf{A}$ , typically appearing in ADE problems. For symmetric systems possessing strong nonlinearities the use of the Newton procedure can also be advantageous, in spite of losing symmetry in the final equation system to be solved.

## 8.19 Derived Quantities

### 8.19.1 Computing First Derivatives at Nodes

We have discussed in Sect. 3.11 the suitably chosen primary variables in form of hydraulic head, species concentration or temperature for solving the governing flow, mass and heat transport equations in porous media. Having known the





**Fig. 8.47** (a) Continuous (smoothed) and (b) discontinuous (unsmoothed) flux component given for an element patch of quadrilaterals (Modified from [251])

spatio-temporal solutions of the primary variables, there is a need to obtain the solution of secondary variables, such as Darcy velocity, mass flux or heat flux, which represent quantities derived from the primary variables. In terms of the prototypical ADE equation (8.3) or (8.5), the FEM leads to the solution of the primary variable  $\phi$  in space and time, which represents an elementwise continuous approximation (cf. Sect. 8.7). A derived quantity would be the flux

$$\mathbf{j} = -\mathbf{D} \cdot \nabla \phi \tag{8.400}$$

where  $\mathbf{D}$  is a dispersion tensor. Since the finite element approximation of the primary variable  $\phi$  is of the form (cf. (8.16))

$$\phi(\mathbf{x}, t) = \sum_j N_j(\mathbf{x}) \phi_j(t) \tag{8.401}$$

we obtain the discrete flux

$$\mathbf{j}(\mathbf{x}, t) = - \sum_j \mathbf{D} \cdot \nabla N_j(\mathbf{x}) \phi_j(t) \tag{8.402}$$

As result of the basic finite element solution, e.g., (8.322),  $\phi_j$  is known at each global node  $j$  of the mesh and given time  $t$  so that  $\mathbf{j}$  can be evaluated in a postprocessing operation. However, we recognize from (8.402) the first derivative in the flux  $\mathbf{j}$  is no more continuous since the used element shape function  $N_j$  satisfies only  $C_0$ -continuity (cf. Sect. 8.7). Indeed, by using lower order elements of linear or quadratic type, the first derivatives do not possess anymore inter-element continuity. The elemental fluxes become discontinuous between elements and no unique fluxes at nodal points result as illustrated in Fig. 8.47.

Unfortunately, the discontinuity of the derived fluxes results in a number of serious drawbacks. Most important are balance errors arising in local flux evaluations. For example, Yeh [578] showed balance errors up to 30 % on a domain interior. On the other hand, the evaluation of streamlines according to (2.94) and pathlines according to (2.98) needs always a basically continuous flow field,

otherwise coherent trajectories in the global flow field cannot be computed. Hence, suitable methods are required to produce continuous and precise fluxes over the finite element mesh, which are referred to as *smoothing* strategies. Most important are global and local smoothing as well as superconvergent patch recovery (SPR) techniques to derive continuous fluxes at internal nodes.

### 8.19.1.1 Global Smoothing

Global smoothing represents a natural approach of FEM to obtain continuous flux values at nodes. Most common is the technique basically proposed by Hinton and Campbell [251], which has proved to be quite widely used [590]. Yeh [578] firstly introduced such type of global smoothing in groundwater modeling to compute precise Darcy fluxes. A global finite element approximation of a smoothed (continuous) flux  $\tilde{j}$  can be written as

$$\tilde{j}(\mathbf{x}, t) = \sum_j N_j(\mathbf{x}) \tilde{j}_j(t) \quad (8.403)$$

Suppose an unsmoothed (discontinuous) flux is given by  $j$  (8.402), then the smooth function which provides a best fit in the least squares sense over the domain  $\Omega$  can be obtained from a minimization of the functional

$$\mathcal{I} = \int_{\Omega} (\tilde{j} - j)^2 d\Omega \Rightarrow \min \quad (8.404)$$

The minimization procedure

$$\begin{aligned} \frac{\partial \mathcal{I}}{\partial \tilde{j}_i} &= \int_{\Omega} 2(\tilde{j} - j) \frac{\partial \tilde{j}}{\partial \tilde{j}_i} d\Omega = \mathbf{0} \\ &= \int_{\Omega} N_i(\tilde{j} - j) d\Omega = \mathbf{0} \end{aligned} \quad \text{for } i = 1, 2, \dots, N_P \quad (8.405)$$

results in a system of linear equations to solve for the nodal vector of smoothed fluxes  $\tilde{j}_d$  for each vector component  $d = 1, \dots, D$  in  $\mathfrak{R}^D$ , viz.,

$$\mathbf{O} \cdot \tilde{j}_d = \mathbf{F}_d \quad (d = 1, \dots, D) \quad (8.406)$$

where  $\mathbf{O}$  represents a mass (smoothing) matrix and  $\mathbf{F}_d$  is the RHS  $d$ -component flux vector involving the unsmoothed relations. They are formed in the finite element assembling procedure as

$$\begin{aligned} \mathbf{O} &= O_{ij} = \sum_e \left( \sum_I \sum_J O_{IJ}^e \Delta_{ii}^e \Delta_{jj}^e \right) \\ \mathbf{F}_d &= F_{id} = \sum_e \left( \sum_I F_{Id}^e \Delta_{ii}^e \right) \end{aligned} \quad (8.407)$$

where the element matrix and element vector are formed by

$$\begin{aligned} O_{IJ}^e &= \int_{\Omega^e} N_I^e N_J^e d\Omega^e \\ F_{Id}^e &= - \int_{\Omega^e} N_I^e \sum_J^{N_{\text{BN}}} \sum_l^D (D_{dl}^e \frac{\partial N_J^e}{\partial x_l} \phi_J^e) d\Omega^e \end{aligned} \quad (d, l = 1, \dots, D) \quad (8.408)$$

It can be easily seen that the least squares global smoothing is equivalent to the Galerkin weak statement directly applied to (8.400). The smoothing matrix  $\mathbf{O}$  and the RHS vector  $\mathbf{F}_d$  may be evaluated using numerical integration as described in Sect. 8.12. However, the complete solution of the linear matrix system (8.406) has been found too costly and furthermore in many cases unnecessary. The following smoothing procedures will be accordingly the preferred techniques.

A cost-effective alternative to (8.406) appears if the element smoothing matrix  $\mathbf{O}$  is lumped by a row-summing technique (see Sect. 8.13.2) for each element  $e$

$$O_{IJ}^e = \delta_{IJ} \int_{\Omega^e} N_I^e d\Omega^e \quad (8.409)$$

In doing so, there is no need anymore to solve the linear equation system (8.406). Instead, the smoothed flux can be explicitly evaluated by

$$\tilde{\mathbf{j}}_d = \mathbf{O}^{-1} \cdot \mathbf{F}_d \quad (d = 1, \dots, D) \quad (8.410)$$

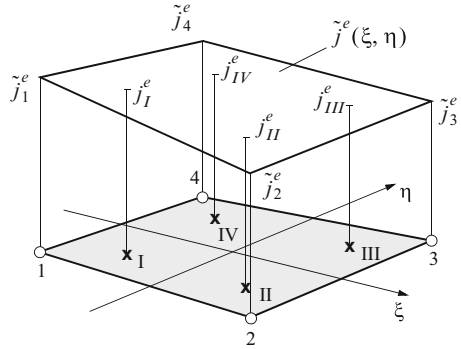
where the inverse of the diagonal matrix  $\mathbf{O}^{-1}$  effects for each node a division by the sum  $\sum_e \int_{\Omega^e} d\Omega^e$  of the surrounding element patch. This lumped form of global smoothing (8.410) can be recognized as an area/volume-weighted averaging for nodal flux values. However, this area/volume-weighting strategy has an essential drawback for irregular meshes: It weights larger elements more than smaller elements notwithstanding that larger elements imply presumably less accurate flux computations. To weight the more accurate smaller elements than the less accurate larger elements, the inverse area/volume-weighted averaging could be chosen instead [209], however, its foundation lies outside of the Galerkin-FEM framework. Thus, the following local smoothing and recovery strategies will be preferred.

