Chapter 1
Introduction

1.1 A model problem

In this first chapter, we introduce the basic principles of adaptivity and moving mesh methods for solving partial differential equations in one spatial dimension. In particular, two adaptive methods are described and used to solve a simple model problem – an initial-boundary value problem consisting of Burgers' equation

$$u_t = \varepsilon u_{xx} - \left(\frac{u^2}{2}\right)_x, \quad x \in (0,1),\ t > 0 \tag{1.1}$$

subject to the boundary conditions

$$u(0,t) = u(1,t) = 0 \tag{1.2}$$

and initial condition

$$u(x,0) = \sin(2\pi x) + \frac{1}{2}\sin(\pi x). \tag{1.3}$$

Here, $\varepsilon > 0$ is a physical parameter. For small ε, the solution has a smooth initial profile and develops a steep front. The front propagates toward the right end and eventually dies out due to the homogeneous Dirichlet boundary condition at $x = 1$. The difficulty with the numerical solution of the problem lies in the resolution of this propagating steep front.

W. Huang and R.D. Russell, *Adaptive Moving Mesh Methods*, Applied Mathematical Sciences 174, DOI 10.1007/978-1-4419-7196-2_1, © Springer Science+Business Media, LLC 2011

1.2 A moving finite difference method

1.2.1 Finite difference method on a fixed mesh

The first numerical method we consider is a standard method of lines for the partial differential equation (PDE) (1.1). Specifically, the PDE is first discretized in the spatial domain, and then the resulting system of ordinary differential equations (ODEs) is integrated using an ODE solver. The main advantage of the method of lines is the separate treatments of the spatial and temporal components of the PDE, so that attention can be focused on each of them in turn.

To illustrate, we consider the finite difference solution of the model problem on a uniform spatial mesh. Given a positive integer N, define the mesh

$$\mathscr{T}_h: \quad x_j = (j-1)h, \quad j = 1,...,N \tag{1.4}$$

where $h = 1/(N-1)$. A semi-discretization of Burgers' equation (1.1) using central finite differences in space is given by

$$\frac{du_j}{dt} = \frac{\varepsilon}{h^2}\left(u_{j+1} - 2u_j + u_{j-1}\right) - \frac{1}{4h}\left(u_{j+1}^2 - u_{j-1}^2\right), \qquad j = 2,...,N-1 \tag{1.5}$$

where $u_j(t)$ is an approximation to the solution $u = u(x,t)$ at $x = x_j$, i.e., $u_j(t) \approx u(x_j,t)$. The discrete boundary and initial conditions become

$$u_1(t) = 0, \quad u_N(t) = 0, \quad t > 0 \tag{1.6}$$

$$u_j(0) = \sin(2\pi x_j) + \frac{1}{2}\sin(\pi x_j), \quad j = 1,...,N. \tag{1.7}$$

The boundary conditions (1.6) are replaced in the actual implementation by the ODE form

$$\frac{du_1}{dt} = 0, \quad \frac{du_N}{dt} = 0. \tag{1.8}$$

The equations (1.5) and (1.8), with initial conditions (1.7), constitute an initial value problem which can in principle be conveniently integrated using an ODE solver.

Figure 1.1 (a) and (b) show computed solutions for Burgers' equation with $\varepsilon = 10^{-2}$ using this approach. The results are obtained using the Matlab ODE solver "ode15i," which is based on the backward differentiation formulas (BDFs) of orders 1–5 [296, 297, 298]. Note that with a uniform mesh of 21 points, oscillations in the computed solution are visible, indicating that the steep front is not adequately resolved on the uniform mesh. These oscillations can be eliminated by using a finer mesh of 81 equidistant points, as shown in Figure 1.1(b).

When ε is smaller, a correspondingly finer mesh has to be used to resolve the steep front. This is illustrated in Figure 1.2, where oscillations are still visible in the

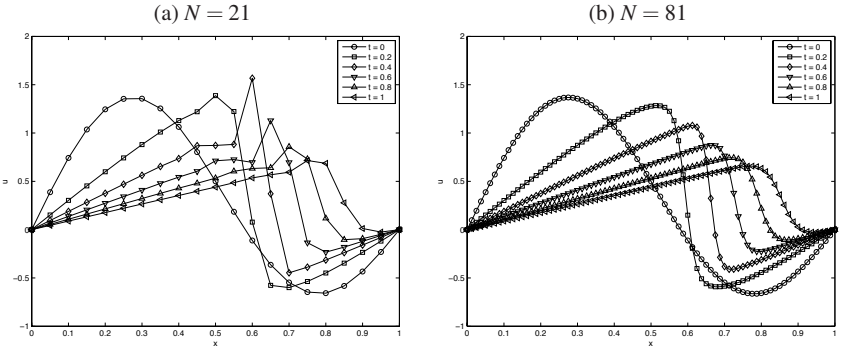

Fig. 1.1 Computed solutions obtained with a uniform mesh for Burgers' equation with $\varepsilon = 10^{-2}$ are shown at $t = 0,\ 0.2,\ 0.4,\ 0.6,\ 0.8$, and 1.0.

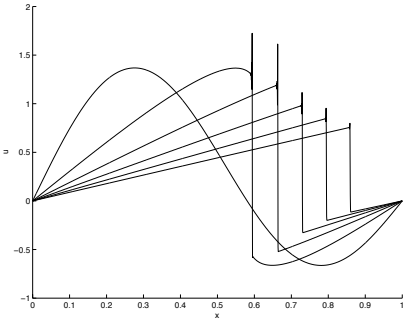

Fig. 1.2 Computed solutions at $t = 0,\ 0.2,\ 0.4,\ 0.6,\ 0.8$, and 1.0 obtained with a uniform mesh of 2001 points for Burgers' equation with $\varepsilon = 10^{-4}$.

solution computed with a uniform mesh of 2001 points for the case $\varepsilon = 10^{-4}$. Use of a very fine uniform mesh is expensive in terms of computer time and memory, and much more so for two- and three-dimensional problems, making mesh adaptation necessary.

1.2.2 Finite difference method on an adaptive moving mesh

The adaptive solution of the model problem requires that the mesh points be concentrated around the steep front and dynamically adjusted to follow the front as it propagates in time. Such a dynamically adjusting mesh is referred to as an *adaptive moving mesh*.

Adaptive mesh movement is often best understood by interpreting the problem in terms of a suitable coordinate transformation. Specifically, we assume for the

moment that a time-dependent coordinate transformation $x = x(\xi,t) : \Omega_c \equiv [0,1] \rightarrow \Omega \equiv [0,1]$ is given, where Ω_c and Ω are the computational and physical domains, respectively. Generally speaking, this transformation is chosen such that the solution in the transformed spatial variable,

$$\hat{u}(\xi,t) = u(x(\xi,t),t),$$

is smooth and in principle economical to approximate using a uniform mesh. A corresponding moving mesh can be described as

$$\mathscr{T}_h(t): \qquad x_j(t) = x(\xi_j,t), \quad j = 1,...,N \tag{1.9}$$

for the fixed, uniform mesh on Ω_c,

$$\mathscr{T}_h^c: \quad \xi_j = \frac{j-1}{N-1}, \quad j = 1,...,N. \tag{1.10}$$

A finite difference discretization of Burgers' equation for $\hat{u}(\xi,t)$ on this moving mesh can be derived using the so-called *quasi-Lagrange approach*. Burgers' equation is transformed from the physical domain to the computational domain using the coordinate transformation as follows: By the chain rule,

$$\hat{u}_\xi = u_x x_\xi, \quad \hat{u}_t = u_t + u_x x_t, \tag{1.11}$$

where $x_t = \frac{\partial x}{\partial t}(\xi,t)$ determines the mesh speed. In the new coordinates (ξ,t) Burgers' equation (1.1) becomes

$$\hat{u}_t - \frac{\hat{u}_\xi}{x_\xi} x_t = \frac{\varepsilon}{x_\xi}\left(\frac{\hat{u}_\xi}{x_\xi}\right)_\xi - \frac{1}{x_\xi}\left(\frac{\hat{u}^2}{2}\right)_\xi. \tag{1.12}$$

Central finite differences can then be used to discretize (1.12) on the uniform computational mesh $\mathscr{T}_h^c$. This yields

$$\begin{aligned}\frac{du_j}{dt} - \frac{(u_{j+1}-u_{j-1})}{(x_{j+1}-x_{j-1})}\frac{dx_j}{dt} = {} & \frac{2\varepsilon}{(x_{j+1}-x_{j-1})}\left[\frac{(u_{j+1}-u_j)}{(x_{j+1}-x_j)} - \frac{(u_j-u_{j-1})}{(x_j-x_{j-1})}\right] \\ & -\frac{1}{2}\frac{(u_{j+1}^2-u_{j-1}^2)}{(x_{j+1}-x_{j-1})}, \quad j = 2,...,N-1\end{aligned} \tag{1.13}$$

where $u_j(t) \approx \hat{u}(\xi_j,t) = u(x_j(t),t)$.

This begs the question of how one determines the coordinate transformation $x = x(\xi,t)$. We defer a detailed discussion of moving mesh methods until Chapter 2, but for now suppose that $x(\xi,t)$ is determined by solving the so-called *moving mesh PDE* (MMPDE)

$$x_t = \frac{1}{\rho\tau}\left(\rho x_\xi\right)_\xi, \tag{1.14}$$

supplemented with the boundary conditions

$$x(0,t) = 0, \quad x(1,t) = 1. \tag{1.15}$$

Here, $\rho = \rho(x,t)$ is called a *mesh density specification function*, or simply *mesh density function*, whose purpose is to control the concentration or density of the mesh, and $\tau > 0$ is a user-specified parameter for adjusting the response time of mesh movement to changes in $\rho(x,t)$. The smaller τ, the more quickly the mesh responds to changes in ρ. Likewise, the mesh moves slowly when a large value of τ is used.

The proper choice of a mesh density function is key to the success of the moving mesh method. One popular choice is

$$\rho = \left(1 + \frac{1}{\alpha}|u_{xx}|^2\right)^{\frac{1}{3}}, \tag{1.16}$$

where α is the intensity parameter given by

$$\alpha = \max\left\{1, \left[\int_0^1 |u_{xx}|^{\frac{2}{3}}\, dx\right]^3\right\}. \tag{1.17}$$

(Such strategies for choosing the mesh density function are explained in Chapter 2.)

Semi-discretization of MMPDE (1.14) on the uniform mesh $\mathscr{T}_h^c$ gives, for $j = 2,\ldots,N-1$,

$$\frac{dx_j}{dt} = \frac{1}{\rho_j \tau \Delta\xi^2}\left[\frac{\rho_{j+1}+\rho_j}{2}(x_{j+1}-x_j) - \frac{\rho_j+\rho_{j-1}}{2}(x_j - x_{j-1})\right], \tag{1.18}$$

and the boundary conditions (1.15) become

$$\frac{dx_1}{dt} = 0, \quad \frac{dx_N}{dt} = 0. \tag{1.19}$$

Here, $\Delta\xi = 1/(N-1)$, and

$$\rho_j = \left(1 + \frac{1}{\alpha_h}\left|u_{xx,j}\right|^2\right)^{\frac{1}{3}}, \quad j = 1,\ldots,N \tag{1.20}$$

$$\alpha_h = \max\left\{1, \left[\sum_{j=2}^{N}\frac{1}{2}(x_j - x_{j-1})\left(\left|u_{xx,j}\right|^{\frac{2}{3}} + \left|u_{xx,j-1}\right|^{\frac{2}{3}}\right)\right]^3\right\}, \tag{1.21}$$

where the second derivative is approximated by

$$\begin{cases} u_{xx,j} = \dfrac{2}{(x_{j+1}-x_{j-1})}\left[\dfrac{(u_{j+1}-u_j)}{(x_{j+1}-x_j)} - \dfrac{(u_j-u_{j-1})}{(x_j-x_{j-1})}\right], \\ \qquad\qquad j = 2,\ldots,N-1 \\ u_{xx,1} = \dfrac{2\left[(x_2-x_1)(u_3-u_1)-(x_3-x_1)(u_2-u_1)\right]}{(x_3-x_1)(x_2-x_1)(x_3-x_2)}, \\ u_{xx,N} = \dfrac{2\left[(x_{N-1}-x_N)(u_{N-2}-u_N)-(x_{N-2}-x_N)(u_{N-1}-u_N)\right]}{(x_{N-2}-x_N)(x_{N-1}-x_N)(x_{N-2}-x_{N-1})}. \end{cases} \tag{1.22}$$

If u is not smooth, the discrete mesh density function computed this way can often change abruptly and unnecessarily slow down the computation. To obtain a smoother mesh and also make the MMPDE easier to integrate, it is common practice in the context of moving mesh methods to smooth the mesh density function. A simple but effective smoothing scheme is weighted averaging, e.g.,

$$\begin{cases} \rho_j := \frac{1}{4}\rho_{j-1} + \frac{1}{2}\rho_j + \frac{1}{4}\rho_{j+1}, \quad j = 2,\ldots,N-1 \\ \rho_1 := \frac{1}{2}\rho_1 + \frac{1}{2}\rho_2, \\ \rho_N := \frac{1}{2}\rho_{N-1} + \frac{1}{2}\rho_N, \end{cases} \tag{1.23}$$

where the symbol ":=" stands for the operation in which the right-hand side terms are calculated and the final value is saved to the variable on the left-hand side. Several sweeps of the scheme may be applied each integration step. (Four sweeps are used for the numerical results presented in this chapter.)

Equations (1.13) and (1.18), supplemented with the boundary conditions (1.8) and (1.19), form a coupled system of $2N$ ordinary differential equations for the physical solution $u_1(t)$, ..., $u_N(t)$ and the mesh $x_1(t)$, ..., $x_N(t)$. Let

$$\boldsymbol{y} = [u_1(t),\ldots,u_N(t),x_1(t),\ldots,x_N(t)]^T, \quad \boldsymbol{y}' = \frac{d\boldsymbol{y}}{dt}.$$

Then the ODE system can be written in the implicit form

$$\boldsymbol{f}(t,\boldsymbol{y},\boldsymbol{y}') = \boldsymbol{0}. \tag{1.24}$$

Performance of an ODE solver can generally be improved by utilizing the nonzero structure of the Jacobian matrices $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}$ and $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}'}$. For the current system, this structure has the form

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}} = \left[\begin{array}{cccccc|cccccc} * & * & & & & & * & * & & & & \\ * & * & * & & & & * & * & * & & & \\ & \ddots & \ddots & \ddots & & & & \ddots & \ddots & \ddots & & \\ & & * & * & * & & & & * & * & * & \\ & & & * & * & & & & & * & * & \\ \hline * & * & \cdots & * & * & * & * & * & \cdots & * & * & \\ * & * & \cdots & * & * & * & * & * & \cdots & * & * & \\ \vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots & \\ * & * & \cdots & * & * & * & * & * & \cdots & * & * & \\ * & * & \cdots & * & * & * & * & * & \cdots & * & * & \end{array}\right] \tag{1.25}$$

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}'} = \left[\begin{array}{cccccc|cccccc} * & & & & & & * & & & & & \\ & * & & & & & & * & & & & \\ & & \ddots & & & & & & \ddots & & & \\ & & & * & & & & & & * & & \\ & & & & * & & & & & & * & \\ \hline & & & & & * & & & & & & * \\ & & & & & & * & & & & & \\ & & & & & & & * & & & & \\ & & & & & & & & \ddots & & & \\ & & & & & & & & & * & & \\ & & & & & & & & & & * & \end{array}\right]. \tag{1.26}$$

Returning to the model problem, the solution and mesh trajectories computed for $\varepsilon = 10^{-4}$ are shown in Figure 1.3. In the computation, an initial uniform mesh with $N = 41$ and the value $\tau = 10^{-2}$ are used. Experience has shown that this value for τ works well for most problems. Oscillations are no longer visible in the computed solution, as the formation and propagation of the steep front are resolved on the adaptive moving mesh of 41 points. The mesh trajectories show how the mesh points respond quickly to the change in the solution.