### 8.19.1.2 Superconvergent Flux Evaluation and Local Smoothing

In FEM there is the phenomenon of *superconvergence* [590], which is referred to optimal sampling points for which derivatives are more accurate than elsewhere. In particular, Gauss quadrature sampling points (cf. Sect. 8.12) exhibit superconvergent behavior and have shown the suited locations  $\mathbf{x}$  to evaluate derived quantities

**Fig. 8.48** Superconvergent flux components

$j_I^e, j_{II}^e, j_{III}^e, j_{IV}^e$  sampled at Gauss points  $\xi_p, \eta_p$  ( $p = 1, \dots, 4$ ) and functional dependency of smoothed flux  $\tilde{j}^e(\xi, \eta)$  to extrapolate to nodal flux components  $\tilde{j}_1^e, \tilde{j}_2^e, \tilde{j}_3^e, \tilde{j}_4^e$  for a linear quadrilateral element  $e$



having best accuracy.<sup>17</sup> Such a superconvergent flux evaluation means that the discontinuous flux  $\mathbf{j}(\mathbf{x}(\boldsymbol{\eta}), t)$  of (8.402) has to be sampled at the Gauss points with the local coordinates  $\boldsymbol{\eta}_p$  ( $p = 1, \dots, m$ ) within each element  $e$ , i.e.,

$$\mathbf{j}(\mathbf{x}(\boldsymbol{\eta}), t) \rightarrow \mathbf{j}^e(\boldsymbol{\eta}_p, t) = - \sum_J^{N_{BN}} \mathbf{D}^e \cdot \nabla N_J^e(\boldsymbol{\eta}_p) \phi_J^e(t) \quad (p = 1, \dots, m) \tag{8.411}$$

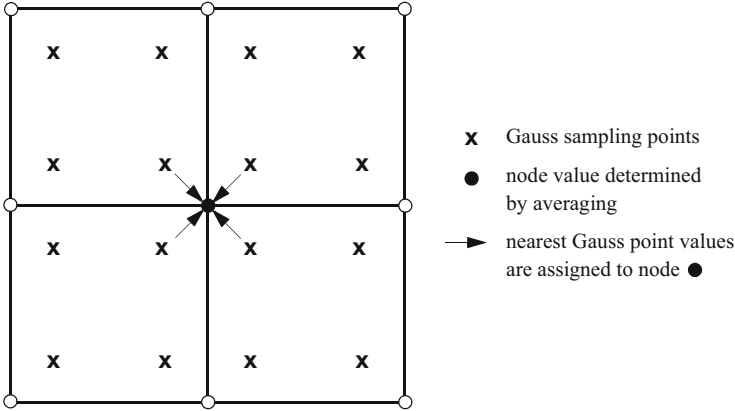
where  $\boldsymbol{\eta}_p$  is the vector of local coordinates, e.g.,  $(\xi_p, \eta_p, \zeta_p)$  for a 3D element, and  $m$  is the total number of Gauss points. Their locations in 2D and 3D elements are displayed in Fig. 8.17. Conveniently,  $m$  is chosen by the same number of element nodes  $N_{BN}$  so that the Gauss sample points can be related to corresponding element nodes (see Fig. 8.48).

Now, the smoothing of the discontinuous flux is considered over individual elements, termed *local smoothing*. It is assumed that the smoothed flux function  $\tilde{\mathbf{j}}^e(\boldsymbol{\eta})$  is a least squares fit to the selected values  $\mathbf{j}^e(\boldsymbol{\eta}_p)$  at the Gauss points  $p = 1, \dots, m$  for each element  $e$  separately. In using the least squares procedure of Sect. 8.19.1.1 to an individual element the following local equation system results

$$\begin{pmatrix} \int_{\Omega^e} N_1^e N_1^e d\Omega^e & \dots & \int_{\Omega^e} N_1^e N_{N_{BN}}^e d\Omega^e \\ \int_{\Omega^e} N_2^e N_1^e d\Omega^e & \dots & \int_{\Omega^e} N_2^e N_{N_{BN}}^e d\Omega^e \\ \vdots & \vdots & \vdots \\ \int_{\Omega^e} N_{N_{BN}}^e N_1^e d\Omega^e & \dots & \int_{\Omega^e} N_{N_{BN}}^e N_{N_{BN}}^e d\Omega^e \end{pmatrix} \cdot \begin{pmatrix} \tilde{j}_{1d}^e \\ \tilde{j}_{2d}^e \\ \vdots \\ \tilde{j}_{N_{BN}d}^e \end{pmatrix} = \begin{pmatrix} \int_{\Omega^e} N_1^e j_d^e(\boldsymbol{\eta}_1) d\Omega^e \\ \int_{\Omega^e} N_2^e j_d^e(\boldsymbol{\eta}_2) d\Omega^e \\ \vdots \\ \int_{\Omega^e} N_{N_{BN}}^e j_d^e(\boldsymbol{\eta}_{m=N_{BN}}) d\Omega^e \end{pmatrix} \tag{8.412}$$

$(d = 1, \dots, D)$

<sup>17</sup>Superconvergence of the derivatives can be shown for the Gauss points, at least for quadrilateral elements [590]. On the other hand, the location of the superconvergent points for triangular elements is not fully known. Zienkiewicz and Zhu [594] propose to use optimal points, for instance the central points for linear triangles.



**Fig. 8.49** Element patch surrounding the particular node ● at which a globally smoothed flux is computed on the basis of extrapolated or shifted superconvergent flux values sampled at Gauss points ×

to solve the smoothed flux  $d$ -components  $\tilde{j}_{Id}^e$  at the local nodes  $I = 1, 2, \dots, N_{BN}$  of an element  $e$  from the superconvergent flux  $d$ -components  $j_d^e(\eta_p)$  according to (8.411) sampled at Gauss points  $p = 1, \dots, m = N_{BN}$ . Simple relations are obtained from (8.412) for elements having a constant Jacobian in  $d\Omega^e = |\mathbf{J}^e|d\eta$ , see derivations in Appendix H. For example, for a rectangular and parallelogram 2D element by using  $2 \times 2$  Gauss points (listed in Table 8.2) the following expression can be derived

$$\begin{pmatrix} \tilde{j}_{1d}^e \\ \tilde{j}_{2d}^e \\ \tilde{j}_{3d}^e \\ \tilde{j}_{4d}^e \end{pmatrix} = \begin{pmatrix} 1 + \frac{\sqrt{3}}{2} & -\frac{1}{2} & 1 - \frac{\sqrt{3}}{2} & -\frac{1}{2} \\ -\frac{1}{2} & 1 + \frac{\sqrt{3}}{2} & -\frac{1}{2} & 1 - \frac{\sqrt{3}}{2} \\ 1 - \frac{\sqrt{3}}{2} & -\frac{1}{2} & 1 + \frac{\sqrt{3}}{2} & -\frac{1}{2} \\ -\frac{1}{2} & 1 - \frac{\sqrt{3}}{2} & -\frac{1}{2} & 1 + \frac{\sqrt{3}}{2} \end{pmatrix} \cdot \begin{pmatrix} j_{dI}^e \\ j_{dII}^e \\ j_{dIII}^e \\ j_{dIV}^e \end{pmatrix} \quad (d = 1, \dots, D) \tag{8.413}$$

to compute directly the smoothed flux components  $\tilde{j}_{Id}^e$  at the corner nodes  $I = 1, 2, 3, 4$  from the superconvergent flux components at Gauss points  $I, II, III, IV$  (illustrated in Fig. 8.48), where  $j_{dI}^e = j_d^e(-\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}})$ ,  $j_{dII}^e = j_d^e(\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}})$ ,  $j_{dIII}^e = j_d^e(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}})$  and  $j_{dIV}^e = j_d^e(-\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}})$ . It can be easily seen that such a formula of local smoothing represents nothing more than a local scheme to interpolate/extrapolate Gauss point values to nodal point values [251].

Unlike global smoothing, the local smoothing strategy does not produce unique flux values at nodes and therefore an appropriate averaging of the superconvergent flux values is needed. Consider for example the element patch surrounding the particular node at which a unique flux has to be computed as shown in Fig. 8.49. For each element of the patch the superconvergent flux values determined at the Gauss sampling points can be either (1) interpolated/extrapolated to the particular

node by using solutions of (8.412), exemplified for 2D rectangular elements in form of (8.413), or (2) without any interpolation, by a simple assignment (shift) of Gauss-point fluxes nearest to the particular node. The latter strategy is commonly acceptable, in particular for linear elements, because the derivatives usually vary only slightly or are even constant within the element. In doing so, each nodal contribution of elements sharing the particular node is summed up and finally averaged in the following ways. The simplest method is the arithmetic mean for the global node  $i$  in the form

$$\tilde{j}_i = \frac{1}{N_\Sigma} \sum_e^{N_\Sigma} j_i^e \quad (8.414)$$

to determine the smoothed flux  $\tilde{j}_i$  at the node  $i$ , where  $N_\Sigma$  is the number of patch elements surrounding the node  $i$  and  $j_i^e$  is the superconvergent flux of element  $e$  assigned or extrapolated to node  $i$ . Alternatively, as discussed above, an inverse area/volume-weighted averaging can be favorable to attain an improved approximation for more irregularly shaped elements of a patch. It reads

$$\tilde{j}_i = \frac{1}{\sum_e^{N_\Sigma} w^e} \sum_e^{N_\Sigma} w^e j_i^e \quad (8.415)$$