1.3 A moving finite element method

1.3.1 Finite element method on a fixed mesh

The finite element discretization in space for the model problem is based on the Galerkin formulation of Burgers' equation (1.1). Let

$$V = H_0^1(0,1) \equiv \{u \mid u \in H^1(0,1),\ u(0) = u(1) = 0\}.$$

(a) Computed solution. (b) Mesh trajectories.

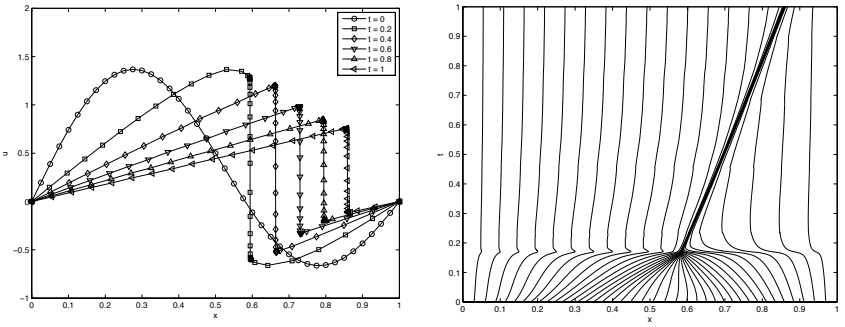

Fig. 1.3 (a) The computed solution for Burgers' equation with $\varepsilon = 10^{-4}$, obtained with an adaptive moving mesh of 41 points, is shown at $t = 0$, 0.2, 0.4, 0.6, 0.8, and 1.0. (b) The corresponding mesh trajectories.

(See Appendix A for the definition of Sobolev space $H^1(0,1)$.) For a given time $T > 0$, the Galerkin formulation of the model problem is the following: Find $u(\cdot,t) \in V$ for $0 < t \le T$ such that

$$\int_0^1 u_t \phi \, dx = \int_0^1 \left(-\varepsilon u_x + \frac{1}{2} u^2 \right) \phi_x dx, \quad \forall \phi \in V, \ 0 < t \le T \tag{1.27}$$

and the initial condition (1.3) holds. The equation (1.27) is obtained by multiplying (1.1) by ϕ, integrating both sides of the resulting equation over the interval $(0,1)$, and integrating by parts on the right-hand side.

We consider the linear finite element approximation on the uniform mesh $\mathscr{T}_h$ in (1.4). Specifically, define the basis functions (hat functions)

$$\phi_j(x) = \begin{cases} \dfrac{x - x_{j-1}}{x_j - x_{j-1}}, & \text{for } x \in [x_{j-1}, x_j] \\ \dfrac{x_{j+1} - x}{x_{j+1} - x_j}, & \text{for } x \in [x_j, x_{j+1}] \\ 0, & \text{otherwise} \end{cases} \qquad j = 1, \ldots, N \tag{1.28}$$

and let V^h be the $(N-2)$-dimensional subspace of V spanned by the basis functions $\phi_2, \ldots, \phi_{N-1}$, i.e.,

$$V^h = \text{ span}\{\phi_2, \ldots, \phi_{N-1}\}.$$

A linear finite element approximation $u^h(\cdot,t) \in V^h$ for $0 < t \le T$ to the exact solution u of the model problem is then required to satisfy

$$\int_0^1 u_t^h \phi \, dx = \int_0^1 \left(-\varepsilon u_x^h + \frac{1}{2} \left(u^h \right)^2 \right) \phi_x dx, \quad \forall \, \phi \in V^h, \ 0 < t \le T \tag{1.29}$$

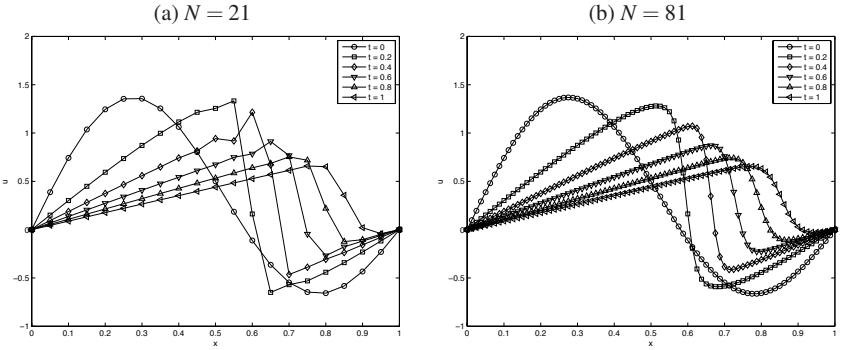

Fig. 1.4 Finite element solutions for Burgers' equation with $\varepsilon = 10^{-2}$, obtained with a uniform mesh, are shown at $t = 0,\ 0.2,\ 0.4,\ 0.6,\ 0.8$, and 1.0.

and

$$u^h(x_j,0) = u(x_j,0), \quad j = 1,...,N. \tag{1.30}$$

Writing

$$u^h(x,t) = \sum_{j=1}^{N} u_j(t)\phi_j(x), \tag{1.31}$$

where $u_j(t) \approx u(x_j,t)$, and taking $\phi = \phi_k(x),\ k = 2,...,N-1$ in (1.29) leads to

$$\sum_{j=1}^{N} \frac{du_j}{dt} \int_0^1 \phi_j(x)\phi_k(x)dx = -\varepsilon \sum_{j=1}^{N} u_j \int_0^1 \phi_j'(x)\phi_k'(x)dx + \frac{1}{2}\int_0^1 \left(\sum_{j=1}^{N} u_j\phi_j(x)\right)^2 \phi_k'(x)dx, \quad k = 2,...,N-1. \tag{1.32}$$

These equations and the boundary conditions (1.8) constitute a nonlinear system of N ordinary differential equations, which can be integrated with N initial conditions (1.30) for the unknown variables $u_1(t),...,u_N(t)$.

An implementation of the finite element method is described below. Figure 1.4 (a) and (b) show the numerical solutions for Burgers' equation with $\varepsilon = 10^{-2}$, obtained using the implementation. The results are comparable to those obtained using the finite difference method in §1.2.1. Once again, oscillations in the solution computed with a uniform mesh of 21 points are significant, unlike for the solution on a finer mesh of 81 points. Like the finite difference method, the finite element method requires a very large number of equidistant points to resolve the steep front when ε is small.

Implementation of finite element method. The initial value problem consisting of the ODE system (1.32) and (1.8), with initial conditions (1.30), can be solved

using a standard ODE solver. The residual of the system, i.e., the function $\boldsymbol{f}(t,\boldsymbol{y},\boldsymbol{y}')$ in the implicit form (1.24) with the unknowns $\boldsymbol{y}=[u_1,...,u_N]^T$ (since the mesh points are fixed), can be easily calculated. From (1.32),

$$\begin{aligned}
f_k(t,\boldsymbol{y},\boldsymbol{y}') &= \int_0^1 u_t^h \phi_k dx - \int_0^1 \left(-\varepsilon u_x^h + \frac{1}{2}\left(u^h\right)^2\right)(\phi_k)_x dx \\
&= \int_{x_{k-1}}^{x_k} \left(\frac{du_{k-1}}{dt}\phi_{k-1} + \frac{du_k}{dt}\phi_k\right)\phi_k dx \\
&\quad - \int_{x_{k-1}}^{x_k} \left(-\varepsilon\left(u_{k-1}\phi_{k-1}+u_k\phi_k\right)_x + \frac{1}{2}\left(u_{k-1}\phi_{k-1}+u_k\phi_k\right)^2\right)(\phi_k)_x dx \\
&\quad + \int_{x_k}^{x_{k+1}} \left(\frac{du_k}{dt}\phi_k + \frac{du_{k+1}}{dt}\phi_{k+1}\right)\phi_k dx \\
&\quad - \int_{x_k}^{x_{k+1}} \left(-\varepsilon\left(u_k\phi_k+u_{k+1}\phi_{k+1}\right)_x + \frac{1}{2}\left(u_k\phi_k+u_{k+1}\phi_{k+1}\right)^2\right)(\phi_k)_x dx
\end{aligned}$$

for $k=2,...,N-1$, and from the boundary conditions (1.8),

$$f_1(t,\boldsymbol{y},\boldsymbol{y}') = \frac{du_1}{dt}, \quad f_N(t,\boldsymbol{y},\boldsymbol{y}') = \frac{du_N}{dt}.$$

A procedure to compute the residual element by element is the following:

(i) Set $f_k = 0$ for $k=1,...,N$.
(ii) For $k=2,...,N$, compute

$$\begin{aligned}
f_{k-1} &:= f_{k-1} + \int_{x_{k-1}}^{x_k} \left(\frac{du_{k-1}}{dt}\phi_{k-1} + \frac{du_k}{dt}\phi_k\right)\phi_{k-1} dx \\
&\quad - \int_{x_{k-1}}^{x_k} \left(-\varepsilon\left(u_{k-1}\phi_{k-1}+u_k\phi_k\right)_x + \frac{1}{2}\left(u_{k-1}\phi_{k-1}+u_k\phi_k\right)^2\right)(\phi_{k-1})_x dx, \\
f_k &:= f_k + \int_{x_{k-1}}^{x_k} \left(\frac{du_{k-1}}{dt}\phi_{k-1} + \frac{du_k}{dt}\phi_k\right)\phi_k dx \\
&\quad - \int_{x_{k-1}}^{x_k} \left(-\varepsilon\left(u_{k-1}\phi_{k-1}+u_k\phi_k\right)_x + \frac{1}{2}\left(u_{k-1}\phi_{k-1}+u_k\phi_k\right)^2\right)(\phi_k)_x dx.
\end{aligned}$$

(iii) Modify f_1 and f_N according to the boundary conditions:

$$f_1 := \frac{du_1}{dt} - 0, \quad f_N := \frac{du_N}{dt} - 0.$$

Due to the simplicity of the model problem, the integrals in the above formulas are simple enough to evaluate exactly, but in general, numerical quadrature is needed. In such a case, it suffices to use the two-point Gaussian quadrature rule satisfying

$$\int_{x_{k-1}}^{x_k} f(x)dx = \frac{x_k - x_{k-1}}{2}\left(f(x_{k,1}) + f(x_{k,2})\right) + O\left((x_k - x_{k-1})^5\right),$$

where $x_{k,1} = x_{k-1} + s_1(x_k - x_{k-1})$, $x_{k,2} = x_{k-1} + s_2(x_k - x_{k-1})$ for the two Gaussian points

$$s_1 = \frac{1}{2}\left(1 - \frac{1}{\sqrt{3}}\right), \quad s_2 = \frac{1}{2}\left(1 + \frac{1}{\sqrt{3}}\right).$$

1.3.2 Finite element method on an adaptive moving mesh

A moving mesh strategy for the finite difference method as discussed in §1.2.2 (e.g., using the MMPDE (1.14)) applies in essentially the same way for the finite element method. For this reason, we focus our discussion in this subsection on the finite element semi-discretization of the model problem on the moving mesh $\mathscr{T}_h(t)$ in (1.9).

Since the mesh is moving, the basis functions and the approximation function space are now time-dependent, viz.,

$$\phi_j(x,t) = \begin{cases} \dfrac{x - x_{j-1}(t)}{x_j(t) - x_{j-1}(t)}, & \text{for } x \in [x_{j-1}(t), x_j(t)] \\ \dfrac{x_{j+1}(t) - x}{x_{j+1}(t) - x_j(t)}, & \text{for } x \in [x_j(t), x_{j+1}(t)] \\ 0, & \text{otherwise} \end{cases} \qquad j = 1, ..., N \quad (1.33)$$

and

$$V^h(t) = \text{ span}\{\phi_2(\cdot,t), ..., \phi_{N-1}(\cdot,t)\}.$$

The finite element approximation can be defined in a similar fashion as for a fixed mesh: Find $u^h(\cdot,t) \in V^h(t)$ for $0 < t \le T$ such that

$$\int_0^1 u_t^h \phi dx = \int_0^1 \left(-\varepsilon u_x^h + \frac{1}{2}\left(u^h\right)^2\right)\phi_x dx \quad \forall \phi \in V^h(t),\ 0 < t \le T, \qquad (1.34)$$

and

$$u^h(x_j(0),0) = u(x_j(0),0), \quad j = 1, ..., N. \qquad (1.35)$$

The time derivative of u^h now requires special attention. Writing

$$u^h(x,t) = \sum_{j=1}^{N} u_j(t)\phi_j(x,t), \qquad (1.36)$$

where $u_j(t) \approx u(x_j(t),t)$, the time derivative becomes

$$u_t^h(x,t) = \sum_{j=1}^{N} \left(\frac{du_j}{dt}(t)\phi_j(x,t) + u_j(t)\frac{\partial \phi_j}{\partial t}(x,t) \right).$$

A direct calculation using the basis function ϕ_j in (1.33) shows that

$$\frac{\partial \phi_j}{\partial t}(x,t) = -\frac{\partial \phi_j}{\partial x}(x,t) X_t(x,t), \tag{1.37}$$

where $X_t(x,t)$ is the linear interpolant of the nodal mesh speeds, i.e.,

$$X_t(x,t) = \sum_{j=1}^{N} \frac{dx_j}{dt}(t)\phi_j(x,t). \tag{1.38}$$

Thus,

$$\begin{aligned} u_t^h(x,t) &= \sum_{j=1}^{N} \left(\frac{du_j}{dt}(t)\phi_j(x,t) - u_j(t)\frac{\partial \phi_j}{\partial x}(x,t)\, X_t(x,t) \right) \\ &= \sum_{j=1}^{N} \frac{du_j}{dt}(t)\phi_j(x,t) - \frac{\partial u^h}{\partial x}(x,t)\, X_t(x,t). \end{aligned}$$

Inserting this into (1.34) yields, for any $t \in (0,T]$ and $\phi \in V^h(t)$,

$$\int_0^1 \left(\sum_{j=1}^{N} \frac{du_j}{dt}\phi_j - \frac{\partial u^h}{\partial x}\, X_t \right) \phi dx = \int_0^1 \left(-\varepsilon u_x^h + \frac{1}{2}\left(u^h\right)^2 \right) \phi_x \, dx. \tag{1.39}$$

Observe that the mesh movement introduces an extra convection term,

$$-\frac{\partial u^h}{\partial x}\, X_t.$$

Interestingly, this term has a similar form to the convection term

$$-\frac{\hat{u}_\xi}{x_\xi} x_t = -\frac{\partial u}{\partial x} x_t$$

in (1.12), derived using the coordinate transformation $x(\xi,t)$.