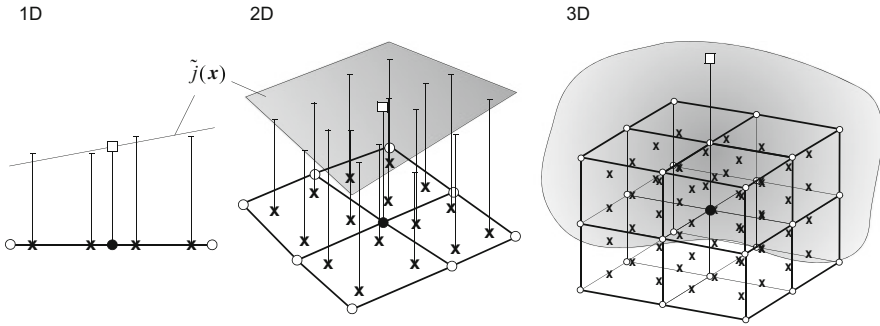
with the weights

$$w^e = \begin{cases} \frac{1}{\Delta x^e} & \text{1D} \\ \frac{1}{A^e} & \text{2D} \\ \frac{1}{V^e} & \text{3D} \end{cases} \quad (8.416)$$

The averaging techniques in combination with local smoothing the superconvergent flux values have proved to be accurate comparable to global smoothing procedures. Instead of nodal averaging, however, an improved method exists in which a polynomial expansion is used on an element patch fitting locally the superconvergent points in a least squares manner, known as superconvergent patch recovery (SPR) to be described next.

### 8.19.1.3 Superconvergent Patch Recovery (SPR)

Zienkiewicz and Zhu [594] have proposed a powerful and accurate method of computing derivatives via a direct polynomial smoothing, which leads to superconvergent flux values at *all* and not only at certain Gauss sampling points within the finite element, called *superconvergent patch recovery* (SPR). In this method a polynomial expansion of the function  $\tilde{j}(\mathbf{x})$  describing the derivatives is used on an element patch surrounding the interelement node at which the nodal derivatives



**Fig. 8.50** Element patches of linear quadrilateral elements in 1D, 2D and 3D. Polynomial expansion of linear function  $\tilde{j}(\mathbf{x})$  describing the derivatives at superconvergent Gauss points  $\times$  is used on the element patch surrounding the node  $\bullet$  at which recovery is desired. Its nodal value  $\square$  is obtained by evaluating the resulting polynomial

have to be determined (recovered). The polynomial is chosen in the same order as occurring in the used finite element approximation of the primary variable, which achieves superconvergent accuracy everywhere if this polynomial is made to fit the superconvergent Gauss sampling points in a least square manner [209, 590]. Let us consider for example the linear polynomial expansion of the derivatives applied to element patches of linear quadrilateral elements in 1D, 2D and 3D as shown in Fig. 8.50. The following three working steps are needed:

1. The derivatives are evaluated at the superconvergent Gauss points  $\times$ .
2. A least-squares fit through the Gauss points is made with a linear polynomial.
3. The superconvergent nodal derivatives  $\square$  are obtained by evaluating the resulting polynomial at the patch node  $\bullet$ .

Having determined the derivatives  $\mathbf{j}(\mathbf{x})$  at the Gauss points according to (8.402) we introduce the linear polynomials for the superconvergent (smooth) derivatives in the form

$$\tilde{\mathbf{j}}(\mathbf{x}) = \begin{cases} \alpha + \beta x & 1D \\ \alpha + \beta x + \chi y + \delta xy & 2D \\ \alpha + \beta x + \chi y + \delta z + \epsilon xy + \phi yz + \gamma zx + \eta xyz & 3D \end{cases} \quad (8.417)$$

where  $\mathbf{x} = (x \ y \ z)^T$  are the Cartesian coordinates and  $\alpha, \beta, \chi, \delta, \epsilon, \phi, \gamma, \eta$  are unknown coefficients to be determined. Note that the mixed terms in (8.417) does not exist for 2D triangular and 3D tetrahedral elements. Now, the method of least-squares is applied to minimize the sum of the squares  $\tilde{\mathbf{j}}(\mathbf{x}_i) - \mathbf{j}(\mathbf{x}_i)$  over all Gauss points  $i = 1, 2, \dots, n$  encountered in the element patch, i.e.,

$$\mathcal{I} = \frac{1}{2} \sum_{i=1}^n [\tilde{\mathbf{j}}(\mathbf{x}_i) - \mathbf{j}(\mathbf{x}_i)]^2 \Rightarrow \min \quad (8.418)$$

where  $n = mN_{\Sigma}$  ( $m$  = number of Gauss points per element,  $N_{\Sigma}$  = number of patch elements). For example, the minimization of  $\mathcal{I} = \frac{1}{2} \sum_{i=1}^n [\alpha + \beta x_i + \chi y_i + \delta x_i y_i - \mathbf{j}(x_i, y_i)]^2$  for the 2D polynomial with respect to the four unknown coefficients yields

$$\begin{aligned} \partial \mathcal{I} / \partial \alpha &= 0 = \sum_i \alpha + \beta x_i + \chi y_i + \delta x_i y_i - \mathbf{j}(x_i, y_i) \\ \partial \mathcal{I} / \partial \beta &= 0 = \sum_i [\alpha + \beta x_i + \chi y_i + \delta x_i y_i - \mathbf{j}(x_i, y_i)] x_i \\ \partial \mathcal{I} / \partial \chi &= 0 = \sum_i [\alpha + \beta x_i + \chi y_i + \delta x_i y_i - \mathbf{j}(x_i, y_i)] y_i \\ \partial \mathcal{I} / \partial \delta &= 0 = \sum_i [\alpha + \beta x_i + \chi y_i + \delta x_i y_i - \mathbf{j}(x_i, y_i)] x_i y_i \end{aligned} \quad (8.419)$$

This leads to the local  $4 \times 4$  linear system ( $\Sigma \equiv \sum_{i=1}^n$ )

$$\begin{pmatrix} n & \sum x_i & \sum y_i & \sum x_i y_i \\ \sum x_i & \sum x_i^2 & \sum x_i y_i & \sum x_i^2 y_i \\ \sum y_i & \sum x_i y_i & \sum y_i^2 & \sum x_i y_i^2 \\ \sum x_i y_i & \sum x_i^2 y_i & \sum x_i y_i^2 & \sum x_i^2 y_i^2 \end{pmatrix} \cdot \begin{pmatrix} \alpha \\ \beta \\ \chi \\ \delta \end{pmatrix} = \begin{pmatrix} \sum \mathbf{j}(x_i, y_i) \\ \sum x_i \mathbf{j}(x_i, y_i) \\ \sum y_i \mathbf{j}(x_i, y_i) \\ \sum x_i y_i \mathbf{j}(x_i, y_i) \end{pmatrix} \quad (8.420)$$

to solve for the polynomial coefficients  $\alpha$ ,  $\beta$ ,  $\chi$  and  $\delta$ . Then, the recovered derivative at an interelement node  $j$  can be easily computed from

$$\tilde{\mathbf{j}}(x_j, y_j) = \alpha + \beta x_j + \chi y_j + \delta x_j y_j \quad (8.421)$$

Similar recovery expressions can be derived for 1D and 3D element patches. The additional numerical cost in SPR is acceptable because the equation system like (8.420) remains small. The total effort usually is smaller than global smoothing and larger than local smoothing, however, in favor of an improved accuracy of the derivatives at the nodes. A robust implementation of SPR requires that the rank of the resulting local equation system, e.g., (8.420), must be equivalent to the number of terms  $a$  used in the polynomial expansion [326]:

$$n \geq a \quad (8.422)$$

where  $a = 2$  in 1D,  $a = 4$  for quadrilateral and  $a = 3$  for triangular elements in 2D and  $a = 8$  for quadrilateral and  $a = 4$  for tetrahedral elements in 3D. Thus, there is a minimal number of elements  $N_{\Sigma}$  in an element patch to make the resulting local equation system solvable. Hence, for linear triangles  $N_{\Sigma}$  has to be greater than or equal to three. To overcome this difficulty in a robust recovery procedure the number of sampling points  $m$  is set at least equal to the number of terms  $a$  in the polynomial expansion regardless of achieving actual superconvergence for the recovered solution.