The system of ODEs for $u_1,\ldots,u_N$ is obtained by substituting $\phi = \phi_k(x,t)$ into (1.39), viz., for $k = 2,\ldots,N-1$,

$$\int_0^1 \left(\sum_{j=1}^{N} \frac{du_j}{dt}\phi_j - \frac{\partial u^h}{\partial x}\, X_t \right) \phi_k dx = \int_0^1 \left(-\varepsilon u_x^h + \frac{1}{2}\left(u^h\right)^2 \right) (\phi_k)_x \, dx, \tag{1.40}$$

and these equations are supplemented with the boundary conditions (1.8).

Combining with the mesh movement conditions, we now have a coupled system consisting of the discrete physical equations (1.40), mesh equations (1.18), and cor-

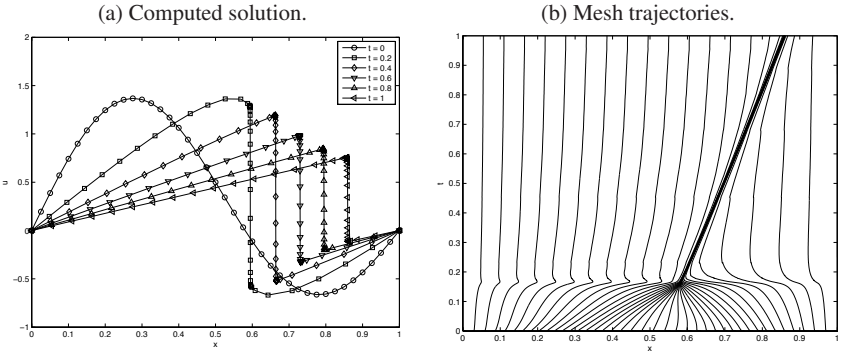

Fig. 1.5 (a) The finite element solution for Burgers' equation with $\varepsilon = 10^{-4}$, obtained with an adaptive moving mesh of 41 points, is shown at $t = 0,\ 0.2,\ 0.4,\ 0.6,\ 0.8$, and 1.0. (b) The corresponding mesh trajectories.

responding boundary conditions for variables $\boldsymbol{y} = [u_1(t), ..., u_N(t), x_1(t), ..., x_N(t)]^T$. It can be cast in the form (1.24), with $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}$ having a nonzero structure as in (1.25) and the Jacobian matrix $\frac{\partial \boldsymbol{f}}{\partial \dot{\boldsymbol{y}}}$ having the slightly different structure

$$\frac{\partial \boldsymbol{f}}{\partial \dot{\boldsymbol{y}}} = \left[\begin{array}{cc|cc} * & * & * & * \\ * & * & * & * \\ \ddots & \ddots & \ddots & \ddots \\ * & * & * & * \\ * & * & * & * \\ \hline & & * & \\ & & & * \\ & & & \ddots \\ & & & * \\ & & & * \end{array}\right]. \tag{1.41}$$

Figure 1.5 shows the solution and mesh trajectories obtained for Burgers' equation with $\varepsilon = 10^{-4}$. An ODE solver has been used to integrate the resulting system of ordinary differential equations in the implicit form (1.24). Four smoothing cycles with the weighted averaging (1.23) are applied to the monitor function each time it is computed. The results are almost identical to those shown in Figure 1.3 for the finite difference method. Once again, the steep front is resolved on an adaptive mesh of 41 points, and the mesh responds well to the change in the solution.

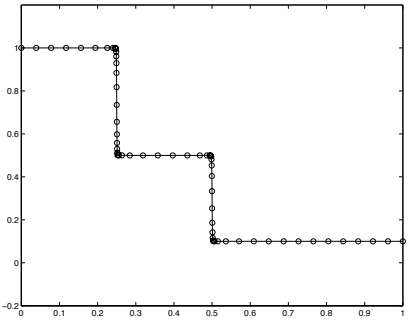

Fig. 1.6 An adaptive initial mesh of 61 points is shown on the graph of the initial solution (1.42).

1.4 Burgers' equation with an exact solution

As another example, an initial-boundary value problem for Burgers' equation with a known solution is solved with the moving finite difference method described in §1.2. (Results for the moving finite element method in §1.3 are similar, and not given.) The problem consists of Burgers' equation (1.1) together with Dirichlet boundary and initial conditions chosen such that the exact solution is

$$u(x,t) = \frac{0.1e^{\frac{-x+0.5-4.95t}{20\varepsilon}} + 0.5e^{\frac{-x+0.5-0.75t}{4\varepsilon}} + e^{\frac{-x+0.375}{2\varepsilon}}}{e^{\frac{-x+0.5-4.95t}{20\varepsilon}} + e^{\frac{-x+0.5-0.75t}{4\varepsilon}} + e^{\frac{-x+0.375}{2\varepsilon}}}, \tag{1.42}$$

where ε is taken as $\varepsilon = 10^{-4}$. Initially, the solution has two steep fronts (physically corresponding to shock waves in a fluid) traveling toward the right end. They merge around $t = 0.55$ and form a steeper shock wave.

Since the initial solution, $u(x,0)$, has two steep fronts, an adaptive initial mesh should be used to start the integration. Such a mesh is generated using a time-continuation method. Specifically, if we define $v(x,t) = tu(x,0)$, where t here denotes the continuation parameter, then v satisfies the differential equation

$$\frac{\partial v}{\partial t} = u(x,0), \quad x \in [0,1],\ 0 < t \le T, \tag{1.43}$$

subject to the initial condition $v(x,0) = 0$. Starting with a uniform mesh, the system consisting of (1.43) and a suitable MMPDE is integrated from $t = 0$ to $t = T$ to obtain an adaptive mesh for $v(x,T) = Tu(x,0)$, and for sufficiently large T, this mesh gives a suitable approximation for $u(x,0)$ (or the multiple $Tu(x,0)$). Here, using $T = 1$ and $N = 61$, the graph of $u(x,0)$ and the points on the graph corresponding to the adaptive mesh obtained this way are shown in Figure 1.6.

Once this adaptive initial mesh for $u(x,0)$ is obtained, Burgers' equation (1.1) with the corresponding boundary conditions is integrated using the moving finite

difference method. Figure 1.7 shows a computed solution and the corresponding mesh trajectories obtained with a moving mesh of 61 points. The corresponding time step size used in the integration is shown in Figure 1.8. The convergence history in Figure 1.9 shows that when a moving mesh is used, the error at $t = 1$ in the H^1 semi-norm (see Appendix A) converges at the rate $O(N^{-1})$. In contrast, when a uniform mesh is used, the method does not converge for the range of values of N considered.

To give a sense of the efficiency or the cost-effectiveness of the adaptive moving mesh method, the H^1 semi-norm of the error at $t = 1$ is plotted in Figure 1.10 as a function of the scaled CPU time for three cases – one with uniform meshes, and the other two with adaptive meshes obtained using the intensity parameter $\alpha = \alpha(u)$ defined in (1.17) and using $\alpha = 1$. The error is smaller with adaptive meshes than with a uniform mesh for the same amount of the CPU time and, to reach the same level of error, more CPU time is required when a uniform mesh is used. In this sense, the adaptive moving mesh method is more efficient than a uniform mesh method.

No special effort has been made to optimize the performance of the methods used for Figure 1.10. Generally speaking, a uniform mesh method runs much faster than an adaptive mesh method for the same number of mesh points. This is because the linear algebraic systems resulting from implicit time discretization are tridiagonal and can be solved extremely fast when a uniform mesh is used. In contrast, for the moving mesh method with $\alpha = \alpha(u)$, the Jacobian matrix has a denser nonzero structure (cf. (1.25)), so computing its finite difference approximations (typically used in an ODE solver) and doing the inversion require more CPU time than for a tridiagonal system. The situation can be improved (cf. Figure 1.10) by using a constant intensity parameter such as $\alpha = 1$ in (1.16) which leads to a Jacobian matrix with a sparser nonzero structure

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}} = \begin{bmatrix} A & A \\ B & B \end{bmatrix}, \tag{1.44}$$

where

$$A = \begin{bmatrix} * & * & & & \\ * & * & * & & \\ & \ddots & \ddots & \ddots & \\ & & * & * & * \\ & & & * & * \end{bmatrix}, \quad B = \begin{bmatrix} * & \cdots & * & & & & \\ \vdots & \ddots & & \ddots & & & \\ * & & \ddots & & & & \\ & \ddots & & \ddots & & \ddots & \\ & & & & \ddots & & * \\ & & & \ddots & & \ddots & \vdots \\ & & & & * & \cdots & * \end{bmatrix}.$$

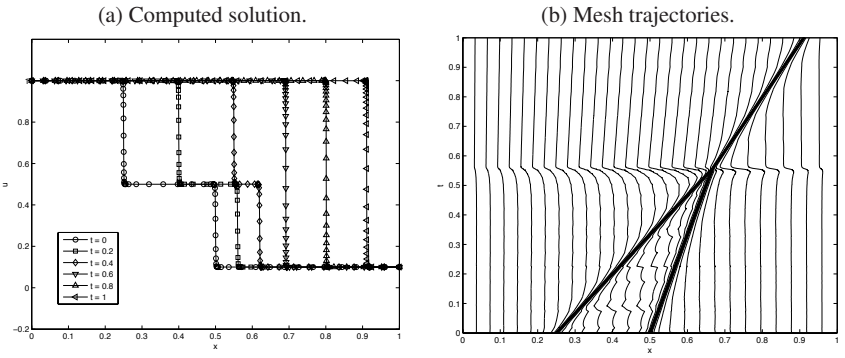

Fig. 1.7 (a) The finite difference solution for Burgers' equation with $\varepsilon = 10^{-4}$, obtained with an adaptive moving mesh of 61 points, is shown at $t = 0$, 0.2, 0.4, 0.6, 0.8, and 1.0. (b) The corresponding mesh trajectories.

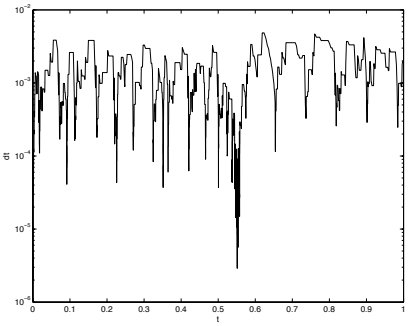

Fig. 1.8 The time step size used in the adaptive moving mesh solution of Burgers' equation with $\varepsilon = 10^{-4}$ and 61 points is plotted as function of time. The relative and absolute tolerances for the time step control are taken as $rtol = 10^{-6}$ and $atol = 10^{-4}$, respectively, for the Matlab ODE solver "ode15i" (using a backward differentiation formula of order 5).

The upper and lower bandwidths of B are equal to 2 plus the number of sweeps used in smoothing the mesh density function. Strategies for choosing the mesh density function and the intensity parameter are considered in detail in Chapter 2.

The difference in time required to solve algebraic systems for the uniform and adaptive mesh methods turns out to be less significant in two and three dimensions, where the algebraic systems for the uniform mesh method are generally much more expensive to solve.

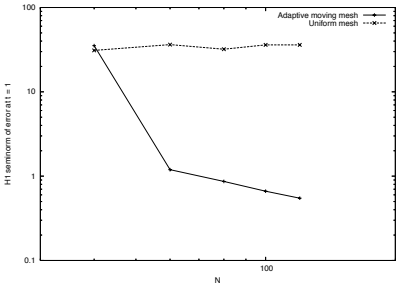

Fig. 1.9 The H^1 semi-norm of the error at $t = 1$ is plotted against the number of mesh points, N, for uniform and adaptive moving meshes.

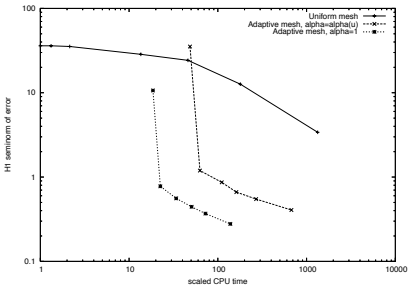

Fig. 1.10 The H^1 semi-norm of the error at $t = 1$ is plotted against the CPU time scaled by the time required for the computation of the case with a uniform mesh of 100 points.

1.5 Basic components of a moving mesh method

Thus far, we have seen that a moving mesh method has three major components: the strategy used to move the mesh, the method employed to discretize the physical PDE, and the approach used to solve the coupled system of physical and mesh equations. In particular, for the finite difference and finite element moving mesh methods described in this chapter, the mesh is moved using the moving mesh PDE (1.14), the physical PDE is discretized with the quasi-Lagrange approach using finite differences or finite elements, and the coupled system of physical and mesh PDEs is solved simultaneously with a general ODE solver. Motivated by this elementary understanding of the moving mesh method for solving Burgers' equation, we complete this chapter with a brief discussion of various options available when implementing these three components of a moving mesh method.

1.5.1 Mesh movement strategies

Mesh movement is usually performed either by solving an elliptic or parabolic system of PDEs involving the mesh coordinate transformation or by doing a direct error-based minimization. The derivation of the moving mesh system is typically motivated by the equidistribution principle. It makes use of an error density function (which is referred to as a mesh density function) which is required to be evenly distributed among all the mesh elements. In one dimension, the equidistribution condition, together with suitable boundary conditions, uniquely determines a mesh for a given mesh density function. However, as we shall see, this is not the case in multidimensions. Generally speaking, a condition regularizing the shape of mesh elements, in addition to equidistribution, is needed to determine a suitable multidimensional mesh. For instance, for isotropic mesh adaptation, mesh elements are required to be nearly equilateral, whereas for anisotropic mesh adaptation, a shape-alignment condition is often imposed on mesh elements. Studies of basic principles of mesh adaptation, including equidistribution and shape-alignment conditions, are formally addressed in Chapter 4. Note that while the formulation of a mesh movement strategy is independent of the type of the physical PDE to be solved, the discretization method and solution strategy are not.

A variational approach is perhaps the most natural one for formulating elliptic or parabolic PDE based mesh generators. (The moving mesh PDE (1.14), for instance, is derived using the variational approach.) For this approach, mesh equations are defined as the Euler-Lagrange equations of a functional specially designed for the purpose of mesh adaptation. A number of adaptation functionals have been developed in the past based on error estimates and geometric considerations. They are discussed in Chapter 6.