## 8.19.2 Computing First Derivatives at Exterior or Interior Boundaries: The Consistent Boundary Flux Method (CBFM) and Budget Analysis

### 8.19.2.1 Consistently Derived Boundary Flux Based on Weak Forms

In the previous Sect. 8.19.1 appropriate postprocessing methods for determining fluxes  $\mathbf{j} = -\sum_j \mathbf{D} \cdot \nabla N_j \phi_j$  at nodal points are introduced. Now, we could assume that those nodal fluxes are also suitable to evaluate *boundary* fluxes  $q_n$  in a way such as

$$q_n = -\sum_j \phi_j (\mathbf{D} \cdot \nabla N_j) \cdot \mathbf{n}|_\Gamma \quad (8.423)$$

where  $\mathbf{n}$  is the unit normal vector to the boundary  $\Gamma$  and  $\phi_j$  is the given solution of the primary variable at nodal points  $j$ . Boundary fluxes are needed for evaluating balance quantities in a budget analysis, for example balanced boundary fluxes through exterior Dirichlet-type boundary  $\Gamma_D$  or Cauchy-type boundary section  $\Gamma_C$  of the model domain  $\Omega$  or through interior boundaries  $\Gamma_I$  of subdomains  $\Omega_I$  as part of  $\Omega$  (Fig. 8.51). However, the numerical differentiation in the form (8.423) is not a sufficiently accurate and reasonable expression of a discrete boundary flux because it does not guarantee a proper balance condition at the local position of the boundary. Additionally, (8.423) requires an actual construction of  $\mathbf{n}$ , which is cumbersome and often quite ambiguous if the boundary is not smooth. In a sum, the discrete boundary flux in the form of (8.423) has not the required quality of a locally balanced flux and is accordingly rather inappropriate for any balance evaluation.

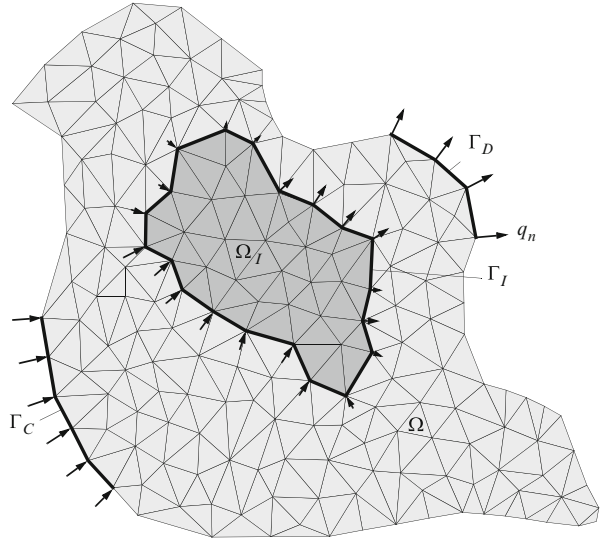
To overcome the difficulties with (8.423) the *consistent boundary flux method* (CBFM) satisfies the requirements for local balance accuracy as suggested by Gresho et al. [213]. It has been shown that CBFM (and related methods) leads to conservative (consistent) flux quantities, e.g., [47, 69, 148, 355, 403]. To obtain a consistent approximation to the boundary flux

$$q_n = \begin{cases} (\phi \mathbf{q} - \mathbf{D} \cdot \nabla \phi) \cdot \mathbf{n}|_\Gamma & \text{for the divergence form of ADE} \\ -\mathbf{D} \cdot \nabla \phi \cdot \mathbf{n}|_\Gamma & \text{for the convective form of ADE} \end{cases} \quad (8.424)$$

we directly utilize the weak statements (8.46) and (8.53) of the governing balance equations for divergence form and convective form, respectively. Applying the Galerkin finite element weighting we find the appropriate weak formulation

$$\begin{aligned} \int_\Gamma N_i q_n d\Gamma &= -\int_\Omega N_i \frac{\partial(\mathcal{R}\phi)}{\partial t} d\Omega + \int_\Omega \phi \mathbf{q} \cdot \nabla N_i d\Omega - \\ &\int_\Omega \nabla N_i \cdot (\mathbf{D} \cdot \nabla \phi) d\Omega - \int_\Omega N_i (\vartheta \phi - H - Q_{\phi w}) d\Omega \end{aligned} \quad (8.425)$$

**Fig. 8.51** Mesh of domain  $\Omega$  with discrete exterior boundary sections  $\Gamma_D$  and  $\Gamma_C$  as well as interior boundary  $\Gamma_I$  enclosing subdomain  $\Omega_I \subset \Omega$



for the divergence form of ADE and

$$\int_{\Gamma} N_i q_n d\Gamma = - \int_{\Omega} N_i \mathcal{R} \frac{\partial \phi}{\partial t} d\Omega - \int_{\Omega} N_i \mathbf{q} \cdot \nabla \phi d\Omega - \int_{\Omega} \nabla N_i \cdot (\mathbf{D} \cdot \nabla \phi) d\Omega - \int_{\Omega} N_i [(\vartheta + \mathcal{Q})\phi - H - \mathcal{Q}_{\phi_w}] d\Omega \quad (8.426)$$

for the convective form of ADE, where the primary variable

$$\phi = \sum_j N_j \phi_j \quad (8.427)$$

is now known from the approximate finite element solution  $\phi_j$  given at each nodal point  $j$  and current time  $t_{n+1}$ . These weak formulations allow a consistent computation of the boundary flux  $q_n$ . In doing so, we expand  $q_n$  in the finite element context as

$$q_n = \sum_j N_j q_{nj} \quad (8.428)$$

where  $q_{nj}$  is the nodal boundary flux to be determined on  $\Gamma$  and at evaluation time  $t_{n+1}$ . Inserting (8.427) and (8.428) into (8.425) and (8.426) the following matrix system results

$$\mathbf{M} \cdot \mathbf{q}_n = -\mathbf{O} \cdot \dot{\phi} - (\mathbf{A} + \mathbf{C} + \mathbf{R}) \cdot \phi + \mathbf{Q} \quad (8.429)$$

where

$$\mathbf{q}_n = q_{nj} = \begin{pmatrix} q_{n1} \\ q_{n2} \\ \vdots \\ q_{nN_p} \end{pmatrix} \quad (8.430)$$

is the nodal vector of the boundary flux and

$$\begin{aligned} \mathbf{M} &= M_{ij} = \sum_e \left( \sum_I \sum_J M_{IJ}^e \Delta_{Ii}^e \Delta_{Jj}^e \right) \\ M_{IJ}^e &= \int_{\Gamma^e} N_I^e N_J^e d\Gamma^e \end{aligned} \quad (8.431)$$

is the boundary mass matrix, which couples  $q_{nj}$  to its nearest neighbors of  $\Gamma$ . The matrices  $\mathbf{O}$ ,  $\mathbf{A}$ ,  $\mathbf{C}$  and  $\mathbf{R}$  as well as the RHS vector  $\mathbf{Q}$  appearing in (8.429) are already given by (8.103)–(8.105). The assembly of (8.429) is done in the usual way at element level, except that only those elements with nodes on  $\Gamma$  need be considered, i.e., the linear matrix system (8.429) is solved only for a subset of  $N_p$  nodes because all contributions to nodes which do not belong to  $\Gamma$  are irrelevant. The linear system (8.429) is solved for the nodal boundary flux  $\mathbf{q}_n$  on  $\Gamma$ , where the RHS of (8.429) is built up with the known solution  $\phi$  and its time derivative  $\dot{\phi}$  at evaluation time  $t_{n+1}$ . We recognize that the CBFM is a strategy in which the ‘forward’ solution system (8.100) is reversely solved on  $\Gamma$ –nodes with known  $\phi$  and  $\dot{\phi}$ . Babuška and Miller [18] have shown that the consistent boundary fluxes exhibit superior convergence behavior, i.e., superconvergence.

*Remark.* The equation (8.429) represents the *consistently derived* flux having the following remarkable properties: (1) If this flux is computed on a Dirichlet boundary  $\Gamma_D$ , it will lead to the same  $\phi$  when imposed as a Neumann-type BC, i.e.,  $\mathbf{q}_n$  and  $\phi$  are equivalent and exchangeable as BC’s. This means that with a known  $\phi$  the domain  $\Omega$  can arbitrarily be subdivided into subdomains  $\Omega_I \subset \Omega$  (Fig. 8.51) forming nonoverlapping interior and/or exterior boundaries of Dirichlet type ( $\phi$  is prescribed there) formed along mesh edges/faces  $\Gamma_I$  on which the consistent boundary flux is computable. (2) The boundary flux guarantees the appropriate approximation to the governing balance equation both globally and locally. The smallest subdomain can be even each single element  $\Omega_I \rightarrow \Omega^e$  so that the boundary flux on  $\Gamma_I \rightarrow \Gamma^e$  also guarantees conservation according to the local balance with (8.429), see Sect. 8.19.3 for further discussion.

### 8.19.2.2 Lumped Solution

To avoid the solution of the linear system (8.429) the cost-effective alternative is to invoke mass lumping for  $\mathbf{M}$ , i.e.,

$$\mathbf{M} = M_{ij} = \delta_{ij} \int_{\Gamma} N_i d\Gamma \quad (8.432)$$

With mass lumping (8.432) the nodal boundary fluxes in (8.429) become uncoupled and can be explicitly computed from

$$\mathbf{q}_n = -\mathbf{M}^{-1} \cdot [\mathbf{O} \cdot \dot{\phi} + (\mathbf{A} + \mathbf{C} + \mathbf{R}) \cdot \phi - \mathbf{Q}] \quad (8.433)$$

### 8.19.2.3 Integral Boundary Flux

Alternatively to (8.433), we can simply sum up the contributions of the system for each row  $i$  to obtain the *integral* boundary balance flux at the boundary node  $i$

$$\begin{aligned} Q_{ni} &= - \int_{\Gamma} N_i q_n d\Gamma \\ &= - \sum_j M_{ij} q_{nj} \\ &= \sum_j [O_{ij} \dot{\phi}_j + (A_{ij} + C_{ij} + R_{ij}) \phi_j - Q_i], \quad (j = 1, \dots, N_P) \end{aligned} \quad (8.434)$$

or in matrix form

$$\begin{aligned} \mathbf{Q}_n &= -\mathbf{M} \cdot \mathbf{q}_n \\ &= \mathbf{O} \cdot \dot{\phi} + (\mathbf{A} + \mathbf{C} + \mathbf{R}) \cdot \phi - \mathbf{Q} \end{aligned} \quad (8.435)$$

where the sign of  $Q_{ni} = \mathbf{Q}_n$  is used in accordance with the definitions of well-type SPC terms (cf. Sect. 6.3), i.e., a positive  $\mathbf{Q}_n$  corresponds to a point sink. The integral boundary flux  $\mathbf{Q}_n = -\mathbf{M} \cdot \mathbf{q}_n$  is used in a *budget analysis* in which the balance quantities on boundaries  $\Gamma$  are determined at evaluation time  $t_{n+1}$ . It is also required in constraint formulations for BC's (see Sect. 6.4).

### 8.19.2.4 Auxiliary Problem Formulation for Convective Form of ADE

In use of the convective form of ADE the boundary flux is dispersion/diffusion-controlled  $q_n^d = -\mathbf{D} \cdot \nabla \phi \cdot \mathbf{n}|_{\Gamma}$  (8.424) according to the basic weak statement. For a budget analysis it is also desired to quantify the missing advective part of a boundary flux  $q_n^a = \phi \mathbf{q} \cdot \mathbf{n}|_{\Gamma}$ , where  $\mathbf{q}$  is the advective flux. We recall that the convective form of ADE results from the substitution of mass conservation, cf. (3.45). To obtain the total boundary flux  $q_n = q_n^a + q_n^d$  we have to retrieve the substituted mass conservation via an auxiliary weak formulation. Let us consider for example the mass conservation equation given in Table 3.7 and multiplying all terms by  $\phi$ . It results

$$\phi \left( s S_o \frac{\partial h}{\partial t} + \varepsilon \frac{\partial s}{\partial t} \right) + \phi \nabla \cdot \mathbf{q} = \phi (Q + Q_{\text{EOB}}) \quad (8.436)$$

Its weak statement reads

$$\int_{\Omega} w \phi \nabla \cdot \mathbf{q} d\Omega = \int_{\Omega} w \phi (Q + Q_{\text{EOB}}) d\Omega - \int_{\Omega} w \phi \left( s S_o \frac{\partial h}{\partial t} + \varepsilon \frac{\partial s}{\partial t} \right) d\Omega \quad (8.437)$$

where  $h$ , the hydraulic head, and  $s$ , the saturation, are another primary variables which are assumed to be known from a separate finite element solution of the flow equation. Using the product rule of differentiation

$$\nabla \cdot (w \phi \mathbf{q}) = \phi \mathbf{q} \cdot \nabla w + w \phi \nabla \cdot \mathbf{q} + w \mathbf{q} \cdot \nabla \phi \quad (8.438)$$

and employing the Gauss's integral theorem (2.77) on the LHS term of (8.438) we obtain from (8.437)

$$\begin{aligned} \int_{\Gamma} w \phi \mathbf{q} \cdot \mathbf{n} d\Gamma &= \int_{\Omega} \phi \nabla w \cdot \mathbf{q} d\Omega + \int_{\Omega} w \nabla \phi \cdot \mathbf{q} d\Omega + \\ &\int_{\Omega} w \phi (Q + Q_{\text{EOB}}) d\Omega - \int_{\Omega} w \phi \left( s S_o \frac{\partial h}{\partial t} + \varepsilon \frac{\partial s}{\partial t} \right) d\Omega \end{aligned} \quad (8.439)$$

Now, using the Galerkin weak formulation  $w \rightarrow N_i$ , invoking the Darcy law to express the flow vector as  $\mathbf{q} = -k_r \mathbf{K} f_{\mu} \cdot (\nabla h + \chi \mathbf{e})$  (cf. Table 3.7) and expanding the known variables  $\phi = \sum_j N_j \phi_j$ ,  $h = \sum_j N_j h_j$  and  $s = \sum_j N_j s_j$  in the finite element context, we find

$$\begin{aligned} \int_{\Gamma} N_i q_n^a d\Gamma &= - \sum_j \int_{\Omega} \nabla N_i \cdot [k_r \mathbf{K} f_{\mu} \cdot (\nabla N_j + \chi \mathbf{e})] h_j \left( \sum_l N_l \phi_l \right) d\Omega - \\ &\sum_j \int_{\Omega} N_i \nabla N_j \phi_j \cdot \left[ \sum_l k_r \mathbf{K} f_{\mu} \cdot (\nabla N_l h_l + \chi \mathbf{e}) \right] d\Omega + \\ &\int_{\Omega} N_i \left( \sum_l N_l \phi_l \right) (Q + Q_{\text{EOB}}) d\Omega - \\ &\int_{\Omega} N_i \left( \sum_l N_l \phi_l \right) \left[ \left( \sum_l N_l s_l \right) S_o \left( \sum_l N_l \frac{\partial h_l}{\partial t} \right) + \varepsilon \left( \sum_l N_l \frac{\partial s_l}{\partial t} \right) \right] d\Omega \end{aligned} \quad (8.440)$$

or with expanding  $q_n^a = \sum_j N_j q_{nj}^a$

$$\mathbf{M} \cdot \mathbf{q}_n^a = -\mathbf{U}(\phi) \cdot \mathbf{h} - \mathbf{V}(\mathbf{h}) \cdot \phi + \mathbf{X}(\phi) - \mathbf{Y}(\phi, \mathbf{s}, \dot{\mathbf{h}}, \dot{\mathbf{s}}) \quad (8.441)$$

to solve the advective boundary flux vector  $\mathbf{q}_n^a$  at evaluation time  $t_{n+1}$ , where the matrices  $\mathbf{U}$  and  $\mathbf{V}$  are related to the 1st and 2nd RHS-terms and the vectors  $\mathbf{X}$

and  $\mathbf{Y}$  are related to the 3rd and 4th RHS-terms of (8.440). It is assumed that the solutions  $\phi$ ,  $\mathbf{h}$ ,  $\mathbf{s}$  and the time derivatives  $\dot{\mathbf{h}}$  and  $\dot{\mathbf{s}}$  are known. Dependence of the solution vectors  $\phi$ ,  $\mathbf{h}$ ,  $\mathbf{s}$ ,  $\dot{\mathbf{h}}$  and/or  $\dot{\mathbf{s}}$  in  $\mathbf{U}$ ,  $\mathbf{V}$ ,  $\mathbf{X}$  and  $\mathbf{Y}$  are shown in parentheses. Finally, we can combine (8.429) and (8.441) to find the expression for solving the total consistent boundary flux  $\mathbf{q}_n = \mathbf{q}_n^a + \mathbf{q}_n^d$  in the form

$$\mathbf{M} \cdot \mathbf{q}_n = -\mathbf{O} \cdot \dot{\phi} - [\mathbf{A} + \mathbf{C} + \mathbf{R} + \mathbf{V}(\mathbf{h})] \cdot \phi - \mathbf{U}(\phi) \cdot \mathbf{h} + \mathbf{X}(\phi) - \mathbf{Y}(\phi, \mathbf{s}, \dot{\mathbf{h}}, \dot{\mathbf{s}}) + \mathbf{Q} \quad (8.442)$$

associated with the convective form of ADE.