Variational strategies are special examples of the so-called *location-based* mesh movement strategies which control directly the location of mesh points. A different group of strategies is *velocity-based* since it targets directly the mesh velocity and obtains the location of mesh points by integrating the velocity field. A majority of methods of this type are motivated by the Lagrangian method in fluid dynamics where the mesh coordinates, defined to follow fluid particles, are obtained by integrating flow velocity. Velocity-based strategies are presented in Chapter 7.

1.5.2 Discretization of PDEs on a moving mesh

Finite differences and finite elements have been used in this chapter for spatial discretization of the physical PDE on a moving mesh. As we have seen, the physical PDE is discretized on the computational domain when finite differences are used and

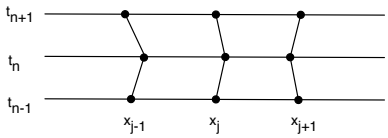

Fig. 1.11 The mesh points are considered to move continuously for the quasi-Lagrange approach of temporal discretization of physical PDEs.

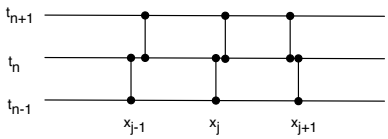

Fig. 1.12 The mesh is considered to vary only at time instants $t = t_n$, $n = 0, 1, ...$ for the rezoning approach of temporal discretization of physical PDEs.

on the physical domain when finite elements are used. This is because normally a finite difference discretization can only be conveniently carried out on a rectangular or a cubic mesh in the computational domain, whereas a finite element discretization can be used directly on a non-uniform mesh in the physical domain. Discretization in the computational domain requires that the physical PDE be transformed into the computational variables, often giving a complicated form. In contrast, discretization on the physical mesh can avoid such a complexity.

The effect of mesh movement in the time discretization of the physical PDE can be treated with either *the quasi-Lagrange approach* or *the rezoning approach*. With the quasi-Lagrange approach, the mesh points are considered to move continuously in time (cf. Figure 1.11), and physical time derivatives are transformed into time derivatives along mesh trajectories supplemented with a convective term reflecting mesh movement. The new time derivative and the extra convection term are typically treated in the same way as other terms in the physical PDE. Alternatively, with the rezoning approach, the mesh points are considered to move in an intermittent manner in time (cf. Figure 1.12). More precisely, the mesh is updated at each time level using certain mesh equations or generators, the physical solution is interpolated from the old mesh to the new one, and the physical PDE is then discretized on the new mesh, which is held fixed for the current time step. Interpolation of the physical solution is a crucial step for the success of this approach, and it is often necessary to use a conservative interpolation scheme which preserves some solution quantities. This issue is discussed in detail in Chapters 2 and 3.

1.5.3 Simultaneous or alternate solution

We have seen that for a moving mesh method, the discrete physical PDE and the mesh equation give a coupled system. Some basic features of the solution procedure for this system, along with the approach used to treat the mesh movement in the physical PDE, are briefly discussed here. By design the rezoning approach involves solving alternately for the physical solution and the mesh. For the quasi-Lagrange approach, the physical PDE and the mesh equation for the moving mesh method can be solved either *simultaneously* or *alternately*.

A simultaneous solution procedure is illustrated in Figure 1.13. For it, the mesh equation and physical PDE are treated as one large system which is solved simultaneously for the mesh and physical solution. This is illustrated in §1.2 and §1.3, where the mesh equation (1.18) is solved together with physical equation (1.13) or (1.32). Simultaneous solution is in principle relatively simple and has the advantage that standard, well-developed ODE solvers can be directly applied to the integration of the extended system arising from applying the method of lines to the mesh and physical equations. Moreover, the physical solution and the mesh are tightly coupled, with the mesh responding promptly to any change occurring in the physical solution. The main disadvantage of simultaneous solution is the highly nonlinear coupling between the physical solution and the mesh. Even a linear physical PDE can result in a highly nonlinear equation in the new variables (cf. (1.13)). The extended system also has a more complicated structure (cf. (1.25)) and often loses features which the physical PDE may have in the physical variables, such as symmetry and positive definiteness. These factors often make the extended system more difficult and expensive to solve.

The alternate solution procedure is illustrated in Figure 1.14. A mesh x^{n+1} at the new time level is first generated using the mesh and the physical solution (x^n, u^n) at the current time level, and the solution u^{n+1} is then obtained at the new time level. Note that this mesh x^{n+1} adapts only to the current solution u^n and thus lags in time. This will not generally cause much trouble if the time step is reasonably small or the solution does not have abrupt changes in time. If the lag of the mesh in time causes a serious problem, several iterations of solving for the mesh and the physical PDE at each new time level can be used (cf. Figure 1.14). The main advantages of the alternate solution procedure are (i) its flexibility (the mesh generation part can be coded separately as a module to incorporate into the PDE solver) and (ii) its potential efficiency at each time step (structures for each of the physical and mesh equations can be fully explored to improve efficiency). Since the mesh adaptation is not tied to the solution process for the physical PDE, the mesh generator does not have to take the form of a differential equation. A minimization-based mesh generator, for example, can suffice equally well. In addition to the above-mentioned disadvantage of the lag of the mesh in time, the alternate solution procedure runs the risk of

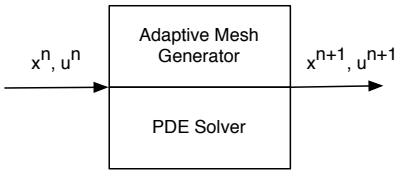

Fig. 1.13 Illustration of a simultaneous solution procedure.

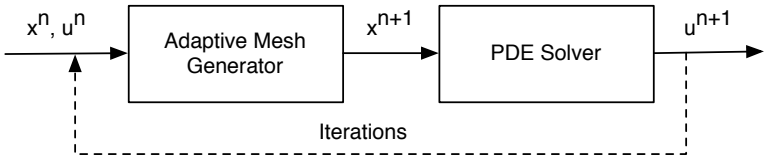

Fig. 1.14 Illustration of an alternate solution procedure.

causing instability in the integration because it does not have a mechanism built in to force the system back on track once the mesh is not generated accurately enough at one time step. Relatively speaking, this risk is smaller with the simultaneous solution method because the physical solution and the mesh are forced to satisfy the physical PDE and the mesh equation simultaneously at each time step.

The simultaneous solution procedure has been limited mainly to one-dimensional problems in space, and most of the existing moving mesh methods for multidimensional computation employ an alternate solution procedure.

1.6 Biographical notes

Roughly speaking, mesh movement algorithms can be classified into the two groups, *velocity-based* algorithms and *location-based* ones [85], cf. §1.5.1. Many of the velocity-based algorithms have been motivated by the Lagrangian method in fluid dynamics (e.g., see Batchelor [38]), and a major consideration in their development has been to avoid mesh tangling, an undesired property of the Lagrangian method. Examples include the method of Yanenko et al. [345] that is of Lagrange-type. In the work of Anderson and Rai [13], the mesh is moved according to attraction and repulsion pseudo-forces between nodes motivated by a spring model in mechanics. The moving finite element method (MFE) of Miller and Miller [258] and Miller [253] has aroused considerable interest. It computes the solution and the mesh simultaneously by minimizing the residual of the PDEs written in a finite element form. Penalty terms are added to avoid possible singularities in the mesh movement equations; see [88, 89]. A way of treating the singularities but without using penalty functions has been proposed by Wathen and Baines [339]. Pet-

zold [274] obtains an equation for mesh velocity by minimizing the time variation of both the unknown variable and the spatial coordinate in computational coordinates and adding a diffusion-like term to the mesh equation. Liao and his coworkers [53, 226, 234, 236, 232, 241] employ a deformation map to move the mesh. In [84], Cao, Huang, and Russell develop the GCL method, which is based on the Geometric Conservation Law. Interestingly, the deformation map method can be viewed as a special example of the GCL method (see §7.1). A similar idea has been used by Baines et al. [31, 32, 33] for the development of the so-called moving mesh finite element method.