### 8.19.2.5 Illustrative Example

To clarify the consistent flux method let us consider a simple, however, quite representative and illustrative example [213]: A steady-state diffusion problem with a varying source in one dimension  $x$ . The corresponding basic PDE is

$$-\nabla^2 \phi = H(x), \quad 0 \leq x \leq 3 \quad (8.443)$$

which has to be solved for  $\phi = \phi(x)$  subject to the BC's

$$\phi = \begin{cases} 0 & \text{at } x = 0 \quad \text{and} \\ 0 & \text{at } x = 3 \end{cases} \quad (8.444)$$

and with the source function

$$H(x) = \begin{cases} 0 & \text{for } 0 \leq x < 2 \quad \text{and} \\ 6 & \text{for } 2 \leq x \leq 3 \end{cases} \quad (8.445)$$

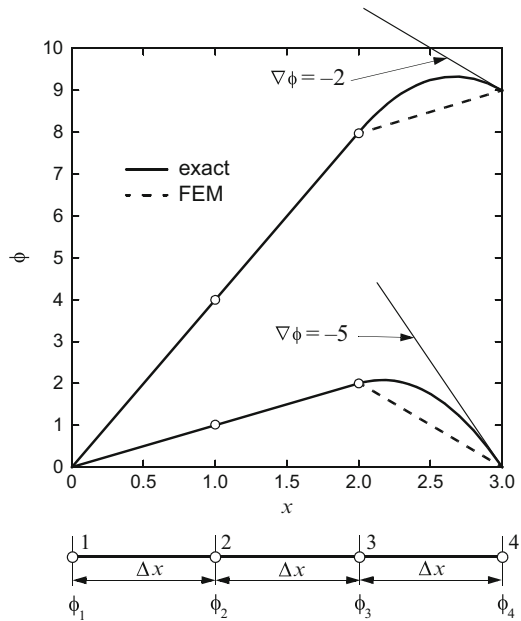
The exact solution is

$$\phi(x) = \begin{cases} x & \text{for } 0 \leq x \leq 2 \\ -3x^2 + 13x - 12 & \text{for } 2 \leq x \leq 3 \end{cases} \quad (8.446)$$

which is plotted as the lower solid curve in Fig. 8.52. The *exact* boundary flux  $q_n = -\nabla \phi \cdot \mathbf{n} = -\partial \phi / \partial x|_r$  through the outer boundary at  $x = 3$  can be simply derived from (8.446) as  $q_n = 5$ .

The problem is approximated by using just three linear elements, each of unit length  $\Delta x = 1$  (Fig. 8.52). The finite element discretization leads to the following matrix system (see Appendix H.1, note that incoming and outgoing gradients are canceled at interior element boundaries due to their opposite signs):

**Fig. 8.52** Steady diffusion problem in one dimension



$$\frac{1}{\Delta x} \begin{pmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{pmatrix} \cdot \begin{pmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \end{pmatrix} = \begin{pmatrix} q_n \\ 0 \\ H \frac{\Delta x}{2} \\ H \frac{\Delta x}{2} - q_n \end{pmatrix} \quad (8.447)$$

The BC's (8.444) giving  $\phi_1 = \phi_4 = 0$  are incorporated in (8.447), cf. Sect. 8.16. Then, the following discrete equations result

$$\frac{1}{\Delta x}(2\phi_2 - \phi_3) = 0 \quad (8.448)$$

and

$$\frac{1}{\Delta x}(-\phi_2 + 2\phi_3) = H \frac{\Delta x}{2} \quad (8.449)$$

Hence, with  $\phi_2 = 1$  and  $\phi_3 = 2$  the finite element solution is exact at the nodes (see the dashed line in the lower curve of Fig. 8.52).

Now, suppose that the flux  $q_n$  at the boundary  $x = 3$  is desired. If the conventional nodal flux evaluation according to (8.423) is employed, we find (cf. Appendix H.1)

$$q_n = -\frac{1}{\Delta x}(-\phi_3 + \phi_4) = 2 \quad (8.450)$$

which is different to the exact solution of 5. In contrast to (8.450), the consistent boundary flux results from (8.447) (resolving the last row for  $q_n$  with the given  $\phi$ -solution):

$$q_n = H \frac{\Delta x}{2} - \frac{1}{\Delta x}(-\phi_3 + \phi_4) = 5 \quad (8.451)$$

which agrees with the exact solution. It is obvious that although (8.450) is in fact the true slope of the approximate solution, the CBFM solution (8.451) yields the correct balanced flux, which properly accounts for both the source term in the finite element and the diffusion at  $x = 3$ . We can even prove the local balance for each element if we solve the consistent boundary flux for a separate element. Consider, for example, the last element  $e$ , ( $2 \leq x \leq 3$ ):

$$\frac{1}{\Delta x^e} \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \cdot \begin{pmatrix} \phi_3 \\ \phi_4 \end{pmatrix} = \begin{pmatrix} H^e \frac{\Delta x^e}{2} \\ H^e \frac{\Delta x^e}{2} \end{pmatrix} - \begin{pmatrix} -q_n^- \\ q_n^+ \end{pmatrix} \quad (8.452)$$

It yields  $q_n^- = -1$  and  $q_n^+ = 5$  as left-sided and right-sided boundary flux of the element, respectively. Thus, the element balance is exactly satisfied with

$$H^e \Delta x^e + q_n^- - q_n^+ = 6 - 1 - 5 = 0 \quad (8.453)$$

Finally, to demonstrate consistency in the finite element solutions, let us solve the problem by using a Neumann-type BC  $q_N$  at  $x = 3$ , where we use the derived value for  $q_n$  from (8.450):  $q_N = q_n = 2$ . The corresponding nodal equations are for this case

$$\begin{aligned} \frac{1}{\Delta x}(2\phi_2 - \phi_3) &= 0 \\ \frac{1}{\Delta x}(-\phi_2 + 2\phi_3 - \phi_4) &= H \frac{\Delta x}{2} \\ \frac{1}{\Delta x}(-\phi_3 + \phi_4) &= H \frac{\Delta x}{2} - q_N \end{aligned} \quad (8.454)$$

The solution to (8.454) for  $q_N = 2$  is  $\phi_2 = 4$ ,  $\phi_3 = 8$  and  $\phi_4 = 9$ , which is displayed as the dashed upper curve in Fig. 8.52 in comparison to the exact solution for  $q_N = 2$  given as  $\phi = 4x$  for  $0 \leq x \leq 2$  and  $\phi = -3x^2 + 16x - 12$  for  $2 \leq x \leq 3$ . On the other hand, using  $q_N = 5$  from (8.451) we retrieve the original result as  $\phi_2 = 2$ ,  $\phi_3 = 3$  and  $\phi_4 = 0$ . Thus, it demonstrates that only the consistently derived flux can be applied as a natural BC to recover the original solution obtained with Dirichlet-type BC's. Although we have shown only a 1D problem, the same essential issues are given in multidimensional and transient problems [209, 213, 277].

### 8.19.3 Continuous Finite Element Approach Is Locally Conservative

The basic model equations which are solved via approximate methods represent balance laws for conserving physical quantities such as mass, momentum and energy.

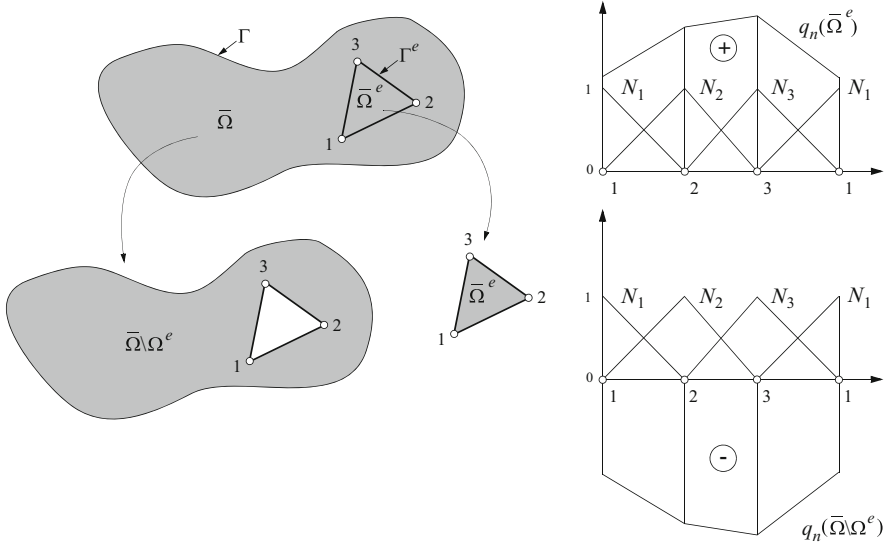


Accordingly, the used numerical approach should also respect these conservation equations both globally and locally. *Conservativity* (see definition in Sect. 1.2.2) enforces that incoming and outgoing fluxes through interior and exterior boundaries of a global domain and its subdivided subdomains have to be conserved and consistent with the source/sink and storage effects occurring in the balance volumes, otherwise the method is nonconservative which can produce artificial sources and sinks, changing the balance both locally and globally. Nonconservative methods are to be declined to avoid erroneous solutions.