In contrast, location-based mesh movement algorithms provide a direct control of the location of mesh points. A natural and important approach for designing this type of algorithm is the variational approach for which the mesh point relocation and movement are determined by minimizing some functional formulated to measure error or difficulty in numerical simulation. Many location-based algorithms have been developed as variational ones, whereas some others have been based on elliptic PDEs or other considerations. For example, Winslow [341] and Thompson et al. [324] use a system of elliptic PDEs for generating boundary-fitted meshes. Winslow [342] proposes to generate adaptive meshes through a variable diffusion model. The idea is generalized by Brackbill and Saltzman [58], who combine functionals representing mesh adaptivity, smoothness, and orthogonality. This is further modified by Brackbill [57] to include directional control in mesh adaptation and to require the terms contained in the functional to be dimensionally homogeneous. The method of Dorfi and Drury [124] is linked to a functional associated with the well-known equidistribution principle [115, 186] while that of Dvinsky [129] is based on the energy of a harmonic mapping for mesh adaptation. Examples of other mesh adaptation functionals can be found in Jacquotte et al. [202, 203, 204] (based on mechanical models), Knupp [211] (using vector fields), Knupp et al. [212, 214] (using a weighted or reference Jacobian matrix), Huang and Russell [189] and Cao et al. [82] (using a generalized variable diffusion functional with a matrix-valued diffusion coefficient), and Huang [176] (based on the so-called equidistribution and isotropy (or alignment) conditions). These functionals are discussed in Chapter 6. The moving mesh PDE (MMPDE) method developed in [81, 185, 186, 189, 190, 282] moves the mesh through the gradient flow equation of an adaptation functional. Tang et al. [228, 229, 316] use the generalized variable diffusion functional (cf. [82, 189]) as their adaptation functional, but discretize the physical PDE in the rezoning approach. Budd and Williams [71] use a parabolic Monge-Ampère equation to move adaptive meshes. The methods of Ren and Wang [280] and Ceniceros and Hou [95] also deserve special attention.

There exist a number of review articles and books addressing (at least partially) moving mesh methods. Review articles include Russell and Christiansen [285], Thompson et al. [326], Thompson [323], Eiseman [132, 133], Hawken et al. [169],

Thompson and Weatherill [327], Huang and Russell [191], Cao et al. [85], Sloan [303], and more recently, Huang [181] and Budd et al. [68]. Hawken et al. [169] give a particularly extensive overview and list of references on moving mesh methods before 1990. Relevant books include Thompson et al. [325], Ascher et al. [16], Knupp and Steinberg [213], Baines [29], Zegeling [347], Carey [86], and Liseikin [238]. Relevant conference proceedings and edited books include Babuška et al. [23], Castillo [92], Shi et al. [299], and Tang and Xu [319].

This book is mainly concerned with the *r*-adaptive, or moving mesh, method; for other types of adaptive mesh methods, especially the *h*-adaptive mesh method and the Adaptive Mesh Refinement (AMR) method, the interested reader is referred to books such as Baden et al. [27], Carey [86], Ern and Guermond [135], Frey and George [150], George [154], Lang [222], Linß [237], Plewa et al. [275], and Sarris [293].

1.7 Exercises

1. Assume that the function $u = u(x)$ is sufficiently smooth around point x. Find the order of the truncation error for the following finite difference approximations:

$$\begin{aligned}
\frac{du}{dx}(x) &\approx \frac{u(x+h)-u(x)}{h},\\
\frac{du}{dx}(x) &\approx \frac{u(x)-u(x-h)}{h},\\
\frac{du}{dx}(x) &\approx \frac{u(x+h)-u(x-h)}{2h},\\
\frac{du}{dx}(x) &\approx \frac{u(x+\frac{h}{2})-u(x-\frac{h}{2})}{h},\\
\frac{d^2u}{dx^2}(x) &\approx \frac{u(x+h)-2u(x)+u(x-h)}{h^2},
\end{aligned}$$

   where h is a small positive number.
2. Derive the three-point central finite difference approximation to the second derivative $\frac{d^2u}{dx^2}(x)$ on a non-uniform mesh. What are the leading terms in the truncation error?
3. Assume that functions $u(x)$ and $p(x)$ are sufficiently smooth around point x. Derive the approximation

$$\begin{aligned}
\frac{d}{dx}\left(p(x)\frac{du}{dx}\right) \approx \frac{1}{h}\Bigg(&\frac{(p(x+h)+p(x))}{2}\frac{(u(x+h)-u(x))}{h}\\
&- \frac{(p(x-h)+p(x))}{2}\frac{(u(x)-u(x-h))}{h}\Bigg),
\end{aligned}$$

where h is a small positive number. Find the leading terms in the truncation error of the approximation.

4. Consider a central finite difference approximation on a uniform mesh to the boundary value problem

$$\begin{aligned} &-u'' + u' = 1, \quad \forall x \in (0,1) \\ &u(0) = u(1) = 0. \end{aligned}$$

(a) Derive the scheme; (b) find the local truncation error; and (c) write down the matrix form of the resulting algebraic system explicitly.

5. Prove (1.11) using the chain rule.
6. For sufficiently smooth functions $u = u(x)$ and $x = x(\xi)$, let $\hat{u} = u(x(\xi))$. Show that

$$\frac{d^2u}{dx^2} = \frac{1}{x_\xi}\frac{d}{d\xi}\left(\frac{1}{x_\xi}\frac{d\hat{u}}{d\xi}\right),$$

where $x_\xi = dx/d\xi$.

7. Derive (1.12).
8. Derive the semi-discrete scheme (1.13).
9. For $\rho = 1$, find the general solution of MMPDE (1.14) and boundary condition (1.15) for any initial coordinate transformation $x(\xi,0) = x_0(\xi)$. Discuss the monotonicity of the solution in space and its asymptotical behavior as $t \to \infty$. (Hint: Set $x = \xi + \phi$ and solve the equation for ϕ using the Fourier series method or the method of separation of variables.)
10. Evaluate the integrals

$$\int_{x_{k-1}}^{x_k} \phi_{k-1}\phi_k dx, \quad \int_{x_{k-1}}^{x_k} \phi_k\phi_k dx,$$

$$\int_{x_{k-1}}^{x_k} \phi_{k-1}^{'}\phi_k^{'} dx, \quad \int_{x_{k-1}}^{x_k} \phi_k^{'}\phi_k^{'} dx,$$

$$\int_{x_{k-1}}^{x_k} \phi_{k-1}^{'}\phi_k dx, \quad \int_{x_{k-1}}^{x_k} \phi_k^{'}\phi_k dx,$$

where ϕ_k and ϕ_{k-1} are the basis functions defined in (1.28).

11. Consider a linear finite element approximation on a uniform mesh to the boundary value problem

$$\begin{cases} -u'' + u' = 1, \quad \forall x \in (0,1) \\ u(0) = u(1) = 0. \end{cases}$$

(a) Using the results in Problem 10, derive the scheme and (b) write down the matrix form of the resulting algebraic system explicitly.

12. Implement on computer the finite difference and finite element schemes in Problems 4 and 11.

13. Use direct calculation to derive (1.37).

http://www.springer.com/978-1-4419-7915-5

Adaptive Moving Mesh Methods

Weizhang, H.; Russell, R.D.
2011, XVIII, 434 p., Hardcover
ISBN: 978-1-4419-7915-5