*Local conservativity* means that conservation is guaranteed for each of the smallest discrete unit, i.e., for each element (or cell), regardless of mesh (grid) size. However, local conservativity does not mean local accuracy. The problem solution may be inaccurate, but will, nevertheless, be conservative. Repeatedly, it is believed that finite element methods are not locally conservative. But, this is a misbelief (and in part a strange discrediting of FEM) which has been refuted in a number of papers, see e.g., [47, 89, 148, 277, 355]. Obviously, there is a misunderstanding on both the basic conservation law structure of the FEM and the computation of local fluxes. A major reason is apparently in the misuse of nodal derivatives in form of (8.402) or (8.423) as balance fluxes. Indeed, those nonconsistent fluxes obtained from a numerical differentiation are not necessarily conservative and can cause significant local balance errors [47, 148, 355, 578]. The simple example of Sect. 8.19.2.5 has evidently shown the importance of a suitable flux computation for balance evaluations.

The present continuous finite element approach is based on an elementwise *continuous* approximation, see Sect. 8.7. It guarantees continuity up to first derivatives (fluxes) even at element interfaces. Having this property for  $C_0$  continuous basis functions, the subdivision of the global integrals into subdomains, elements and subboundaries can be done via (8.62) without any interelement residual. As a consequence, fluxes between adjacent elements cancel since the flux is contained within the element, while fluxes exposed on the external boundary do not. With other word, the fluxes appears only on external (global) boundaries, while fluxes interchanging between adjacent elements remain hidden during the usual computation. However, we can evaluate this type of flux as consistent boundary flux  $q_n$ . In Sect. 8.19.2 the CBFM is described which provides precise boundary fluxes at any exterior or interior boundaries of a meshed domain coinciding with the element edges or faces. If the external boundary  $\Gamma$  is used, the consistent boundary fluxes determine the global conservation of the domain  $\bar{\Omega} = \Omega \cup \Gamma$ , if the boundary  $\Gamma_I$  refers to a subdomain  $\Omega_I \subset \Omega$ , such as illustrated in Fig. 8.51, the boundary flux measures the exchange between the adjacent subdomain and accordingly determines the conservation of the subdomain  $\bar{\Omega}_I = \Omega_I \cup \Gamma_I$ , and finally if the boundary is even chosen as the element boundary  $\Gamma^e$ , the resulting boundary flux measures the conservation of the single element  $\bar{\Omega}^e = \Omega^e \cup \Gamma^e$  (Fig. 8.53).

Now, we can utilize the weak formulations (8.425) and (8.426) to find the boundary flux  $q_n^e$  of each single element  $e$  in form of *element conservation laws*:



**Fig. 8.53** Consistent boundary flux  $q_n(\bar{\Omega}^e)$  of element  $\bar{\Omega}^e$  in equilibrium with boundary flux  $q_n(\bar{\Omega} \setminus \Omega^e)$  of subdomain  $\bar{\Omega} \setminus \Omega^e$  (Modified from [277])

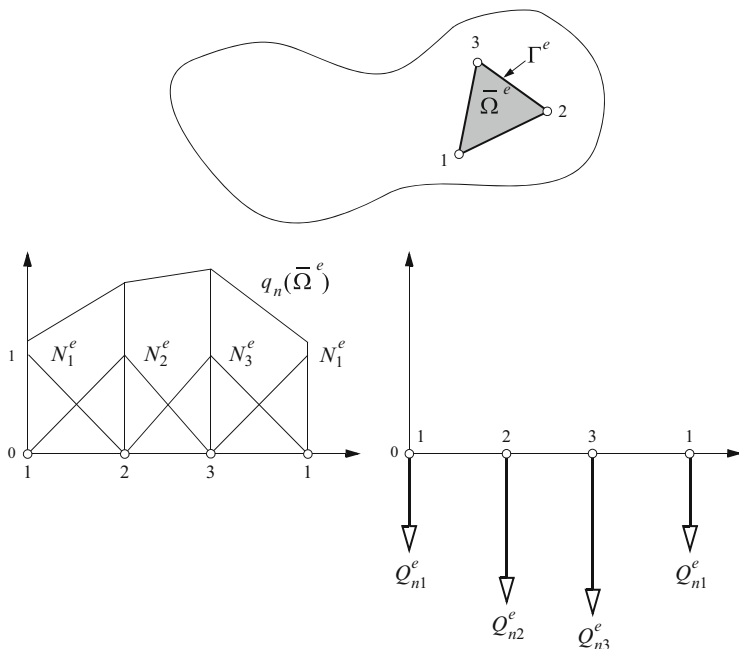
$$\int_{\Gamma^e} N_I^e q_n^e d\Gamma^e = - \int_{\Omega^e} N_I^e \frac{\partial(\mathcal{R}^e \phi^e)}{\partial t} d\Omega^e + \int_{\Omega^e} \phi^e \mathbf{q}^e \cdot \nabla N_I^e d\Omega^e - \int_{\Omega^e} \nabla N_I^e \cdot (\mathbf{D}^e \cdot \nabla \phi^e) d\Omega^e - \int_{\Omega^e} N_I^e (\vartheta^e \phi^e - H^e - Q_{\phi_w}^e) d\Omega^e \quad (8.455)$$

for the divergence form of ADE and

$$\int_{\Gamma^e} N_I^e q_n^e d\Gamma^e = - \int_{\Omega^e} N_I^e \mathcal{R}^e \frac{\partial \phi^e}{\partial t} d\Omega^e - \int_{\Omega^e} N_I^e \mathbf{q}^e \cdot \nabla \phi^e d\Omega^e - \int_{\Omega^e} \nabla N_I^e \cdot (\mathbf{D}^e \cdot \nabla \phi^e) d\Omega^e - \int_{\Omega^e} N_I^e [(\vartheta^e + Q^e) \phi^e - H^e - Q_{\phi_w}^e] d\Omega^e \quad (8.456)$$

for the convective form of ADE, where  $I = 1, \dots, N_{\text{BN}}$ . Similar to (8.434) and (8.435), respectively, we can summarize (8.455) and (8.456) as follows

$$\begin{aligned} Q_{nt}^e &= - \int_{\Gamma^e} N_I^e q_n^e d\Gamma^e \\ &= - \sum_J M_{IJ}^e q_{nJ}^e \\ &= \sum_J [O_{IJ} \dot{\phi}_J^e + (A_{IJ}^e + C_{IJ}^e + R_{IJ}^e) \phi_J^e - Q_I], \quad (J = 1, \dots, N_{\text{BN}}) \end{aligned} \quad (8.457)$$



**Fig. 8.54** Element conservation: The consistent boundary flux  $q_n(\bar{\Omega}^e)$  is the conservative redistribution of the integral element flux  $Q_{nl}^e$  in terms of the element basis functions  $N_I^e, I = 1, \dots, N_{BN}$  (Modified from [277])

or in matrix form

$$\begin{aligned}
 \mathbf{Q}_n^e &= -\mathbf{M}^e \cdot \mathbf{q}_n^e \\
 &= \mathbf{O}^e \cdot \dot{\phi}^e + (\mathbf{A}^e + \mathbf{C}^e + \mathbf{R}^e) \cdot \phi^e - \mathbf{Q}^e
 \end{aligned}
 \tag{8.458}$$

with

$$\mathbf{M}_{IJ}^e = \int_{\Gamma^e} N_I^e N_J^e d\Gamma^e
 \tag{8.459}$$

where  $\mathbf{q}_n^e$  is the element boundary flux and  $\mathbf{Q}_n^e = \mathbf{O}^e \cdot \dot{\phi}^e + (\mathbf{A}^e + \mathbf{C}^e + \mathbf{R}^e) \cdot \phi^e - \mathbf{Q}^e$  is the integral element flux (Fig. 8.54). By summing (8.457) over  $I = 1, \dots, N_{BN}$ , we see that

$$\int_{\Gamma^e} q_n^e d\Gamma^e + \sum_I Q_{nl}^e = 0
 \tag{8.460}$$

which represents the element conservation. Thus, the sum of the integral element fluxes is a conserved quantity. The corresponding element boundary flux  $q_n^e$  of element  $\bar{\Omega}^e$  is in equilibrium with the boundary fluxes of the adjacent complementary subdomain  $\bar{\Omega} \setminus \bar{\Omega}^e$  (see Fig. 8.53), viz.,

$$q_n(\bar{\Omega}^e) = -q_n(\bar{\Omega} \setminus \Omega^e) \quad (8.461)$$

It becomes clear that the nodal fluxes  $Q_{nl}^e$  and their continuous redistribution  $q_n^e$  in terms of the element basis functions are different but equivalent representations of the same information [277], viz.,

$$\sum_J \int_{\Gamma^e} N_I^e N_J^e q_{nJ}^e d\Gamma^e = -Q_{nl}^e \quad (8.462)$$

$$Q_{nl}^e = - \int_{\Gamma^e} N_I^e q_n^e d\Gamma^e \quad (8.463)$$

Once  $Q_{nl}^e$  is known,  $q_n^e = \sum_J N_J^e q_{nJ}^e$  is uniquely defined by (8.462). Likewise, if  $q_n^e = q_n(\bar{\Omega}^e)$  is known, the nodal fluxes  $Q_{nl}^e$  are uniquely defined by (8.463). These quantities are fundamental to the local conservativity of the continuous FEM.

### 8.19.4 Note on Mixed Finite Element Formulations

So far we have considered the basic balance equation in a form in which the governing flux  $\mathbf{j} = -\mathbf{D} \cdot \nabla \phi$  has been suitably substituted so that only one unknown function  $\phi$ , the primary variable, remains in the scalar governing equations (8.3) or (8.5). This elimination of  $\mathbf{j}$  leads to a mathematically well-defined problem with appropriate BC's expressed in terms of  $\phi$  or its gradients (8.4) or (8.6). In the FEM context it leads to an approximation of only one unknown variable  $\phi_i$  per node  $i$  of a mesh, i.e., degrees of freedom are  $N_{\text{DOF}} = 1$ , and the resulting matrix system becomes usually easily solvable. However, in this approach the required knowledge of the secondary variable in form of the flux  $\mathbf{j}$  must be obtained as a derived quantity, which naturally implies a loss of accuracy compared to the accuracy attainable for the primary variable, notwithstanding the precise evaluation techniques for deriving  $\mathbf{j}$  such as described in the preceding Sect. 8.19.1.

The FEM does not restrict per se the formulation to governing equations in which the flux is eliminated. It is also possible to refer to a formulation where both  $\phi$  and  $\mathbf{j}$  are chosen as primary variables. This is called as a *mixed finite element formulation*, e.g., [56, 84, 436, 590]. Mixed finite element methods are inevitable in CFD for solving the coupled system of Navier-Stokes equations [209], where the elimination of fluxes (velocities) from the basic equations is not possible or restricted. This is quite different to Darcy-based flow equations in porous media, where a mixed formulation appears as a useful but commonly nonessential alternative [75, 152, 378]. To illustrate the mixed finite element formulation for the present class of problems, let us write the governing ADE (8.5) in the alternative form as

$$\begin{aligned} \mathcal{R} \frac{\partial \phi}{\partial t} + \mathbf{q} \cdot \nabla \phi + \nabla \cdot \mathbf{j} + (\vartheta + Q)\phi &= H + Q_{\phi w} \\ \mathbf{j} &= -\mathbf{D} \cdot \nabla \phi \end{aligned} \quad (8.464)$$

and introduce the finite element approximation for both primary variables  $\phi$  and  $\mathbf{j}$  as (cf. (8.16))

$$\begin{aligned} \phi(\mathbf{x}, t) &= \sum_j N_j(\mathbf{x}) \phi_j(t) \\ \mathbf{j}(\mathbf{x}, t) &= \sum_l M_l(\mathbf{x}) \mathbf{j}_l(t) \end{aligned} \quad (8.465)$$

where  $N_j$  and  $M_d$  represent basis functions at global nodes  $j$  and  $d$ , respectively, which must not coincide, and  $\phi_j$  and  $\mathbf{j}_d$  are the corresponding nodal vectors of the unknowns  $\phi$  and  $\mathbf{j}$ , respectively. Now, taking (8.465) and applying the GFEM to (8.464), in which the weighting functions are in accordance with the basis functions, we can find the following matrix system

$$\begin{pmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{C}^\dagger & \mathbf{0} \end{pmatrix} \cdot \begin{pmatrix} \phi \\ \mathbf{J} \end{pmatrix} = \begin{pmatrix} \mathbf{F} \\ \mathbf{0} \end{pmatrix} \quad (8.466)$$

to solve simultaneously  $\phi = \phi_j$  and  $\mathbf{J} = \mathbf{j}_d$ , where

$$\begin{aligned} \mathbf{A} = A_{ij} &= \sum_e \left( \int_{\Omega^e} \mathcal{R}^e N_i N_j \frac{\partial}{\partial t} d\Omega^e + \int_{\Omega^e} N_i \mathbf{q}^e \cdot \nabla N_j d\Omega^e + \right. \\ &\quad \left. \int_{\Omega^e} (\vartheta^e + Q^e) N_i N_j d\Omega^e + \int_{\Gamma_C^e} \Phi^e N_i N_j d\Gamma^e \right) - \\ &\quad \delta_{ij} Q_w(t) \Big|_i \\ \mathbf{C} = C_{id} &= \sum_e \int_{\Omega^e} N_i \nabla M_d d\Omega^e \\ \mathbf{C}^\dagger = C_{lj}^\dagger &= - \sum_e \int_{\Omega^e} M_l \mathbf{D} \cdot \nabla N_j d\Omega^e \\ \mathbf{F} = F_i &= \sum_e \left( \int_{\Omega^e} N_i H^e d\Omega^e + \int_{\Gamma_C^e} N_i \Phi^e \phi_C^e d\Gamma^e - \int_{\Gamma_N^e} N_i q_N^e d\Gamma^e \right) - \\ &\quad \phi_w Q_w(t) \Big|_i \end{aligned} \quad (8.467)$$

in which the indices  $i, j$  and  $l, d$ , respectively, run over the same nodal points. The mixed finite element formulation in the form of (8.467) includes the following properties:

1. The simultaneous solution of  $\phi$  and  $\mathbf{J}$  leads to a higher accuracy of the flux  $\mathbf{J}$  compared to the standard formulation (at the same mesh resolution) in which  $\mathbf{J}$  has been eliminated and appears as secondary variable. Indeed,  $\mathbf{J}$  resulting from the mixed formulation satisfies implicitly local conservativity.

2. The higher accuracy of  $\mathbf{J}$  must be paid by a significant increase in the computational effort because the increased degrees of freedom (e.g.,  $N_{\text{DOF}} = 4$  in 3D) considerably enlarge the final equation system to be solved.
3. The resulting matrix system (8.466) forms a saddle point problem, where the total matrix is not positive definite. It can lead to difficulties in the solving the equations.
4. The formulation of BC's for the flux is restricted.
5. The mixed interpolation for  $\phi$  and  $\mathbf{j}$  must satisfy a compatibility condition, known as LBB (Ladyshenkaya-Babuška-Brezzi) condition, see e.g., [56, 149, 209], otherwise the mixed formulation does not guarantee stability. The basis functions  $N_j$  and  $M_d$  are differently chosen. Once subjected to 1st-order derivatives they have to be  $C_0$  continuous functions, otherwise no continuity is needed. A well-known stable element is the Taylor-Hood element [209], in which the flux  $\mathbf{j}$  is interpolated quadratically and the scalar variable  $\phi$  is interpolated by a linear continuous function such as used by Diersch [130] in free convection flow modeling in porous media. Other useful stable elements are discussed in [30, 56, 149, 209, 436], where the Raviart-Thomas element [75] appears suitable for the present class of porous-media problems [31, 46, 359, 476].

The mixed finite element formulation offered the possibility for obtaining a potentially higher accuracy in the flux computations compared to a standard formulation (simulated at the same mesh resolution), however, at a significant increase in the computational effort and at the expense of a reduced robustness and flexibility. This limits the mixed FEM to only specific (often academic, small-size) problems. In practical modeling of flow and transport processes in porous and fractured media, mixed finite element formulations are neither feasible nor necessary in general. Indeed, the accuracy of fluxes achieved from the standard formulations by using the derived quantity evaluations as discussed in Sect. 8.19.1 are usually able to provide equivalently precise flux computations. In the following, we need not to resort to mixed finite element formulations.